****初始化：创建一个git仓库，创建之后就会在当前目录生成一个.git的文件****

****git init****

****添加文件：把文件添加到缓冲区****

****git add filename****

****添加所有文件到缓冲区（从目前掌握的水平看，和后面加“.”的区别在于，加all可以添加被手动删除的文件，而加“.”不行）：****

****git add .****

****git add --all****

****删除文件****

****git rm filename****

****提交：提交缓冲区的所有修改到仓库(注意：如果修改了文件但是没有add到缓冲区，也是不会被提交的)****

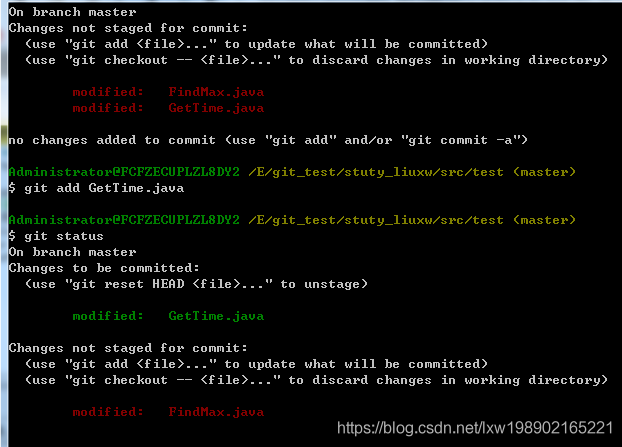
****git commit -m "提交的说明"****

****commit可以一次提交缓冲区的所有文件****

****查看git库的状态，未提交的文件，分为两种，add过已经在缓冲区的，未add过的****

****git status****

****从图中可以看出，绿色的就是已经add过的****



比较：如果文件修改了，还没有提交，就可以比较文件修改前后的差异

git diff filename

查看日志

git log

版本回退：可以将当前仓库回退到历史的某个版本

git reset

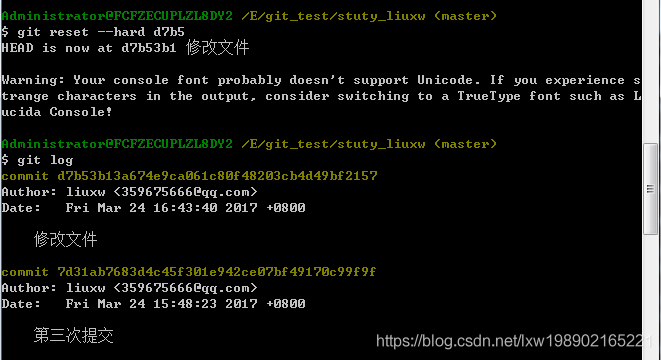
第一种用法：回退到上一个版本（HEAD代表当前版本，有一个^代表上一个版本，以此类推）

git reset --hard HEAD^

第二种用法：回退到指定版本(其中d7b5是想回退的指定版本号的前几位)

git reset --hard d7b5

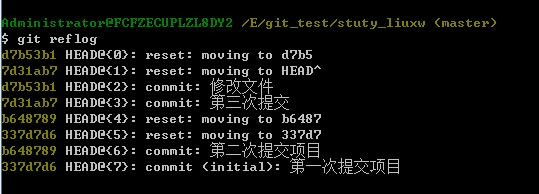
具体的效果如下





查看命令历史：查看仓库的操作历史

git reflog



git分支管理

查看分支的情况，前面带\*号的就是当前分支

git branch

创建分支

git branch 分支名

切换当前分支到指定分支

git checkout 分支名

创建分支并切换到创建的分支

git checkout -b 分支名

合并某分支的内容到当前分支

git merge 分支名

删除分支

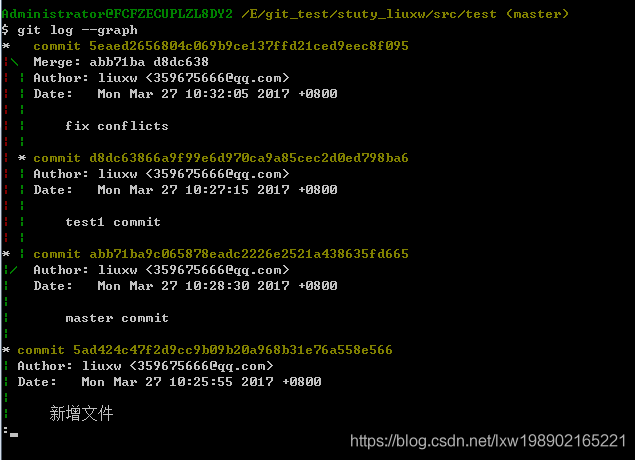
git branch -d 分支名

如果两个分支同时进行了同一个文件的修改和提交，在merge时就会产生冲突，首先要手动打开文件解决冲突，再提交，就相当于进行了merge

从下图可以看到，红色和绿色的虚线分别代表两个分支，两个分支分别进行了commit，最新的一次提交将两个分支内容进行了合并

查看分支合并图

git log --graph



新建标签，默认为最新版本，后面加上版本号参数则可指定版本增加标签

git tag 标签名 版本号

查看所有标签：

git tag

查看标签的详细信息

git show 标签名

将tag提交到远端仓库

推送所有tag:

git push origin --tags

推送某个tag：

git push origin v1.0

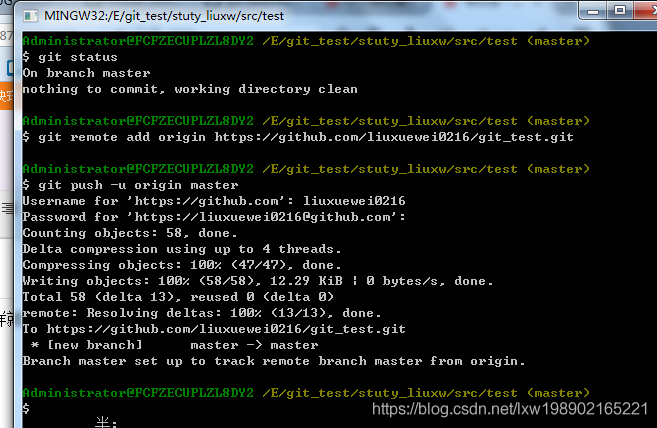
git远端库相关

git remote add origin git://127.0.0.1/abc.git 这样就增加了远程仓库abc。

git remote remove origin移除远端仓库

将本地仓库内容推送到远端仓库(-u 表示第一次推送master分支的所有内容，后面再推送就不需要-u了)，跟commit的区别在于一个是提交到本地仓库，一个是提交到远程仓库

git push -u origin master



从远端库更新内容到本地（相当于svn的update），

git pull

tips:如果push的时候，本地和文件和远端文件有冲突，就要先pull、然后手动解决冲突，才能继续push

git记住用户名密码

push的时候默认每次都需输入GitHub的用户名和密码，在git仓库根目录下.git文件夹的config文件末尾增加如下内容，即可记住用户名密码，无需每次推送都输入了

[credential]

helper = store

提交到git时，忽略部分IDE产生的文件

在根目录下创建.gitignore文件， 注意：新加.gitignore只能忽略那些原来没有被提交过的文件，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。

解决方法就是先把本地缓存删除（改变成未track状态），然后再提交:

git rm -r --cached .

git add .

git commit -m 'update .gitignore'