# 

//在windows 命令中我尝试过,如果上一层用../是不起动作的,而用..\其动作的,而且在后面的路径也用\,入..\..\game\resource

Cd D: 到D盘 cd FireEngineer d盘文件夹FireEngineer

Ssh

$ cd ~/.ssh 检查有没有没有则

在“用户目录”下，右击选择“Git Bash hear”,效果见下图：

输入以下命令并执行，然后一路按“回车”即可，效果见图：

使用ssh-keygen命令生成ssh 密钥可以使用以下命令：

ssh-keygen -t rsa

也可以使用以下命令在生成时添加注释：

ssh-keygen -t rsa -**C** "我的SSH密钥"

需要注意的是ssh-keygen是相连的不能空格

小功能记录一下

git diff HEAD – redme.txt //查看文件的差别



gitbash命令的简单使用--Git Bash简单使用

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2016-03-15 14:14:52)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

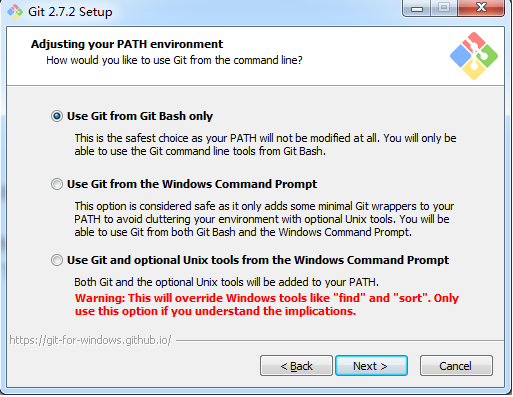
|  |  |
| --- | --- |
| 标签：  [git-bash](http://search.sina.com.cn/?c=blog&q=git-bash&by=tag" \t "_blank)    [git](http://search.sina.com.cn/?c=blog&q=git&by=tag" \t "_blank) |  |

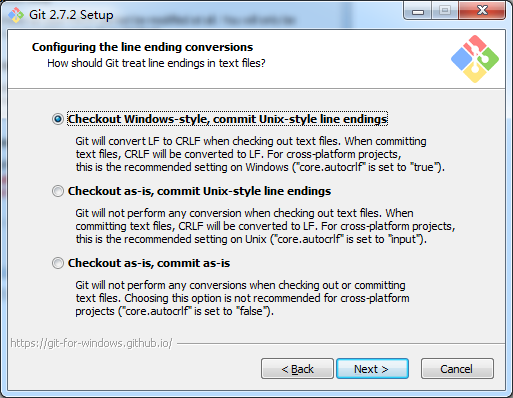
现简述一下git的安装和使用。

**安装：**

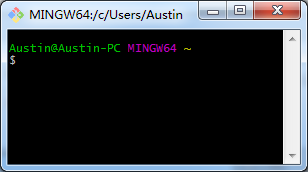
有关git于svn区别及其优势不再赘述，现简介其使用方法，因为我的电脑是win7系统，所以需要到官网（https://git-for-windows.github.io/）下载msysgit。(图形界面的话还需安装TortoiseGit)

下载完成，双击安装，一路选择默认设置即可。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706FOxfcLcb)

（选择使用git的命令行模式，选择默认git Bash模式即可）  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706FOURf036)  
（选择换行格式，默认选第一个windows下使用的跨平台样式即可）。

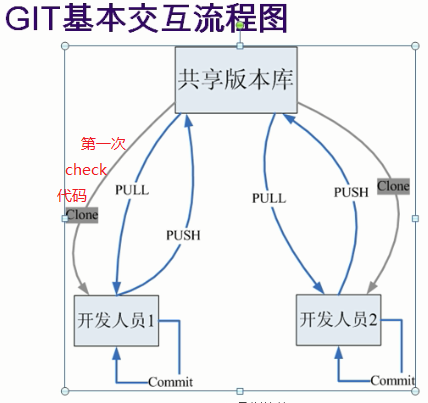
安装完成后，找到刚刚安装的git ，其中有一个git Bash，打开它即可使用了。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706G5fsKCd2)

其中使用的命令和Linux的常用命令一样，比如ls、pwd、cd等。

--------------------------------------------------------------------------------------------

**使用：**

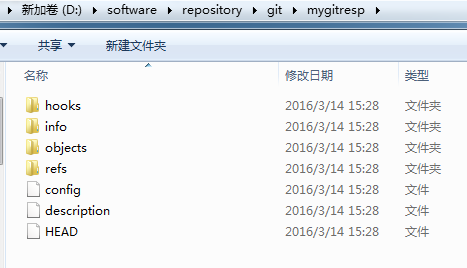
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706J4VjGbdb)

好了，git已经安装完毕，现在打算在D盘software/repository/git/目录下创建一个git仓库，名字是mygitresp

使用命令git init --bare <仓库名>

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQzy706HUyJLRdb)

然后在d/software/respository/git目录下就看到了刚刚创建的仓库。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQzy706Izwq8m9e)

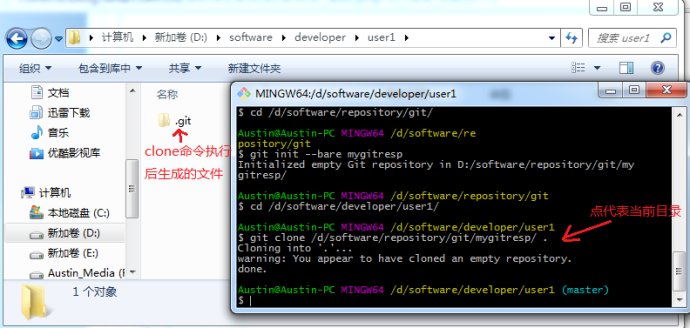
目录简单说明：

1. hooks:这个目录存放一些shell脚本，可以设置特定的git命令后触发相应的脚本；在搭建gitweb系统或其他git托管系统会经常用到hook script
2. info:包含仓库的一些信息
3. objects:所有的Git对象都会存放在这个目录中，对象的SHA1哈希值的前两位是文件夹名称，后38位作为对象文件名
4. refs:这个目录一般包括三个子文件夹，heads、remotes和tags，heads中的文件标识了项目中的各个分支指向的当前commit
5. config:这个是GIt仓库的配置文件
6. description:仓库的描述信息，主要给gitweb等git托管系统使用
7. HEAD:这个文件包含了一个档期分支（branch）的引用，通过这个文件Git可以得到下一次commit的parent

新用户-->代码clone

    我们创建一个新的目录d/software/developer/user1来模拟用户1 第一次从仓库check代码操作。

（*主要注意的是第一次需要使用clone命令将整个项目down到本地（类似svn的check），不是首次的话就可以直接使用git pull来同步更新代码了。*）  
命令git clone <仓库地址或目录> /d/software/respository/<仓库名> <用户目录>

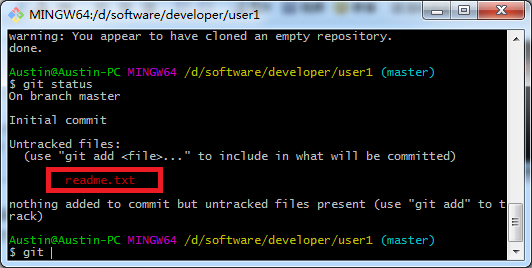
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706JPIJpQa6)

新建文件，上传代码：

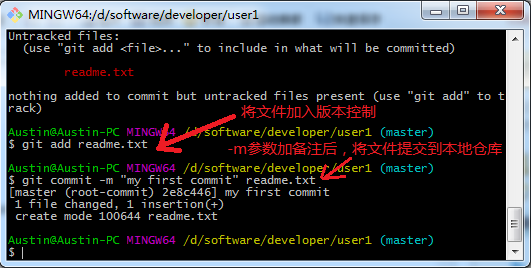
在user1目录新建文本文件 readme.txt，打开写入一测试语句“user1 create file.”后保存；然后在控制台中键入

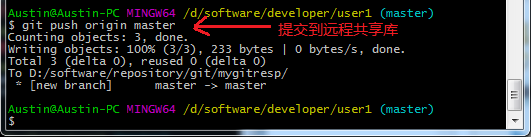
命令 git status

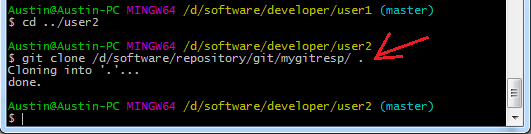
告知有未提交的文件 readme.txt需要提交。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706PCWXMm52)

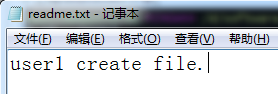
现在将其加入到git并做本地提交和共享库提交。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706Qd1x0t69)

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQzy706QDuZFV9b)  
  
好了，至此，我们已经将readme.txt文件提交到了远程共享仓库，我们切换到一个新的目录(d/software/developer/user2 相当于新用户)来将user1提交的代码更新到本地, 同样，首次需要使用clone将整个工程项目down下来。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706RrEooUb8)

OK, 打开user2目录，你会发现user1提交的readme.txt成功获得，打开后发现user1输入的文本“user1 create file.”显示OK。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy706S3HuMnc7)

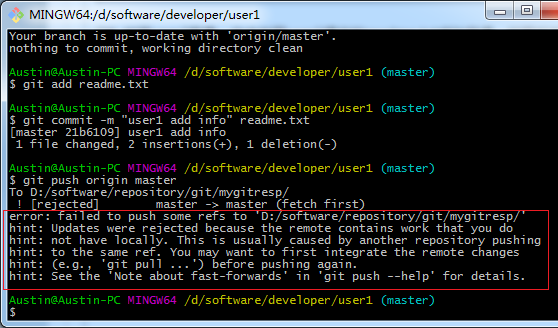
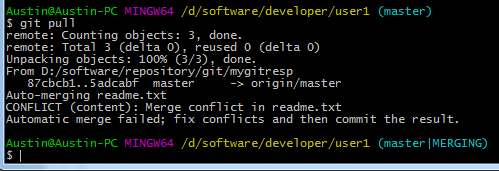
---------------------------------------------------------------------------------------------

**冲突解决：**

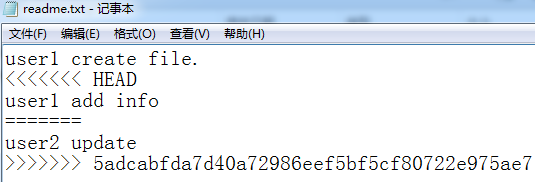
现在模拟一种冲突现象，user2在readme.txt文件第二行加入自己的一句话“user2 update"后保存，

然后在控制台按照以上命令提交代码。

然后user1在没有更新代码的前提下在第二行写入了“user1 add info",那么当user1提交到远程时，将报错，说有冲突。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy708fOkLV32d)  
接下来根据上图提示我们先使用命令 git pull更新一下  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy708g4lSmxe5)

提示自动合并文件失败，此时user1下的readme.txt内容已经被打上冲突标记了：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQgy708gdBSkk6c)  
我们先不用管它，继续回到user1的控制台，执行

命令 git mergetool后回车，要求再键入一个冲突解决工具的名称，直接回车则打开默认的vimdiff工具(git当然也可以自己配置其他如beyond compare等工具)

代码修改无误确认冲突解决后，退出编辑器，这时候再执行

命令 git commit -a 填写备注后将最终结果提交到本地git库，然后

命令 git push origin master 来提交到远程库。这里可以直接将目录下自动生成的.orig文件删除了

OK。

----------------------------------------------------------------------------------------------

**总结：**

**git常用命令：**

1.创建git资源库

    git init --bare 库名称

2.在用户文件夹下把资源clone下来

    git clone <仓库地址或目录> /d/software/respository/<仓库名> <用户目录>

3.创建一个文件，纳入到版本控制中。

    git add <文件名>

4.提交到本地版本库中。

 git commit <文件名>

5 推送到远程共享库中

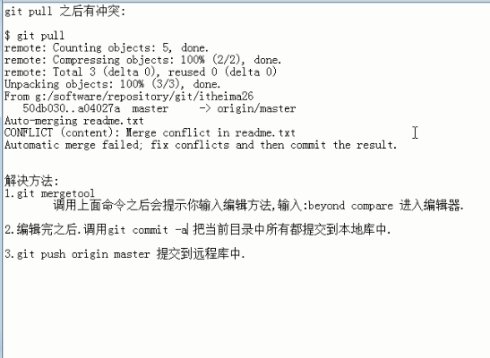
   git push origin master

6.切换目录(新用户)，获取最新的文件。

   git pull

7.查看git仓库中各文件的当前状态

git status

[](http://photo.blog.sina.com.cn/showpic.html#blogid=4d9c3fec0102w7g9&url=http://album.sina.com.cn/pic/001q7pWQzy708nJG2r335)

###### Github 完整学习教程

2017年05月11日 08:54:09

阅读数：394

# 前言

本文可能不会是非常详细的关于 GitHub 的教程，只是把基本的步骤和重要的地方记录下来，方便查阅。关于 GitHub 的学习，建议可以首先阅读完参考资料中的 stormzhang [从 0 开始学习 GitHub](http://blog.csdn.net/googdev/article/category/6306065" \t "_blank)。

说道 GitHub，可能很多人也常听到 Git，那这两是同一个东西吗？答案不是的。很早的我、懵懂的我，虽然经常听到这两家伙，但是不是很清楚区别。我相信很多人也是的，认为 GitHub 就是 Git，其实这是一个理解的误区。

我们先看下 Wiki 百科上关于 GitHub 说的吧：

GitHub 是个共享虚拟主机服务，于存放使Git 版本控制的软件代码和内容项它由 GitHub 公司（曾称 Logical Awesome）的开发者 ChrisWanstrath、 PJ Hyett 和 Tom Preston-Werner 使⽤ Ruby on Rails 编写成。

可能这么说大家不能理解其作用。

说下什么是 Git？

Git 是一款免费、开源的分布式版本控制系统，他是著名的 Linux 发明者 Linus Torvalds 开发的。说到版本控制系统，估计很多人都用过 SVN ，只不过 Git 是新时代的产物，如果你还在用 SVN 来管理你的代码，那就真的有些落伍了。

其实就和 SVN 一样， Git 是代码版本控制系统，免费开源的，并且是一个分布式版本控制系统。不管是学GitHub，还是以后想从事编程行业，可以说 Git 算是必备技能了。而 GitHub 上面说了，主要提供基于 git 的版本托管服务。也就是说现在 GitHub 上托管的所有项目代码都是基于 Git 来进行版本控制的，所以 Git 只是 GitHub 上用来管理项目的一个工具而已，但 GitHub 的功能可远不止于此！ 只要你愿意慢慢玩你还能知道它的用处：

• 学习优秀的开源项目   
• 多人协作   
• 搭建博客、个人网站或者公司官网   
• 写作   
• 个人简历   
• 其他

比如基于 Hexo + Github Page 可以搭建免费博客。关于 Github 和 Git 以及 git 和 svn 详细区别可以看这篇文章：[Git系列之初识Git与Github](http://dkylin.com/archives/2016/git-github-base.html" \t "_blank)

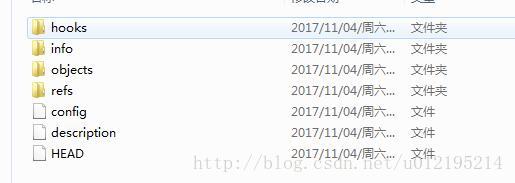
# 首先是 Git 的简单使用

好了，废话不多说，下面说下自己的使用教程，本文的教程首要目标就是带你使用 git 上传自己的项目代码到 github 网站。

1. git 下载（有两个网站都可以下载：[https://git-scm.com/downloads](https://git-scm.com/downloads" \t "_blank)、[https://git-for-windows.github.io/](https://git-for-windows.github.io/" \t "_blank)）

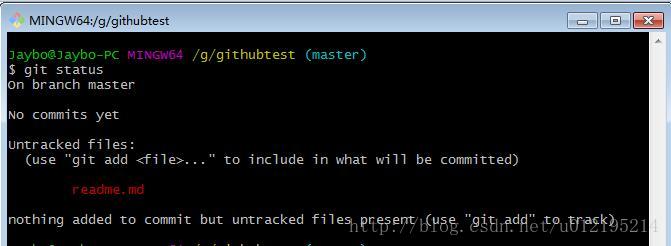
毋庸置疑，提到这么多次git，肯定首先需要安装 git 版本控制工具系统。下载完毕就是进行安装了，安装教程就不多说了，就类似 Java 开发需要安装 JDK 一样，这里也是，不安装 git 怎么进行代码版本管理。安装详细教程网上搜索下。注意的地方，有个安装页面可以选择勾选 git 自动加入 系统 path ，也可以不选，安装完毕，自己加入 git 安装目录下 bin 文件目录于 电脑 path 下。 path的作用：就是可以使你在电脑 cmd 命令窗口下能直接使用相关命令。判断是否安装成功：cmd 命令行下，输入 git 可以看到一些信息，则证明安装 ok。

1. 随便在哪个磁盘创建一个文件夹，比如 githubtest，然后在该目录下，鼠标右键选择 GIt Bash Here 打开git，输入 git init 初始化 git 仓库。这样能在”不显示隐藏文件”的情况下，能看到 .git 文件夹。内容如下图：



此文件夹保存了版本控制的所有相关信息。

1. 在 githubtest 文件夹下随便创建一个文件，如创建一个名叫 readme.md 文件，然后输入 git status 可以查看当前版本库状态，可以看到如下图：



在Untracked files(未跟踪文件)下，会出现红色的readme.txt，代表此文件还未被Git所管理。

1. 使用 git add readme.md，将该文件加入**缓冲区**（这里的缓冲区可以后面再好好了解下），如果你确定所有的修改都需要提交，可以使用 git add . 来加入所有修改。现在用git status查看，将看到文件名变为绿色。
2. 使用 git commit -m "my first commit!" 来提交修改，-m 后面所带的参数是本次提交的注释说明信息，一般用来记录本次提交的主要意图。

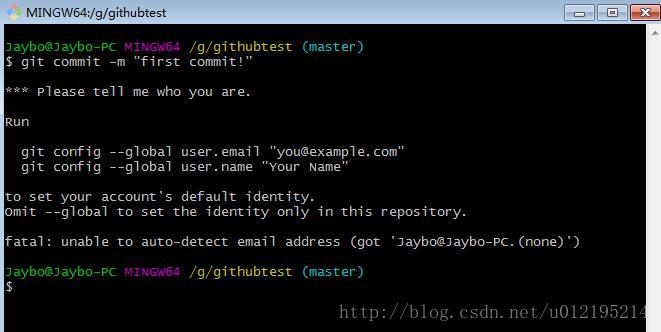
注：为什么先要再 add 一次呢？首先 git add 是先把改动添加到一个「暂存区」 ，你可以理解成是一个缓存区域，临时保存你的改动，而 git commit 才是最后真正的提交。这样做的好处就是防止误提交，当然也有办法把这两步合并成一步。**这个命令是**：

git commit -am "my first commit!"

* + 1

该命令用于将工作区中的代码直接提交到本地仓库，而无需手动添加到缓冲区。

但是执行后，出现如下图：



未提交成功，提示设置邮箱和用户名。

是的，为了保证提交日志的准确性，在提交的时候 user.name 和 user.email 会进入日志，这些信息，是追踪代码变更的关键，比如是谁修改的。以后会随更新内容一起被永久纳入历史记录。

PS：在设置用户名的时候，可以使用任意的字符。Git实际上是使用email进行关联每次提交的，只不过使用username作为标示符来进行显示。当你的email地址和github上的email地址一致时，则会使用Github上面的name来进行显示。

**全局配置：**

如果工作中只涉及一个 git 服务器，用一个全局配置就可以了。

git config --global user.name "strivebo"

git config --global user.email "ishuzb@gmail.com"

* + 1
  + 2

**工作在多个git项目：**

但是我们可能同时工作在多个项目中，公司内部用自有的git管理项目，我们在github上还有自己的项目。对于使用不同的用户身份，需要设置不用的sshkey，具体的配置可以看这里：[多个sshkey配置](https://www.liaohuqiu.net/cn/posts/git-setup-and-setting/" \t "_blank)。这个时候，对于user.name和user.email我们不能采用全局的配置。而是要对各个项目单独配置。

某个项目下的配置（去掉 --global）：

git config user.name "strivebo"

git config user.email "ishuzb@gmail.com"

* + 1
  + 2

可以使用命令来查看修改后的配置：

git config --global user.name 或 git config user.name

git config --global user.email 或 git config user.email

* + 1
  + 2

**取消全局配置：**

git config --global --unset user.name

git config --global --unset user.email

git config --global user.name #(查看)全局配置账户已经移除

git config --global user.email #(查看)全局配置邮箱已经移除

* + 1
  + 2
  + 3
  + 4
  + 5

参考资料：

​ [设置 Git 账户及邮箱](http://daemon369.github.io/git/2015/03/11/setting-email-in-git" \t "_blank)

​ [git: 提交前强制检查各个项目用户名邮箱设置](https://www.liaohuqiu.net/cn/posts/using-diffrent-user-config-for-different-repository/" \t "_blank)

​ [Git中的username的设置](https://blog.spoock.com/2016/03/29/git-username-setting/" \t "_blank)

多说几句，关于 github 页面的设置哪个邮箱接收哪些消息，看下这篇文章：[如何正确接收 GitHub 的消息邮件](https://github.com/cssmagic/blog/issues/49" \t "_blank)

1. 提交成功后，可以用 git log 查看历史提交记录。每个记录都会有提交id，作者和提交日期。

现在我想使用版本回退操作，我想把当前的版本回退到上一个版本，要使用什么命令呢？可以使用如下2种命令，第一种是：

git reset --hard HEAD^

* + 1

那么如果要回退到上上个版本只需把 HEAD^ 改成 HEAD^^ 以此类推。那如果要回退到前100个版本的话，使用上面的方法肯定不方便，我们可以使用下面的简便命令操作：git reset --hard HEAD~100 即可。

**假设**： 我进行了两次修改，第一次readme.txt文件添加了2222，第二次添加了3333，我已经使用回退操作回到了第一次的修改，即现在文本内容为2222，但我其实又想回到第二次的修改，该怎么办呢（如何恢复3333内容呢）？这里可以：

可以通过如下命令即可获取到版本号：git reflog，可以看到增加内容3333的版本号是多少比如为 6fcfc89，我们现在可以命令：

git reset --hard 6fcfc89

* + 1

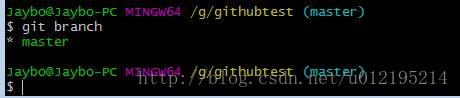
来恢复了。

1. 你可以用 git branch 查看当前有哪些分支，当然，因为我们没有创建任何分支，目前只会有一个master分支。

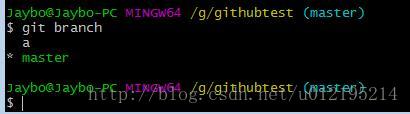
注：branch 即分支的意思，分支的概念很重要，尤其是团队协作的时候，假设两个人都在做同一个项目，这个时候分支就是保证两人能协同合作的最大利器了。举个例子，A, B俩人都在做同一个项目，但是不同的模块，这个时候A新建了一个分支叫a， B新建了一个分支叫b，这样A、B做的所有代码改动都各自在各自的分支，互不影响，等到俩人都把各自的模块都做完了，最后再统一把分支合并起来。

​

执行 git init 初始化git仓库之后会默认生成一个主分支 master ，也是你所在的默认分支，也基本是实际开发正式环境下的分支，一般情况下 master 分支不会轻易直接在上面操作的，你们可以输入 git branch 查看下当前分支情况，如图：



1. 如果我们想在此基础上新建一个分支，很简单，执行 git branch a 就新建了一个名字叫 a 的分支，这时候分支 a 跟分支 master 是一模一样的内容，我们再输入 git branch 查看的当前分支情况：



但是可以看到 master 分支前有个 \* 号，即虽然新建了一个 a 的分支，但是当前所在的分支还是在 master 上，如果我们想在 a 分支上进行开发，首先要先切换到 a 分支上才行，所以下一步要切换分支

git checkout a

* + 1

有人就说了，我要先新建再切换，未免有点麻烦，有没有一步到位的，有的：

git checkout -b a

* + 1

​

1. 以下为常用的 Git 命令：

git merge

* + 1

A同学在a分支代码写的不亦乐乎，终于他的功能完工了，并且测试也都ok了，准备要上线了，这个时候就需要把他的代码合并到主分支master上来，然后发布。git merge 就是合并分支用到的命令，针对这个情况，需要先做两步，第一步是切换到 master 分支，如果你已经在了就不用切换了，第二步执行 git merge a ，意思就是把a分支的代码合并过来，不出意外，这个时候a分支的代码就顺利合并到 master 分支来了。为什么说不出意外呢？因为这个时候可能会有冲突而合并失败。

​

git branch -d

* + 1

有删除分支，假如这个分支新建错了，或者a分支的代码已经顺利合并到 master 分支来了，那么a分支没用了，需要删除，这个时候执行 git branch -d a 就可以把a分支删除了。

​

git branch -D

* + 1

有些时候可能会删除失败，比如如果a分支的代码还没有合并到master，你执行 git branch -d   
a 是删除不了的，它会智能的提示你a分支还有未合并的代码，但是如果你非要删除，那就执   
行 git branch -D a 就可以\*\*强制删除\*\*a分支。

​

git tag

* + 1

我们在客户端开发的时候经常有版本的概念，比如v1.0、v1.1之类的，不同的版本肯定对应不同的代码，所以我一般要给我们的代码加上标签，这样假设v1.1版本出了一个新bug，但是又不晓得v1.0是不是有这个bug，有了标签就可以顺利切换到v1.0的代码，重新打个包测试了。所以如果想要新建一个标签很简单，比如 git tag v1.0 就代表我在当前代码状态下新建了一个v1.0的标签，输入 git tag 可以查看历史 tag 记录。 想要切换到某个 tag，执行：

git checkout v1.0

* + 1

就可以切换到 v1.0 的代码状态。

还有很多命令，待大家去找相关资料学习和练习了。

# 上传代码至 Github

以上都是讲的本地仓库关于 Git 的操作，下面需要讲的是项目代码提交至 GitHub 开源社区。

关于 GitHub 网站的操作教程包括注册、功能、设置、新建代码仓库等，请先看这篇文章吧：[Git系列之Github基础设置及使用详解](http://dkylin.com/archives/2016/git-github-base-usage.html" \t "_blank)，写的很详细很好。我自个写的，也只是简单记录下步骤而已。

## 以自己 Github 的项目为例

1. 首先注册 github 账户

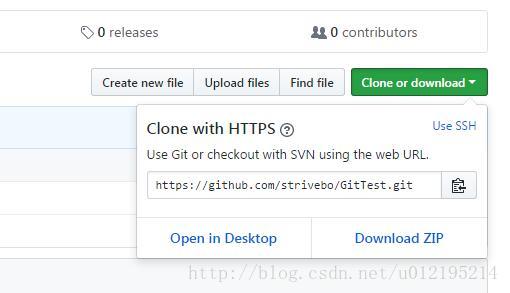
这个我就不用说了吧。

1. 新建仓库，即新建项目文件

如果想要新建私有仓库，即别人不能看到的仓库，则需要选择收费选项，否则我们就选择公共模式就可以。好像 github 有一个针对高校和学生有一个叫什么学生包申请，可以去了解下，我在网上也随便找了几个文章，先看下：[Github 学生包申请教程](https://www.mf8.biz/github-education-pack/" \t "_blank)、[利用学生身份可以享受到的相关学生优惠权益…](https://github.com/ivmm/Student-resources" \t "_blank)

1. 然后就是克隆(下载)该项目

如图：



复制该地址。

在 Git Bash 中输入：

git clone https://github.com/strivebo/GitTest.git

* + 1

下载项目源码。这个时候该本地项目本身已经是一个 git 库了，不需要执行 git init 进行初始化，甚至已经关联好了远程仓库。

1. 此时，就可以去开发里面的代码了，添加新的功能，修改明显的bug…

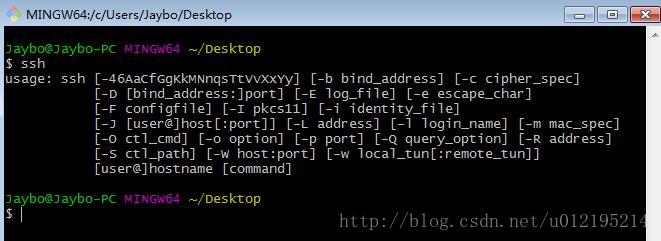
关于这部分已经在前面 Git 本地库讲了相关命令。

1. 最后就是要把本地的改动提交到 github 账户了。(\*\*\*\*\*\*\*\*我提交不成功原因)

**注：** 你拥有了一个 GitHub 账号之后，就可以自由的 clone 或者下载其他项目，也可以创建自己的项目，但是你没法提交代码。仔细想想也知道，肯定不可能随意就能提交代码的，如果随意可以提交代码，那么GitHub 上的项目岂不乱了套了，所以提交代码之前一定是需要某种授权的，而 GitHub 上一般都是基于 SSH 授权的。那么什么是 SSH 呢？ 简单点说，SSH是一种网络协议，用于计算机之间的加密登录。目前是每一台 Linux 电脑的标准配置。而大多数 Git 服务器都会选择使用 SSH 公钥来进行授权，所以想要在 GitHub 提交代码的第一步就是要先添加 SSH key 配置。

## SSH 设置

* + Linux 与 Mac 都是默认安装了 SSH ，而 Windows 系统安装了 Git Bash 应该也是带了 SSH 的。大家可以在终端（win下在 Git Bash 里） 输入 ssh 如果出现以下提示证明你本机已经安装 SSH， 否则请搜索自行安装下。

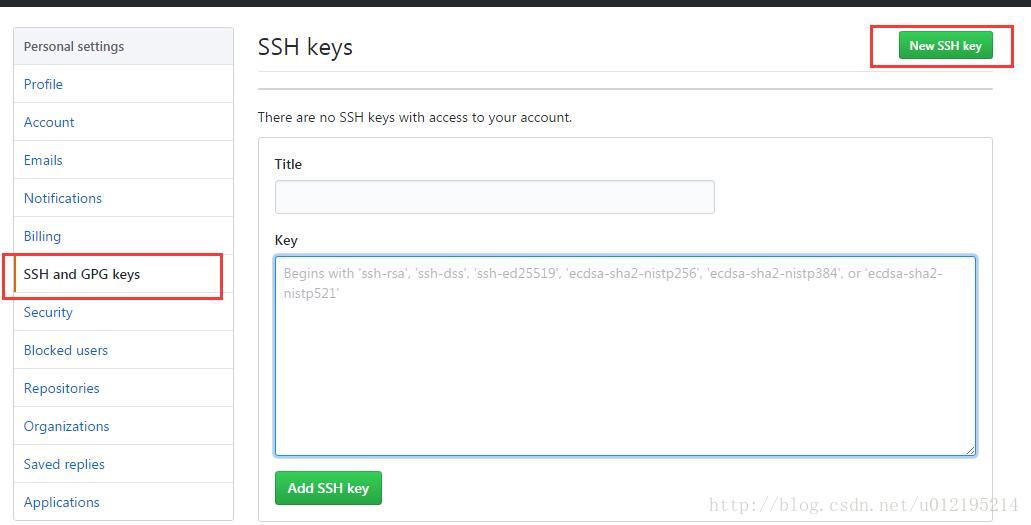


* + 紧接着输入 ssh-keygen -t rsa ，什么意思呢？就是指定 rsa 算法生成密钥，接着连续三个回   
    车键（不需要输入密码） ，然后就会生成两个文件 id\_rsa 和 id\_rsa.pub ，而 id\_rsa 是密钥，id\_rsa.pub 就是公钥。这两文件默认分别在如下目录里生成：   
    Linux/Mac 系统 在 ~/.ssh 下，win系统在 /c/Documents and Settings/username/.ssh 下，   
    都是隐藏文件，相信你们有办法查看的。



**id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。**

* + 接下来要做的是把 id\_rsa.pub 的内容添加到 GitHub 上，这样你本地的 id\_rsa 密钥跟 GitHub   
    上的 id\_rsa.pub 公钥进行配对，授权成功才可以提交代码。 在设置页面打开 SSH and GPG keys 菜单栏：

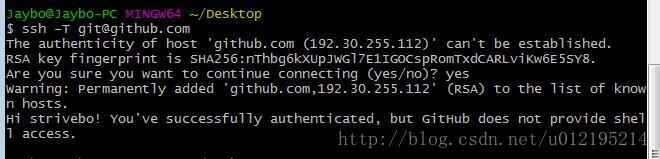


需要做的只是在 Key 那栏把 id\_rsa.pub 公钥文件里的内容复制粘贴进去就可以了，Title 那栏不需要填写，点击 Add SSH key 按钮就ok了。

* + SSH key 添加成功之后，输入 ssh -T git@github.com 进行测试，如果出现以下提示：



输入 yes 出现如下图则证明添加成功了。



​

1. 在提交之前熟悉两个命令：Push 和 Pull

Push ：直译过来就是「推」 的意思，什么意思呢？如果你本地代码有更新了，那么就需要把本地代码推到远程仓库，这样本地仓库跟远程仓库就可以保持同步了。

git push origin master

* + 1

意思就是把本地代码推到远程 master 分支。

​

Pull：直译过来就是「拉」 的意思，如果别人提交代码到远程仓库，这个时候你需要把远程仓库的最新代码拉下来，然后保证两端代码的同步。

git pull origin master

* + 1

意思就是把远程最新的代码更新到本地。一般我们在 push 之前都会先 pull ，这样不容易冲突。

1. 提交代码

添加 SSH key 成功之后，我们就有权限向 GitHub 上我们自己的项目提交代码了！执行：

git push origin master

* + 1

进行代码提交，这种是最简单方便的一种方式。

**注:**如果我们本地已经有一个完整的 git 仓库，并且已经进行了很多次 commit，这个时候第一种方法就不适合了。

**假设** ：我们本地有个 test2 的项目，我们需要的是在 GitHub 上建一个 test 的项目，然后把本地 test2 上的所有代码 commit 记录提交到 GitHub 上的 test 项目。

第一步就是在 GitHub 上建一个 test 项目，这个想必大家都会了，就不用多讲了。

第二步把本地 test2 项目与 GitHub 上的 test 项目进行关联，切换到 test2 目录，执行如下命   
令：

git remote add origin git@github.com:strivebo/test.git

* + 1

什么意思呢？就是添加一个远程仓库，他的地址是 git@github.com:stormzhang/test.git ，而 origin 是给这个项目的远程仓库起的名字，是的，名字你可以随便取，只不过大家公认的只有一个远程仓库时名字就是 origin ，为什么要给远程仓库取名字？因为我们可能一个项目有多个远程仓库？比如 GitHub 一个，比如公司一个，这样的话提交到不同的远程仓库就需要指定不同的仓库名字了。

查看我们当前项目有哪些远程仓库可以执行如下命令： git remote -v ，接下来，我们本地的仓库就可以向远程仓库进行代码提交了：

git push origin master

* + 1

就是默认向 GitHub 上的 test 目录提交了代码，而这个代码是在 master 分支。当然你可以提交到**指定的分支**。

## 以别人开源项目为例

和上面很多步骤类似的。

1. 首先 github 上搜索别人开源项目，单击右上角的fork按钮，就把这个项目拉到你的账户下了，你就可以加入到这个项目中了；
2. 页面已经跳转到你的账户下的该项目里了，单击右下方的复制按钮，将这个地址就复制下来了；
3. 在你的磁盘选择一个合适的位置专门存放源代码，比如D盘下新建 gittest 文件件，进入 gittest 文件夹，鼠标右键进入 Git Bash，输入：git clone 接着项目地址，回车，就开始将你的 github 上的该源码下载到你的电脑里了；
4. 此时就可以去开发里面的代码了，添加新的功能，修改bug等等操作了；
5. git push -u origin master 将你本地的仓库提交到你的github账号里，如果此时需要你输入你的github的账号和密码，输入就是了；
6. 此时你在你的本地任务就完成了，进入到你的 github 上面，选择到这个该项目名，单击进入，右侧有个pull request，单击进入跳转的页面单击右侧的 New pull Request 按钮；
7. 此时，你就能看到你改动的方了，核对下，没有问题后，就单击 View pull request 按钮；
8. 此时，你可以在页面中输入你的本次提交的说明信息，输入完后，单击提交按钮 comment；
9. ,到此，你的任务就完成了，等到开源项目的管理人员审核，通过了，他就把你的改动合并到相应的开发分支上。

**参考资料：**

github 学习：

* 知乎：[怎样使用 GitHub？](https://www.zhihu.com/question/20070065" \t "_blank)
* [入门级：GitHub和Git超超超详细使用教程！](http://blog.csdn.net/javaandroid730/article/details/53522872" \t "_blank)
* [手把手教你如何加入到github的开源世界！](http://www.cnblogs.com/wenber/p/3630921.html" \t "_blank)
* [Git结合GitHub常用命令学习手册](https://www.kancloud.cn/thinkphp/git-github-study" \t "_blank)
* 菜鸟教程：[Github 简明教程](http://www.runoob.com/w3cnote/git-guide.html" \t "_blank)
* phodal：[GitHub 漫游指南](https://github.com/phodal/github-roam" \t "_blank)
* stormzhang：[从 0 开始学习 GitHub](http://blog.csdn.net/googdev/article/category/6306065" \t "_blank)

git 学习：

* 廖雪峰Git教程：[Git教程](http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/" \t "_blank)
* [猴子都能懂的GIT入门](https://backlog.com/git-tutorial/cn/intro/intro1_1.html" \t "_blank)
* [Git使用教程](http://www.cnblogs.com/tugenhua0707/p/4050072.html" \t "_blank)
* happypeter：[Git 北京](https://haoqicat.com/gitbeijing" \t "_blank)

加载秘钥

## 1.使用 git 客户的生成公私钥：id\_rsa、id\_rsa.pub

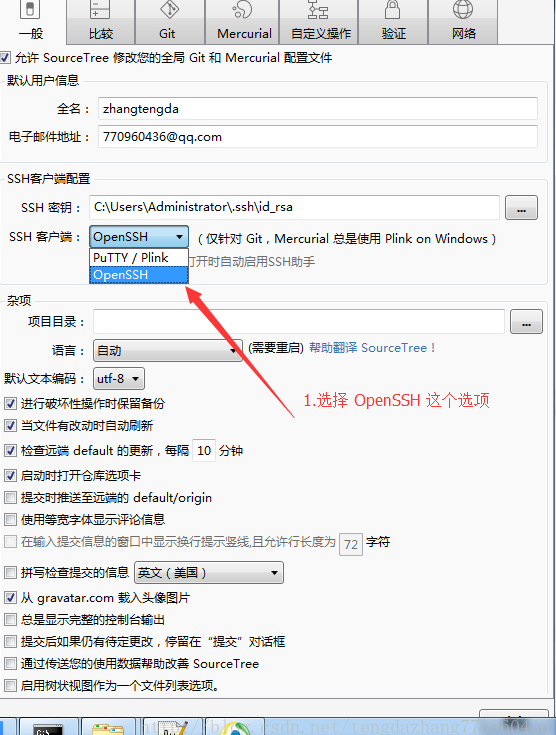
* 1.1设置Git的user name和email：

$ git config --global user.name "xxx"

$ git config --global user.email "xxx.mail@xxx.com"

* 1
* 2
* 1.2.生成SSH密钥过程：   
  + 1.2.1.检查是不是已经存在密钥（能进去说明已经存在，就删掉文件夹，重新创建）：   
    - cd ~/.ssh
* 1.3.生成 SSH 密钥：   
  + $ ssh-keygen -t rsa -C “xxx.mail@xxx.com”
  + 按3个回车，密码为空。
* 1.4.文件存放位置 ~/.ssh，如果是window的话就在：C:\Users\Administrator.ssh 下面，当然如果你不是 Administrator 用户的话，需要换成对应的用户。

## 2.设置 SourceTree 的 SSH客户端

* 1.配置SourceTree 的 SSH 客户的为：OpenSSH   
  + 1.1.工具->选项
  + 1.2.设置 OpenSSH,这时候，SSH 密钥这一栏自然会去选择当前用户下的 .ssh 目录下的 id\_rsa 这个私钥：   
    

## 3.添加 ~/.ssh/id\_rsa.pub 文件内容到 git 服务器里面去

* 3.1.比如你的 git 服务是 github，那么你需要在 [https://github.com/settings/keys](https://github.com/settings/keys" \t "_blank) 里面添加 SSH key
* 3.2.SourceTree 来下载 git 项目   
  + 3.2.1.复制你的 git 地址：git@github.com:ztd770960436/justgame.git
  + 3.2.2.从 SourceTree 里面新建一个地址，这时候你发现你本地已经可以下载远程的 git 代码了

## 4.解释：

* 1.ssh-keygen 是公钥私钥的非对称加密方式：   
  + 1.1.公钥：用于向外发布，任何人都能获取。
  + 1.2.私钥：要自己保存，切勿给别人
* 2.公钥私钥加解密的原理   
  + 2.1.客户端把自己的公钥存放到要链接的远程主机上（相当于我们把自己的 id\_rsa.pub 存放到 git 服务器上）
  + 2.2.客户端要链接远程主机的时候，远程主机会向客户的发送一条随机的字符串，客户的收到字符串之后使用自己的私钥对字符串加密然后发送到远程主机，远程主机根据自己存放的公钥对这个字符串进行解密，如果解密成功证明客户端是可信的，直接允许登录，不在要求登录。

//仓库->检出->检出新分支

# 5在进行一次提交后,马上切换到其他分支,导致提交到临时分支丢失处理方法



# 6 用git直接提交文件

## 1使用git clone命令将新建的git项目信息下载到本地

git clone git addresss

## 2 拷贝自己的项目到git文件中

1 git status

2 git add -A (all 要大写)

git commit -a –“xxx”

//如果git commit -a –“xcxx”失败的话,直接用git commit -m”就这个方法”

git push

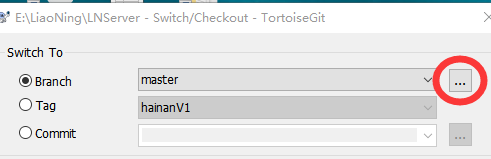
# 3 git add 上传本地变化的命令三种

1 git add -A 提交所有变化

2 git add -u提交修改(modified)和被删除(deleted),不包含新文件(new)

3 git add . 提交新文件(new)和被修改(modified),不包含被删除(deleted)文件

## TortoiseGit的使用

1. 删除远程标签,
2. 

## 

右击即出现删除选项

记录一些脚本运用到的命令

1. bat 脚本运行 添加子模块

git submodule add git@192.168.67.10:LiaoNing\_New/area\_tianjin.git ./area\_tianjin

1. 拉最新代码

cd area\_liaoning

git checkout master

1. 统一推送提交记录

cd xini

git add -u

git commit -m"xini"

git push origin master//主要下各命令时候需要cd ../ ,自己看情况