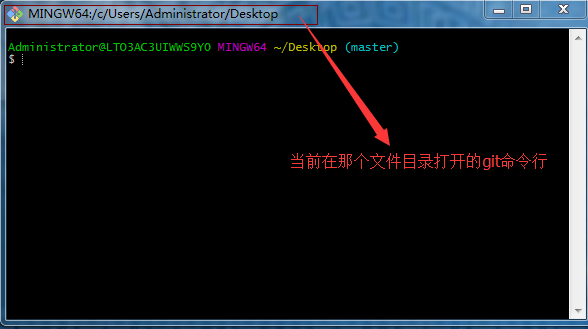
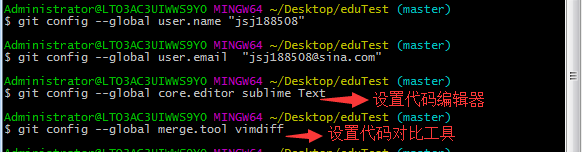
**一、安装和配置**

下载网址：<https://git-for-windows.github.io/>学习参考：[http://www.runoob.com/w3cnote/git-five-minutes-tutorial.html   
http://blog.jobbole.com/78960/](http://www.runoob.com/w3cnote/git-five-minutes-tutorial.html)

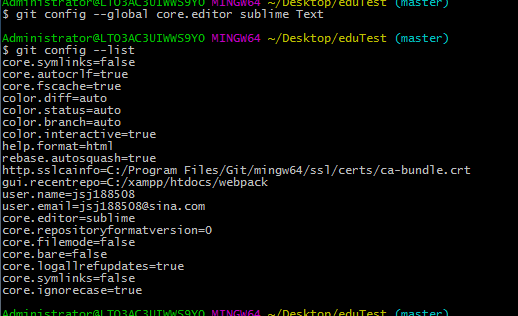
1. 需要从网上下载一个，然后进行默认安装即可。
2. 安装完成后，在桌面的空白处右击：选择 Git Bash Here 弹出git命令行；

  
3、安装完成后，还需要进一步的设置github用户名+注册邮箱+编辑器，在命令行输入如：

因为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。

Vimdiff 使用教程网址：https://www.ibm.com/developerworks/cn/linux/l-vimdiff/

**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的不同的用户名和邮箱。

设置完后可以试用：git config --list 查看设置信息  


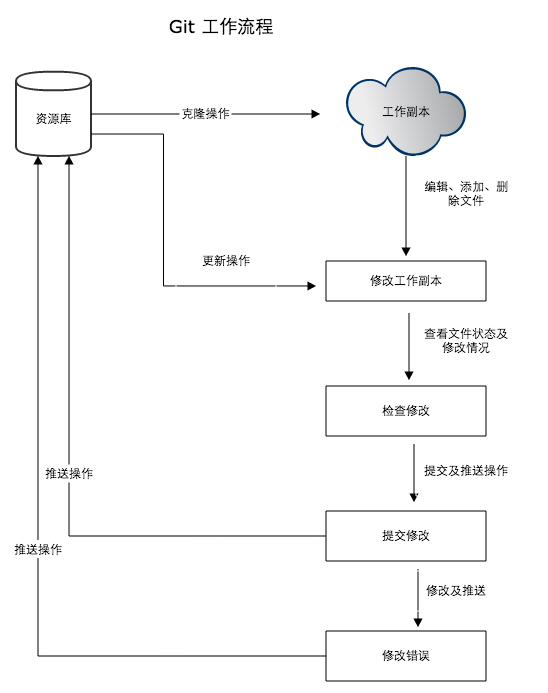
1. **了解工作流程**

**一般工作流程如下：**

1. 克隆 Git 资源作为工作目录。
2. 在克隆的资源上添加或修改文件。
3. 如果其他人在远程上修改了一些代码，可以使用(git pull)更新资源。
4. 在提交前，使用(git status)查看修改后的文件状态。或使用(git diff)查看修改文件的内容
5. 如果当前分支不是提交的分支，可以使用(git checkout“分支名”)切换分支
6. 以上完成好了可以使用(git add .)将修改后的文件添加到本地仓库的分支中去。
7. 然后使用(git commit -m “提交的注释”)命令提交修改后的文件。
8. 提交成功后可以使用(git log或git reflog)查看提交的日志信息;
9. 在修改完成后，如果发现错误，可以使用(git reset --hard “指定的版本”)撤回提交

(10)在使用git reset --hard \*\*\* **回滚代码版本较低时**,可以使用git reflog 查看到所有git分支 提交的代码版本号**(包括已经被删除的commit 记录和git reset的操作**)，并使用git reset --hard \*\*\* 再次回滚到高版本的代码。

(11)排查错误，修改错误，并再次提交



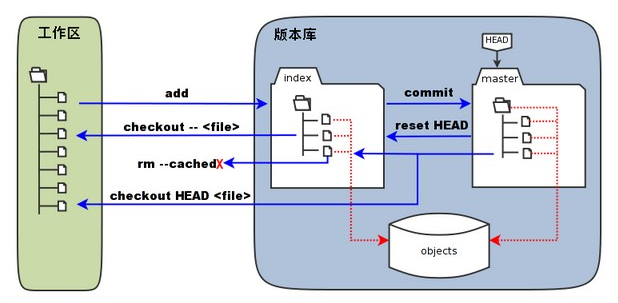
**三、关键件词解析：Git 工作区、暂存区stage (git add)、本地仓库(git commit)**

工作区：就是你在电脑里能看到的目录。

版本库：工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

暂存区：英文叫stage, 或index。一般存放在隐藏的git目录下的index文件（.git/index）中， 所以我们把暂存区有时也叫作索引（index）。

**下面这个图展示了工作区、版本库中的暂存区和本地仓库之间的关系：**



图中左侧为工作区，右侧为版本库。

在版本库中标记为 "index" 的区域是暂存区（stage, index）；标记为Head的区域为本地仓库的一个游标，master是本地仓库的主干分支。图中我们可以看出此时 "HEAD" 实际是指向 master 分支的一个"游标"。

图中的 objects 标识的区域为 Git 的对象库，实际位于 ".git/objects" 目录下，里面包含了创建的各种对象及内容。

**git add fileName.filetype 或git add –all 或git add .**

当对工作区修改（或新增）的文件执行 "git add" 命令时，暂存区(stage)的目录树被更新，同时工作区修改（或新增）的文件内容被写入到对象库中的一个新的对象中，而该对象的ID被记录在版本库的文件索引中。

**git commit -m “fileName.filetype”**

当执行提交操作（git commit）时，版本库中的目录树写到本地仓库中，**当前的分支**会做相应的更新。

**git reset --hard head^**

当执行 "git reset HEAD" 命令时，版本库的目录树会被本地仓库的**(当前的分支**)文件目录重写，但是工作区不受影响。

当执行 "**git rm --cached <file>**" 命令时，**会直接从本地的暂存区删除文件**，工作区则不做出改变(忽略文件的提交)。

当执行 "git checkout ." 或者 "git checkout -- <file>" 命令时，会用暂存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到暂存区的改动。

**git rm –r --cached <file> 直接从远程仓库的暂存区删除文件**

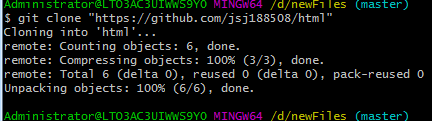
当执行 "git checkout HEAD ." 或者 "git checkout HEAD <file>" 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换暂存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除暂存区中未提交的改动。

**四、基础操作**

**（1）文件克隆 git clone url(仓库地址) directory(本地目录)**

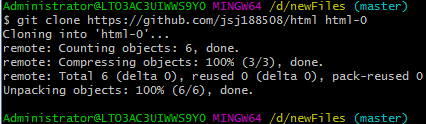
1、例如执行：git clone "https://github.com/jsj188508/html" 命令

-->将远程上的html仓库，克隆在当前的文件中克隆html

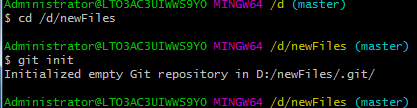


2、例如执行：git clone "https://github.com/jsj188508/html" html-0 命令

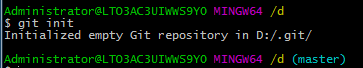
-->将远程上的html仓库，克隆在当前的文件中克隆html-0

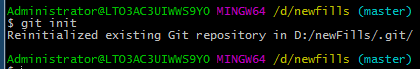


1. **创建版本库**

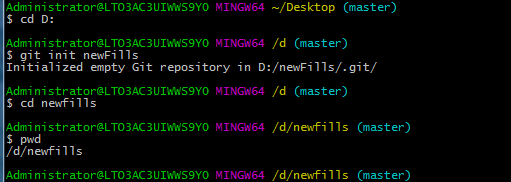
**一种：**①使用cd命令进入文件目录；②使用git init直接初始化这个项目  
 

使用git init 初始化本地仓库

显示这个表明管理成功

显示这个表明已经管理了

**二种：**①使用cd命令进入文件目录；②使用git init newFills直接创建一个newFills新的本地仓库；  
 ③使用cd进入仓库

  
当前文件目录   
显示这个Reinitialized existing Git repository in D:/newFills/.git/ 表明本地仓库创建成功  
显示这个Reinitialized existing Git repository in D:/newFills/.git/表明本地仓库已经创建过了

pwd 命令是用于显示当前的目录。**注意：如果当前目录不是仓库目录 应当使用cd 命令进入仓库**

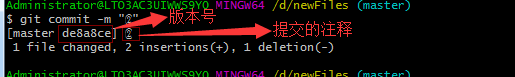
**（3）使用**git add --all**或**git add <file>**或git add . 添加到本地仓库的分支中**

--->先创建01.txt文件 内容：0111111

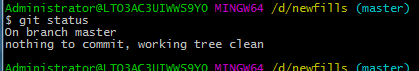
显示这01.txt添加成功

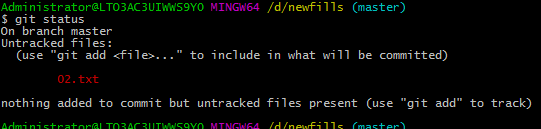
显示这个文件不存在或文件已经添加

**（4）再使用git commit -m “提交注释”将01.txt文件提交到仓库的master（默认）分支上**

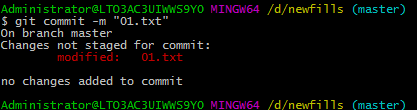
**例如执行git commit -m “②”命令，** 

**（5）再使用git status 查看修改的或者是未提交的文件**

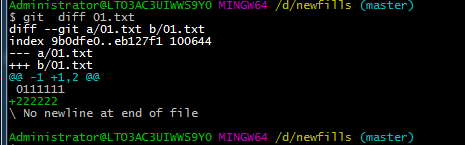
显示这个表明文件都已经提交了。

显示这个表明还有文件没有提交

这时我们再次提交02.txt文件后就会显示没有将02.txt文件添加到暂存区内这时就需要再次使用git add 命令， 将02.txt文件添加到暂存区内后，才可以提交到仓库中的分支上。

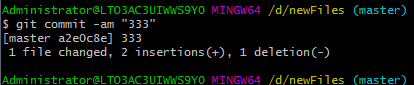
当修改01.txt文件后,再次使用git commit提交; 就会显示： 

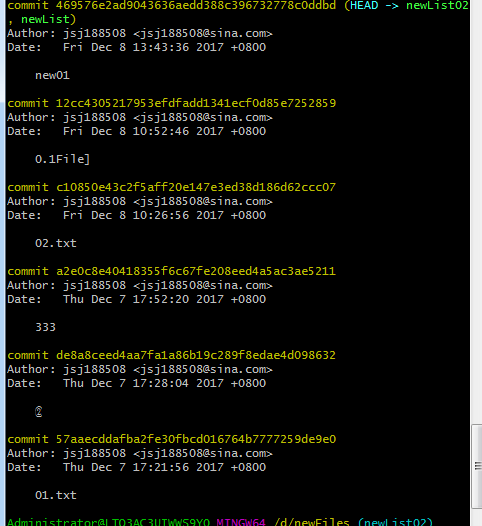
意思是修改后的01文件，需要再次的使用git add 将文件提交到暂冲区内，才可以提交到仓库中的分支上。

**（6）这时再使用git diff 01.txt 或者git diff查看文件修改的 有哪些内容** 

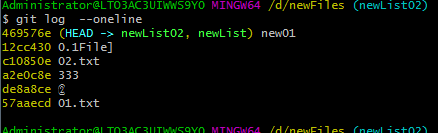
**（7）修改过的文件想提交（注意不是新增文件）**

**如果想跳过git add命令，就可以用git commit -am “提交注释”进行直接提交到仓库的分支上**



**(8)使用**git reset --hard**命令版本回退**  
 ①接着上面的步走，再使用git log 查看所有提交的记录-----由于多次提交，所以历史记录比较多 

或使用git log --oneline查看简洁的版本所有记录



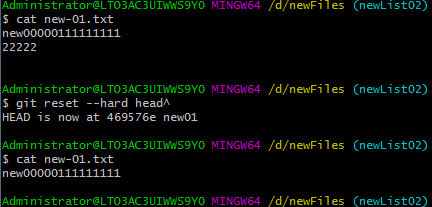
或者使用git log --author=Linus --oneline -5 查看指定用户提交的前5次记录

或者使用git log --oneline --before={2017-12-07} --after={2017-12-08} 查看那一时间段的记录  
②使用git reset --hard head^ 命令让修改的文件版本回退到上次修改   
 （比如：最后一次给文件01.txt文件，添加了一行，这次版本回退,就会将上次添加的一行删掉）。  
 此外版本会退的命令还有：  
 git reset --hard head^^ 回退到上上个版本

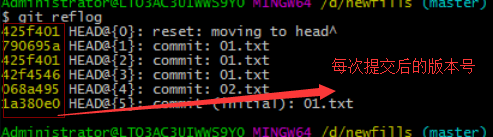
git reset --hard head^^^ 回退到上上上个版本

----------- 以此类推

git reset  –hard HEAD~100 即可。未回退之前的01.txt内容如下

例如：git reset --hard head^回复到上一个版本  
 

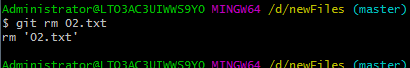
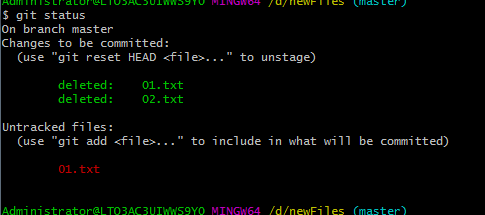
命令解析： cat <file> 查看某个文件内容

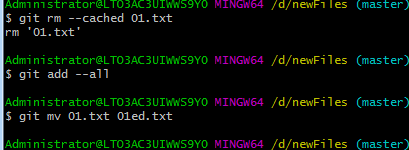
**③使用git reflog查看版本号**   
 ④使用git reset --hard <版本号> 命令，可以恢复某一个的版本  
 

1. **使用git rm <file>命令将文件删除**

**①**如果只是简单地从工作目录中手工删除文件，运行 **git status** 时就会在 **Changes not staged for commit** 的提示。

例如：执行git rm 02.txt 命令可以从 Git 中移除02.txt文件，

②使用**git rm --cached <file>**命令可以把文件从暂存区域移除，但仍然保留在当前工作目录中  
 例如执行git rm --cached 01.txt命令后，再执行git status命令，可以查看文件的状态。   
 **（10）使用git mv 命令用于移动或重命名一个文件、目录、软连接** 例如：执行git mv 01.txt 01ed.txt命令将01.txt文件名改为01ed.txt

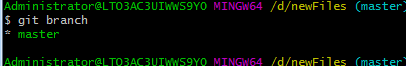
  
  **注意：在执行git mv 命令时，一定要先将文件添加到暂存区，才可以执行此命令**

1. **分支管理**

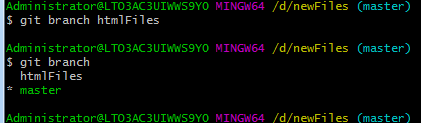
**https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/001375840038939c291467cc7c747b1810aab2fb8863508000**

**解析git分支：当前项目需要开发新的功能时，就需要一个新的B分支（基于master的分支）来管理新功能的代码。当功能开发完成后，可以合并到主分支master中；若功能还没有开发完成又有新功能急于研，可以暂且搁置B分支中的代码并使用**git stash功能进行保存**，后再建一个新的C分支（基于master的分支）来管理新功能的代码。当功能开发完成后，可以合并 到主分支master中去进行发布新的版本（可删除c分支），这时可以回头开发B分支中继续研发B 分支中的功能。**

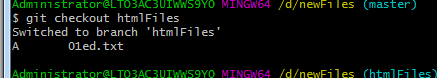
1. **使用git branch 列出本地有那些分支**



1. **使用git branch <分支名> 命令新建分支**

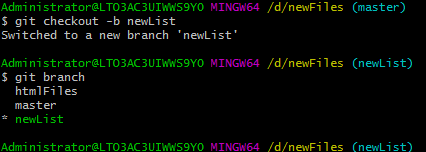


1. **使用git checkout <分支名>命令切换分支**

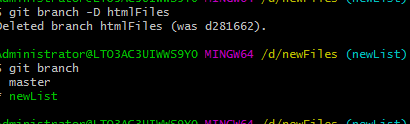
  
 **注意：当你切换分支的时候，git会用该分支的最后提交的快照替换你的工作目录的内容，**

**所以多个分支不需要多个目 录。合并分支命令git merge**

1. **结合git branch <分支名>和git checkout <分支名> 形成git checkout -b <分支名>命令  
    意思是:基于当前的分支，复制一个新的分支，并且换到新的分支上；**



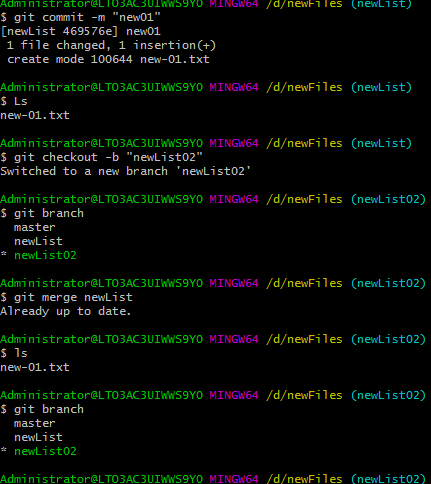
1. **使用git branch -D <要删除的分知名> 删除分支**



1. **使用git push origin --delete <要删除的远程分知名> 删除远程分支**

**再用git branch -a 进行查看远程分支**

**或使用git fetch -p 进行扫描远程的仓库**

1. **使用git merge <等待与当前分支合并的分知名> 命令 —— 合并分支**   
    **命令解析：Ls 查看当前分支里有哪些文件和文件夹**

**六、使用ssh-keygen配置本地仓库与gitHub远程仓库的关系**

**注意：①本地仓库与github远程仓库建立关联后，其他的本地仓库就无法与github中已有旧的仓库进行关联在一起**

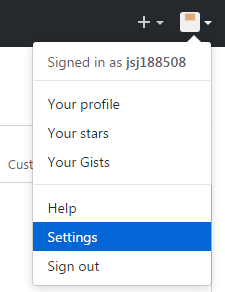
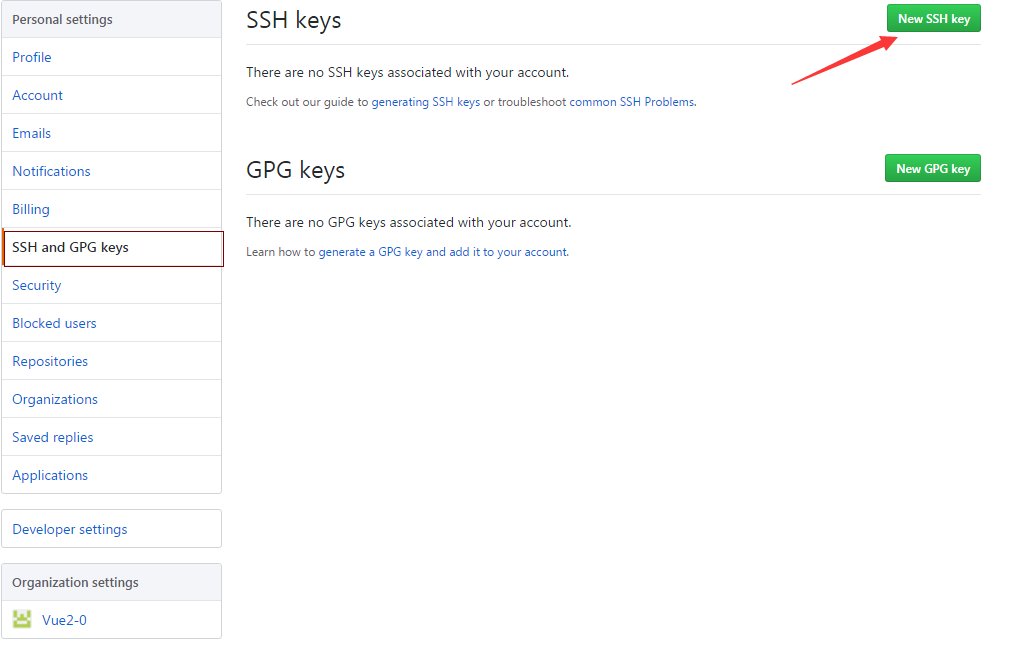
1. **使用ssh-keygen -t rsa -C “邮箱”命令，配置本地用户的ssh-keygen**

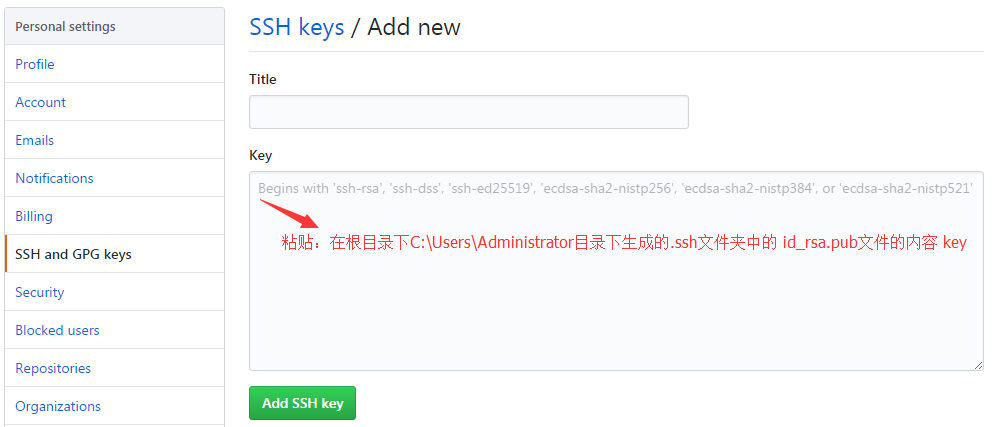
**用途：用于本地git与github链接用的密钥key**

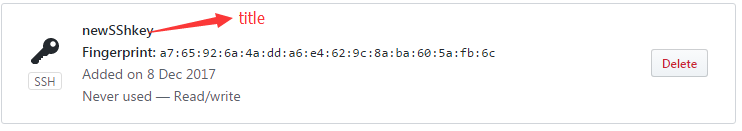
**①会在C:\Users\Administrator（根目录下）目录下生一个.ssh文件夹，进去，打开 id\_rsa.pub，复制里面的内容（key）**

**或：在项目文件夹的根目录下右击选择 Git GUI Here 点击里面的Help 选择Show SSH Key 后，进行复制**

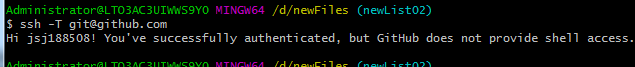
**②回到 github 上，进入 Account => Settings（账户配置）**

①②

③左边选择 SSH and GPG keys，然后点击 New SSH key 按钮,title可以随便填，在key框内粘贴上面复制的key  
 



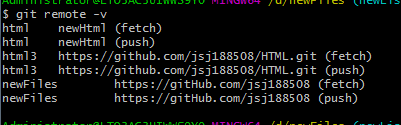
④使用**ssh -T** [**git@github.com 命令让本地的git链接远程gitHub**](mailto:git@github.com命令链接远程gitHub) -----------》yes

  
也可以使用ssh -T -v git@github.com命令查看详细信息

1. **使用git remote add [仓库名] [url]命令添加新的远程仓库**

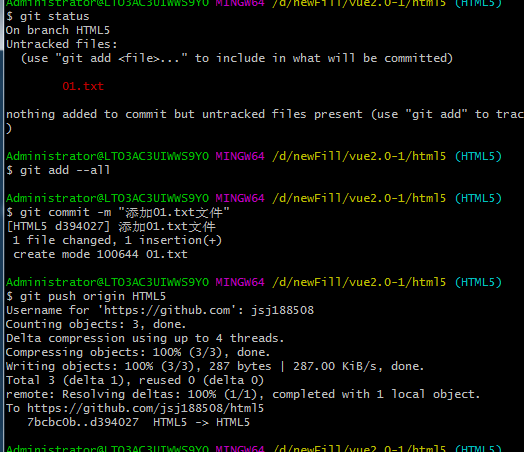


1. **使用git remote -v列出远程仓库分支详细信息**



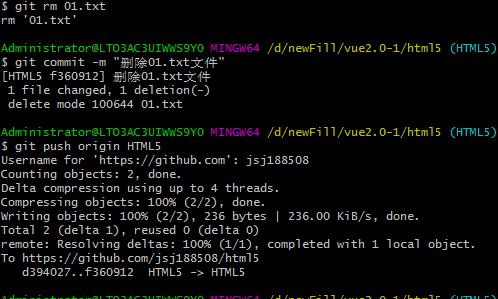
1. **使用git push -u <远程仓库名> <本地仓库的一个分支名>命令将本地仓库的一个分支提交到远程仓库注意：执行git push 命令，让你输入gitHub账号和密码**

比如：在本地添加一个01.txt文件，想要同步到github上；需要执行以下命令

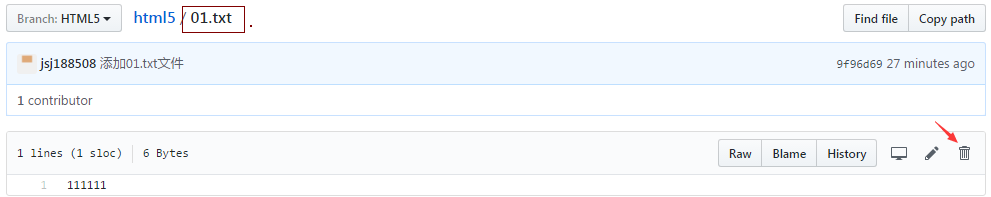




比如：在本地删除一个01.txt文件，想要同步到github上；需要执行以下命令

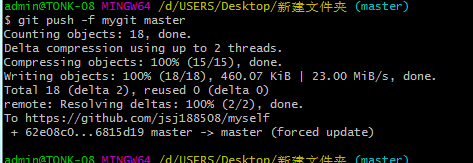


或：在github账户上的这个里可以删除添加的文件



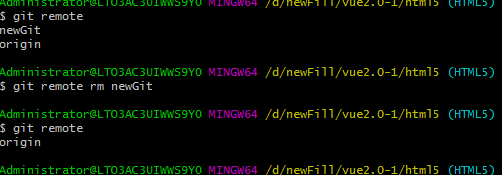
1. **使用git push -f mygit master -->会将本地的项目强制更新到远程的github仓库中，**

**并覆盖或删除有差异的文件**



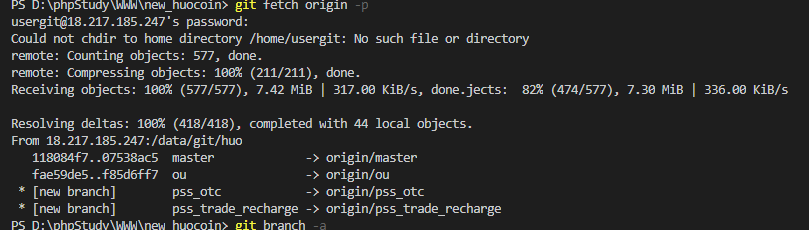
**注意：如果使用这个将会删除git仓库中原有的内容，再进行更新。**

**（6）使用git remote rm <仓库名>命令删除远程仓库**

  
**（7）使用git fetch命令提取远程仓库的分支到本地仓库的分支**

**①git branch –a 查看远程仓库有哪些分支**

**②如果，没有查看到已有的远程仓库分支，可以使用git fetch origin -p 扫描远程仓库中的分支**

****

**③再进行提取远程仓库中的相应的分支git fetch origin <远程仓库名>:<提取到的分支> 提取远程仓库中的代码**

**④可以使用git branch 查看本地仓库有哪些分支  
六git服务器搭建**

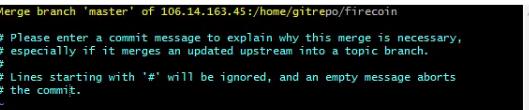
**参考网址：**

[**https://jingyan.baidu.com/article/3a2f7c2e16ca2a26afd611e2.html**](https://jingyan.baidu.com/article/3a2f7c2e16ca2a26afd611e2.html)**；**

**http://www.runoob.com/git/git-server.html**

**七、git 命令提交常出现的错误：**

**1、使用git pull拉取代码出现的错误**

****

**在这时，应该输入:wq(不是大写的WQ)推出当前模式；**

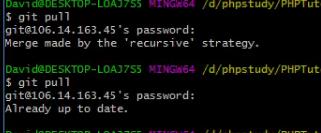
****

**若时：wq退不出当前模式，可以先按Esc键，再输入：wq**

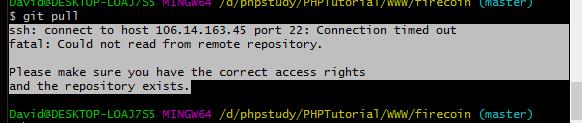
**2、标红提示--->提交到远程仓库的主干master**

****

**3、Already up to date 表示代码在本地拉去更新成功**

****

**4、表示请求超时，-出现的原因-看网络太慢或断网**

****

**5、当使用git diff查看修改的文件后 可以按q键推出**



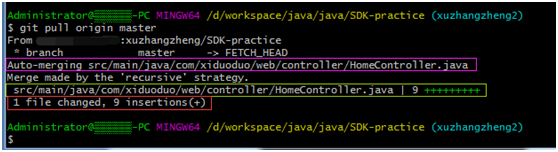
1. **在执行git pull从远程仓库中拉取代码导致**

**远程仓库中拉取代码与本地代码有冲突--而终止git pull命令**

**注意：**

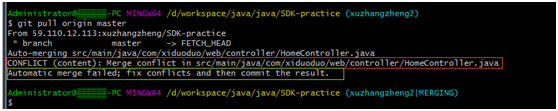
**每次使用该命令前，需要保证本地工作区是没有任何修改代码的，也就是说需要将本地工作区编辑过的文件添加到暂存区（git add .），或提交到本地仓库中(git commit)，才可以使用该命令拉取指定分支的代码合并到当前分支中。**

**每次在操作完git commit命令后，必须拉取一下master分支代码，保持本地正在开发功能逻辑的代码分支代码是最新的，避免后续在提交时冲突过多或覆盖掉其他人的代码的问题出现。**



粉色框框选部分为自动合并代码的文件

**绿色框框选部分为本次更新（包含新增、删除、修改）的文件及具体信息，文件名后面的9 表示有9行有更新， +++++++++表示这些更新是新增（+）还是删除（-）。**



红色框框选部分：提到了一个关键词CONFLICT(content)，这个词的出现表明某一个具体文件在合并过程中发生了冲突。

绿色框框选部分：为提示语，自动合并失败，修改冲突然后提交修改后的结果.

**7、使用分支正确的开发流程：**

1.创建个人功能分支

git status ----查看当前工作区是否干净。保证当前工作分支干净

git branch ----查看当前所在分支，如果不再master 分支，切换到master分支

git checkout master ----切换到master 分支

git checkout -b <自己的分支> --------基于master 分支创建功能分支并切换到该分支下

1. 功能分支开发完成后

git add 修改完成后的文件 ----将修改过的文件添加到本地的暂存区

git commit -m “此次注释” ----提交修改后文件到本地仓库

git push origin <自己的分支> ----将个人分支推送到远程仓库，上到测试环境中进行测试

1. 测试完成后上线

git checkout master ----切换到master 分支

git pull origin master ---- 更新最新master

git merge <自己的分支> ---- 在本地将自己的分支合并到master分支上，如有冲突解决冲突。

git push origin master ---- 将解决完冲突的master 推送的远程仓库 ，然后上线

多人协作使用分支开发新的功能;

.gitignore只能忽略那些原来没有被track的文件，如果某些文件已经被纳入了版本管理中，则修改.gitignore是无效的。  
正确的做法是在每个clone下来的仓库中手动设置不要检查特定文件的更改情况。  
git update-index --assume-unchanged FILE 在FILE处输入要忽略的文件。  
如果要还原的话，使用命令：  
git update-index --no-assume-unchanged FILE