Lesson3

第1步：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：



如果一切顺利的话，可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是SSH Key的秘钥对，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。

第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容：



点“Add Key”，你就应该看到已经添加的Key：



**添加远程库**

首先，登陆GitHub，创建一个新的仓库

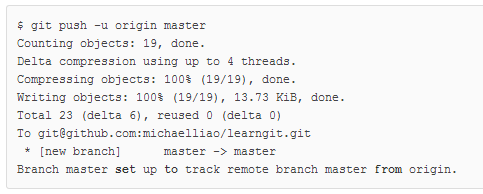
现在，我们根据GitHub的提示，在本地的learngit仓库下运行命令：



在本地关联自己的远程库

添加后，远程库的名字就是origin，这是Git默认的叫法，可以更改，但是origin这个名字一看就知道是远程库。

下一步，就可以把本地库的所有内容推送到远程库上：



注：实际操作中，发现要先进行下载，才能推送。使用 git pull origin master 命令

把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

推送成功后，可以立刻在GitHub页面中看到远程库的内容已经和本地一模一样。

从现在起，只要本地作了提交，就可以通过命令：



把本地master分支的最新修改推送至GitHub，现在，你就拥有了真正的分布式版本库！

### 小结

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

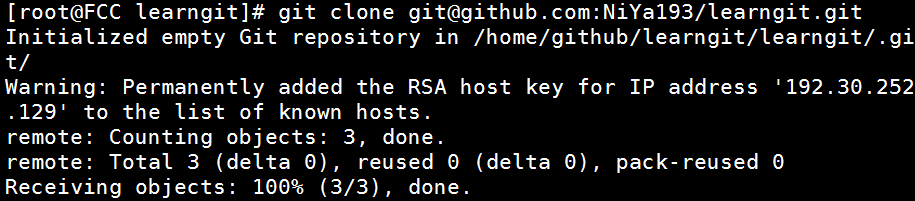
分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在，也就是有没有联网都可以正常工作，而SVN在没有联网的时候是拒绝工作的！当有网络的时候，再把本地提交推送一下就完成了同步！

**从远程库克隆**

首先，登陆GitHub，创建一个新的仓库

我们勾选Initialize this repository with a README，这样GitHub会自动为我们创建一个README.md文件。创建完毕后，可以看到README.md文件：

现在，远程库已经准备好了，下一步是用命令git clone克隆一个本地库：



github clone [git@github.com:用户名/](mailto:git@github.com:用户名/)项目名称

如果有多个人协作开发，那么每个人各自从远程克隆一份就可以了。

你也许还注意到，GitHub给出的地址不止一个，还可以用https://github.com/michaelliao/gitskills.git这样的地址。实际上，Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。

使用https除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用ssh协议而只能用https。

**个人总结：**

Push 用于推送至服务器，Pull用于推送至本地。与克隆的区别是，克隆可以直接克隆一个

本地库，而Pull类似于Push，只推送修改部分。

每次推送，都要进行git add 和 git commit 才可以。