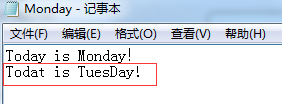
# （1）git status 和git diff 查看修改记录

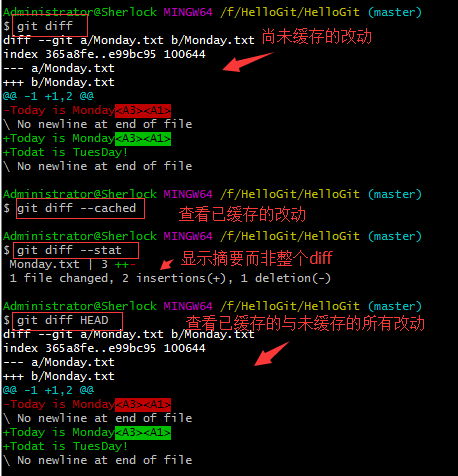
打开HelloGit下的Monday.txt文件，在第一行下面输入“Todat is Tuesday！”，保存关闭。



在gitbash分别输入git status 和 git status –s ,比较这两个命令的不同：

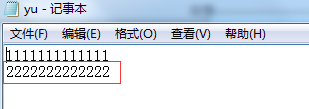


git diff 是对git status 输出结果的进一步说明，显示了文件被改动的内容：

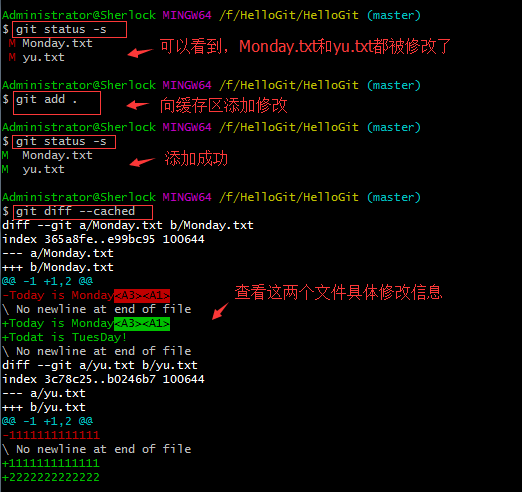


# （2）git reset HEAD 取消已缓存的内容（即撤回git add的动作）

打开HelloGit下的yu.txt文件，在第一行下面输入“222222222222”，保存关闭。

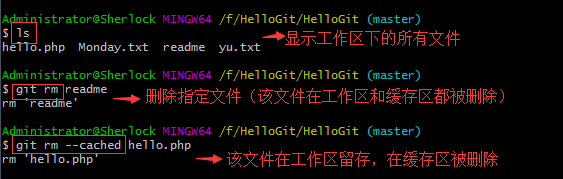


用git status 查看修改并用git add . 向缓存区提交修改：

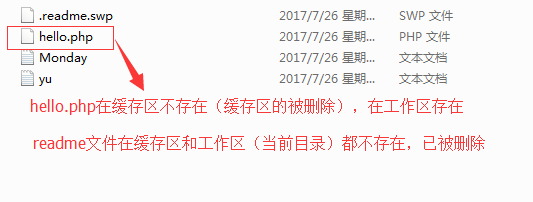




# （3）git rm删除指定文件

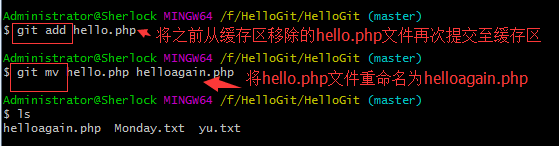


删除结果：

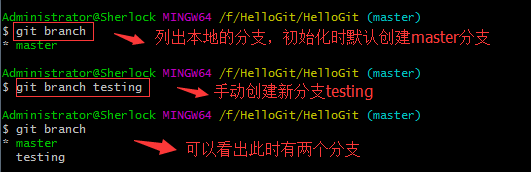


**注意：git rm 和git reset HEAD 不同，git rm 是删除文件，而git reset HEAD 是撤销提交到缓存区的修改，恢复之前的缓存状态。**

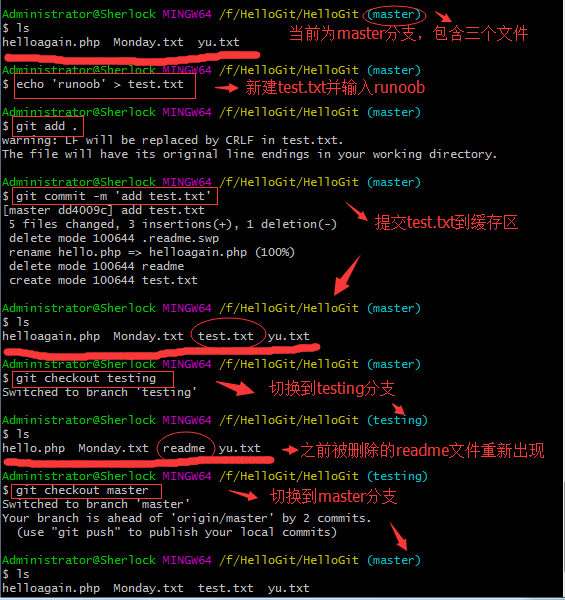
# （4）git mv 重命名文件



# （5）git branch创建并查看分支



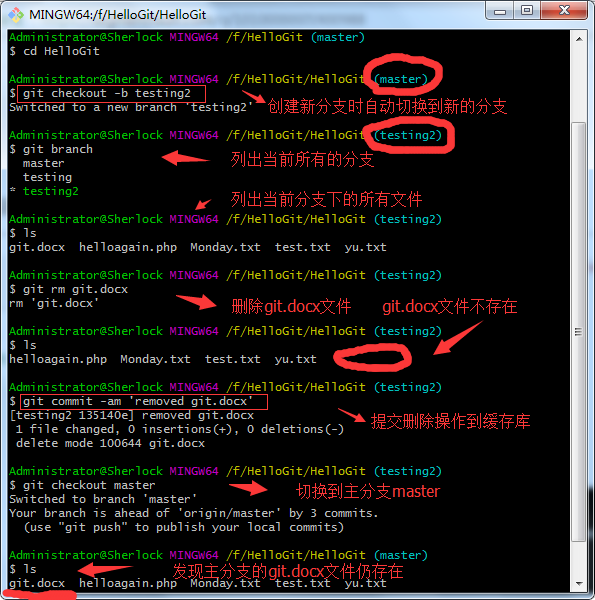
# （6）git checkout （branch）切换分支



**对下图的p.s.**

**git checkout –b (branch)创建新分支时自动切换到新创建的分支，可以看做git branch （branch）和git checkout （branch）的组合技；**

**git commit –am ‘注释’可以看做是 git add 和git commit –m 的组合技。**



**为什么要使用分支？使用分支后，可以将工作切分开来，即你可以从开发主线上分离开来，然后在不影响主线的同时继续工作。**

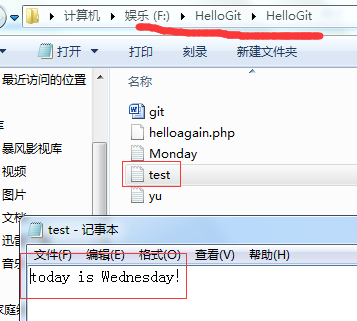
# （7）git merge (branch) 合并分支

在合并分支的时候，有时候会遇到合并冲突的问题。原因是两条分支对同一个文件的同一个文本块进行了不同的修改，而Git不能自动合并。

上面这段话有人可能看不懂，好吧，那我讲人话，意思就是说，有这么一个初始的test.txt文件，里面的内容是“aaaaa”，在同一时间，A把“aaaaaa”修改为“bbbbb”，B在“aaaaa”下面新增一行“111111”，等A和B都改完了，这个时候B要把A修改的东西合并过来，但是这个时候git不能自动合并，因此需要B手动修改。在B手动修改的时候，A修改的东西和B修改的东西都会呈现在test.txt里面，B只需要把相应不需要的东西删去就可以了。

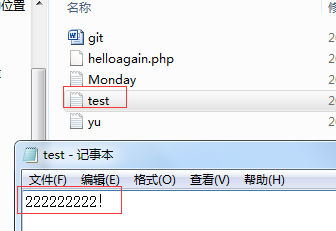
示例如下：

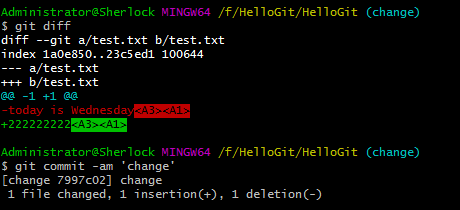
先在master分支下面新建一个test.txt，在第一行输入“today is Wednesday！”；



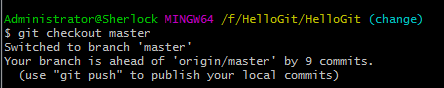
然后新建分支change并切换到change；在这个分支下修改test.txt，删掉第一行，输入“222222222222！”。修改后要通过git commit 提交到缓存区。

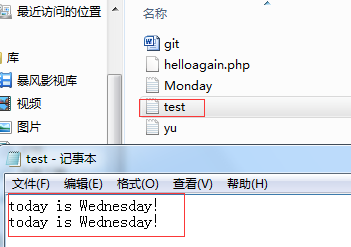


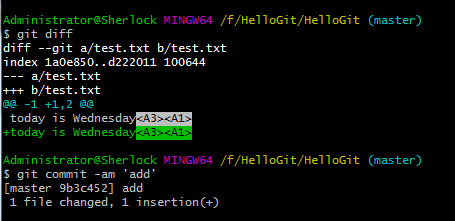




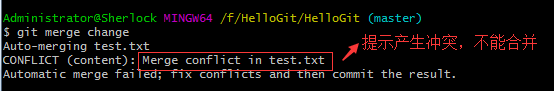
切换到master分支，再次修改test.txt的内容，增加一行“today is Wednesday！”修改后通过git commit 提交到缓存区。



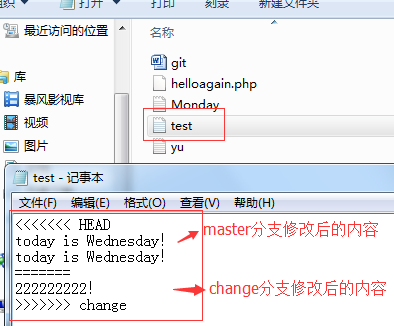




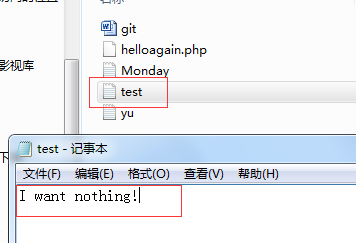
这个时候将change分支合并到master主分支，就会产生冲突。



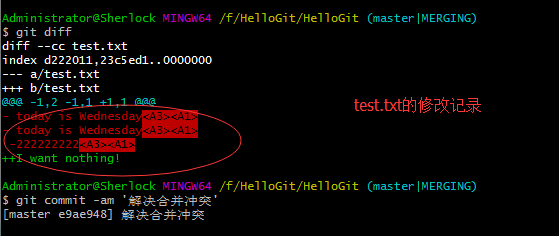
打开test.txt，发现两次修改的内容放在了一起。这个时候需要人工处理，去判断哪些信息是自己需要的，哪些是自己不需要的。



在这里，我把上面test.txt文件里面的内容改成了“I want nothing”。



修改后通过git commit 提交到缓存区。



提交成功就表示合并成功。



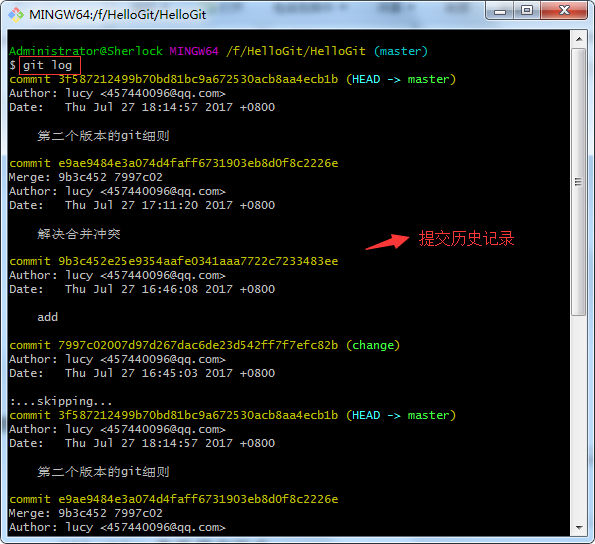
# （8）git branch –d (branch) 删除分支

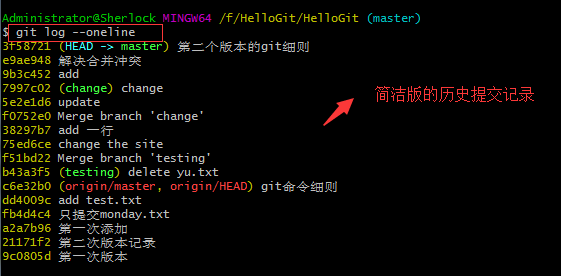
**git btanch -D (branch) 强制删除分支**



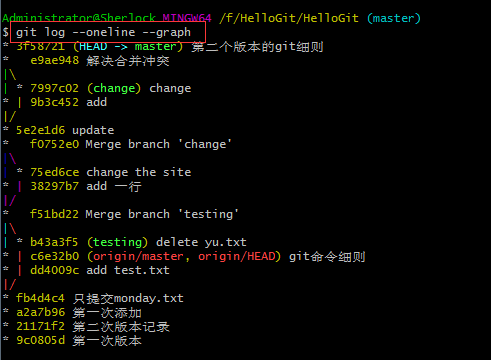
# （9）git log 查看详细的提交历史

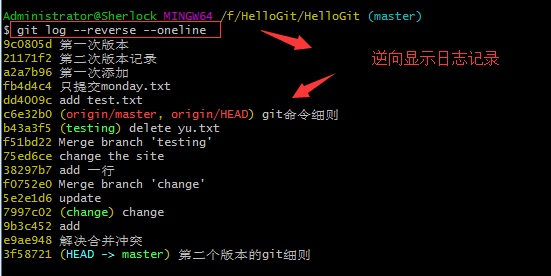
**git log –oneline 查看简洁的提交历史**





用 --graph 选项，查看历史中什么时候出现了分支、合并。



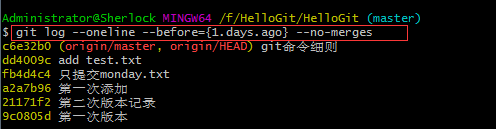


查找指定用户的提交日志:

$ git log --author=指定用户名 --oneline -5

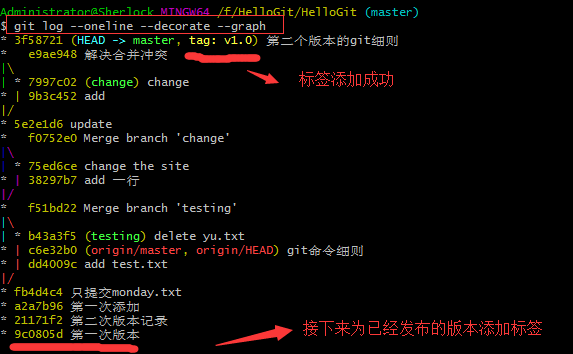
查看指定日期的提交日志（以3周前且在6月18后提交的所有日志为例）：

$ git log --oneline --before={3.weeks.ago} --after={2017-06-18} --no-merges

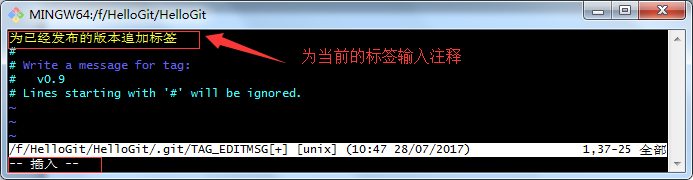
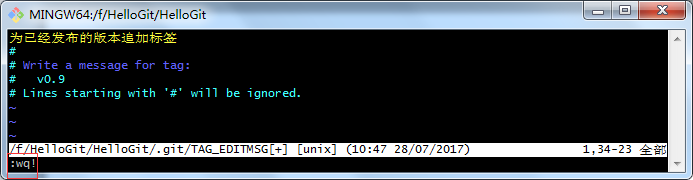


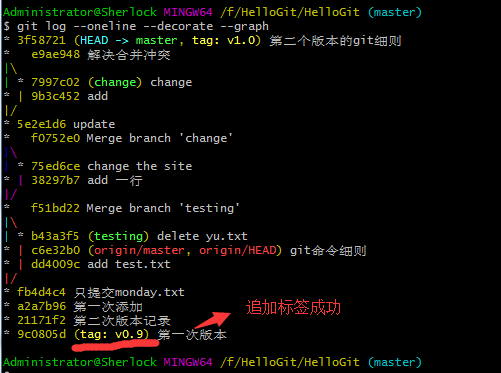
# （10）git tag标签创建、查看和删除

 输入上述语句时，git会自动打开vim编辑器，编辑文本添加注释。因此，需要会使用简单的命令编辑文本。

在键盘上按“i”，就可以在vim编辑器编写内容；内容输完后，按esc，再按“：wq!”保存退出或者按“：q！”不保存退出。



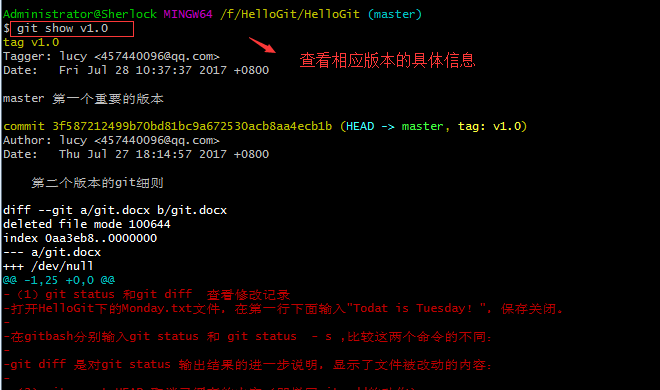
 



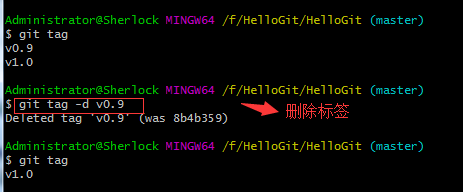
**git tag 查看有哪些标签**

**git show （标签版本号）查看版本的修改信息**



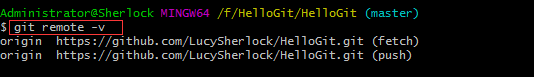


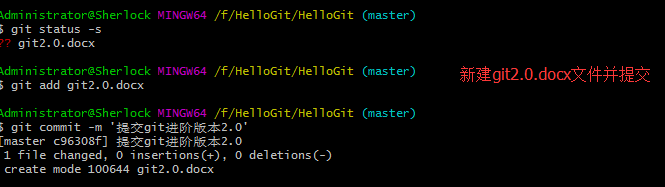
**git tag –d （标签版本号）删除指定标签**

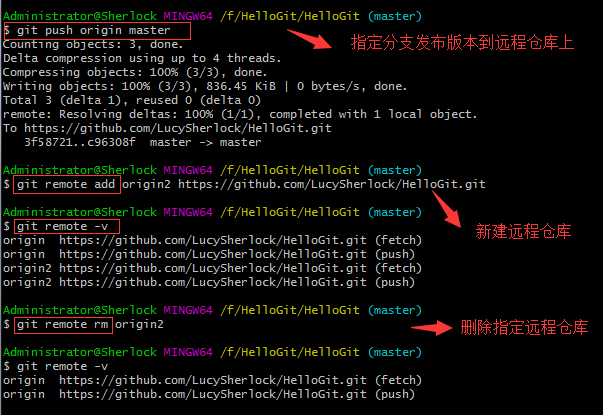


# （11）git remote 远程仓库

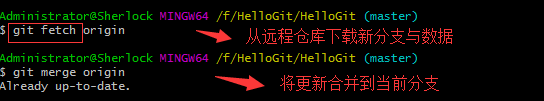
git remote –v 查看当前配置有哪些远程仓库







提取远程仓库：



还有另外一种方式是git pull。