## ****一、如何安装git****

下载地址：

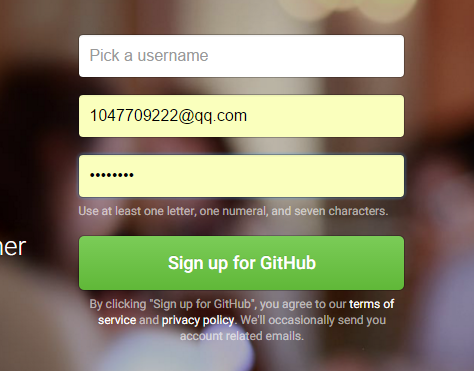
<https://git-scm.com/download/win>

根据自己的电脑选择是32位的还是64位的。下载完后直接运行，之后一直next就好了。安装成功后，会有这样一个标志。

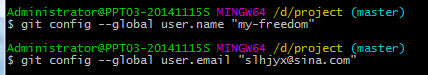


## ****二、如何使用git****

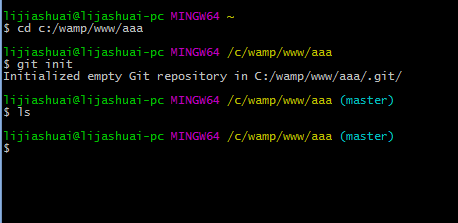
**（1）**首先你要点进去<https://github.com/>这个网站，注册一个github账号。注册好之后，要记住邮箱和密码。



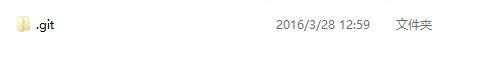
**（2**） 打开之前的那个图标就是这样一个窗口，首先Git是分布式版本控制系统，所以需要填写用户名和邮箱作为一个标识，分别输入这两个命令，用户名和邮箱需要换成自己的。



**（3）**  打开之前的那个图标就是这样一个窗口。现在比如我在我的C:/wamp/www/aaa有一个项目，我要对它进行管理怎么办呢。直接在控制台里面输入  cd   c:/wamp/www/aaa  之后你就会在aaa这个目录里面了。那如何对它进行管理呢。直接在控制台输入git  init



      之后你去查看aaa目录里面发现里面多了这样一个东西。



      这是一个隐藏文件夹，有点电脑看不见，是因为文件夹选项设置的隐藏文件不可见，这个去控制面板里的文件夹选择设置一下就好了。

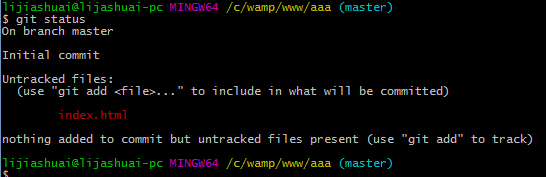
下面我们接着来操作，试着开始写我们的项目。

首先我写一个首页叫index.html

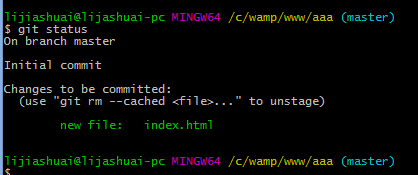


     现在我在我的页面里写好了这样一段话，保存之后我们接着来进行操作。

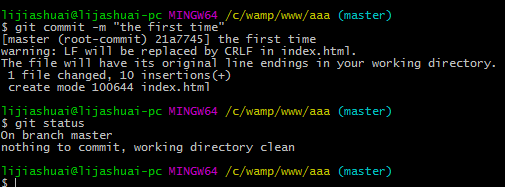
现在我们开始在git里面输入一段话 git status



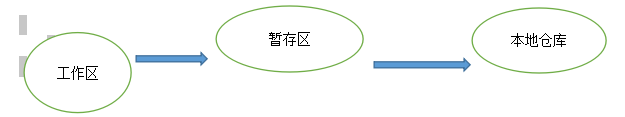
我们可以看到index.html这个文件是红色，这是说明我们这个文件已经做了修改，但是还没保存到本地仓库里面。现在我们接着执行操作 git add  index.html  敲完这个命令接着git status发现index.html变成绿色的啦。



接着我们敲命令  git commit   -m  “the first time”



     敲完之后，接着使用git status发现上面说工作区很干净，没有文件要被提交。现在我来解释一下这几个命令。

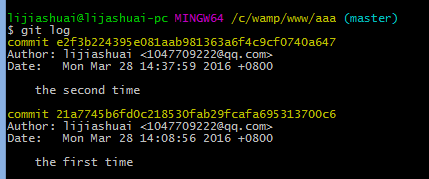


      首先你得明白这几个概念，工作区就是你现在编辑器所处的那个工程里面，在这个指的就是aaa这个文件夹。刚才你写了一个index.html写完之后就是往工作区增添了一个文件，然后你用git status查看状态发现，index.html是红色的。git status是什么意思呢，就是查看你工作区和暂存区有没有文件没被提交到本地仓库，如果有工作室未向暂存区保存的就显示红色，如果有暂存区没提交到本地仓库的就是绿色。那么什么是本地仓库呢，就是一开始说的那个不可见的文件夹，你一执行git init命令就会有一个本地仓库出来。

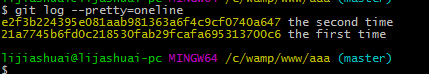
     现在我们接着来看，如何从工作区把文件提交到暂存区，就是使用命令git add index.html就可以了。把文件从暂存区提交到本地仓库呢，就是使用命令git  commit  –m “the first time” 这个引号里面的内容是随意的，就是自己添加一个备注，比如自己改动了什么东西。理解了这句话，我们接着进行操作，我们我的index.html里面继续添加一段话。我再git status查看一下当前的状态，发现有未向暂存区提交的保存，接着我们采用 git commit –m “the second time”命令将暂存区的文件提交到本地仓库。之后再用git status来查看一下，发现已经没有文件要提交了。

## ****（4）版本回退****

      到目前为止，我们已经向工作区提交了两次修改。Git给我们提供了一个git log 可以查看我们最近的提交历史。

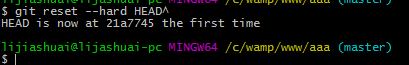


      如果看的很乱，可以使用这个命令，git log -–pretty=oneline



     可以看出来我们的两次提交，前面那串黄色的就是我们的版本号。现在我发现我第二次提交的代码运行不好使，想回到我原来的那个版本怎么办呢

可以使用命令git reset –hard HEAD^这个命令是回退到上一个版本。



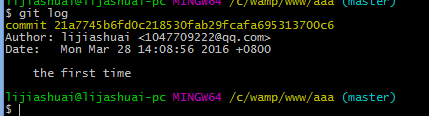
     执行完之后，我再回头看看我的index.html发现已经变成了我第一次提交时的样子。



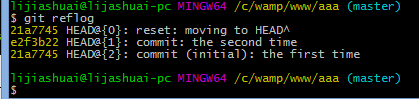
     接着继续。万一要回退到上上个版本呢，可以使用命令git reset –hard HEAD^^，那要是回到到一百个版本就要写100个^吗，也太麻烦了吧

现在我们通过git log –pretty=oneline已经获取到了每次修改的版本号。那么我们就可以使用git reset –hard 版本号就好了。比如刚才我要回退到上个版本就可以使用git reset –hard 21a77

发现也同样有效果。这里版本号不用写全，写一部分能表示这个就行。现在我们再用git log来查看一下现在的状态，发现我们现在的版本只剩下一个了



     但是我现在又发现，我知道原来新版本的解决方法了，我又想回到新版本去怎么办呢，现在用git log已经看不到新版本的版本号了。只能采用新的命令了叫做git reflog查看命令历史。



      在这里发现他记录了我们所有的操作，现在我们继续使用git reset –hard e2f3发现我们又回到新版本了。我们发现index.html已经回到原来的状态了。



      现在我们接着往下面进行。在index.html里面我们接着添加一行。



     后来发现这句话不是很合适，当然这个时候我们可以把这句话直接删掉，然后再git add ,就好了。但也可以通过git checkout – index.html。此时我们发现那句话已经没有了。这种情况适用于我们没有git add到暂存区之前。如果我们已经git add到暂存区了，怎么撤销修改呢。我们可以使用命令 git  reset HEAD index.html先撤销暂存区的修改，然后我们可以通过命令git diff查看暂存区和工作区有什么不同，之后我们可以通过命令git checkout –index.html 发现已经修改好了。但是万一我们已经提交到仓库里呢怎么办呢，也就是我们git commit了，那就要使用我们上面所说的版本回退了。

### ****（5）删除文件****

      现在我们在aaa目录下新建一个文件叫test.txt。我们在里面随便写几个字之后，通过git add text.txt 已经git commit –m “the third time”就已经把test.txt已经写入本地库里了。

假设现在我们发现我们已经不需要test.txt这个文件了，所以我们把它给删了。这个时候你通过git status发现工作区和本地库里的文件不一样了。

     现在我们有两种选择，一种是我们确实要删除。使用git rm test.txt 之后再git commit –m “remve test.txt”发现我们的本地库里的文件已经被删除了。那么另外一种情况就是我删错了，我想还原回来怎么办呢，使用命令git checkout – test.txt就发现这个文件已经还原了。

## ****（6）远程仓库****

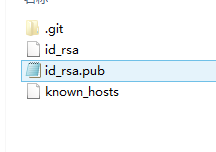
     到目前为止，我们已经学会了如何在本地库和工作区之间进行操作，那这时你就会说我没有看到如何分工协作呢，别着急，远程仓库还帮你解决这个问题。

首先你要点进去<https://github.com/>这个网站，登录你刚才注册的github账号。

ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com)

     把这个邮箱替换成自己的

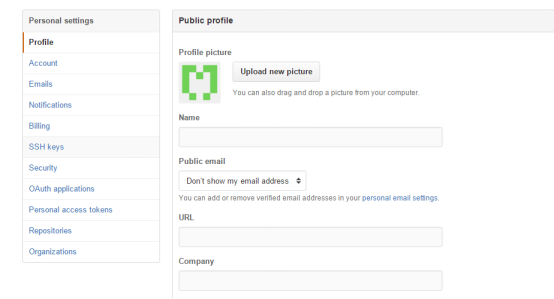
之后一路回车，如果一切顺利的话你去你的C盘下去找你的user文件夹，有的叫用户，你在里面搜索一下有没有一个.ssh的文件夹，把它打开之后，里面有这几个文件



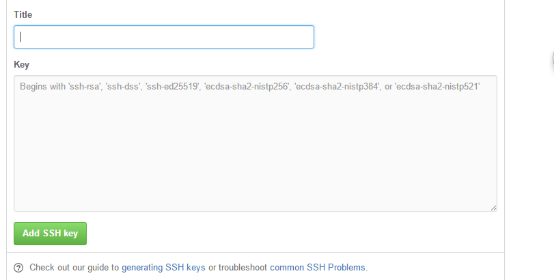
      之后接着打开那个<https://github.com/>网站，登录进去。

IMG_275

     点击右边这个绿色的图表，点开下拉框之后里面有个setting。



    之后会出来这样一个页面，点击左边的ssh keys，



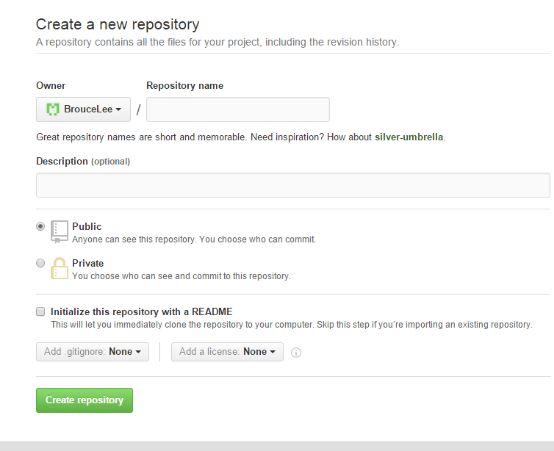
     打开之后是这样的一个页面，随便写入一个title，然后打开你刚才找到的再C盘里的那个id\_rsa.pub.把里面的内容复制到那个key里面去，之后点击添加即可。

之后回到一开始的这个页面，

IMG_278

   点击中间的那个加号

   选择第一个，这就相当于你自己在远程建了一个仓库。



     点击之后，会出来这样一个页面。现在我们在那个名字哪里写上aaa，直接点击创建就好。

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aaa仓库下运行命令：

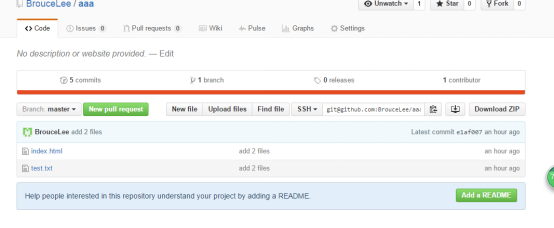
$ git remote add origin git@github.com:BrouceLee/aaa.git

请千万注意，把上面的BrouceLee替换成你自己的GitHub账户名，否则，你在本地关联的就是我的远程库。

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

接着执行命令git push -u origin master

执行完之后，我们发现我们的远程库和本地仓库已经同步了。



     从现在起，只要本地作了提交，就可以通过命令：

$ git push origin master

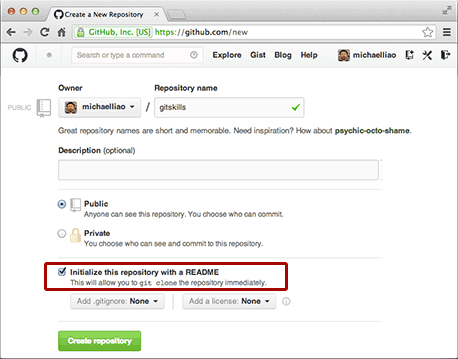
把本地master分支的最新修改推送至GitHub，现在，你就拥有了真正的分布式版本库！

## ****（7）从远程库克隆****

     上次我们讲了先有本地库，后有远程库的时候，如何关联远程库。

现在，假设我们从零开发，那么最好的方式是先创建远程库，然后，从远程库克隆。

首先，登陆GitHub，创建一个新的仓库，名字叫gitskills：



      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：

我们在想要创建项目的地方执行下面的命令

git clone [git@github.com:BrouceLee/gitskills.git](mailto:git@github.com:BrouceLee/gitskills.git)

     现在去你你想要建工程的地方开始，就可以发现大体上已经建好了。

## ****（8）分支管理****

      分支在实际中有什么用呢？假设你准备开发一个新功能，但是需要两周才能完成，第一周你写了50%的代码，如果立刻提交，由于代码还没写完，不完整的代码库会导致别人不能干活了。如果等代码全部写完再一次提交，又存在丢失每天进度的巨大风险。

     现在有了分支，就不用怕了。你创建了一个属于你自己的分支，别人看不到，还继续在原来的分支上正常工作，而你在自己的分支上干活，想提交就提交，直到开发完毕后，再一次性合并到原来的分支上，这样，既安全，又不影响别人工作。

首先我来创建一个分支

执行命令git checkout -b dev 这样就创建好了一个dev分支。

我们来介绍一下，最开始的时候我们的所有操作都是默认在一个叫做master的分支上操作的，包括提交到远程库，等等。关联远程库的时候也是从远程关联master分支。当我们用dev这个分支写完一个项目之后，我们要把这个分支所做的工作向master分支进行靠拢.

我们现在可以通过命令git branch来查看一下所有的分支。

git branch命令会列出所有分支，当前分支前面会标一个\*号。

然后，我们就可以在dev分支上正常提交，比如对index.html做个修改，加上一行



      然后我们执行git add index.html以及 git commit –m “use the dev first”

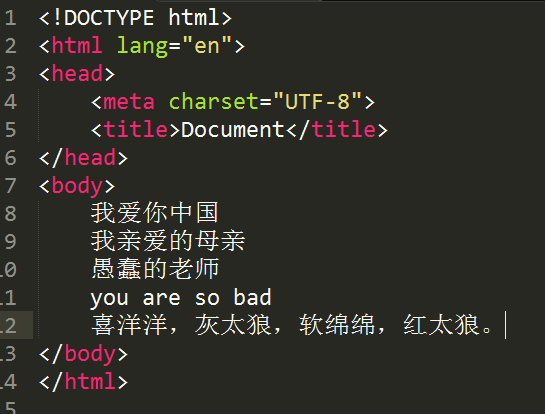
现在我dev的分支工作已经做完，现在我可以切换回master分支。利用命令

git  checkout master

       切换回master分支之后，我们发现index.html里面我们新加的那句话不在了。因为我们刚才是在dev分支上做的提交。现在我们要把分支合到master上，就利用命令 git merge dev就可以了，此时再去查看index.html发现我们在dev里面做的操作已经在master里面了。现在我们可以使用命令git branch –d dev来删除dev这个分支，之后再使用git branch来查看分支发现只有master一个分支了。因为创建、合并和删除分支非常快，所以Git鼓励你使用分支完成某个任务，合并后再删掉分支，这和直接在master分支上工作效果是一样的，但过程更安全。

## ****三、解决冲突****

      现在我们考虑这样一种情况，我们新建了一个分支feature1,利用feature1来进行把index.html里面再加一行，首先切换到feature1这个分支，然后修改index.html,之后提交。



     然后我们切换回master分支，现在我们在master里的index.html里面加一句话,之后进行提交。之后我们按照刚才的那种方式合并两个分支。git merge feature1会发现有冲突出现

IMG_284

此时应该怎么解决呢，此时必须手动解决冲突。

## ****四．分支策略****

在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；

那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

所以，团队合作的分支看起来就像这样：

