

假设目前情况如图所示，开发过程中在C2出新建一个dev分支开发，并有两次提交C5 C6到dev分支； 但是与此同时master端upstream 上也有两个更新，pull 到本地为C3 C4。

现在dev开发完成准备发起一个PR 到upstream 的master上，但是在发PR之前需要在本地进性合并和冲突处理，这样处理好之后发PR到upstream 就不需要处理冲突了。

我们有两种方式来进性合并，一种是使用merge 另一种是使用rebase，下面分析使用两者的区别。

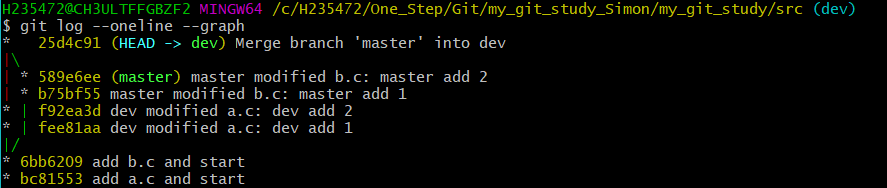
首先我们假设dev 和master 上述改动是改动不同的文件，也就不会产生冲突。

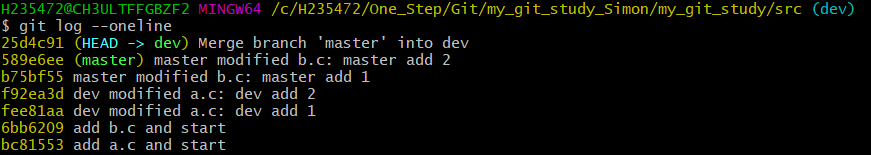
merge： git checkout dev //先切到master 分支上

git merge master //把master的改动合并到dev上

这时候会弹出文本输入界面，让你输入commit comments，此时输入 Merge branch ‘master ’ into dev

git log --oneline --graph //以图表的形式显示log



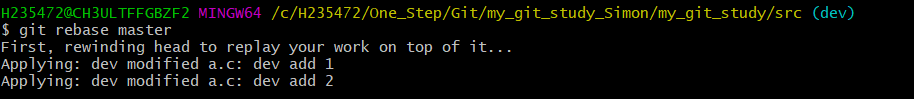


可以看出merge 的时候其实是有一次新的提交的。

rebase：

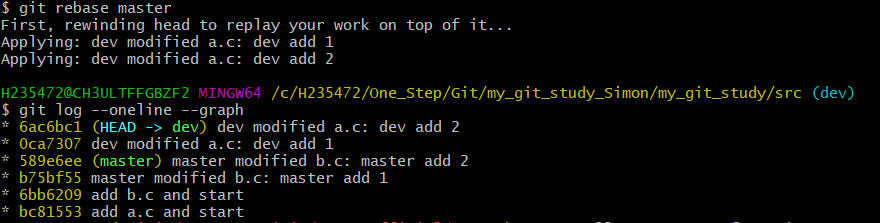
使用 git reset --hard f92ea3d 将dev 分支 强制设置到f92ea3d dev modified a.c: dev add 2

然后使用 git rebase master ，效果如下：



因为没有冲突所以 rebase 就直接结束了。

然后我们看一下log：

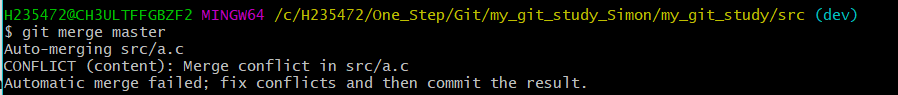


不管使用 还是不使用 --graph 发现没有合并的提交，感觉好像dev 的两次修改和提交是在最新的master上面执行的一样。这样就少了一次合并的提交过程看上去更清晰了。

再次我们假设dev 和master 分别改动提交两次都是相同的位置，也就是产生两次冲突。

merge:

同样执行 git merge master //将master 合并到dev分支上



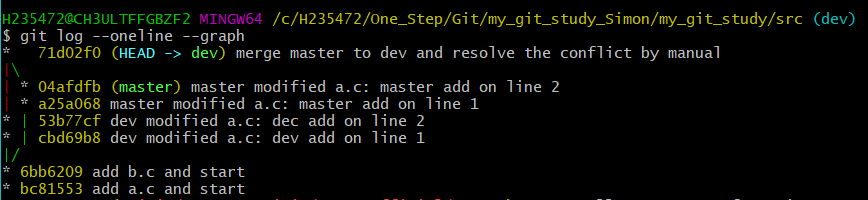
Git 显示产生了冲突，自动merge失败，需要我们自己解决冲突后重新提交。

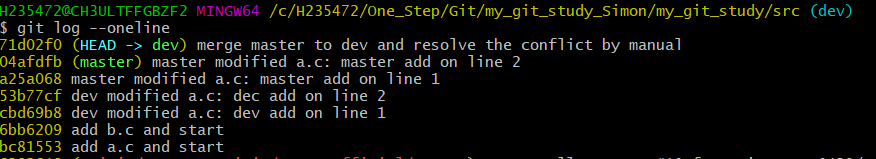
使用git mergetool 进行合并，我设置的mergetool 是beyond compare4。

然后使用beyond compare4进行合并之后并删除合并产生的临时文件a.c.orig ，再次提交



提交之后查看log：

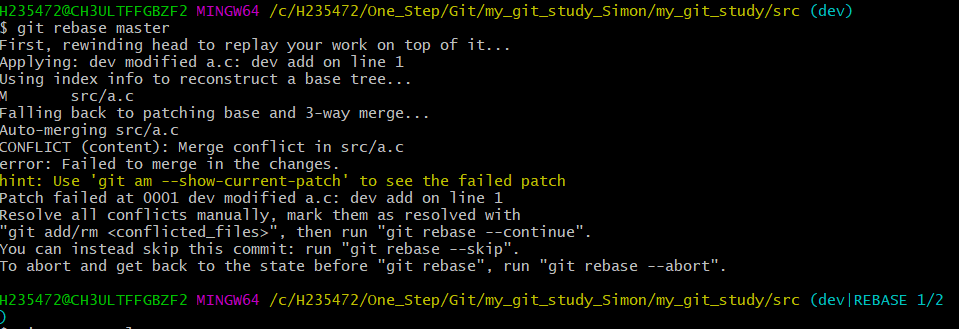




可以看出除了我们手动合并了冲突，剩下的和没有冲突的merge基本上一样，没有什么区别，还是会多出合并的一次commit。

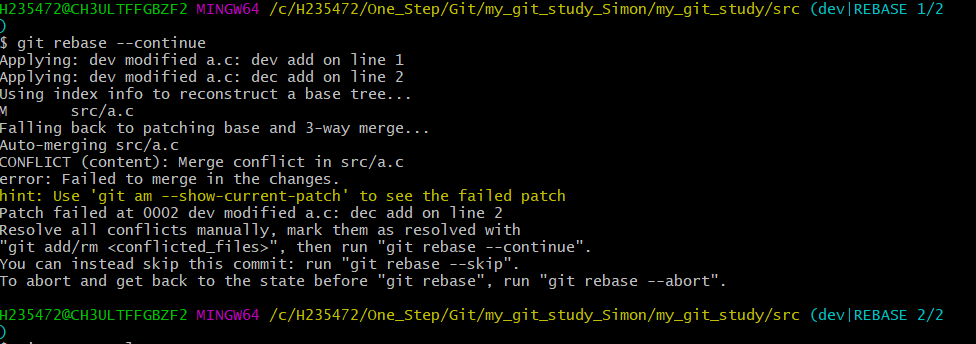
rebase:

使用 git rebase master

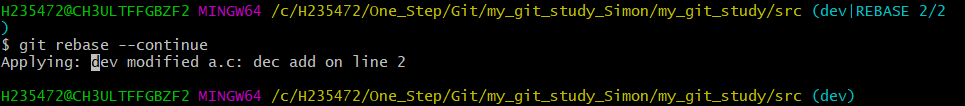


会产生冲突，我们解决冲突之后 使用 git rebase --continue 会继续进行rebase操作，注意这里并没有commit 行为！！

在预料之中还是会产生第二次冲突

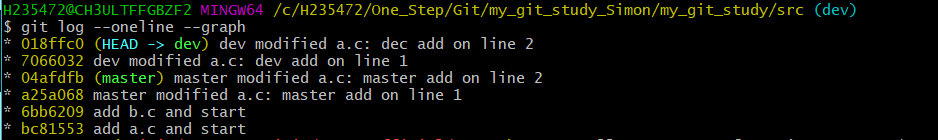


我们合并之后再次使用git rebase --continue



这次rebase 就结束了。

然后我们查看一下log



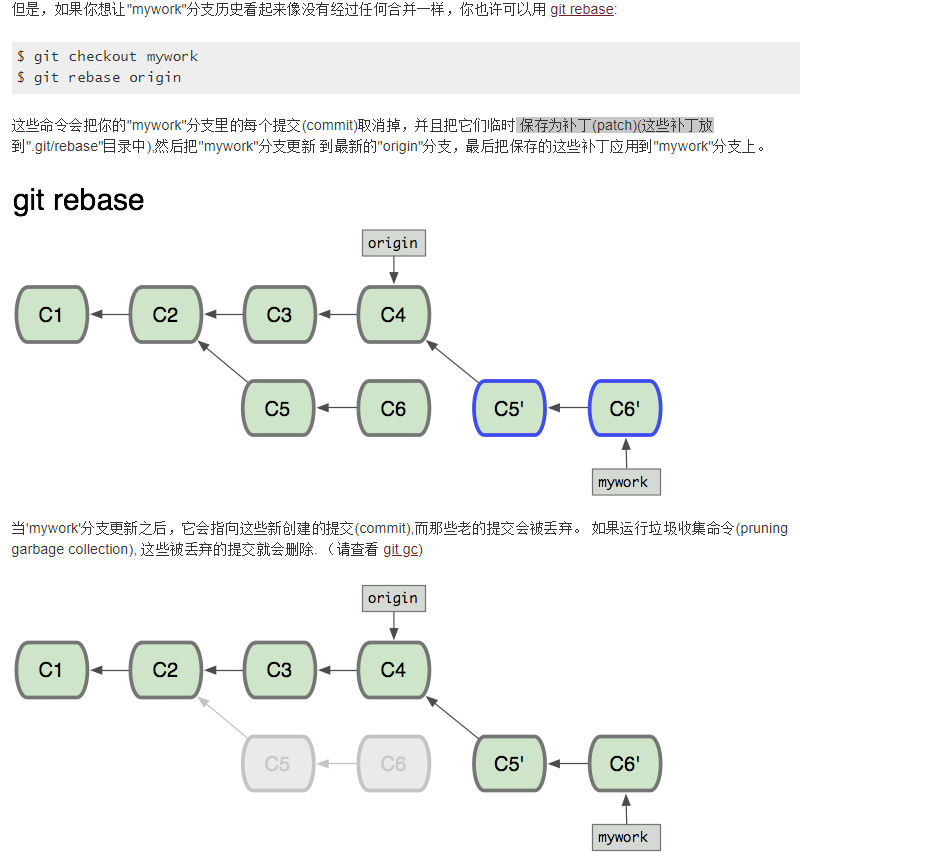
和之前没有冲突的是一样的，加不加 -- graph 一样。都没有merge 的commit，其实你处理冲突的过程是不可见的，给人的感觉好像你的改动是在最新的master 上面的改动。

总结：其实我们开发的时候 master 会同步upstream/master 是会在变的， 我们一般都会新建一个分支例如 dev ， 我们在dev 上做开发和commit， 然后定期把upstream/master 上的更新 pull 到master 上，这样master 和 upstream/master 在保持同步。 很显然我们 dev是基于更早的master版本开发的，我们需要把master上最新的改动映射到dev上，这样发PR到upstream上才不会有冲突（或者我们暂时不发PR，只是做一个日常更新，有冲突的话可以提前处理，免得到发PR冲突很多等等问题）。 那么所谓映射可以有两个方式 merge 和 rebase。

那么到底这两个使用哪个好呢？ 我认为其实使用rebase 更好一些。 如果使用merge的话，你的dev 上合并merge上的改动都会产生一次commit，这个commit可能是没什么用的，因为你只是日常更新一下分支，如果这样最终发PR并合并到upstream上的时候，查看log 就会有很多无用的commit，影响upstream上的log整洁，会比较乱。

这个问题使用rebase 就可以解决了，其实你在rebase 的时候没有产生多余的commit， 给我们的感觉就像dev 上的改动是在最新的master上面改的，这样发PR后整个upstream的log也会比较清晰。

rebase 和 merge的实现原理可以参考这个文章：<http://gitbook.liuhui998.com/4_2.html>



更多可以参考这篇blog：<https://www.cnblogs.com/xueweihan/p/5743327.html>