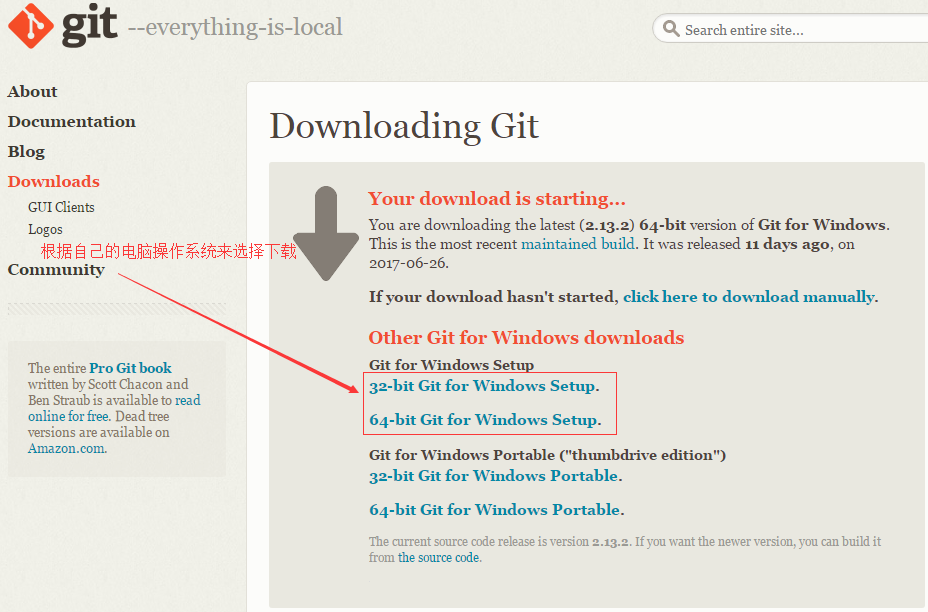
# Git安装和使用

## ****一、下载安装Git****

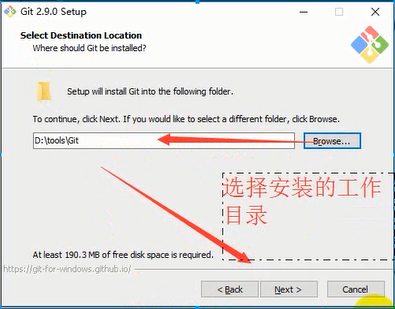
1、下载Git  官方地址为：https://git-scm.com/download/win



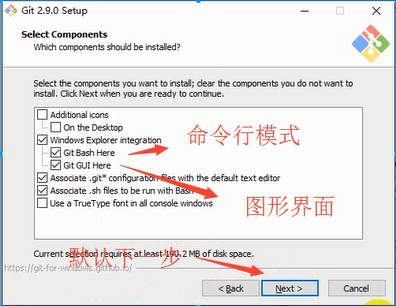
**2、下载完之后，双击安装**



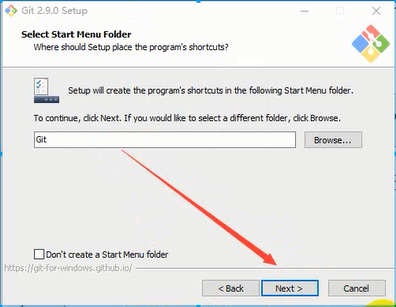
**3、选择安装目录**



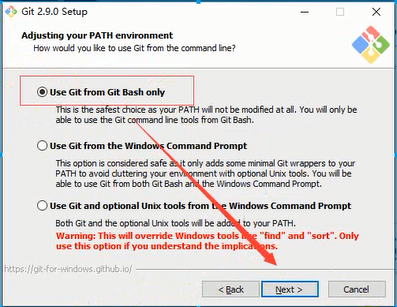
**4、选择组件**



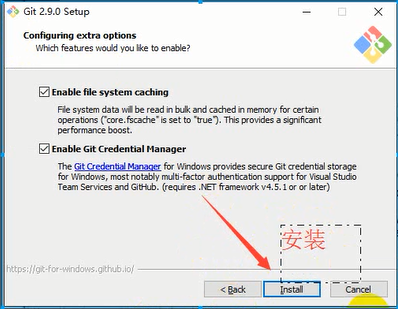
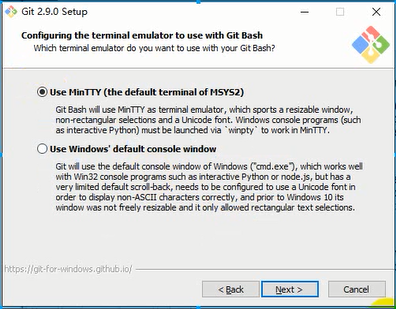
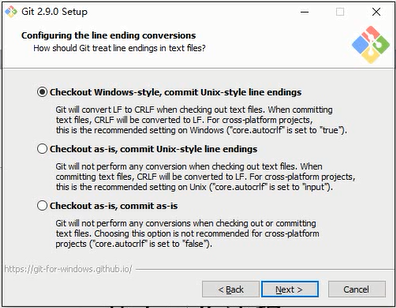
**5、开始菜单目录名设置**



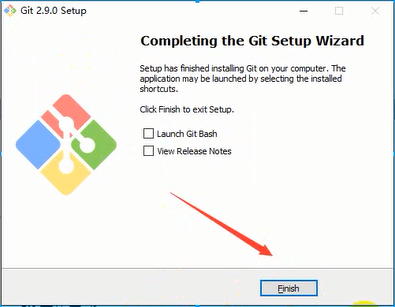
**6、选择使用命令行环境**



**7、以下三步默认，直接点击下一步**



**8、安装完成**



**9、检验是否安装成功**

回到电脑桌面，鼠标右击如果看到有两个git单词则安装成功

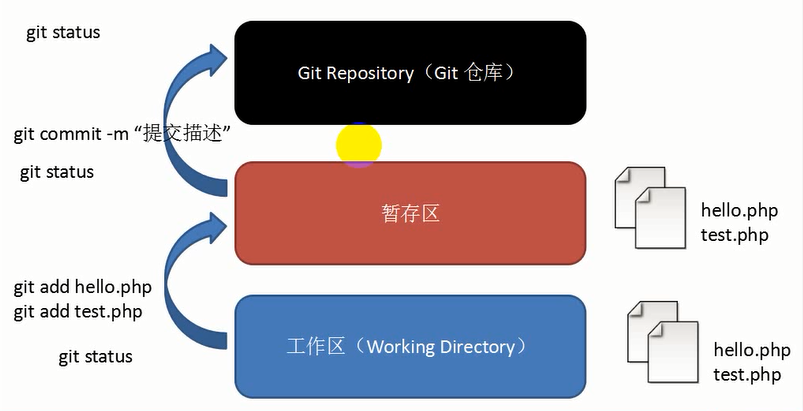


## ****二、Git基本工作流程****

**1、Git工作区域**



**2、向仓库中添加文件流程**



## ****三、Git初始化及仓库创建和操作****

**1、Git安装之后需要进行一些基本信息设置**

　　a、设置用户名：git config --global  user.name 'github上注册的用户名';

　　b、设置用户邮箱：git config --global user.email '注册时候的邮箱';

注意：该配置会在github主页上显示谁提交了该文件

 　c、配置ok之后，我们用如下命令来看看是否配置成功

　　git config --list

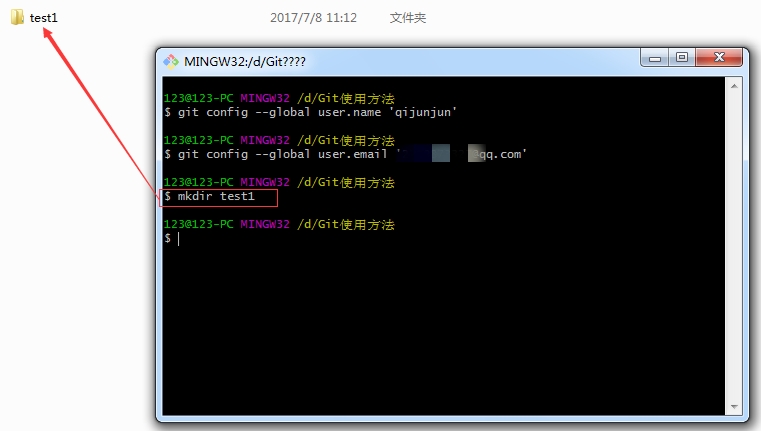
注意：git config --global 参数，有了这个参数表示你这台机器上所有的git仓库都会使用这个配置，当然你也可以对某个仓库指定不同的用户名和邮箱

**2、初始化一个新的git仓库**

**a、创建文件夹**

　　　　方法一：可以鼠标右击-》点击新建文件夹test1

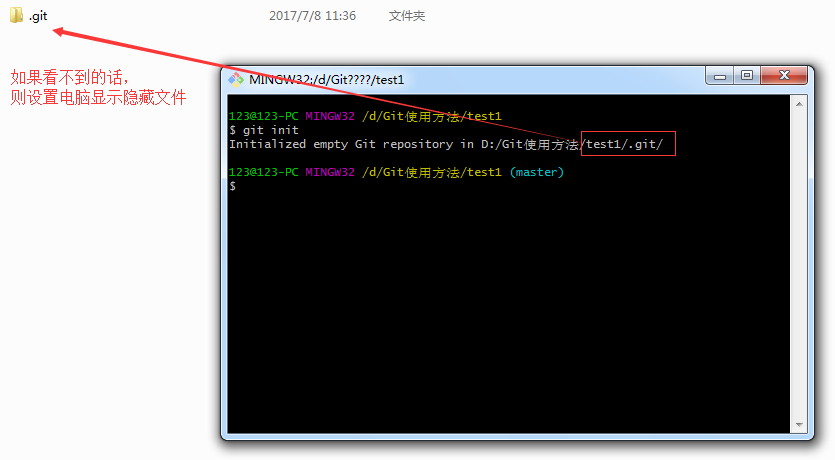
　　　　方法二：使用git新建：$ mkdir test1



**b、在文件内初始化git（创建git仓库）**

　　　　方法一：直接输入 $ cd test1

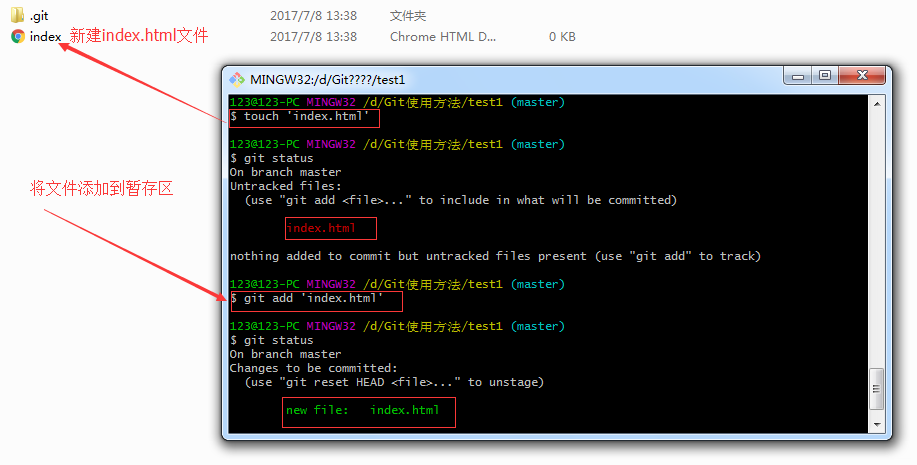
　　　　方法一：点击test1文件下进去之后-》鼠标右击选择Git Bash Here->输入$ git int

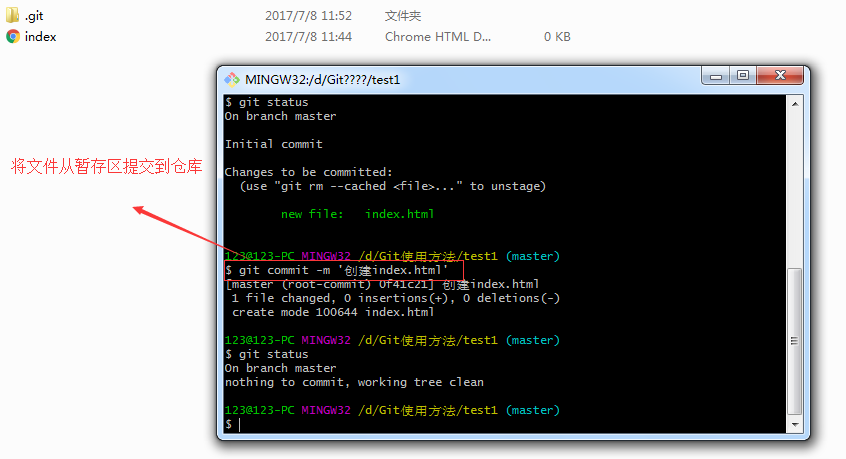


**3、向仓库中添加文件**

**方法一**：**用打开编辑器新建index.html文件**

**方法二：使用git命令。$ touch '文件名'（新建文件），然后把文件通过$ git add '文件名'添加到暂存区，最后提交操作**

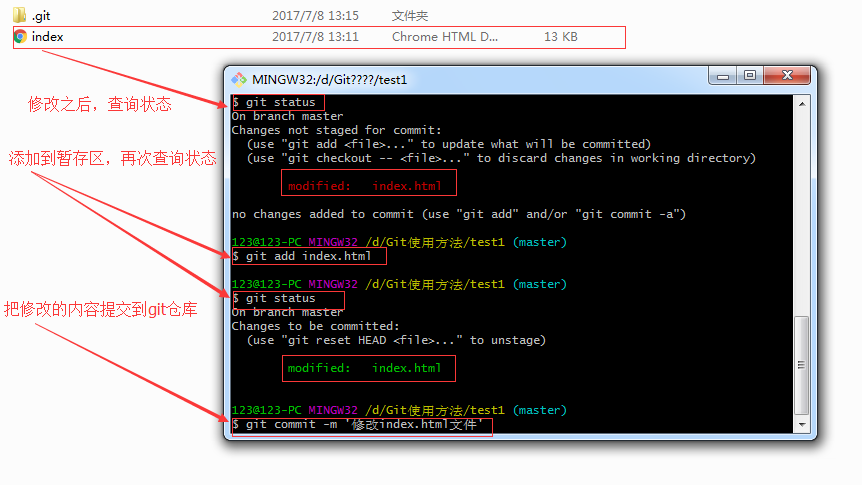




**4、修改仓库文件**

**方法一**：**用编辑器打开index.html进行修改**

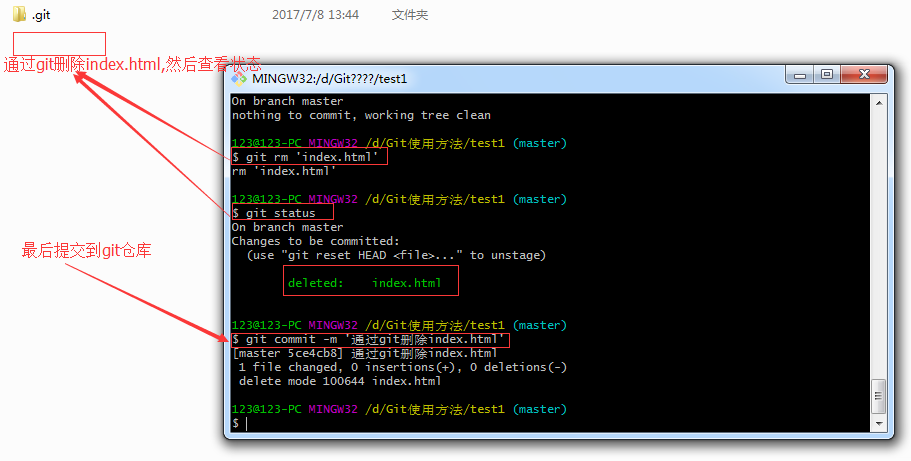
**方法二：使用git命令。$ vi '文件名'，然后在中间写内容，最后提交操作**



**5、删除仓库文件**

**方法一：在编辑器中直接把要删除的文件删除掉**

**方法二：使用git删除：$ git rm '文件名'，然后提交操作**



## 四、本地仓库和远程仓库同步

1、假设你已经创建了本地创库“test”，并且是第一次向github提交更新。a.$ git remote add origin <https://github.com/vitah-lin/test.git>

// //连接远程github项目地址

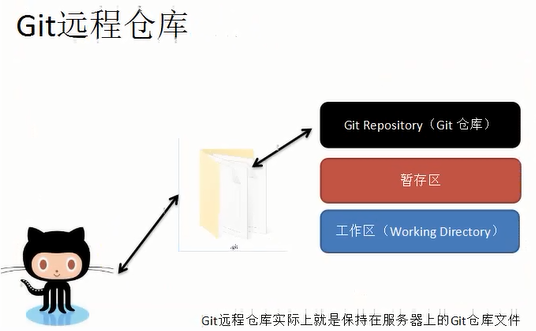
b.$ git push -u origin master //将本地项目更新到github项目上去

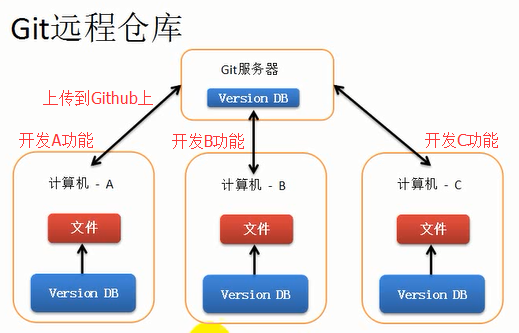
注意：完成第一次提交后，a步骤就不需要。

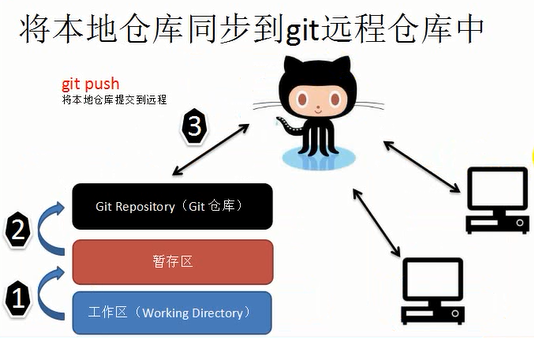
## ****五、Git管理远程仓库****

**1、使用远程仓库的目的**：**备份、实现代码共享集中化管理**

**Git远程仓库实际上就是保持在服务器上的git仓库文件**





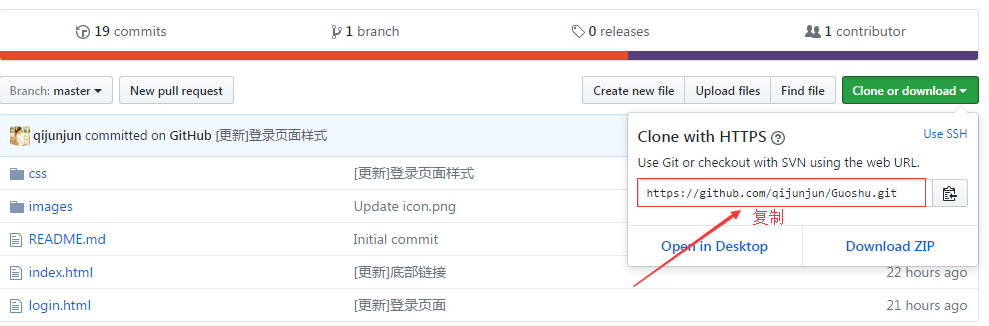


## ****六、Git克隆操作****

**目的：将远程仓库（github上对应的项目）复制到本地**

**1、代码：**git clone 仓库地址

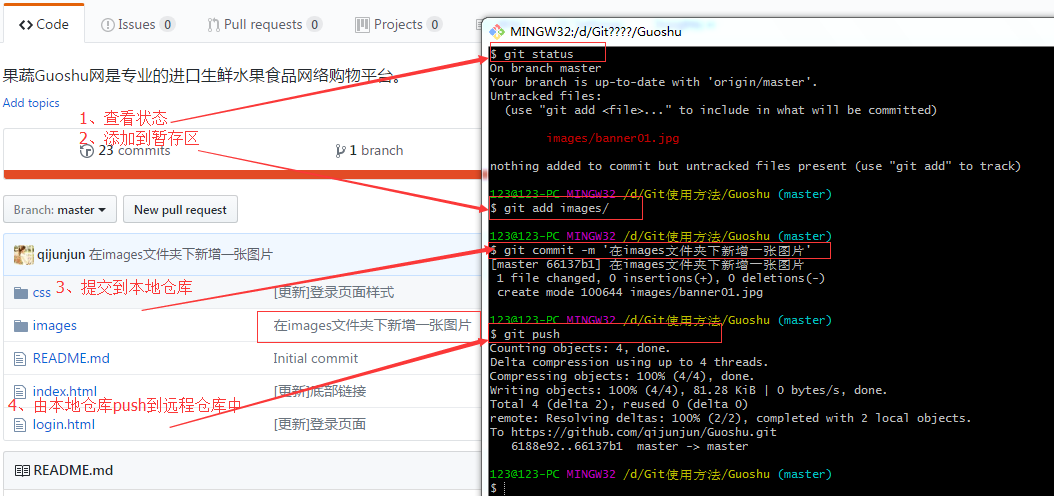
仓库地址由来如下：



**2、克隆项目**

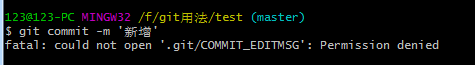


3、将本地仓库同步到git远程仓库中：git push

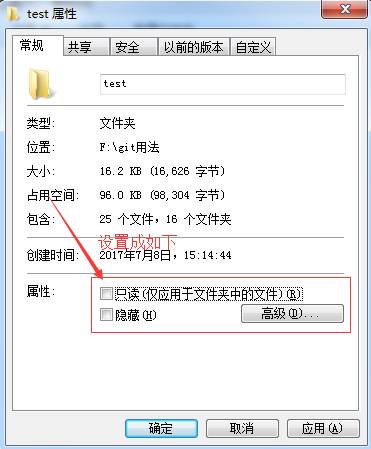
****

**期间出现错误的情况有：**

**a、出现提交错误**



　　解决：这是通过Git GUI进行提交时发生的错误，由 .git 文件夹中的文件被设为“只读”所致，将 .git 文件夹下的所有文件、文件夹及其子文件的只读属性去掉即可。



**b、如果出现无法同步或没有权限，解决方法如下：**

**用户名和密码一定要和github上的一致。**



# 配置用户名和邮箱

安装完成之后，在开始菜单栏会出现Git Bash，点击Git Bash。会弹出一个类似cmd命令行的窗口，就说明Git已经安装成功了，安装成功之后我们还有一件事情，配置用户名(username)和邮箱(email)，先配置然后一会在解释说为什么需要配置用户和邮箱，因为我的机器上之前已经安装过Git，并且也配置过username和email，在这里先使用一个命令，查看Git当前所有的配置git config --list:



如果你是第一次安装的话，如何配置username和email呢，可以使用下列命令:

git config --global user.name "username"

git config --global user.email "email"

但是如果你要是想重新设置用户名和邮箱也可以使用上面的那两条命令:



如果你想要在原有的基础上对原有设置的用户名和邮箱进行修改的话可以使用以下命令:

git config --global -e  查看当前用户名和邮箱:

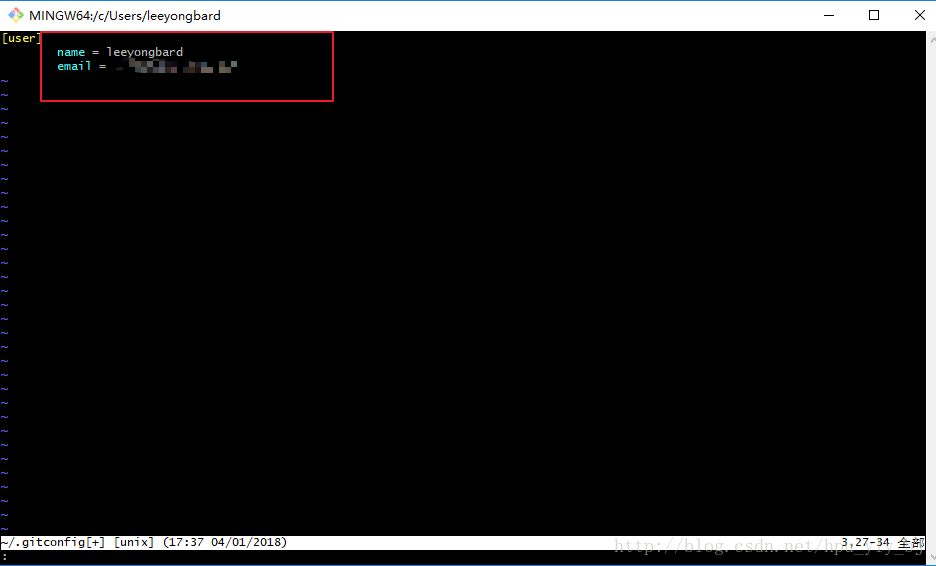
i  命令进入编辑状态

esc 推出编辑

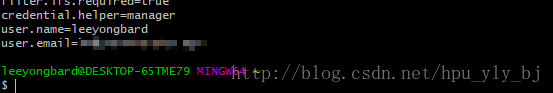
然后再同时点击shift+；号

在输入wq就ok了

如下进行修改:



修改之后的用户名和邮箱：



那下面开始说一下为什么要配置用户名和邮箱

因为Git是分布式版本控制系统，所以说每一个机器都要自报家门(你的邮箱地址和用户名)

在这里还有一个需要注意的地方就是我们使用git config命令的时候后面都是--global参数，这个参数是全局的意思，表示的意思是在你的这台机器上所有的Git仓库使用的都是这个配置，当然我们可以为不同的仓库指定不同的用户名和邮箱

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

作者：leeyongbard

来源：CSDN

原文：https://blog.csdn.net/hpu\_yly\_bj/article/details/78972610

版权声明：本文为博主原创文章，转载请附上博文链接！

# 添加ssh密钥

1、配置github的ssh密钥:

(1)打开Git Bash查看电脑上是否已经存在SSH密钥：

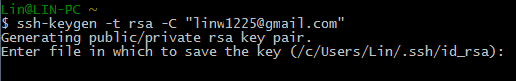
输入 cd ~/.ssh

https://images0.cnblogs.com/i/335529/201405/122329362033539.png

若如上图显示无法找到该文件则要创建新的ssh key;

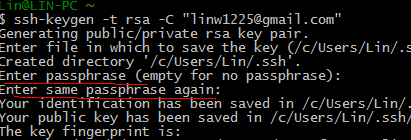
(2)创建新的ssh key:

输入 ssh-keygen -t rsa -C "your\_email@youremail.com"



执行这条命令会如上图提示文件保存路径，可以直接按Enter，

然后提示输入 passphrase（密码），输入两次（可以不输直接两次Enter），



然后会在 .ssh 目录生产两个文件：id\_rsa和id\_rsa.pub

用记事本打开.ssh目录下的id\_rsa.pub文件，复制里面的内容；

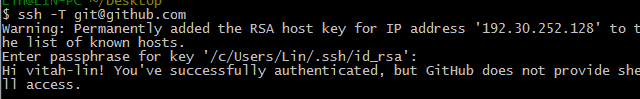
2、复制ssh key到github：

On the GitHub site Click “Account Settings” > Click “SSH Keys” > Click “Add SSH key”

打开github网站，点击右上角扳手图标，然后点击左边菜单的 ssh key， 然后右边页面的 add ssh key，将复制的内容粘贴到github的key中，title可以不填，直接保存即可。

3、测试 ssh 链接 github：

输入 ssh -T git@github.com



出现Successfully就OK；

4、设置自己的git信息：

输入

git config --global user.name "Firstname Lastname" （此处name可修改也不是用于登录github的登录名）

git config --global user.email "your\_email@youremail.com"

设置自己的git信息即完成安装和设置，可以输入git config --list查看自己的git信息。

# github常见操作和常见错误

一、如果输入$ git remote add origin [git@github.com:djqiang（github帐号名）/gitdemo（项目名）.git](mailto:git@github.com:djqiang/gitdemo.git)

提示出错信息：fatal: remote origin already exists.

解决办法如下：

1、先输入$ git remote rm origin

2、再输入$ git remote add origin [git@github.com:djqiang/gitdemo.git](mailto:git@github.com:djqiang/gitdemo.git)

就不会报错了！

3、如果输入$ git remote rm origin 还是报错的话，error: Could not remove config section 'remote.origin'. 我们需要修改gitconfig文件的内容

4、找到你的github的安装路径，我的是C:\Users\ASUS\AppData\Local\GitHub\PortableGit\_ca477551eeb4aea0e4ae9fcd3358bd96720bb5c8\etc

5、找到一个名为gitconfig的文件，打开它把里面的[remote "origin"]那一行删掉就好了！

6、如果输入$ ssh -T [git@github.com](mailto:git@github.com)

出现错误提示：Permission denied (publickey).因为新生成的key不能加入ssh就会导致连接不上github。

解决办法如下：

a、先输入$ ssh-agent，再输入$ ssh-add ~/.ssh/id\_key，这样就可以了。

  b、如果还是不行的话，输入ssh-add ~/.ssh/id\_key 命令后出现报错Could not open a connection to your authentication agent.解决方法是key用Git Gui的ssh工具生成，这样生成的时候key就直接保存在ssh中了，不需要再ssh-add命令加入了，其它的user，token等配置都用命令行来做。

c、最好检查一下在你复制id\_rsa.pub文件的内容时有没有产生多余的空格或空行，有些编辑器会帮你添加这些的。

7、如果输入$ git push origin master

提示出错信息：error:failed to push som refs to .......

解决办法如下：

a、先输入$ git pull origin master //先把远程服务器github上面的文件拉下来

  b、再输入$ git push origin master

  c、如果出现报错 fatal: Couldn't find remote ref master或者fatal: 'origin' does not appear to be a git repository以及fatal: Could not read from remote repository.

d、则需要重新输入

$ git remote add origin[git@github.com:djqiang/gitdemo.git](mailto:git@github.com:djqiang/gitdemo.git)

二、使用git在本地创建一个项目的过程

  $ makdir ~/hello-world    //创建一个项目hello-world  
  $ cd ~/hello-world       //打开这个项目  
  $ git init             //初始化   
  $ touch README  
  $ git add README        //更新README文件  
  $ git commit -m 'first commit'//提交更新，并注释信息“first commit”  
  $ git remote add origin [git@github.com:defnngj/hello-world.git](mailto:git@github.com:defnngj/hello-world.git)     //连接远程github项目    
  $ git push -u origin master  //将本地项目更新到github项目上去

三、gitconfig配置文件

Git有一个工具被称为git config，它允许你获得和设置配置变量；这些变量可以控制Git的外观和操作的各个方面。这些变量可以被存储在三个不同的位置：

1./etc/gitconfig 文件：包含了适用于系统所有用户和所有库的值。如果你传递参数选项’--system’ 给 git config，它将明确的读和写这个文件。

2.~/.gitconfig 文件 ：具体到你的用户。你可以通过传递--global 选项使Git 读或写这个特定的文件。

3.位于git目录的config文件 (也就是 .git/config) ：无论你当前在用的库是什么，特定指向该单一的库。每个级别重写前一个级别的值。因此，在.git/config中的值覆盖了在/etc/gitconfig中的同一个值。在Windows系统中，Git在$HOME目录中查找.gitconfig文件（对大多数人来说，位于C:\Documents and Settings\$USER下）。它也会查找/etc/gitconfig，尽管它是相对于Msys 根目录的。这可能是你在Windows中运行安装程序时决定安装Git的任何地方。

配置相关信息：

2.1　当你安装*Git*后首先要做的事情是设置你的用户名称和*e-mail*地址。这是非常重要的，因为每次*Git*提交都会使用该信息。它被永远的嵌入到了你的提交中：

*$ git config --global user.name "John Doe"*

*$ git config --global user.email*[*johndoe@example.com*](mailto:johndoe@example.com)

2.2你的编辑器*(Your Editor)*

　　现在，你的标识已经设置，你可以配置你的缺省文本编辑器，*Git*在需要你输入一些消息时会使用该文本编辑器。缺省情况下，*Git*使用你的系统的缺省编辑器，这通常可能是*vi*或者*vim*。如果你想使用一个不同的文本编辑器，例如*Emacs*，你可以做如下操作：

*$ git config --global core.editor emacs*

*2.3 检查你的设置(Checking Your Settings)*

　　如果你想检查你的设置，你可以使用*git config --list*命令来列出*Git*可以在该处找到的所有的设置*:*

*$ git config --list*

*你也可以查看Git认为的一个特定的关键字目前的值，使用如下命令 git config {key}:*

*$ git config user.name*

*2.4 获取帮助(Getting help)*

　　如果当你在使用*Git*时需要帮助，有三种方法可以获得任何git命令的手册页*(manpage)*帮助信息*:*

*$ git help <verb>*

*$ git <verb> --help*

*$ man git-<verb>*

　　例如，你可以运行如下命令获取对*config*命令的手册页帮助*:*

*$ git help config*