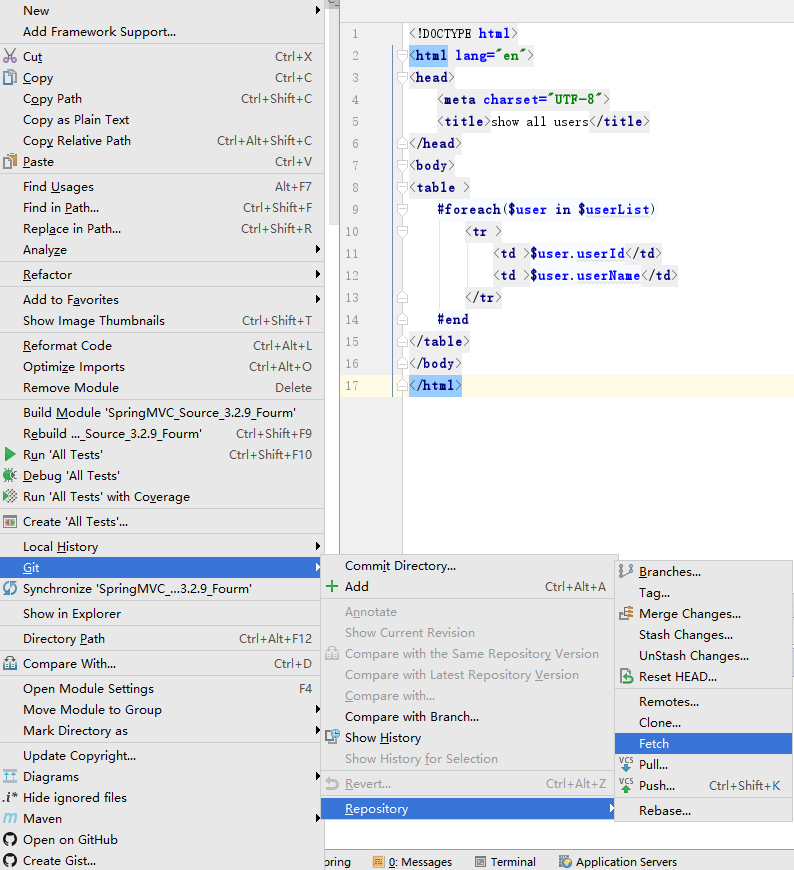
**Git-2.10.0-64-bit.exe是git客户端**

**GitHub for windows 3.0.6.4.zip是GitHub平台项目管理工具**

**Git常用命令**

## Fetch

在idea中，可以使用如下操作，拉取远程分支代码



**3.远程分支就是本地分支push到服务器上。比如master就是一个最典型的远程分支（默认）。**

git push origin 2.0.1.20120806

**4.远程分支和本地分支需要区分好，所以，在从服务器上拉取特定分支的时候，需要指定远程分支的名字。**

git checkout --track origin/2.0.1.20120806

注意该命令由于带有--track参数，所以要求git1.6.4以上！这样git会自动切换到分支。

**5.提交分支数据到远程服务器**

git push origin <local\_branch\_name>:<remote\_branch\_name>

例如：

git push origin 2.0.1.20120806:2.0.1.20120806

一般当前如果不在该分支时，使用这种方式提交。如果当前在 2.0.1.20120806 分支下，也可以直接提交

git push

git clone

**[语法：git](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/u013370834/article/details/_blank) clone 服务器代码仓库地址**

示例：git clone <https://github.com/dwqs/blog.git>

示例：git clone git@gitlab.alibaba-inc.com:tmall-fmcg/smart-store-platform.git

说明：

## **git branch**

**查看当前本地分支：git branch**

**创建本地分支：git branch 新分支名**

示例：git branch 2.0.1.20120806

说明：基于当前文件创建一个分支，仅仅只是创建，本地文件还是对应当前分支

**查看所有分支信息：git branch -a**

**切换本地文件到指定分支：git checkout 新分支名**

示例：git checkout 2.0.1.20120806

**完整示例：**

wb-whz291815@MININT-3U7OIUK MINGW32 ~/Desktop/新建文件夹

**$ git clone https://github.com/whz/testDemo.git**

wb-whz291815@MININT-3U7OIUK MINGW32 ~/Desktop/新建文件夹

**$ cd testDemo**

wb-whz291815@MININT-3U7OIUK MINGW32 ~/Desktop/新建文件夹/testDemo (master)

**$ git branch**

\* master

wb-whz291815@MININT-3U7OIUK MINGW32 ~/Desktop/新建文件夹/testDemo (master)

**$ git branch -a**

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/fenzhi\_v1\_20170626

remotes/origin/master

wb-whz291815@MININT-3U7OIUK MINGW32 ~/Desktop/新建文件夹/testDemo (master)

**$ git checkout fenzhi\_v1\_20170626**

Switched to a new branch 'fenzhi\_v1\_20170626'

Branch fenzhi\_v1\_20170626 set up to track remote branch fenzhi\_v1\_20170626 from origin.

## **删除远程分支和tag**

方法一：

删除远程分支：**git push origin --delete <branchName>**

删除tag：**git push origin --delete tag <tagname>**

说明：在Git v1.7.0 之后，可以使用这种语法删除远程分支

方法二：

删除远程分支：**git push origin :<branchName>**

这是删除tag的方法，推送一个空tag到远程tag：

**git tag -d <tagname>**

**git push origin :refs/tags/<tagname>**

说明：推送一个空分支到远程分支，其实就相当于删除远程分支

# 详细教程

**上面这张图可以直观的表示以下要介绍的命令间的关系：**

* workspace 即工作区，逻辑上是本地计算机，还没添加到repository的状态；
* staging 即版本库中的stage，是暂存区。修改已经添加进repository，但还没有作为commit提交，类似于缓存；
* Local repository 即版本库中master那个地方。到这一步才算是成功生成一个新版本；
* Remote repository 则是远程仓库。用来将本地仓库上传到网络，可以用于备份、共享、合作。本文将使用Github作为远程仓库的例子。

所有命令前都要加git，以下是常用的git命令，变量使用“[xxx]”表示

| **行为** | **命令** | **备注** |
| --- | --- | --- |
| 初始化 | **git [init](http://www.cnblogs.com/schaepher/p/5561193.html" \l "init)** | 在本地的当前目录里初始化git仓库 |
|  | **git [clone [地址]](http://www.cnblogs.com/schaepher/p/5561193.html" \l "clone)** | 从网络上某个地址拷贝仓库(repository)到本地 |
| 查看当前状态 | **git [status](http://www.cnblogs.com/schaepher/p/5561193.html" \l "status)** | 查看当前仓库的状态。碰到问题不知道怎么办的时候，可以通过看它给出的提示来解决问题 |
| 查看不同 | **git [diff](http://www.cnblogs.com/schaepher/p/5561193.html" \l "diff)** | 查看当前状态和最新的commit之间不同的地方 |
|  | **git diff [版本号1] [版本号2]** | 查看两个指定的版本之间不同的地方。这里的版本号指的是commit的hash值 |
| 添加文件 | **git [add -A](http://www.cnblogs.com/schaepher/p/5561193.html" \l "add)** | 这算是相当通用的了。在commit之前要先add |
| 撤回stage的东西 | **git [checkout -- .](http://www.cnblogs.com/schaepher/p/5561193.html" \l "checkout)** | 这里用小数点表示撤回所有修改，在--的前后都有空格 |
| 提交 | **git [commit -m "[提交信息]"](http://www.cnblogs.com/schaepher/p/5561193.html" \l "commit)** | 提交信息最好能体现更改了什么 |
| 删除未tracked | **git [clean -xf](http://www.cnblogs.com/schaepher/p/5561193.html" \l "clean)** | 删除当前目录下所有没有track过的文件。不管它是否是.gitignore文件里面指定的文件夹和文件 |
| 查看提交记录 | **git [log](http://www.cnblogs.com/schaepher/p/5561193.html" \l "log)** | 查看当前版本及之前的commit记录 |
|  | **git [reflog](http://www.cnblogs.com/schaepher/p/5561193.html" \l "reflog)** | HEAD的变更记录 |
| 版本回退 | **git [reset --hard [版本号]](http://www.cnblogs.com/schaepher/p/5561193.html" \l "reset)** | 回退到指定版本号的版本，该版本之后的修改都被删除。同时也是通过这个命令回到最新版本。需要reflog配合 |
| 设置用户名 | **git [config --global user.name "你的用户名"](http://www.cnblogs.com/schaepher/p/5561193.html" \l "config_user)** |  |
| 查看用户名 | **git config user.name** |  |
| 设置邮箱 | **git [config --global user.email "你的邮箱"](http://www.cnblogs.com/schaepher/p/5561193.html" \l "config_email)** |  |
| 查看邮箱 | **git config user.email** |  |
| 查看所有配置项 | **git config --list** |  |
| 生成ssh key | **[ssh-keygen -t rsa -C "你的邮箱"](http://www.cnblogs.com/schaepher/p/5561193.html" \l "ssh_key)** | 这条命令前面不用加git |
| 添加远程仓库 | **git [remote add origin 你复制的地址](http://www.cnblogs.com/schaepher/p/5561193.html" \l "remote_add)** | 设置origin |
| 上传并指定默认 | **git [push -u origin master](http://www.cnblogs.com/schaepher/p/5561193.html" \l "push_origin)** | 指定origin为默认主机，以后push默认上传到origin上 |
| 提交到远程仓库 | **git [push](http://www.cnblogs.com/schaepher/p/5561193.html" \l "push)** | 将当前分支增加的commit提交到远程仓库 |
| 从远程仓库同步 | **git pull** | 在本地版本低于远程仓库版本的时候，获取远程仓库的commit |

# windows下GIt Bash中文乱码解决办法

进入git安装目录，分别修改以下几个文件，然后关闭git bash窗口，重新开启一次即可

1. 编辑etc\gitconfig文件，在文件末尾增加以下内容：

[gui]

encoding = utf-8 #代码库统一使用utf-8

[i18n]

commitencoding = GB2312 #log编码,windows默认gb2312,声明后发到服务器才不会乱码

[svn]

pathnameencoding = GB2312 #支持中文路径

2. 编辑etc\git-completion.bash文件,在文件末尾增加以下内容，解决ls命令不能显示中文目录的问题：

alias ls='ls --show-control-chars --color=auto' #ls能够正常显示中文

3. 编辑etc\inputrc文件，修改output-meta和convert-meta属性值，解决git commit命令不能提交中文注释的问题：

set output-meta on #bash可以正常输入中文

set convert-meta off

4. 编辑profile文件，在文件末尾添加如下内容，解决git log不能显示中文注释的问题：

export LESSHARESET=utf-8

1. 如果出现[git status命令中文文件名编码问题](http://blog.csdn.net/mlq8087/article/details/52174834)，可以通过将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，如：

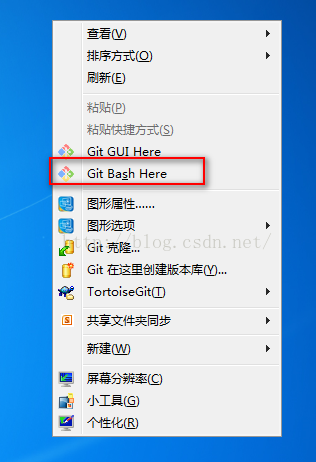
$ git config --global core.quotepath false

说明：以上是网上参考的，这里要修改的四个文件可能是根据操作系统的不同分布在不同的位置，但是一定是在git安装目录下。

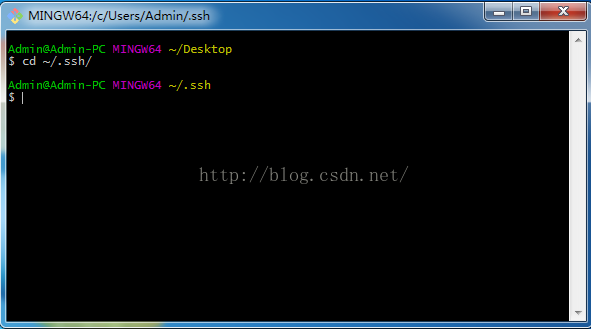
**[Windows 7下Git SSH 创建Key的步骤](http://blog.csdn.net/lsyz0021/article/details/52064829)**

1、首先你要安装[Git](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/lsyz0021/article/details/_blank)工具

2、右键鼠标，选中 “[git](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/lsyz0021/article/details/_blank) Bash here”，当然你也可以在windows的 “开始”--->“所以程序”，或者安装目录打开它



1. 输入指令，进入.ssh文件夹，命令：**cd ~/.ssh/**



如果提示 “ No such file or directory”，你可以手动的创建一个 .ssh文件夹即可

命令为：**mkdir ~/.ssh**

1. 配置全局的name和email，这里是的你github或者bitbucket的name和email

**git config --global user.name "xkwg"**

**git config --global user.email "xkwg@163.com"**

5、生成key

**ssh-keygen -t rsa -C "xkwg@163.com"**  
Your identification has been saved in /User/Admin/.ssh/id\_rsa.  
Your public key has been saved in /User/Admin/.ssh/id\_rsa.pub.  
The key fingerprint is:  
………………  
  
连续按三次回车，这里设置的密码就为空了，并且创建了key，最后得到了两个文件：id\_rsa和id\_rsa.pub

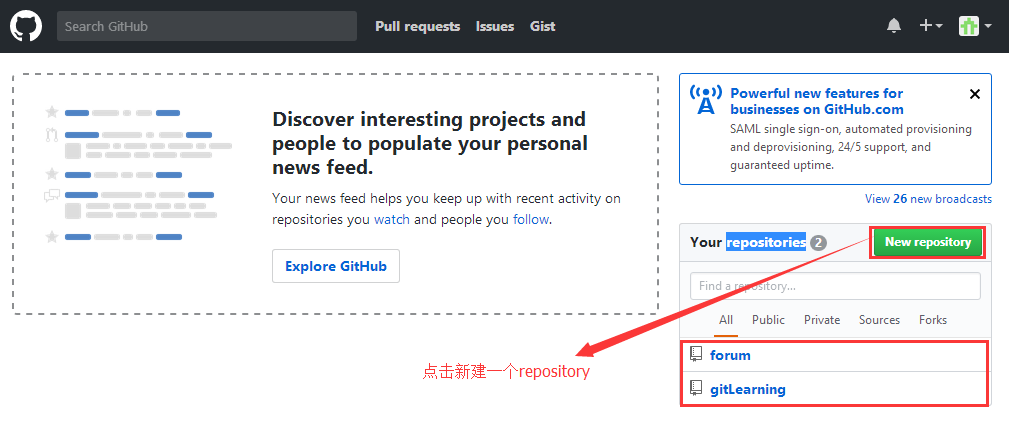
6、打开Admin目录进入.ssh文件夹，用记事本打开id\_rsa.pub，复制里面的内容添加到你github或者bitbucket ssh设置里即可

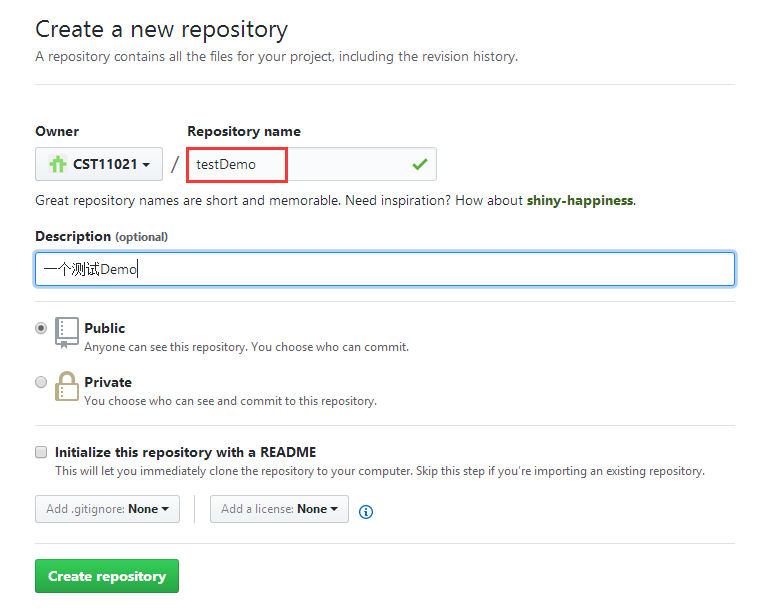


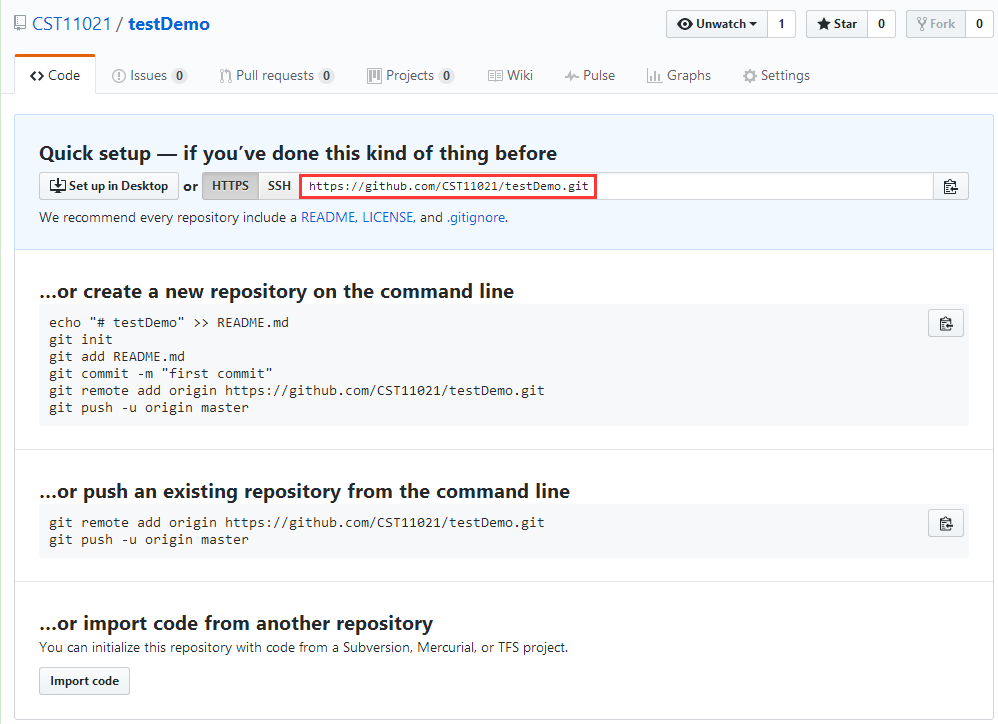
# 如何通过Git GUI将自己本地的项目上传至Github

## 在GitHub上创建仓库

首先要在GitHub上创建一个repository





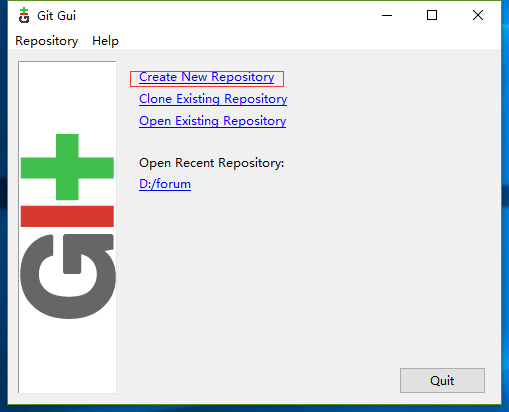


## 使用Git GUI上传

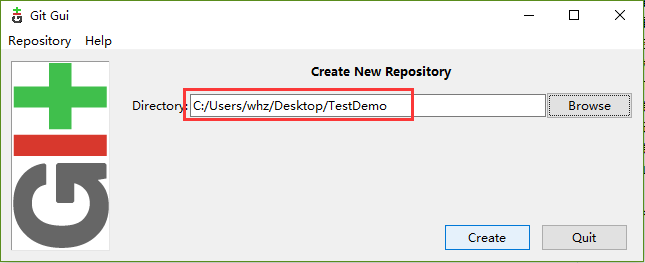
安装完**Git-2.10.0-64-bit.exe客户端后，右键选择Git GUI Here选项。**

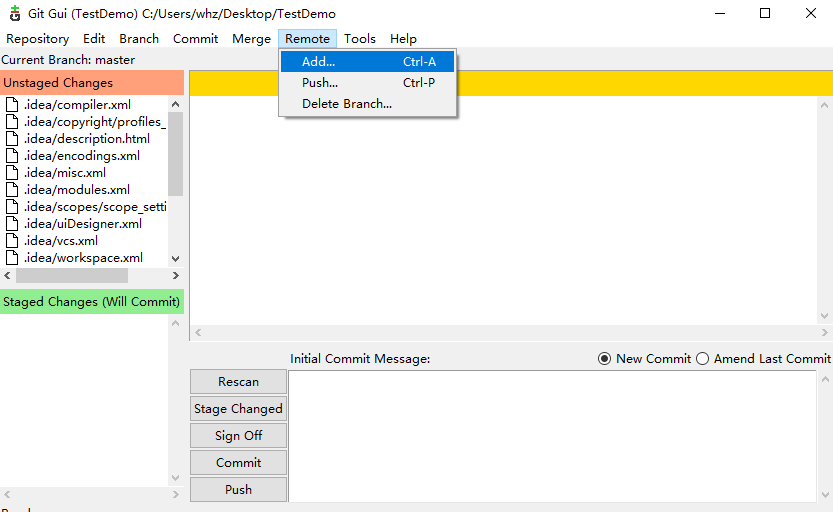


选择Create New Repository

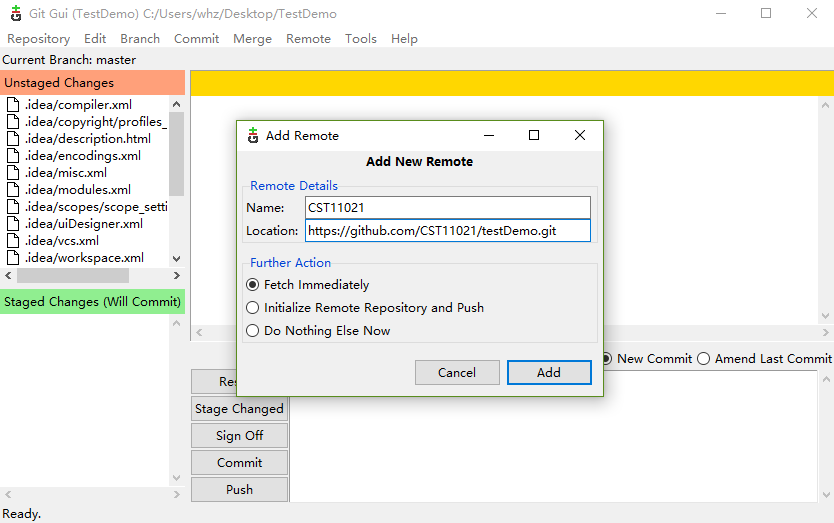


选择要上传的工程

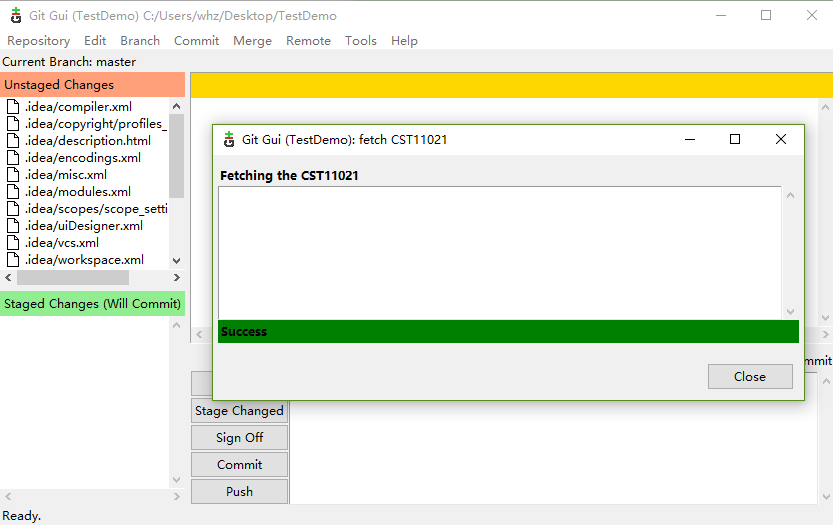


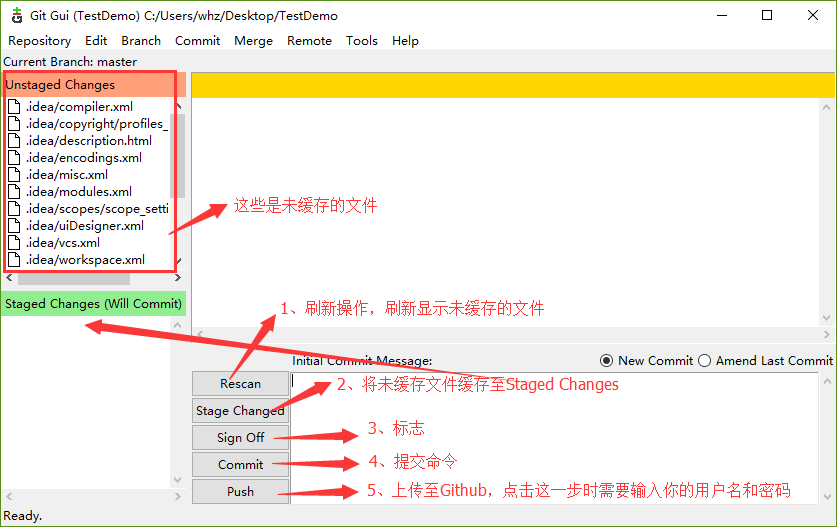


输入GitHub的用户名和仓库地址

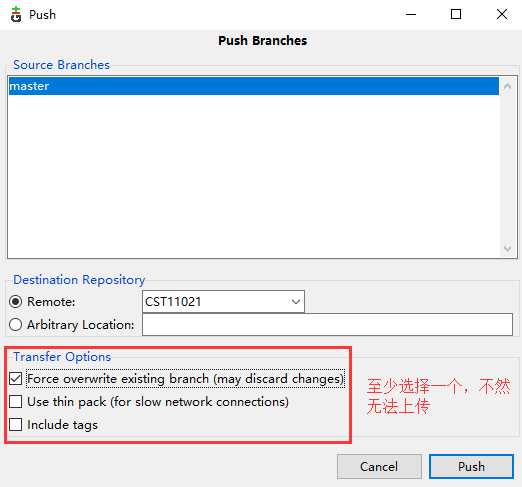


显示Success说明关联成功

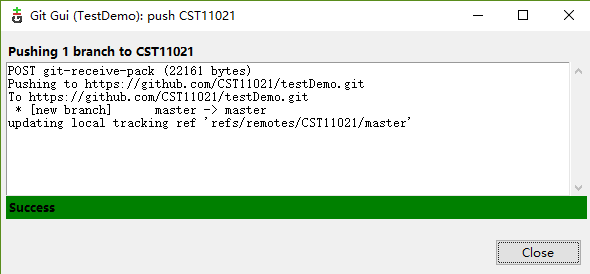




依次点击如上5个步骤，点击Push时，弹出如下窗口



首次上传需要输入用户名和密码，最后显示如下则上传成功



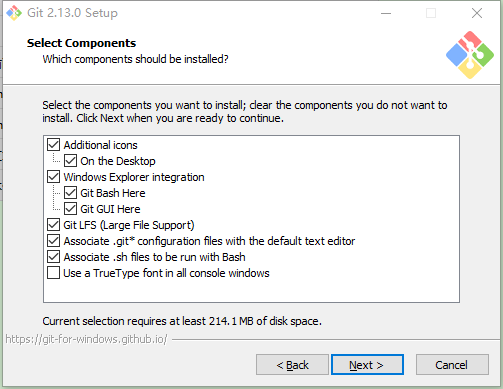
# git安装

On the Desktop：是否添加桌面图标

Git Bash Here：命令行模式

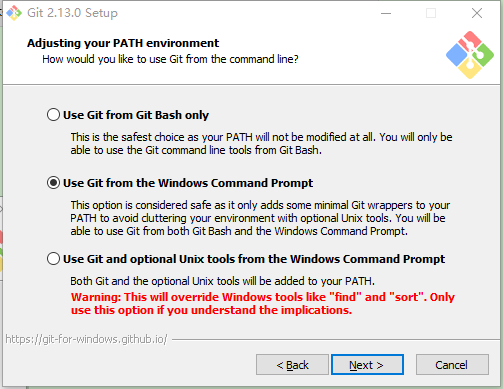
Git GUI Here：图形界面

use a truetype font in all console windows：在所有[控制台](https://www.baidu.com/s?wd=%E6%8E%A7%E5%88%B6%E5%8F%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YYnW6krHwhmWw9Pju9uWf40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1m1nWbsrHnLrHbsnj03rjDz" \t "https://zhidao.baidu.com/question/_blank)窗口使用[truetype字体](https://www.baidu.com/s?wd=truetype%E5%AD%97%E4%BD%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YYnW6krHwhmWw9Pju9uWf40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1m1nWbsrHnLrHbsnj03rjDz" \t "https://zhidao.baidu.com/question/_blank)

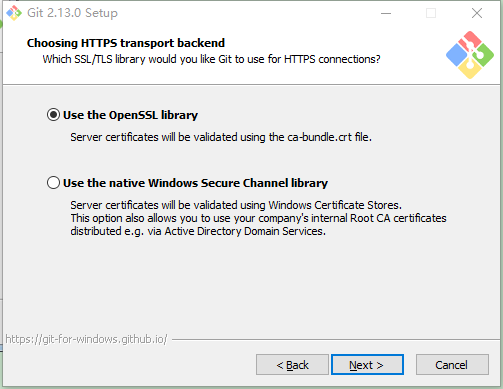


Use Git from Git Bash only，单独的命令行，而不是和Windows命令行混用

Use Git from the Windows Command Prompt， 可以在Windows的命令行cmd中git命令



在Choosing HTTPS transport backend 中选择默认

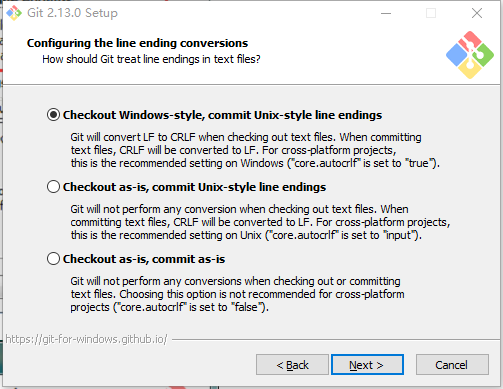


在“Configuring the line ending conversions”选项中：

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Configuring the terminal emulator to use with Git Bash中选择默认

