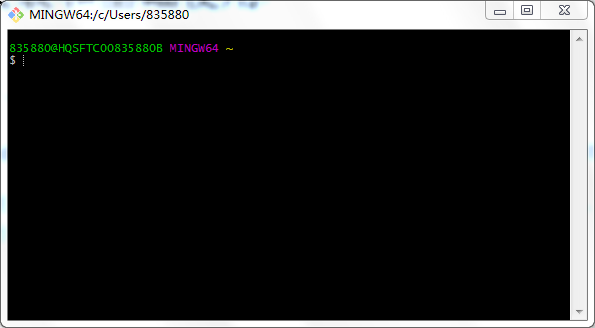
# Git的安装和基础使用

## Windows上安装Git

msysgit是Windows版的Git，从[http://msysgit.github.io/](http://msysgit.github.io/" \t "_blank)下载，然后按默认选项安装即可。

安装完成后，在开始菜单里找到“Git”->“Git Bash”，蹦出一个类似命令行窗口的东西，就说明Git安装成功！



安装完成后，还需要最后一步设置，在命令行输入：

$ git config --global user.name "Your Name"

$ git config --global user.email "email@example.com"

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。你也许会担心，如果有人故意冒充别人怎么办？这个不必担心，首先我们相信大家都是善良无知的群众，其次，真的有冒充的也是有办法可查的。

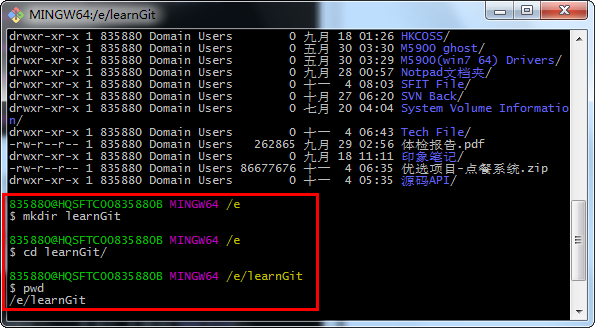
注意git config命令的--global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

## 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”。

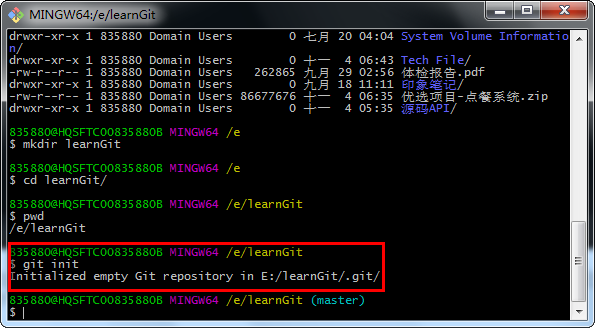
所以，创建一个版本库非常简单：

第一步：选择一个合适的地方，创建一个空目录：命令：mkdir learnGit



Pwd 显示当前的目录为本机 E盘下的learnGit文件夹

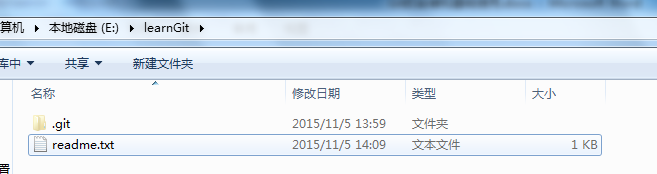
第二步，通过git init命令把这个目录变成Git可以管理的仓库：



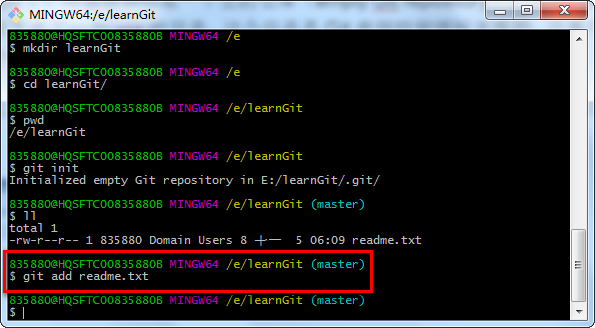
瞬间Git就把仓库建好了，而且告诉你是一个空的仓库（empty Git repository），细心的读者可以发现当前目录下多了一个.git的目录，这个目录是Git来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。

## 添加文件到仓库

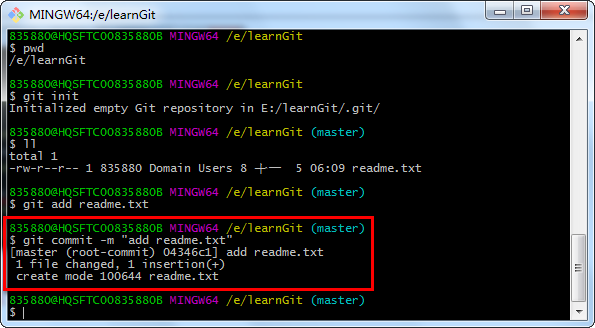
建立好仓库之后，建立一个文件readme.txt 在该文件夹中。之后将该文件用add追加到git的暂存区中，用commit提交到仓库中去。



第一步，用命令git add告诉Git，把文件添加到仓库：



第二步，用命令git commit告诉Git，把文件提交到仓库：



简单解释一下git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

## 删除文件

在Git中，删除也是一个修改操作，我们实战一下，先添加一个新文件test.txt到Git并且提交：

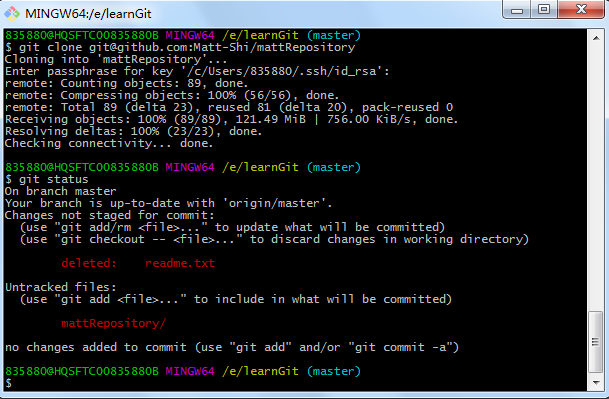
$ git add readme.txt

$ git **commit** -m "add readme.txt"

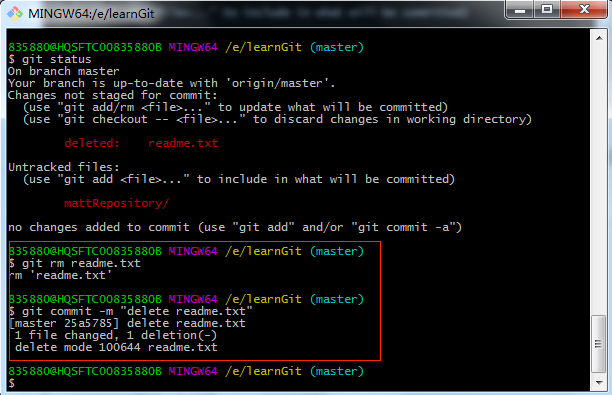
一般情况下，你通常直接在文件管理器中把没用的文件删了，或者用rm命令删了：

$ rm readme.txt

这个时候，Git知道你删除了文件，因此，工作区和版本库就不一致了，git status命令会立刻告诉你哪些文件被删除了：



现在你有两个选择，一是确实要从版本库中删除该文件，那就用命令git rm删掉，并且git commit：



现在，文件就从版本库中被删除了。

另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：

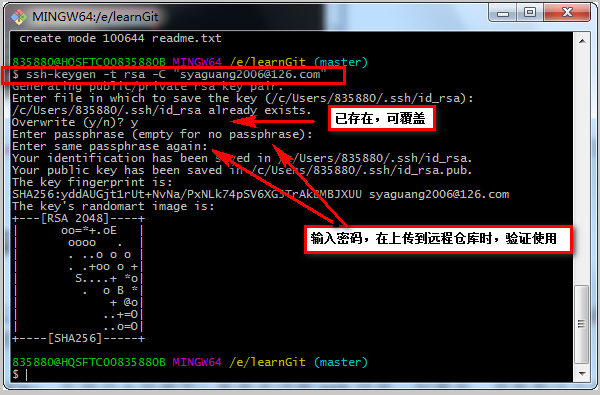
$ git checkout -- readme.txt

git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

## 创建远程仓库

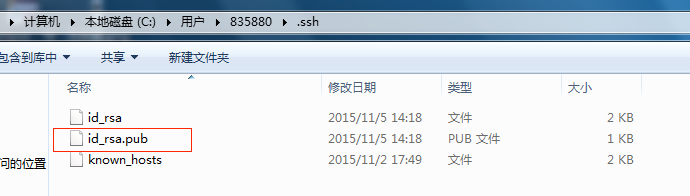
请登录<https://github.com/> 自行注册GitHub账号。由于你的本地Git仓库和GitHub仓库之间的传输是通过SSH加密的，所以，需要一点设置：

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Git Bash，创建SSH Key：

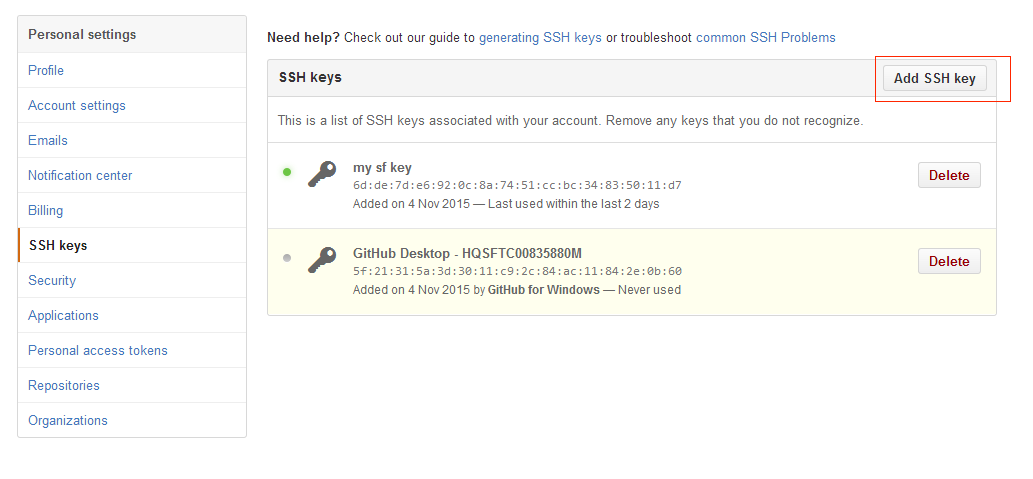
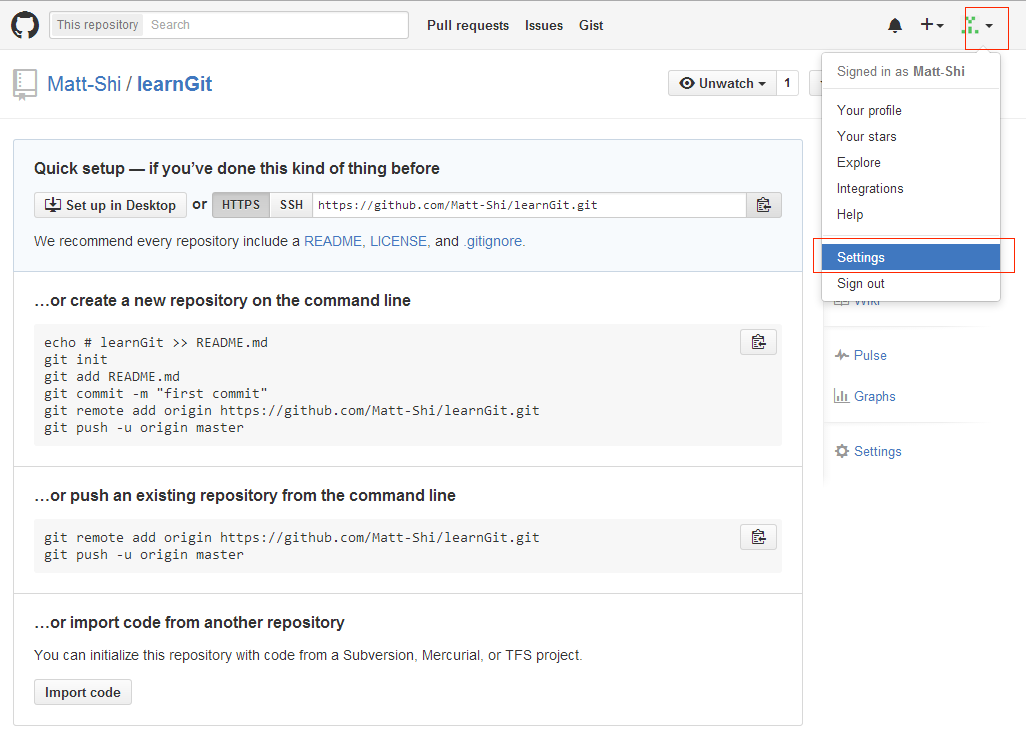


你需要把邮件地址换成你自己的邮件地址，然后一路回车，使用默认值即可，由于这个Key也不是用于军事目的，所以也无需设置密码。

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。

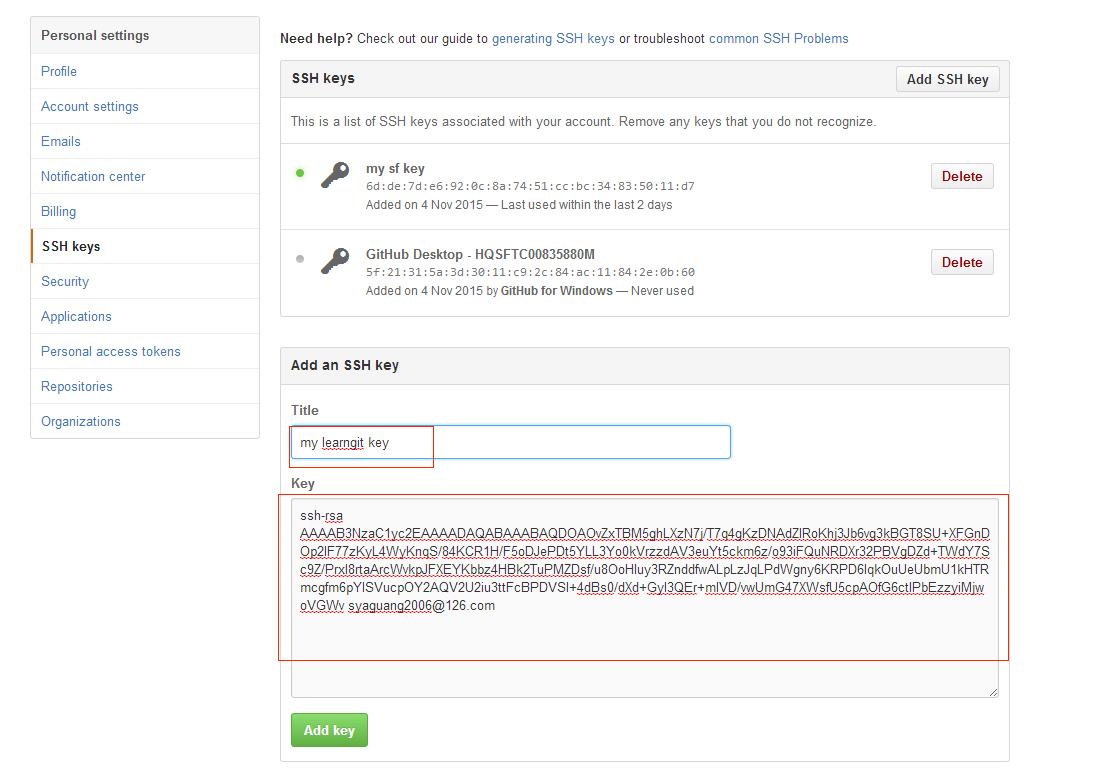


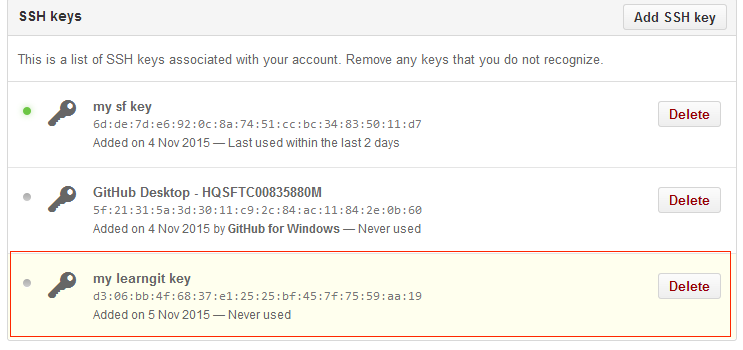
第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：



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：



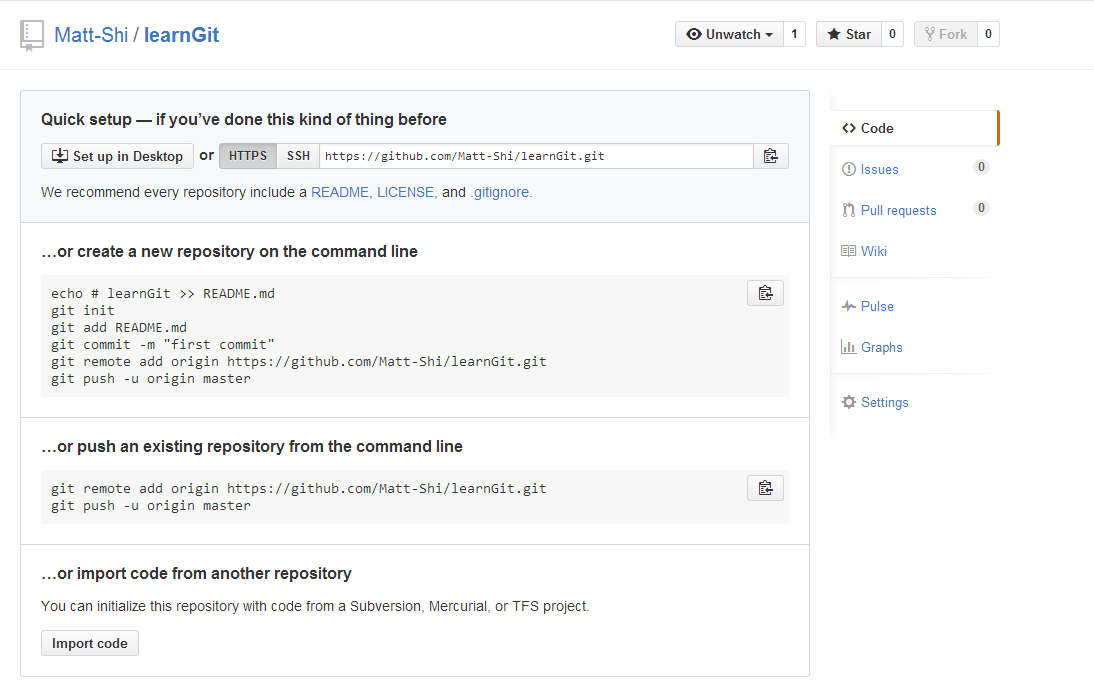


为什么GitHub需要SSH Key呢？因为GitHub需要识别出你推送的提交确实是你推送的，而不是别人冒充的，而Git支持SSH协议，所以，GitHub只要知道了你的公钥，就可以确认只有你自己才能推送。

当然，GitHub允许你添加多个Key。假定你有若干电脑，你一会儿在公司提交，一会儿在家里提交，只要把每台电脑的Key都添加到GitHub，就可以在每台电脑上往GitHub推送了。

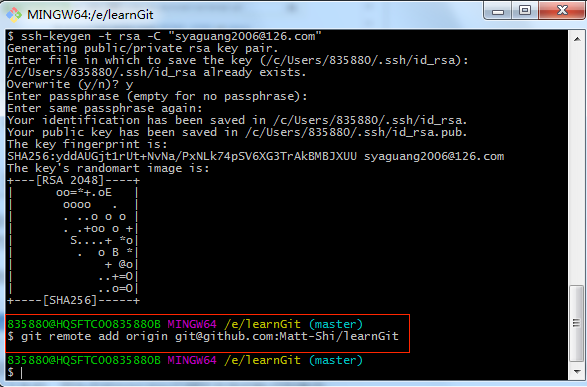
## 添加远程库

目前，在GitHub上的这个learnGit仓库还是空的，GitHub告诉我们，可以从这个仓库克隆出新的仓库，也可以把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到GitHub仓库。



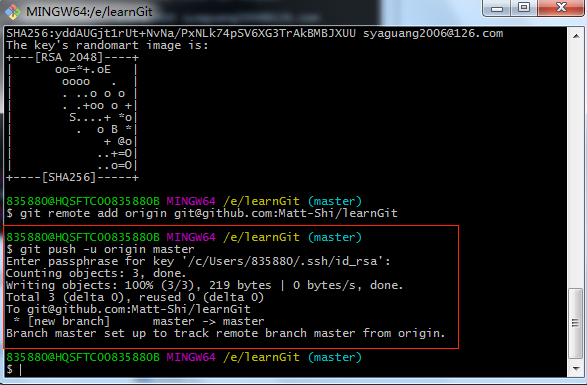
现在，我们根据GitHub的提示，在本地的learnGit仓库下运行命令：

$ git remote add origin git@github.com:Matt-Shi/learnGit



下一步，就可以把本地库的所有内容推送到远程库上：

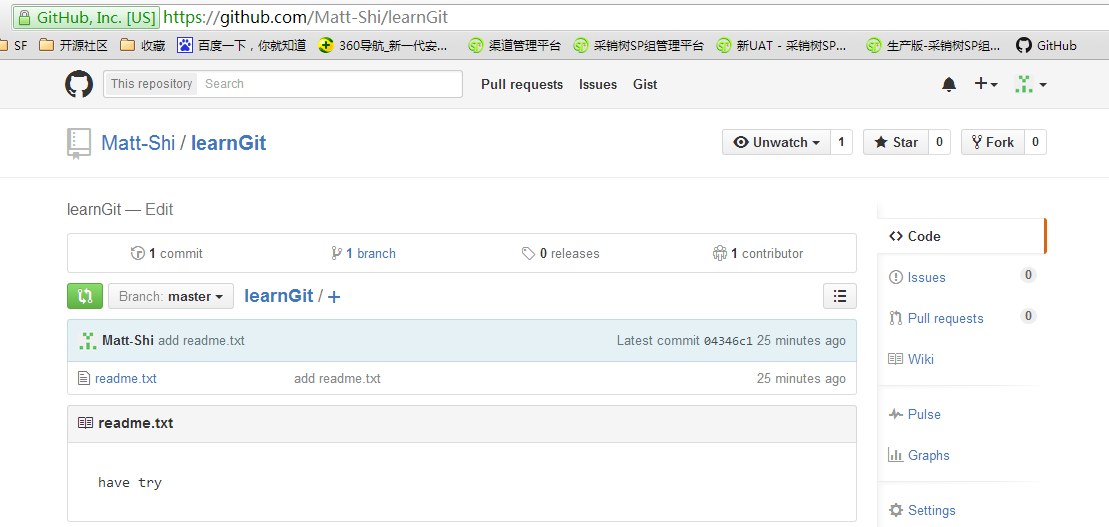
$ git push -u origin master



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：

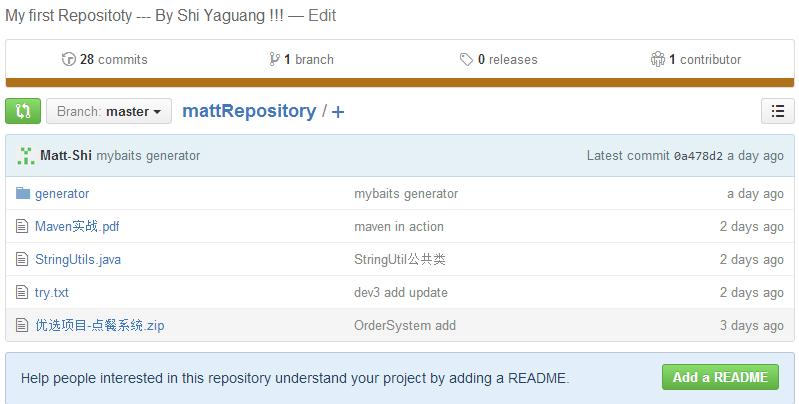
$ git push origin master

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！

在每次push之前 要pull一下远程库中的代码，以获取最新的远程库代码，然后在提交本地库中自己修改的代码

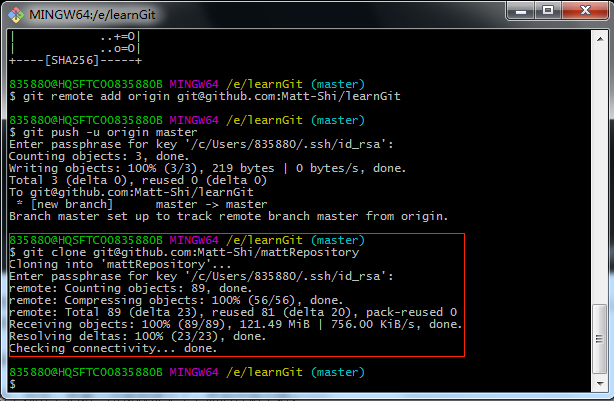
## 从远程库克隆

首先，登陆GitHub，创建一个新的仓库，名字叫mattRepository：

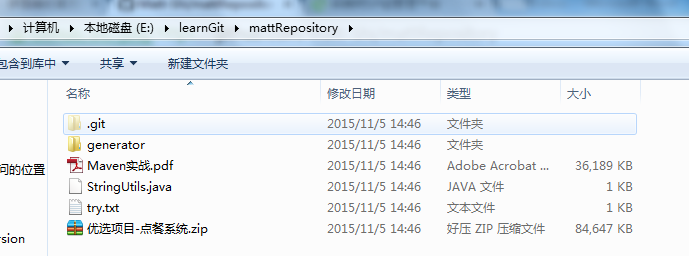


现在，远程库已经准备好了，下一步是用命令git clone克隆一个本地库：

$ git clone git@github.com:Matt-Shi/mattRepository



这样在本地的learnGit仓库中就会存在一个MattRepository的文件夹，包含从远程库中clone下来的仓库内容



参考博客：

<http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000>

## 注意事项

### git revert 和git reset的区别

git revert 是撤销某次操作，此次操作之前的commit都会被保留

git reset 是撤销某次提交，但是此次之后的修改都会被退回到暂存区

具体一个例子，假设有三个commit， git st:

commit3: add test3.c

commit2: add test2.c

commit1: add test1.c

当执行git revert HEAD~1时， commit2被撤销了

git log可以看到：

commit1：add test1.c

commit3：add test3.c

git status 没有任何变化

如果换做执行git reset --soft(默认) HEAD~1后，运行git log

commit2: add test2.c

commit1: add test1.c

运行git status， 则test3.c处于暂存区，准备提交。

如果换做执行git reset --hard HEAD~1后，

显示：HEAD is now at commit2，运行git log

commit2: add test2.c

commit1: add test1.c

运行git st， 没有任何变化

另外：

git revert <commit log string>是撤消该commit，作为一个新的commit。

### git fetch 和git pull的区别

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

git fetch origin master  
git log -p master..origin/master  
git merge origin/master

    以上命令的含义：  
   首先从远程的origin的master主分支下载最新的版本到origin/master分支上  
   然后比较本地的master分支和origin/master分支的差别  
   最后进行合并  
   上述过程其实可以用以下更清晰的方式来进行：

git fetch origin master:tmp  
git diff tmp   
git merge tmp

    从远程获取最新的版本到本地的test分支上  
   之后再进行比较合并  
2. git pull：相当于是从远程获取最新版本并merge到本地

git pull origin master

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

结束

### 请求连接错误

如果出现这个问题：

[ssh: connect to host github.com port 22: Bad file number(git客户端链接github时候出现的错误)](http://www.xnbing.org/?p=759)

使用该命令尝试连接

$ ssh -T [git*@github*.com](mailto:git@github.com)

## 文章总结

<http://www.yangzhiping.com/tech/github.html>

<http://www.zhihu.com/question/34544815>

<http://www.zhihu.com/question/20866683>

<http://www.zhihu.com/question/36974348>

1、Java资料库：  
[akullpp/awesome-java · GitHub](https://github.com/akullpp/awesome-java)/  
2、Spring 项目  
[spring-projects/spring-framework · GitHub](https://github.com/spring-projects/spring-framework)  
3、设计模式  
[iluwatar/java-design-patterns · GitHub](https://github.com/iluwatar/java-design-patterns)  
4、mybatis  
<https://github.com/mybatis/mybatis-3>  
5、spring mvc showcase  
[spring-projects/spring-mvc-showcase · GitHub](https://github.com/spring-projects/spring-mvc-showcase)  
6、maven的几个项目  
[sxyx2008/maven-framework-project · GitHub](https://github.com/sxyx2008/maven-framework-project)  
7、Jenkins插件开发  
[jenkinsci/jenkins · GitHub](https://github.com/jenkinsci/jenkins)