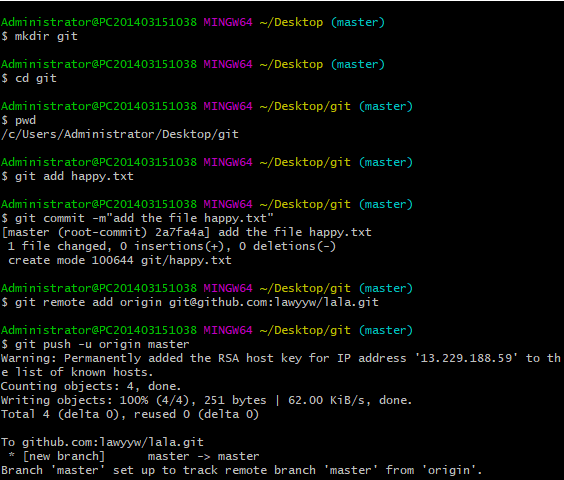
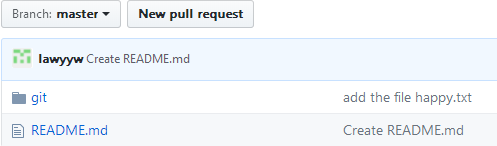
# 工作总结

### GitHub

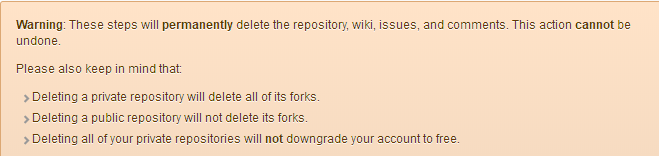
1.在本机与Github网站均重建了仓库，并进行了连接，终于操作成功！



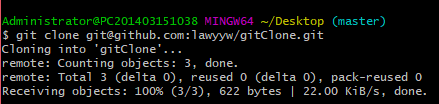


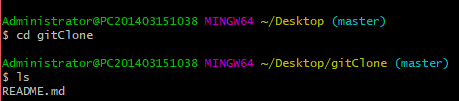
1. Clone

在进行克隆的时候，打算用第一个仓库git来进行操作，只不过它的命名与本机的git文件名重合，需要删除网站的repository或更改其名称

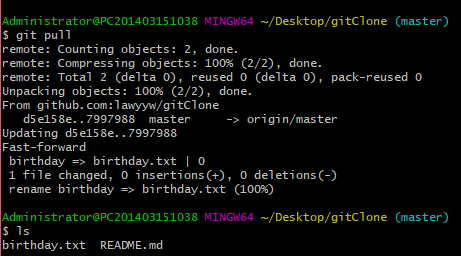


将其名称更改为gitClone，通过git clone命令已在桌面生成了gitClone文件夹，且含有README.md





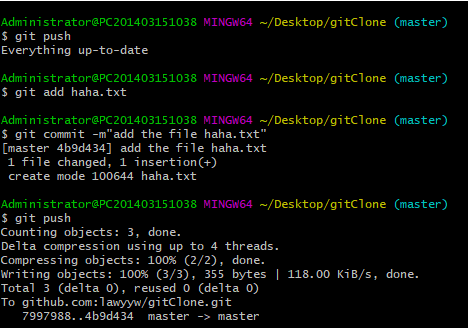
在网站中gitClone添加了birthday文本，通过git pull命令即可在本机中更新

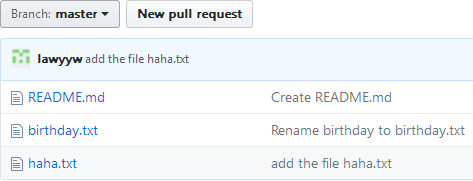


同理，在本机中增添新的文件，使用git push命令即可在网站中更新，只是一开始忘记对haha.txt进行git add和git commit命令，故显示

Everything up-to-date.

如果有多个人协作开发，那么每个人各自从远程克隆一份就可以了。





3.在[版本回退](https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000" \t "https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/_blank)里，每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

一开始的时候，master分支是一条线，Git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点：



每次提交，master分支都会向前移动一步，这样，随着你不断提交，master分支的线也越来越长。

当我们创建新的分支，例如dev时，Git新建了一个指针叫dev，指向master相同的提交，再把HEAD指向dev，就表示当前分支在dev上：



不过，从现在开始，对工作区的修改和提交就是针对dev分支了，比如新提交一次后，dev指针往前移动一步，而master指针不变：



假如我们在dev上的工作完成了，就可以把dev合并到master上。最简单的方法，就是直接把master指向dev的当前提交，就完成了合并：



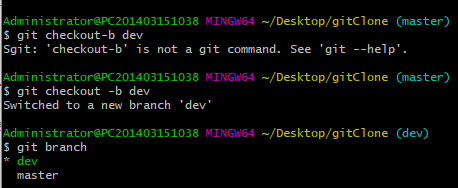
合并完分支后，甚至可以删除dev分支。删除dev分支就是把dev指针给删掉，删掉后，我们就剩下了一条master分支：



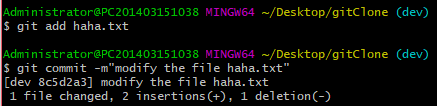
4.命令git checkout加上-b参数表示创建并切换，相当于以下两条命令：

创建分支：$ git branch

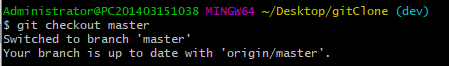
切换分支：$ git checkout

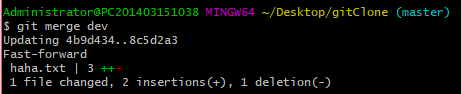


可以看到后缀也变成了(dev).

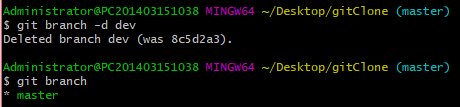


将分支重新切换至master后，再查看haha.txt，发现并没有修改痕迹，



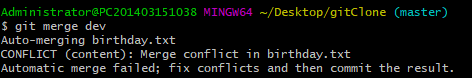


命令git merge用于合并指定分支到当前分支。合并后，再查看haha.txt的内容，就可以看到，和dev分支的最新提交是完全一样的，此时就可以删除分支dev（其中若未合并需要删除分支，需要命令：git branch -D dev）

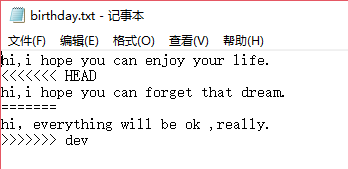


1. 冲突

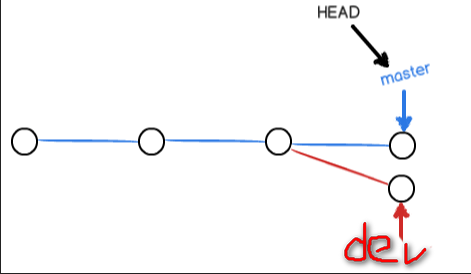
在新分支dev上修改了birthday.txt，并进行了提交命令；接着转向了master分支，仍修改了birthday.txt，进行了提交，此时进行git merge命令，反馈为：



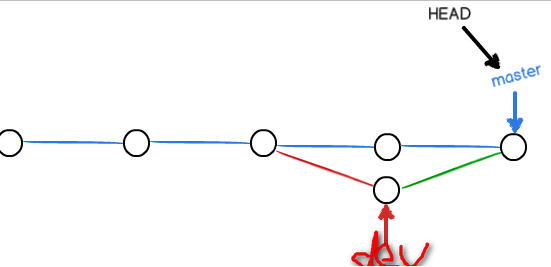
查看birthday.txt，显示：



此时分支情况为：



然后修改文件后再提交，分支变成了如下所示，最后可删除dev.



后来尝试了两次修改输入一样的内容也会出现冲突，在实际应用中应该不会出现两个人同时修改同一个文件，或者将一个文件放置在不同的分支里？

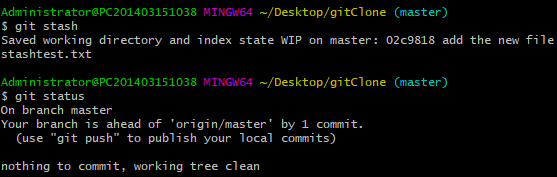
6.软件开发中经常会遇见bug，有了bug就需要修复。在Git中，每个bug都可以通过一个新的临时分支来修复，修复后，合并分支，然后将临时分支删除。

当接到一个修复bug的任务时，就会想创建一个分支来修复它，但是如果当前正在dev上进行的工作只进行到一半，还没有办法提交，但是必须在两个小时内修复该bug，怎么办？此时可以用命令git stash把当前工作现场“储藏”起来，等以后恢复现场后继续工作。

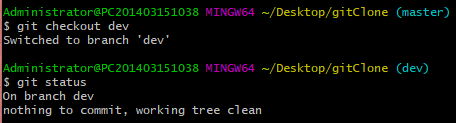
①在文件夹gitClone中新增了文件stashtest.txt，需要先进行add/commit命令，再操作git stash，否则显示结果：



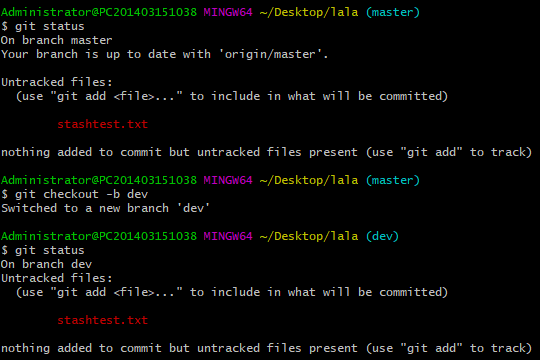
②在分支master中，修改了stashtest.txt，不必进行add/commit，然后使用命令git stash：



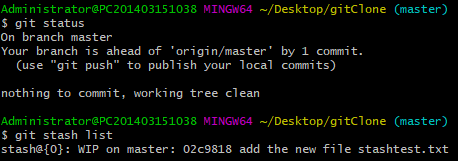
接下来创建了一个新分支dev，使用git status命令显示当前分支干净。



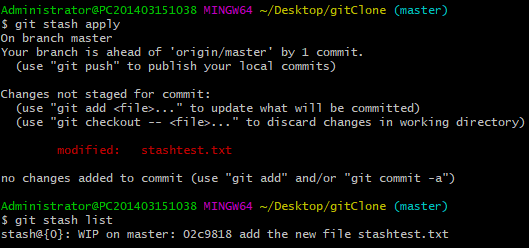
其中下图为未使用git stash命令显示的结果：



③切换至master，仍显示当前分支干净，需要其他命令来进行恢复。



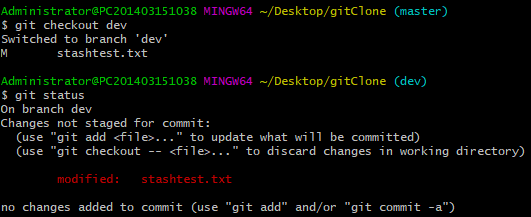
一是用命令git stash apply恢复，但是恢复后需要用命令git stash drop来删除stash的内容；二是用命令git stash pop，恢复的同时把stash内容删除；此时再用git stash list查看，就看不到任何stash内容了。



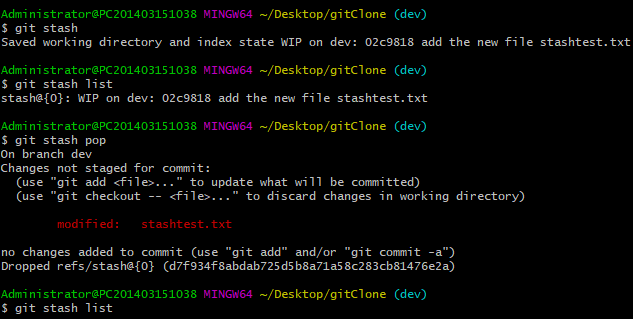




但再接着转向分支dev，仍会有未提交的新文件：



接着使用git stash命令来验证git stash pop：



可以多次使用命令git stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令git stash apply stash@{0}