1、Git历史

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 2005 年的时候，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

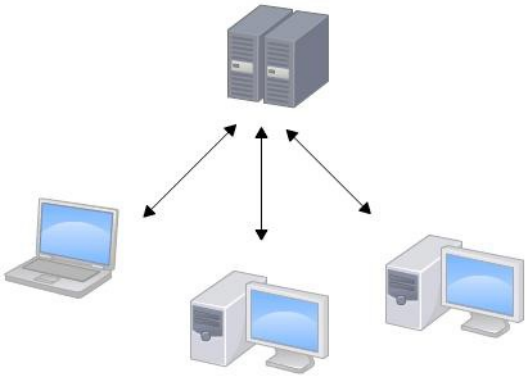
• 有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）

2、Git与svn对比

2.1、Svn

SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而干活的时候，用的都是自己的电脑，所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自己做完的活推送到中央服务器。集中式版本控制系统是必须联网才能工作，如果在局域网还可以，带宽够大，速度够快，如果在互联网下，如果网速慢的话，就郁闷了。

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：

• 服务器单点故障（其实也不是什么缺点，重启服务就好了）

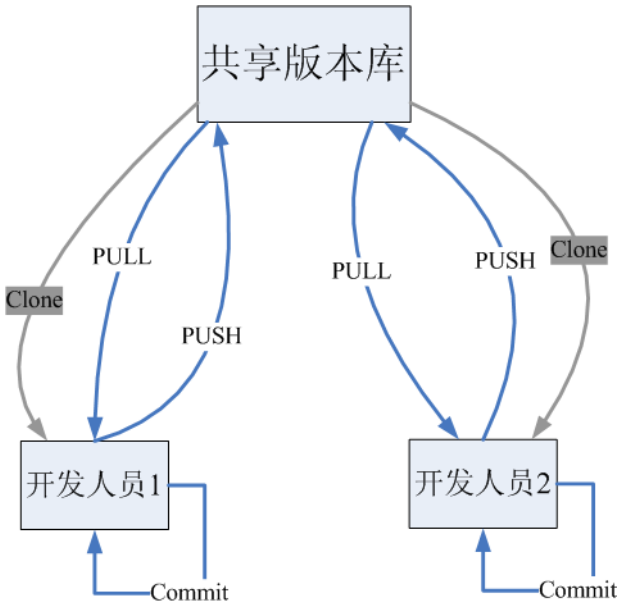
• 容错性差（没有多差）

SVN现在很多企业也在用。

2.2、Git

Git是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整的版本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。既然每个人的电脑都有一个完整的版本库，那多个人如何协作呢？比如说自己在电脑上改了文件A，其他人也在电脑上改了文件A，这时，你们两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。

下图就是分布式版本控制工具管理方式：



3、Git工作流程

一般工作流程如下：

1．从远程仓库中克隆 Git 资源作为本地仓库。

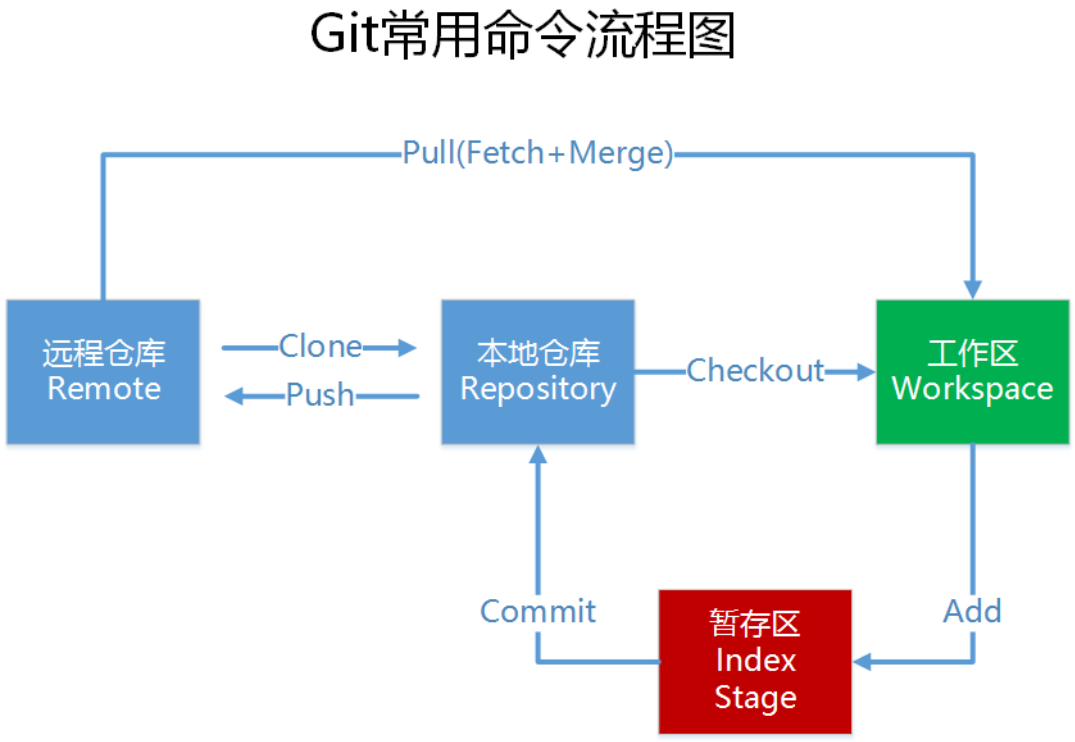
2．从本地仓库中checkout代码然后进行代码修改

3．在提交前先将代码提交到暂存区。

4．提交修改。提交到本地仓库。本地仓库中保存修改的各个历史版本。

5．在修改完成后，需要和团队成员共享代码时，可以将代码push到远程仓库。

下图展示了 Git 的工作流程：



4、安装Git

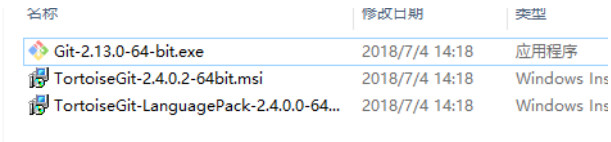
最早Git是在Linux上开发的，很长一段时间内，Git也只能在Linux和Unix系统上跑。不过，慢慢地有人把它移植到了Windows上。现在，Git可以在Linux、Unix、Mac和Windows这几大平台上正常运行了。由于开发机大多数情况都是windows，所以我只讲解windows下的git的安装及使用，以后随着公司业务水平上升，我们会使用Linux当作服务器。

4.1、下载

下载地址：https://git-scm.com/download

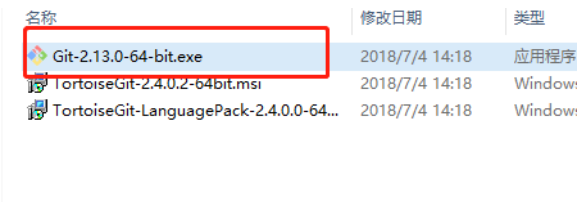
（我们不用这个版本的git，用2.13.0）

参考我发的资料中安装包已经下载完毕，根据不同的操作系统选择对应的安装包。

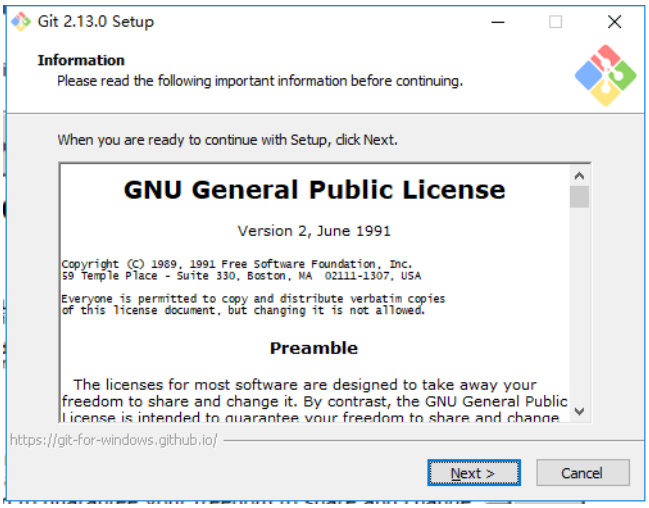


4.2、安装

4.2.1、安装Git for Windows

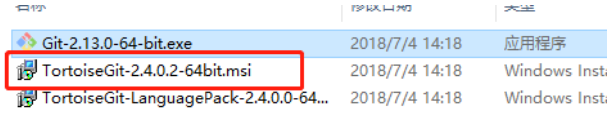


双击运行，安装程序



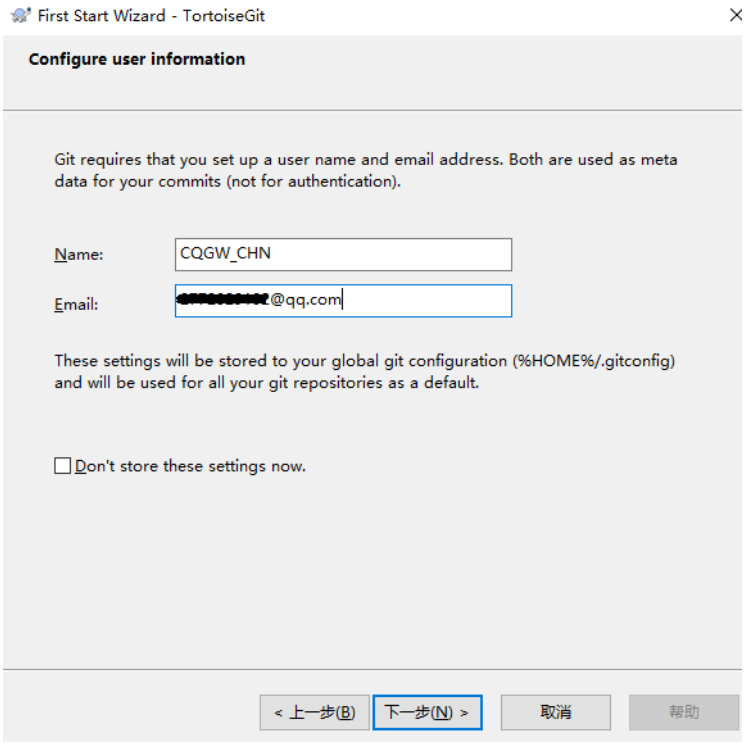
使用默认设置即可（一直点击“下一步”，直到完成安装）

4.2.2安装TrotoiseGit



双击运行，点击下一步，使用默认设置即可。

配置开发者姓名及邮箱，每次提交代码时都会把此信息包含到提交的信息中。 命名规则CQGW_自己名字的首字母大写。

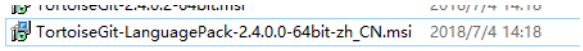


使用默认配置，点击“完成”按钮完成配置。

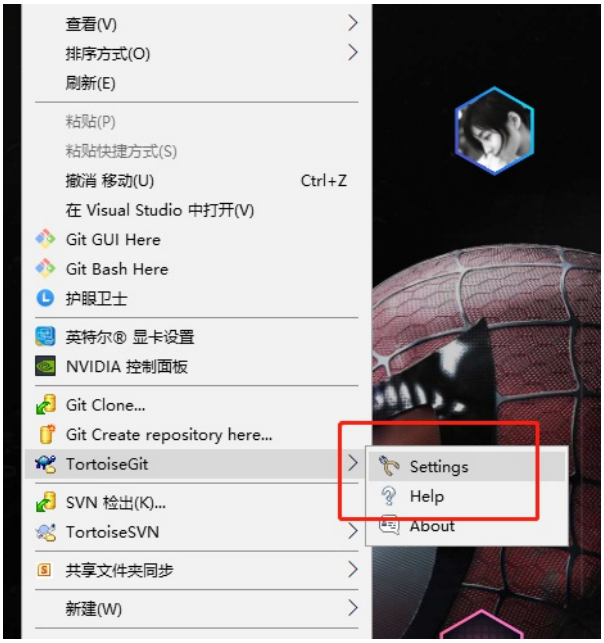
完整完毕后在系统右键菜单中会出现git的菜单项。

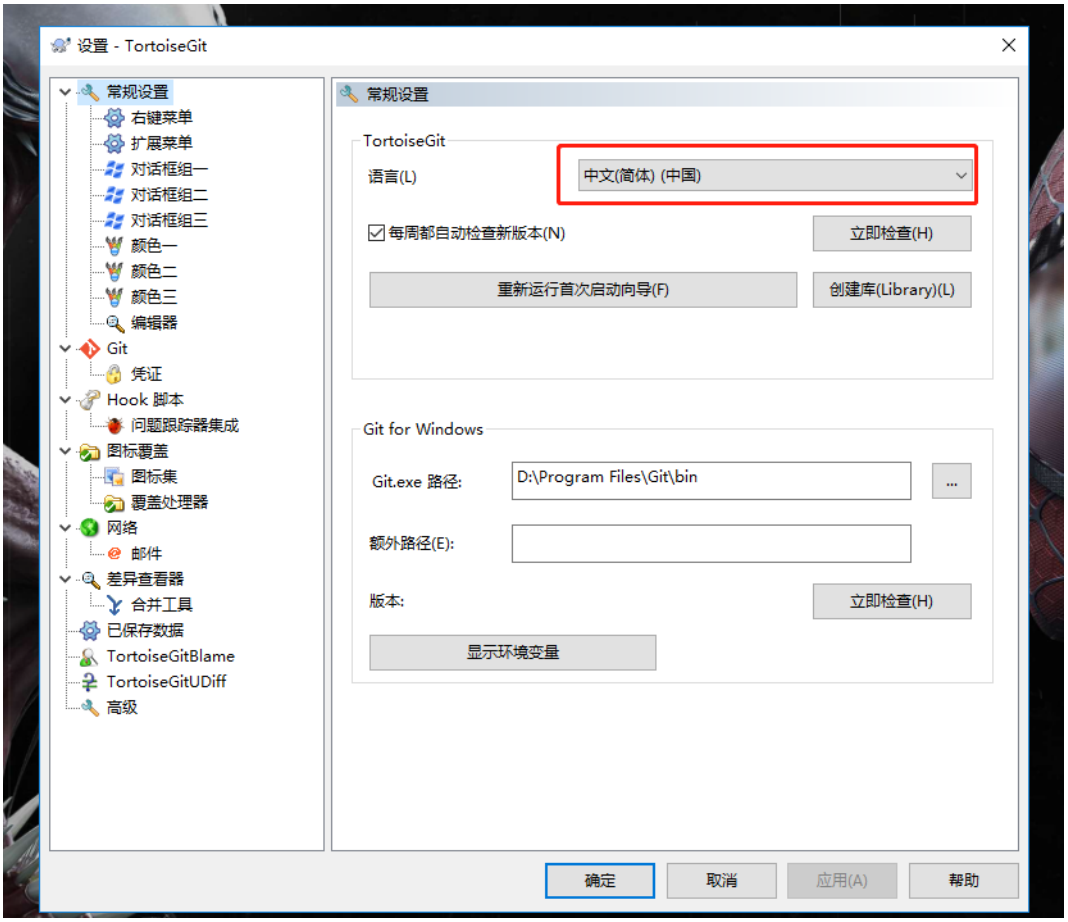
4.2.3安装中文语言包

安装中文语言包并不是必选项。可以根据个人情况来选择安装。



双击运行，保持默认配置。语言包安装完毕后可以在TortoiseGit的设置中调整语言





5、使用Git管理文件版本

5.1、创建版本库

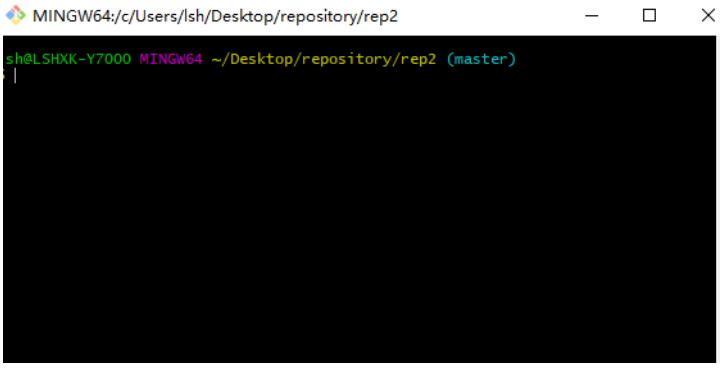
什么是版本库呢？版本库又名仓库，英文名repository，你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。由于git是分布式版本管理工具，所以git在不需要联网的情况下也

具有完整的版本管理能力。

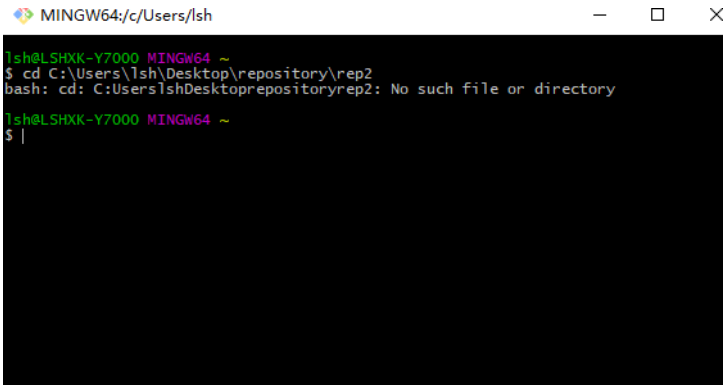
创建一个版本库非常简单，可以使用git bash也可以使用tortoiseGit。首先，选择一个合适的地方，创建一个空目录，如：（C:\Users\lsh\Desktop\repository\rep2）。

5.1.1使用GitBash

在当前目录中点击右键中选择Git Bash来启动。



或者在开始菜单中启动。注意如果是从开始菜单启动的gitbash需要切换目录到仓库所在的目录。



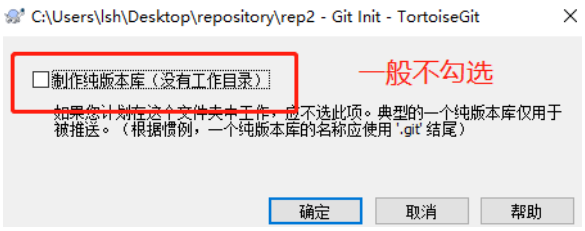
创建仓库执行命令：

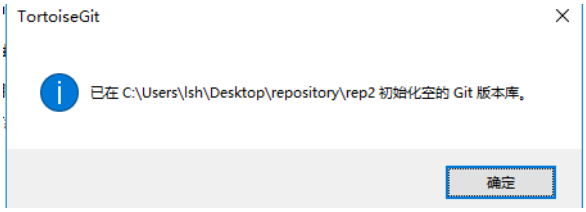
$ git init

就会出现.git的文件

5.1.2、使用TrotoiseGit

使用TortoiseGit时只需要在目录中点击右键菜单选择“在这里创建版本库”





版本库创建成功，会在此目录下创建一个.git的隐藏目录，如下所示：

概念：

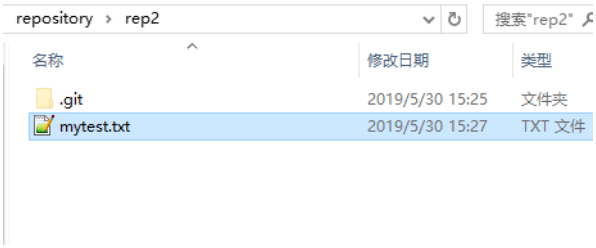
版本库：“.git”目录就是版本库，将来文件都需要保存到版本库中。

工作目录：包含“.git”目录的目录，也就是.git目录的上一级目录就是工作目录。只有工作目录中的文件才能保存到版本库中。

5.2、添加文件

5.2.1、添加文件过程

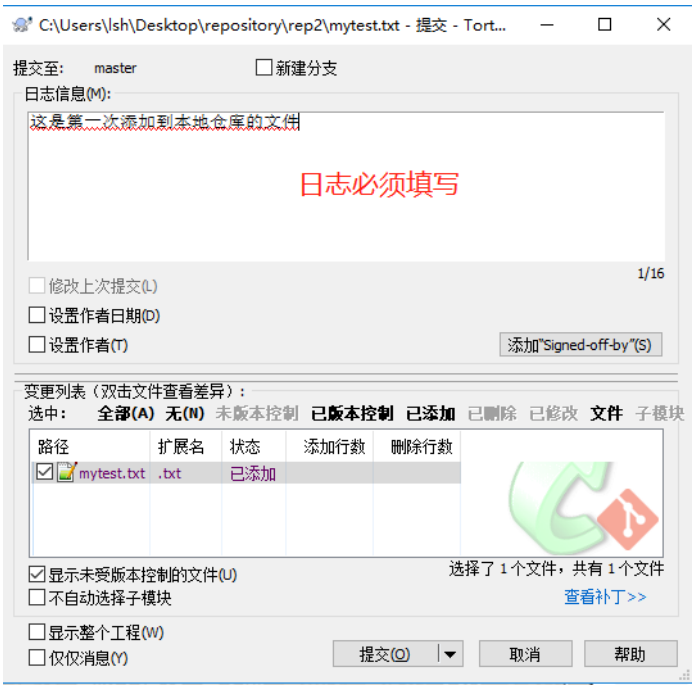
在C:\Users\lsh\Desktop\repository\rep2目录下创建一个mytest.txt文件

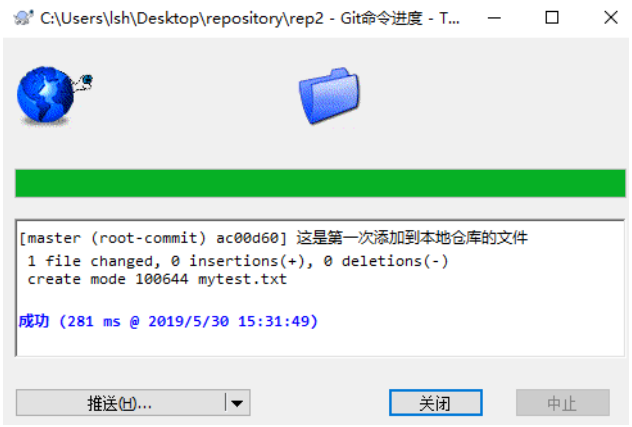


把mytest.txt添加到暂存区

添加后文本文件变为带“+”号的图标：

提交文件：在mytest.txt上再次点击右键选择“提交”，此时将文件保存至版本库中。





5.2.2、工作区和暂存区

Git和其他版本控制系统如SVN的一个不同之处就是有暂存区的概念。

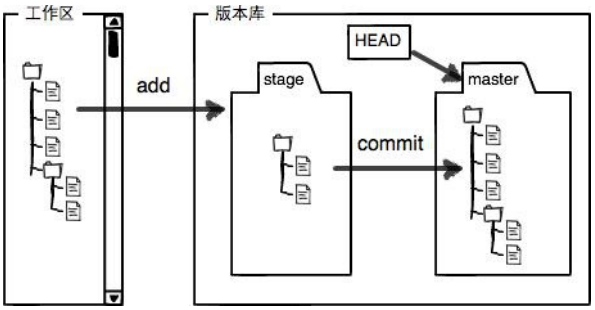
什么是工作区（Working Directory）？

工作区就是你在电脑里能看到的目录，比如我的reporstory文件夹就是一个工作区。

repository不是版本库，是工作区了，repository目录是工作区，在这个目录中的“.git”隐藏文件夹才是版本库。

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。

如下图所示：



分支和HEAD的概念我们稍后细说。前面讲了我们把文件往Git版本库里添加的时候，是分两步执行的：

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。

因为我们创建Git版本库时，Git自动为我们创建了唯一一个master分支，所以，现在，git commit就是往master分支上提交更改。

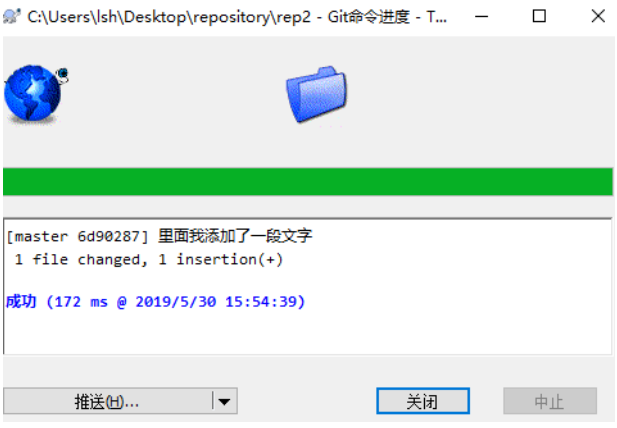
你可以简单理解为，需要提交的文件修改通通放到暂存区，然后，一次性提交暂存区的所有修改。

5.3、修改文件

5.3.1、提交修改

被版本库管理的文件不可避免的要发生修改，此时只需要直接对文件修改即可。修改完毕后需要将文件的修改提交到版本库。

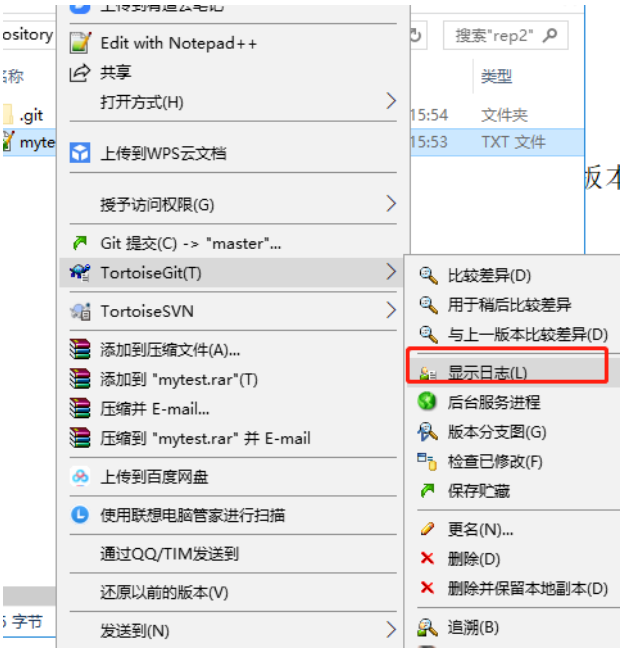
在mytest.txt里面加入一些文字，再在mytest.txt文件上点击右键，然后选择“提交”



5.3.2、查看修改历史

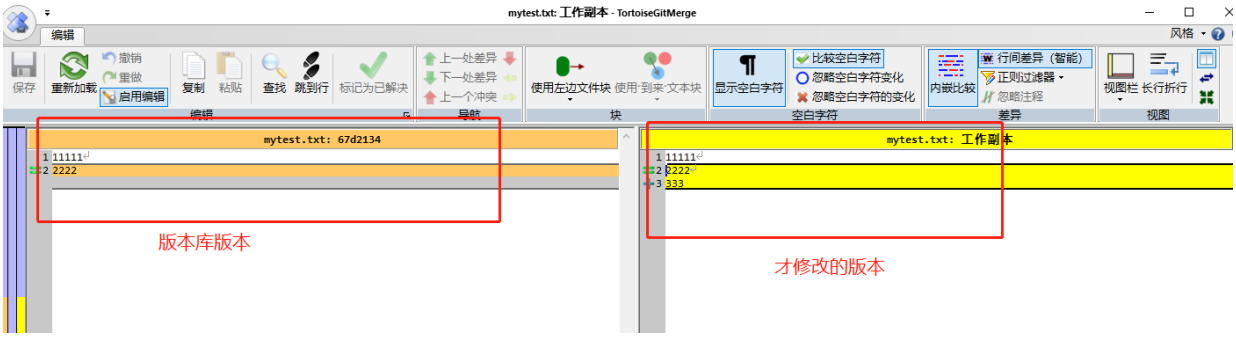
在开发过程中可能会经常查看代码的修改历史，或者叫做修改日志。来查看某个版本是谁修改的，什么时间修改的，修改了哪些内容。

可以在文件上点击右键选择“显示日志”来查看文件的修改历史。



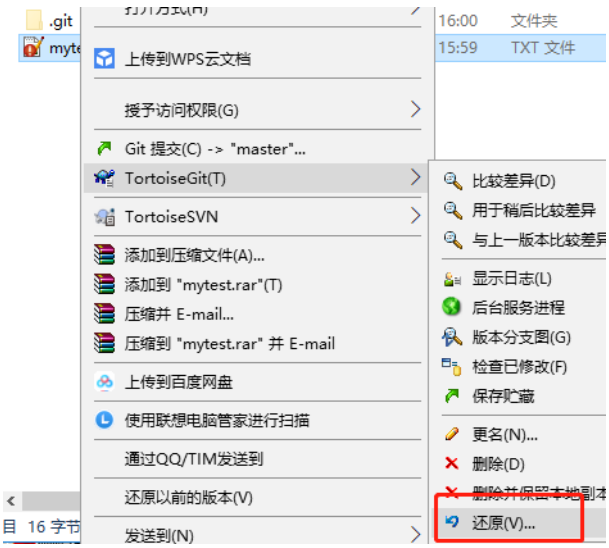
5.3.3、差异比较

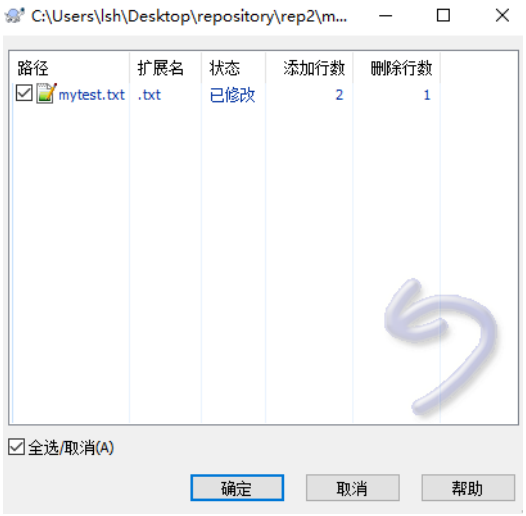
当文件内容修改后，需要和修改之前对比一下修改了哪些内容此时可以使用“比较差异功能”

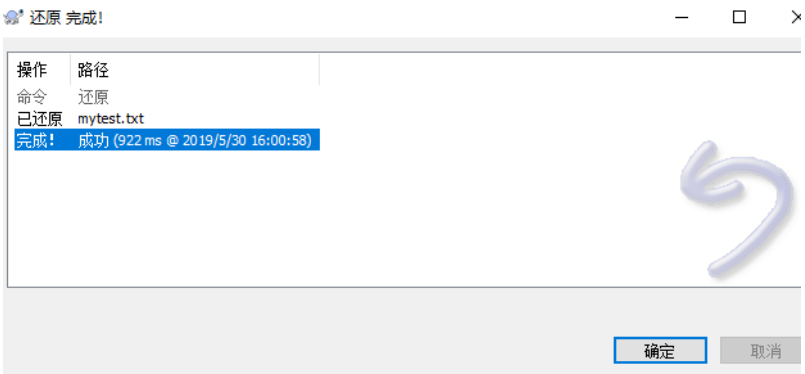


5.3.4、还原修改

当文件修改后不想把修改的内容提交，还想还原到未修改之前的状态。此时可以使用“还原”功能



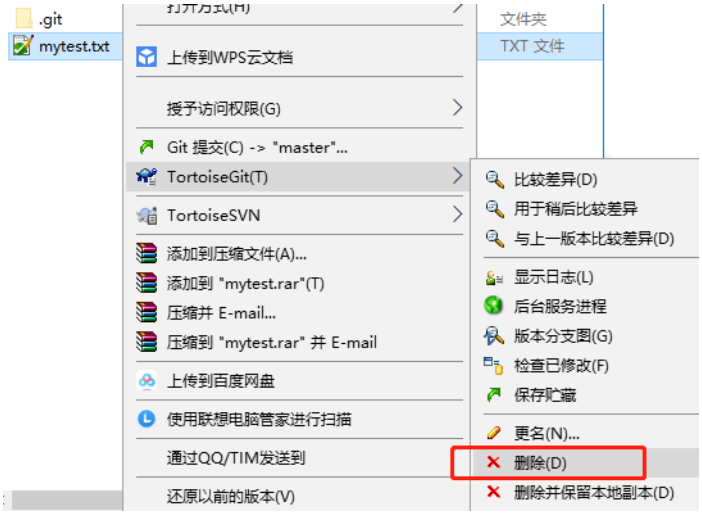




注意：此操作会撤销所有未提交的修改，所以当做还原操作是需要慎重慎重！！！

5.4、删除文件

需要删除无用的文件时可以使用git提供的删除功能直接将文件从版本库中删除。



这样删除在工作空间删除，版本库中是不会被删除的，除非你提交了。如过只想删除版本库中，本地工作空间想要保留，选着删除并保留本地副本，并且提交。

6、远程仓库

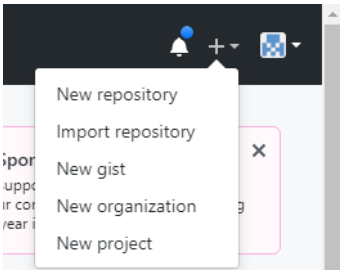
6.1、添加远程仓库

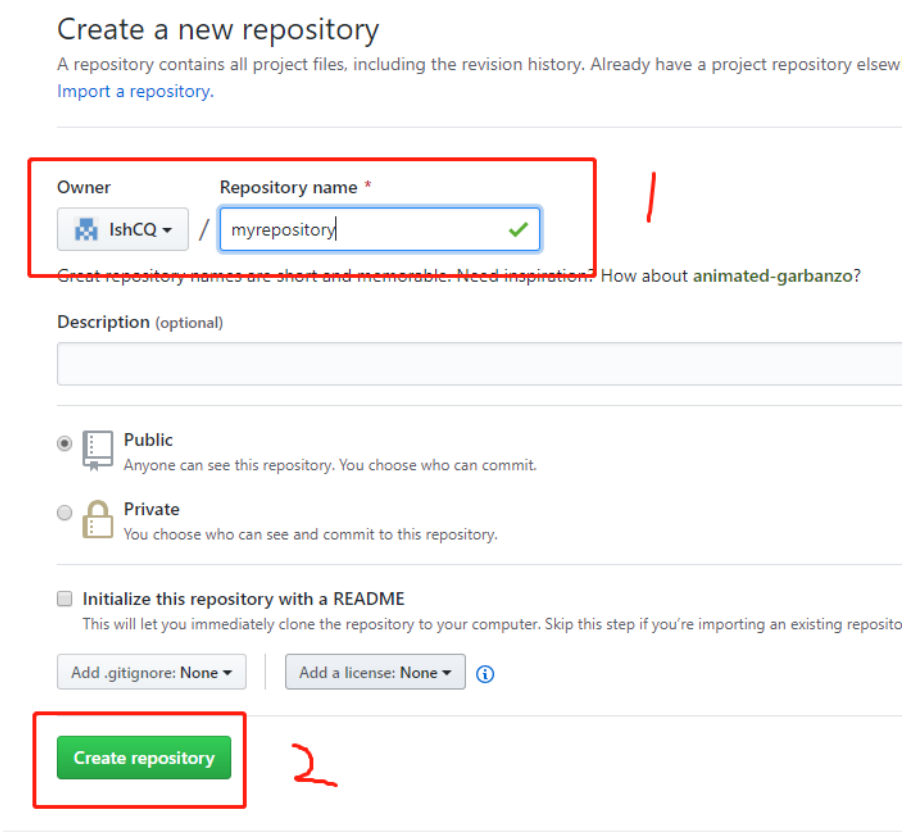
现在我们已经在本地创建了一个Git仓库，又想让其他人来协作开发，此时就可以把本地仓库同步到远程仓库，同时还增加了本地仓库的一个备份。

常用的远程仓库就是github：https://github.com/，接下来我们来看看如何将本地代码同步到github。

6.1.1、在github上创建仓库

首先你得在github上创建一个账号，这个就不演示了。然后在github上创建一个仓库：



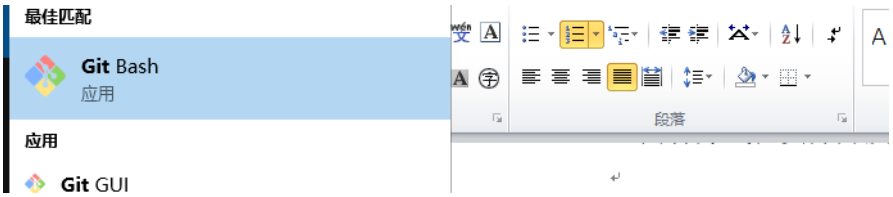


点击“create repository”按钮仓库就创建成功了。

Github支持两种同步方式“https”和“ssh”。如果使用https很简单基本不需要配置就可以使用，但是每次提交代码和下载代码时都需要输入用户名和密码。如果使用ssh方式就需要客户端先生成一个密钥对，即一个公钥一个私钥。然后还需要把公钥放到githib的服务器上。这两种方式在实际开发中都需要应用。接下来我们先看ssh方式。

6.1.1.1、ssh密钥生成

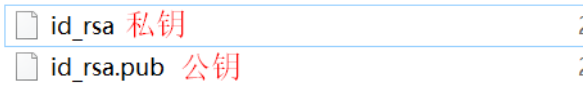
在windows下我们可以使用 Git Bash.exe来生成密钥，可以通过开始菜单或者右键菜单打开Git Bash



git bash 执行命令,生命公钥和私钥

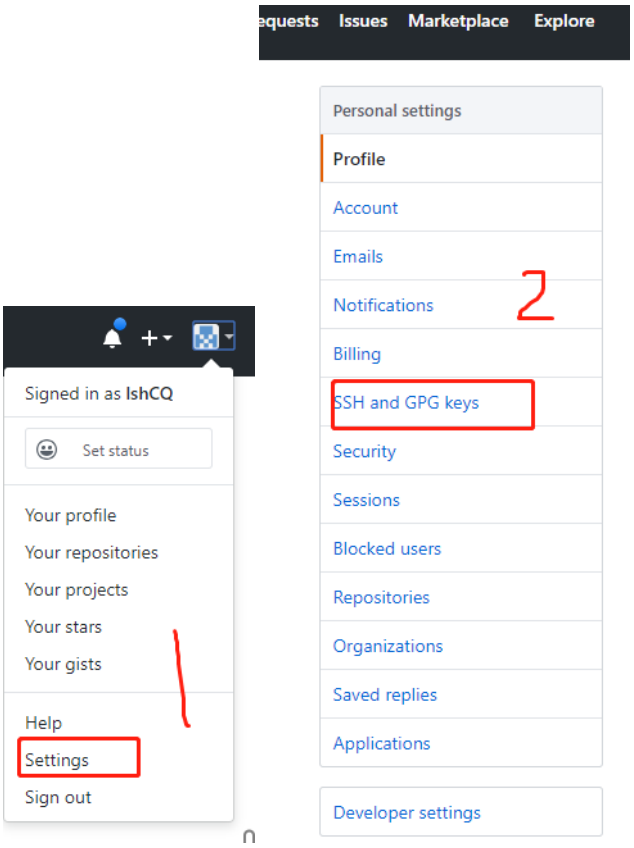
命令: ssh-keygen -t rsa

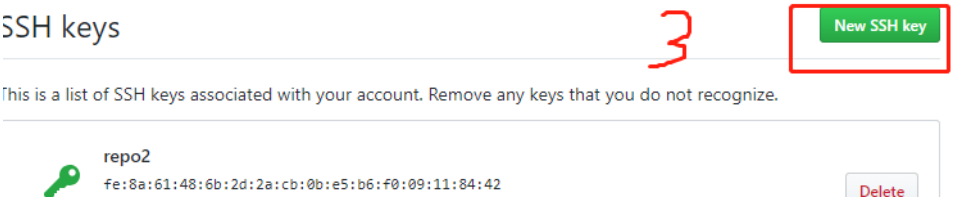
执行命令完成后,在window本地用户.ssh目录C:\Users\用户名.ssh下面生成如下名称的公钥和私钥:



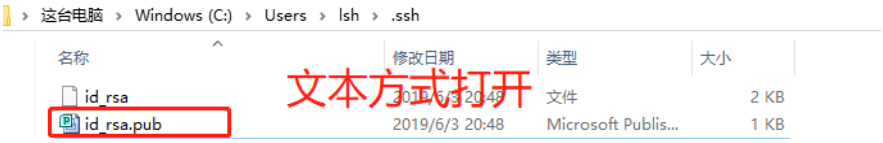
6.1.1.2、ssh密钥配置

密钥生成后需要在github上配置密钥本地才可以顺利访问。

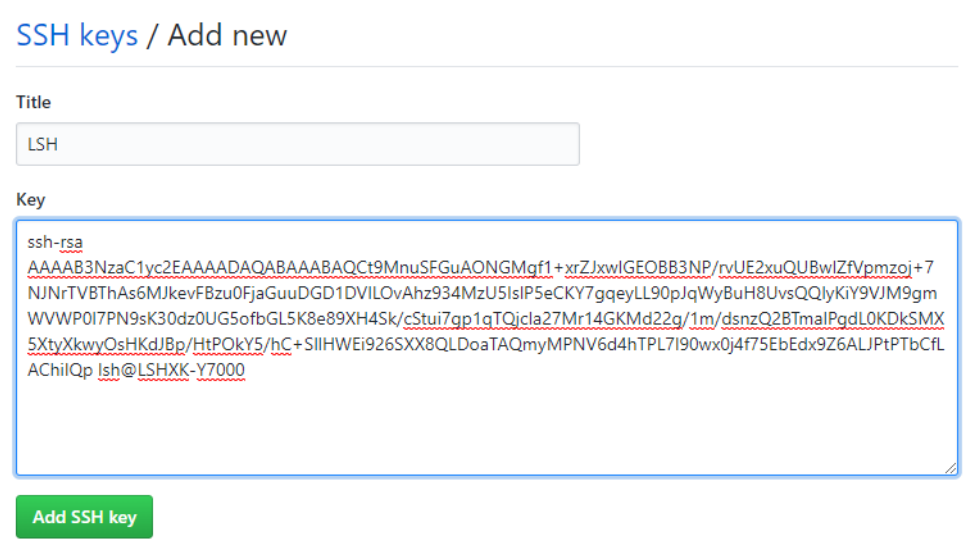




然后找到公钥



复制后粘贴到



6.1.2、同步到远程仓库

同步到远程仓库可以使用git bash也可以使用tortoiseGit

6.1.2.1、使用git bash

在仓库所在的目录点击右键选择“Git Bash Here”，启动git bash程序。

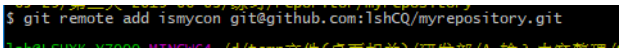
然后在git bash中执行如下语句：

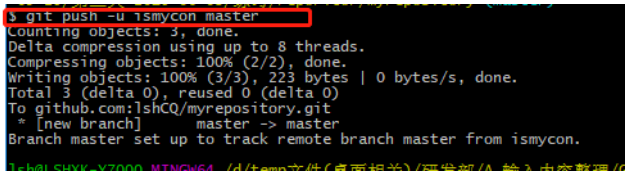
git remote add origin git@github.com:**lshCQ**/myrepository.git

git push -u origin master

注意：其中加粗字体部分需要替换成个人github的用户名。

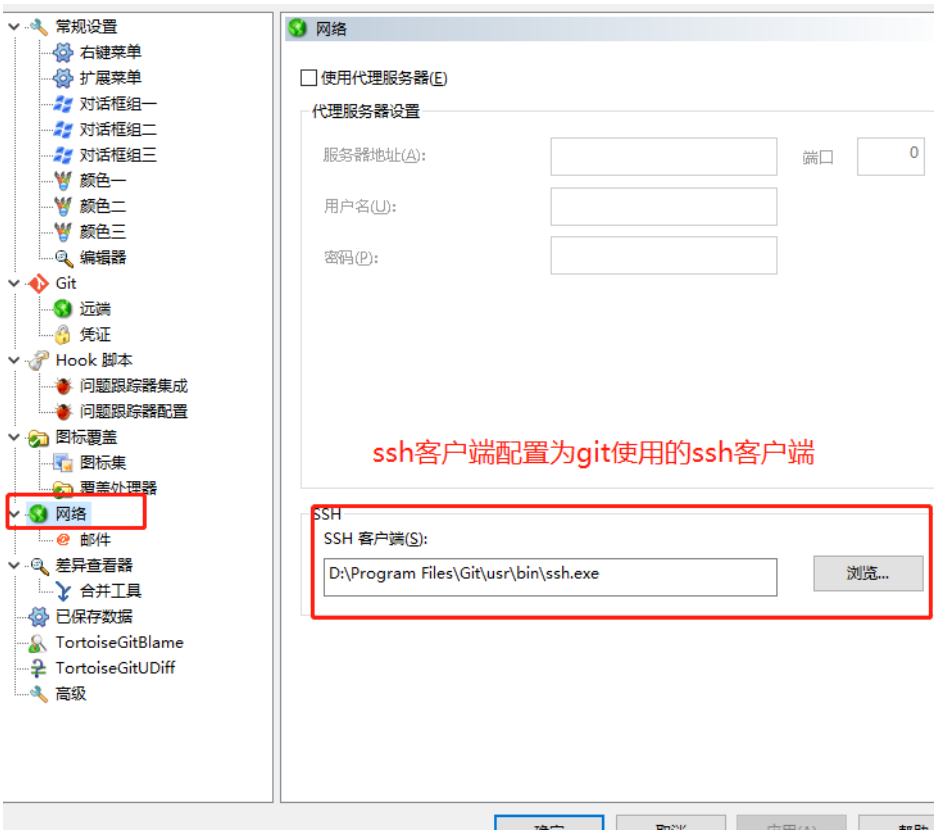
建立链接和上传

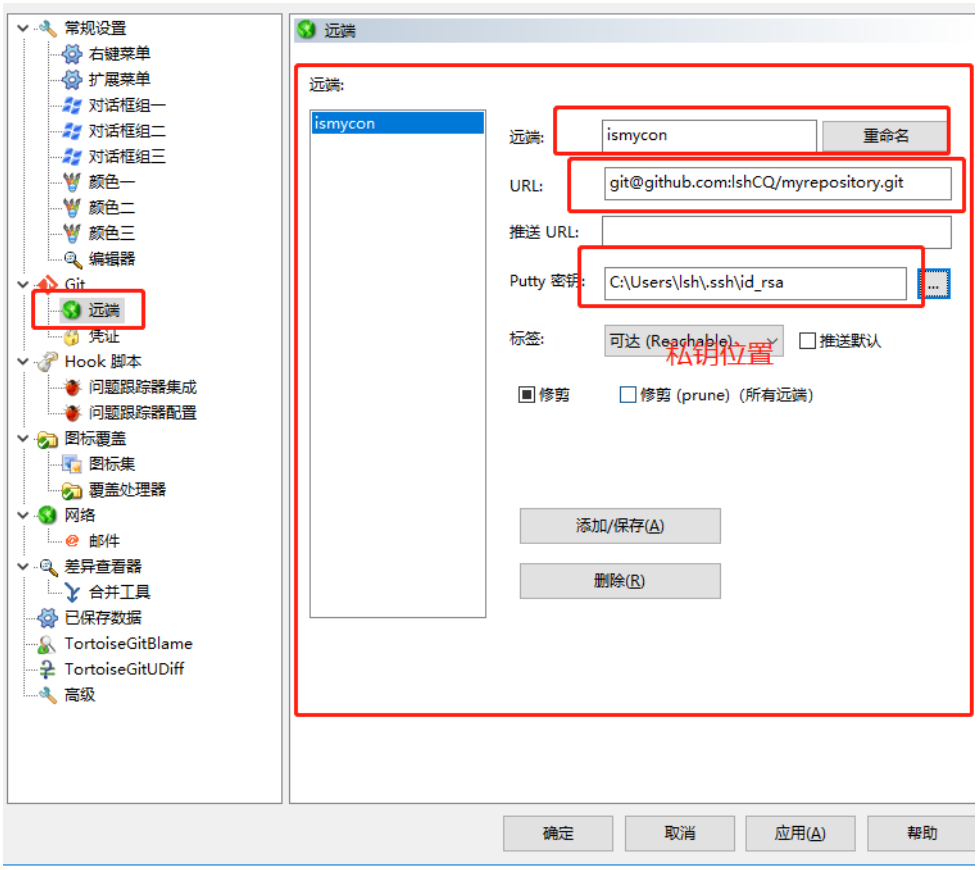




6.1.2.2、使用TortoiseGit同步

一、由于TortoiseGit使用的ssh工具是“PuTTY”git Bash使用的ssh工具是“openSSH”，如果想让TortoiseGit也使用刚才生成的密钥可以做如下配置：



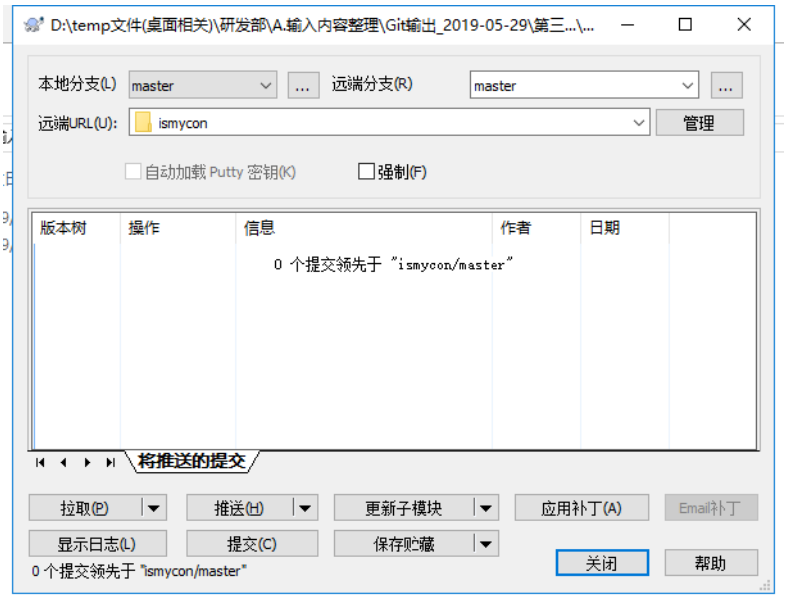


Url：远程仓库的地址

推送URL：也是相同的（可以不填写）

Putty密钥：选择刚才生成的密钥中的私钥

二、同步。在本地仓库的文件夹中单击右键，选择“Git同步”



点击推送即可完成。

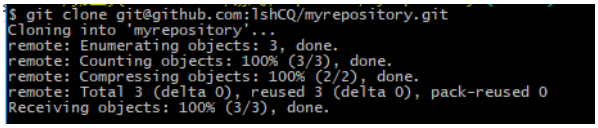
6.2、从远程仓库克隆

克隆远程仓库也就是从远程把仓库复制一份到本地，克隆后会创建一个新的本地仓库。选择一个任意部署仓库的目录，然后克隆远程仓库。

6.2.1、使用git bash

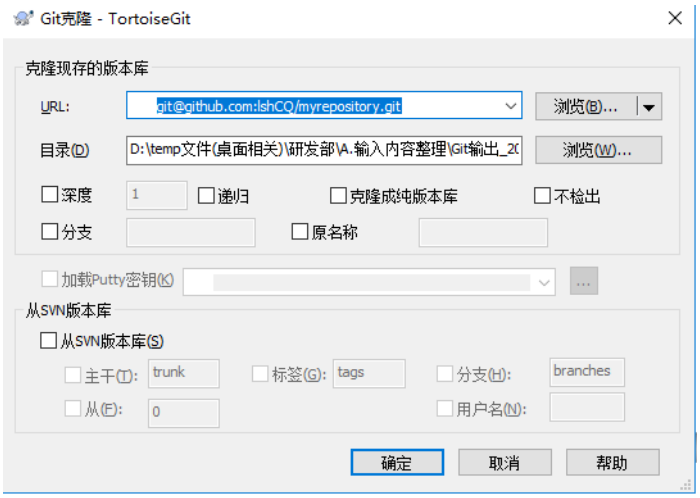
$ git clone git@github.com:**IshCQ**/myrepository.git

注意：其中加粗字体部分需要替换成个人github的用户名。



6.2.2、使用TrotoiseGit

在任意目录点击右键：



点击确定即可

6.3、从远程仓库获取代码

Git中从远程的分支获取最新的版本到本地有这样2个命令：

1.git fetch：相当于是从远程获取最新版本到本地，不会自动merge（合并代码）

2.git pull：相当于是从远程获取最新版本并merge到本地

上述命令其实相当于git fetch 和 git merge，在实际使用中，git fetch更安全一些。因为在merge前，我们可以查看更新情况，然后再决定是否合并。

如果使用TortoiseGit的话可以从右键菜单中点击“拉取”（pull）或者“获取”（fetch）

然后输入需要拉取的远程仓库的地址