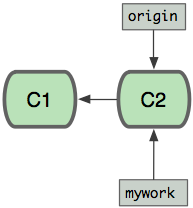
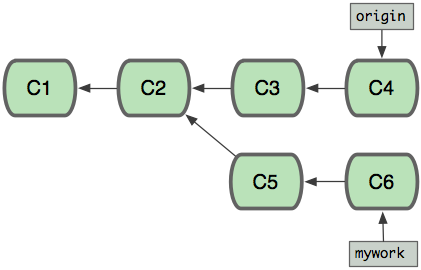
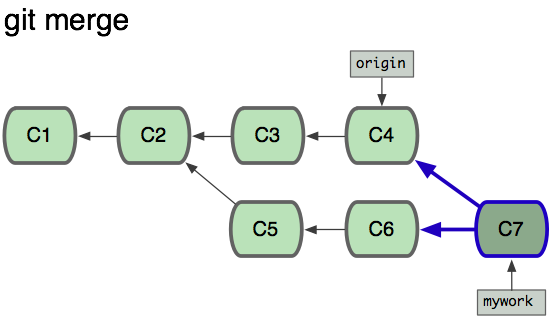
1.在c2建了mywork分支



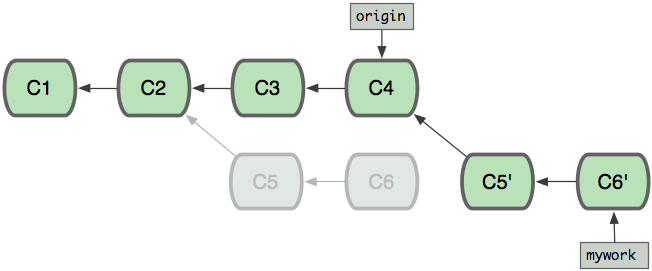
2然后origin myword各提交了两次



git merge origin



Git rebase origin



这些命令会把你的"mywork"分支里的每个提交(commit)取消掉，并且把它们临时 保存为补丁(patch)(这些补丁放到".git/rebase"目录中),然后把"mywork"分支更新 为最新的"origin"分支，最后把保存的这些补丁应用到"mywork"分支上。

**把"mywork"分支更新到最新的"origin"分支 （不是更新到mywork）**

**Mywork的接到了origin后**

git rebase 的黄金法则便是，**绝不要在公共的分支上使用它**。

rebase将master分支上的所有提交都移到了个人分支后面。问题是它只发生在你的代码仓库中，其他所有的开发者还在原来的master上工作。因为rebase引起了新的提交，Git会认为你的master分支和其他人的master已经分叉了。

即 git checkout origin

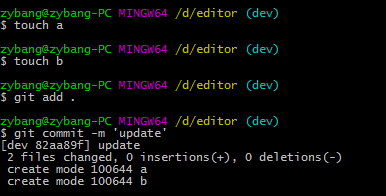
Git rebase mywork

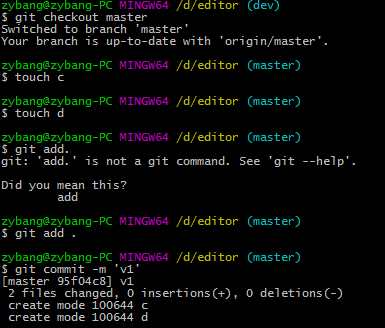
这样可以保证一个干净的分支

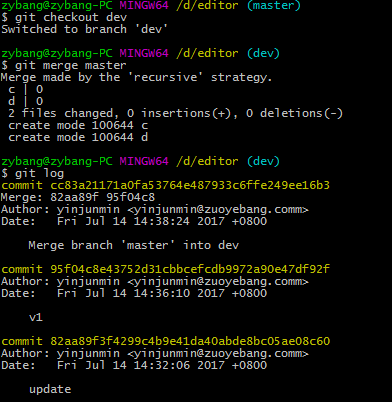
rebase在解决完冲突后，用"git-add"命令去更新这些内容的索引(index), 无需像merge一样执行 git-commit,只要执行:

$ git rebase --continue

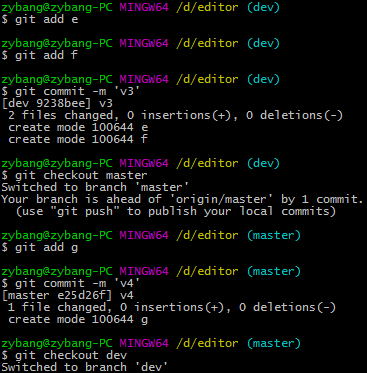
Merge

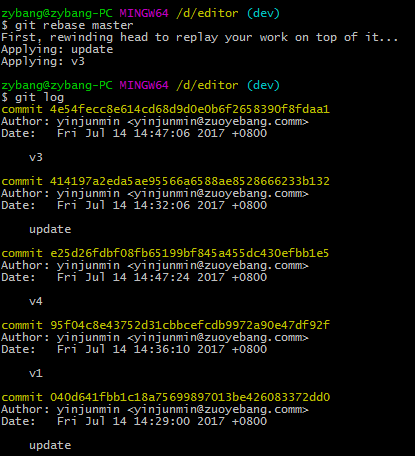






merge





V1 v4是matser的，update v3是dev的 （最下面一个update是初始化的）

结合上一页的log看后。Rebase master以master作为所在的dev主干，将dev的提交 重写生成commit作为补丁一样打到的dev分支上，甚至取消了merge的记录。

相当于，在master v4提交后开始checkout –b dev,然后dev上提交了update和v3