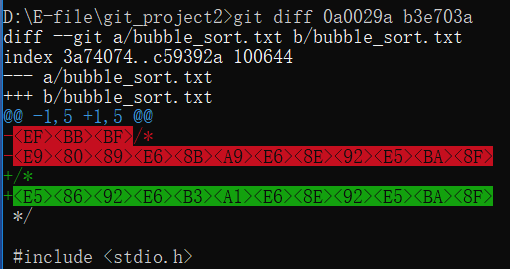
# 学习日报—Git学习

王丹凤

2018/08/22

### 一、Git命令

**1、git diff:比较暂存区域和工作目录，将两个文件合并起来打印**

****

结果：第一行：比较改变前后的两个文件；

第二行：文件的ID 指定文件的类型和权限；

第三、四行：---旧文件，存放在暂存区域，+++存放在工作目录的新文件；

第五行：-号旧文件，+号新文件后面的数字表示改变开始的行号，连续的行数；

旧文件如果没有自己独有的内容，那就不必显示旧文件的行号和行数，已经被覆盖到新文件中了。

注意：windows中，将文档保存到暂存区域再改变文档，此时git diff比较变化，改变的地方会出现乱码。

git diff 修改前id 修改后id号：比较历史两个快照

git diff git仓库中的快照id：比较当前工作目录和git仓库中的内容

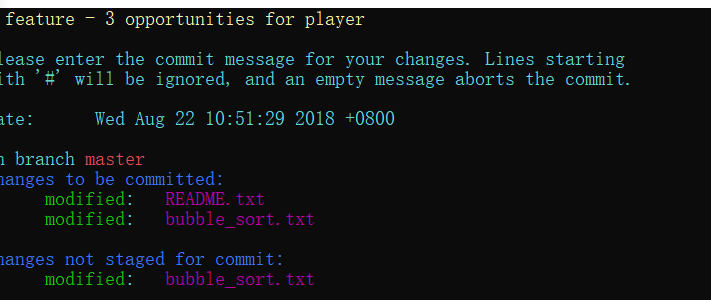
git diff HEAD:比较仓库中最新快照与当前工作目录的内容

git diff --cached：比较暂存区域和仓库的快照

**2、添加或者修改最后一次已经提交到仓库的内容**

（1）git commit --amend:出现新的界面vim编辑器

黄色的是最新快照的提交说明，点击i可以修改提交说明，按下esc，再保存:wq



（2）git commit --amend -m "bubble sort modify"直接修改提交说明，不用再进入新的编辑界面编写

**3、删除文件**

手动删除文件后，使用git checkout -- README.txt就可以恢复被删文件

把已经提交的文件删除：git rm 文件名，该命令删除的只是工作目录和暂存区域的文件，取消跟踪，但是仓库里的快照文件不会连带删除

git status快照能发现记录，git log暂存区域的记录已经不存在了

git rm -f test.txt:强制删除指定文件

git rm --cached 文件名：只删除暂存区域的文件，保留工作目录，

**4、文件重命名**

git mv 文件老名字 新名字

### 二、Git分支（通过指针来实现）

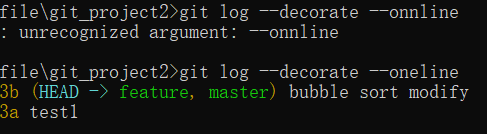
**1、git branch feature(分支名)：创建分支**

git log --decorate：让log显示指向提交的所有引用，比如说这个分支、标签

C:\Users\dell\AppData\Local\Temp\WeChat Files\a57a1f9895e7ea7142cff06d18b2482.png

master：git的默认分支，主分支，feature是刚创建的分支

切换分支：git checkout feature（HEAD本来默认指向主分支master，现在指向feature）



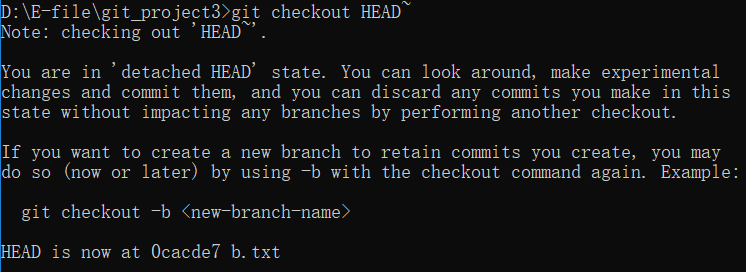
git log --decorate --oneline：一行显示一个快照，只显示快照的id和说明，精简化显示

git log --decorate --oneline（精简化） --graph(图形化显示) --all（所有分支）

git checkout -b feature2：创建并切换到feature2分支里去；

git checkout HEAD~：切换分支的时候没有给分支名字，此时切换的分支是匿名的，以后不能再找到这个分支，适合做实验用，不妨碍其他；

指针HEAD会由c.txt切换到上一个快照b.txt。



**2、实际开发中的分支**

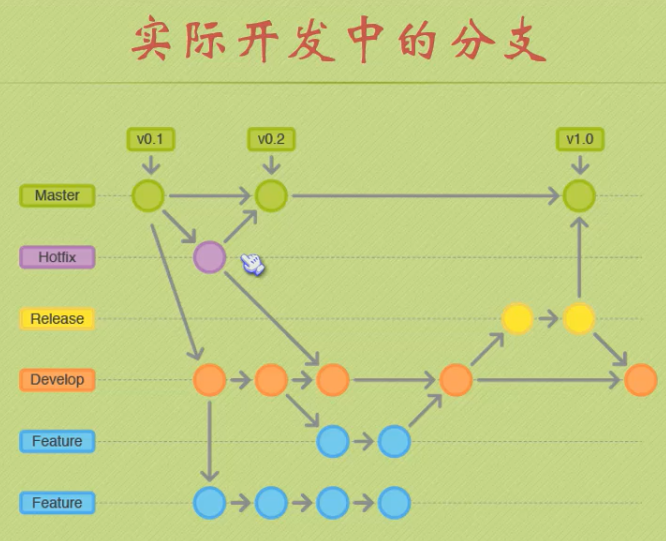
Master:用来发布

hotfix：用来修补程序中出现的bug

release：内部发布版，测试完再master发布

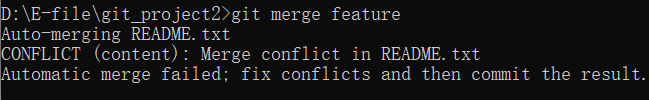
develop：开发版，开发完编译成一个可执行程序，就是release

feature：相关的功能



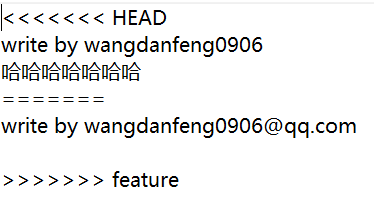
**3、合并分支**

git merge 分支名：将指定的分支名合并到当前分支中来



失败：Automatic merge failed; fix conflicts and then commit the result.

合并分支出现冲突，两个分支中存在重名 的文件



注意：有冲突的文件会在合并时被加入一些标记，删除标记，再重新提交文件，消除冲突

**4、删除分支**

git branch -d 分支名

删除过后，两个分支名已经消失，但是具体的快照还是存在。

**5、checkout和reset命令的区别**：

（1）恢复文件

两者都可以用于恢复指定快照的指定文件，且都不会改变HEAD指针的指向；

区别：reset命令只将指定文件恢复到暂存区域，而checkout是同时覆盖暂存区和工作目录；

注意：如果reset命令后面跟的是文件的话，--hard与--soft都不可使用。

（2）恢复快照

区别：①checkout更安全，checkout命令不只是切换分支，同时在切换分支前尝试进行合并，检查当前的工作状态，如果不是“clean”的话，git不允许这么做

reset --hard则是直接覆盖所有数据

②reset命令会移动HEAD所在分支（feature/master之类的分支）的指向，checkout命令只会移动HEAD自身来指向另一个分支

**6、移动命令：**

j：向下移动一行

k:向上移动一行

f：一页一页的向下移

b:向上移

u、d:半页

G:最后一行

g:第一行

3g:第三行

**7、搜索命令**：

/print,匹配文件中含有print的，:n下一个，:N上一个

:h帮助文档

:q退出,直接退出，不保存内容

：wq保存退出