[SegmentFault](https://segmentfault.com/)

* [头条](https://segmentfault.com/news)
* [问答](https://segmentfault.com/questions)
* [专栏](https://segmentfault.com/blogs)
* [职位](https://segmentfault.com/jobs)
* [活动](https://segmentfault.com/events)
* [App](https://segmentfault.com/app)

窗体顶端

搜索

窗体底端

* [消息](https://segmentfault.com/tour)
* [注册 · 登录](https://segmentfault.com/user/login)
* [home](https://segmentfault.com/)
* [javascript](https://segmentfault.com/t/javascript)
* [php](https://segmentfault.com/t/php)
* [python](https://segmentfault.com/t/python)
* [java](https://segmentfault.com/t/java)
* [mysql](https://segmentfault.com/t/mysql)
* [ios](https://segmentfault.com/t/ios)
* [android](https://segmentfault.com/t/android)
* [node.js](https://segmentfault.com/t/node.js)
* [html5](https://segmentfault.com/t/html5)
* [linux](https://segmentfault.com/t/linux)
* [c++](https://segmentfault.com/t/c%2B%2B)
* [css3](https://segmentfault.com/t/css3)
* [git](https://segmentfault.com/t/git)
* [golang](https://segmentfault.com/t/golang)
* [ruby](https://segmentfault.com/t/ruby)
* [vim](https://segmentfault.com/t/vim)
* [docker](https://segmentfault.com/t/docker)
* [mongodb](https://segmentfault.com/t/mongodb)

问

[团队开发里频繁使用 git rebase 来保持树的整洁好吗?](https://segmentfault.com/q/1010000000430041)

* [git](https://segmentfault.com/t/git)

[**题叶**](https://segmentfault.com/u/jiyinyiyong) 2014年03月11日提问

* 关注 **32** 关注
* 收藏 **54** 收藏，35.4k 浏览

问题对人有帮助，内容完整，我也想知道答案5问题没有实际价值，缺少关键内容，没有改进余地

用了以后, 树可以非常清晰, 某种程度上便于追踪, 但是 push --force 就多多了,  
不用呢, 合并没有远程仓库被修改的麻烦, 可是追踪又不清晰...

怎样取舍? 团队里一般怎么取舍?

* [2014年03月11日提问](https://segmentfault.com/q/1010000000430041)

* [评论](javascript:void(0);)

* [邀请回答](javascript:;)

* [编辑](javascript:;)

* [更多](javascript:void(0);)

[默认排序](https://segmentfault.com/q/1010000000430041#answers-title)[时间排序](https://segmentfault.com/q/1010000000430041?sort=created#answers-title)

11个回答

答案对人有帮助，有参考价值18答案没帮助，是错误的答案，答非所问

git rebase是对commit history的改写。当你要改写的commit history还没有被提交到远程repo的时候，也就是说，**还没有与他人共享之前**，commit history是你私人所有的，那么想怎么改写都可以。

而一旦被提交到远程后，这时如果再改写history，那么势必和他人的history长的就不一样了。git push的时候，git会比较commit history，如果不一致，commit动作会被拒绝，唯一的办法就是带上-f参数，强制要求commit，这时git会以committer的history覆写远程repo，从而完成代码的提交。虽然代码提交上去了，但是这样可能会造成别人工作成果的丢失，所以使用-f参数要慎重。

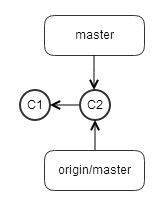
**楼主遇到的问题，就是改写了公有的commit history造成的。**要解决这个问题，就要从提交流程上做规范。

举个正确流程的栗子：

假设楼主的team中有两个developer：tom和jerry，他们共同使用一个远程repo，并各自clone到自己的机器上，为了简化描述，这里假设只有一个branch：master。

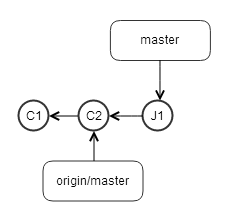
这时tom机器的repo有两个branch  
master, origin/master  
而jerry的机器上也是有两个branch  
master, origin/master

均如下图所示

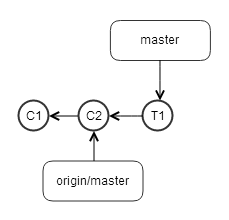


tom和jerry分别各自开发自己的新feature，不断有新的commit提交到他们各自私有的commit history中，所以他们的master指针不断的向前推移，分别指向不同的commit。而又由于他们都没有git fetch和git push，所以他们的origin/master都维持不变。

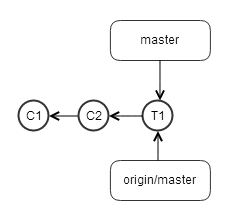
jerry的repo如下



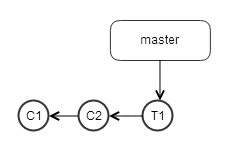
tom的repo如下，注意T1和上图的J1，分别是两个不同的commit



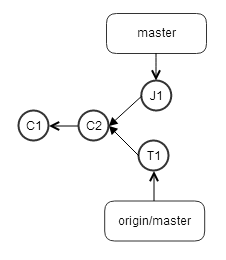
这时Tom首先把他的commit提交的远程repo中，那么他本机origin/master指针则会前进，和master指针保持一致，如下



远程repo如下

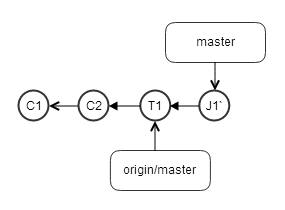


现在jerry也想把他的commit提交到远程repo上去，运行git push，毫无意外的失败了，所以他git fetch了一下，把远程repo，也就是之前tom提交的T1给拉到了他本机repo中，如下



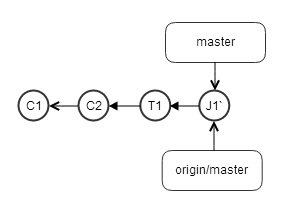
commit history出现了分叉，要想把tom之前提交的内容包含到自己的工作中来，有一个方法就是git merge，它会自动生成一个commit，既包含tom的提交，也包含jerry的提交，这样就把两个分叉的commit重新又合并在一起。但是这个自动生成的commit会有两个parent，review代码的时候必须要比较两次，很不方便。

jerry为了保证commit history的线性，决定采用另外一种方法，就是git rebase。jerry的提交J1这时还没有被提交到远程repo上去，也就是他完全私有的一个commit，所以使用git rebase改写J1的history完全没有问题，改写之后，如下

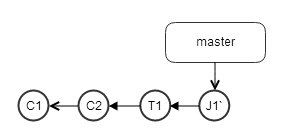


注意J1被改写到T1后面了，变成了J1`

git push后，本机repo



而远程repo



异常的轻松，一条直线，没有-f

所以，在不用-f的前提下，想维持树的整洁，方法就是：在git push之前，先git fetch，再git rebase。

**git** fetch origin master

git rebase origin/master

git push

强烈推荐阅读

* [a successful git branching model](http://nvie.com/posts/a-successful-git-branching-model/)
* [Git-分支-分支的衍合](http://git-scm.com/book/zh/Git-%E5%88%86%E6%94%AF-%E5%88%86%E6%94%AF%E7%9A%84%E8%A1%8D%E5%90%88)
* [2014年03月19日回答](https://segmentfault.com/q/1010000000430041/a-1020000000439630)

* [6 评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/chengbo)

[Roy\_370884](https://segmentfault.com/u/chengbo)269 声望

答案对人有帮助，有参考价值10答案没帮助，是错误的答案，答非所问

除非只有自己一个人用，不然用 push --force 的都该去死。

* [2014年03月11日回答](https://segmentfault.com/q/1010000000430041/a-1020000000430149)

* [6 评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/lilydjwg)

[依云](https://segmentfault.com/u/lilydjwg)21.5k 声望

+1

我现在觉得不是他们用 rebase 的问题, 而是他们用了 rebase 那么容易导致 push -f是有问题

[**题叶**](https://segmentfault.com/u/jiyinyiyong)· 2014年03月12日

+2

回复 [题叶](https://segmentfault.com/u/jiyinyiyong)：

只 rebase 未 push 的部分

[**Hsiaoming\_Yang**](https://segmentfault.com/u/lepture)· 2014年03月19日

[展开评论](javascript:void(0);)

答案对人有帮助，有参考价值2答案没帮助，是错误的答案，答非所问

本地分支和远程分支的绑定（tracking)，加上 rebase 策略：

[branch "master"]

remote = origin

**merge** = refs/heads/**master**

rebase = true

这样一来，更新代码（pull）的时候就会自动应用 rebase 而不是产生 merge commit，除非有其他情况产生，比如三方合并造成了冲突需要人共去干预。大部分时候还是很聪明的，只要团队里的习惯都良好，那么可以保持一个非常干净漂亮的树形。

其实想让树形结构漂亮些清晰些是有很多办法的，但是首先要取决于团队用的是什么样的 Git Model，对症下药即可。在这里就无法一言以蔽之了。

**另外，楼上说得对，慎用 push -f！**

* [2014年03月11日回答](https://segmentfault.com/q/1010000000430041/a-1020000000430200) · [2014年03月11日更新](https://segmentfault.com/q/1010000000430041/a-1020000000430200/revision)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/nightire)

[n͛i͛g͛h͛t͛i͛r͛e͛](https://segmentfault.com/u/nightire)27k 声望

答案对人有帮助，有参考价值2答案没帮助，是错误的答案，答非所问

这应该是一个git workflow的问题，我们团队也一直在使用rebase保证commit信息的整洁，但不会用到push -f这样的操作。

关于git workflow就是一个见仁见智的问题了，下面几篇文章你可以看下，再找出一套适合自己团队的就可以了，不过最重要的是要保证团队的每个人都熟悉git，防止犯愚蠢的错误。

* [构造干净的 Git 历史线索](http://codecampo.com/topics/379?comment_id=1354#comment-1354)
* 流传甚广的[A successful Git branching model](http://nvie.com/posts/a-successful-git-branching-model/)
* [Github workflow](http://guides.github.com/overviews/flow/)

如果使用github来团队合作的话，用好[pull request](https://github.com/blog/1124-how-we-use-pull-requests-to-build-github" \t "_blank)，它可以解决push -f这种愚蠢问题！

* [2014年03月12日回答](https://segmentfault.com/q/1010000000430041/a-1020000000430652)

* [2 评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/rockybean)

[rockybean](https://segmentfault.com/u/rockybean)597 声望

+1

回复 [题叶](https://segmentfault.com/u/jiyinyiyong)：

我觉得你们开发的工作流(就是上面的workflow)很有问题，所有的开发都应该在本地分支上开发，不要直接基于远程分支来改。应该是本地分支开发完成后，在尝试使用rebase合并到远程分支，并且尽快push，防止远程分支更改。你详细看下我列出的文章，然后将workflow加入日常开发，就能完全避免这些问题。

[**rockybean**](https://segmentfault.com/u/rockybean)· 2014年03月13日

[展开评论](javascript:void(0);)

答案对人有帮助，有参考价值1答案没帮助，是错误的答案，答非所问

每个人提交前，都应该把自己的修改rebase到服务器的最新代码之上，遵守这个规则就不会有任何问题。如果你需要force push，说明你做反了，把服务器代码rebase到你本地分支之上才会需要force push，这是错误的用法。

* [2014年03月12日回答](https://segmentfault.com/q/1010000000430041/a-1020000000430784)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/alvin_1)

[Alvin\_135272](https://segmentfault.com/u/alvin_1)34 声望

答案对人有帮助，有参考价值1答案没帮助，是错误的答案，答非所问

建议参考 Pro Git 中关于 rebase 的章节 <http://git-scm.com/book/zh/Git-%E5%88%86%E6%94%AF-%E5%88%86%E6%94%AF%E7%9A%84%E8%A1%8D%E5%90%88>

**衍合的风险**  
呃，奇妙的衍合也并非完美无缺，要用它得遵守一条准则：

**一旦分支中的提交对象发布到公共仓库，就千万不要对该分支进行衍合操作。**

如果你遵循这条金科玉律，就不会出差错。否则，人民群众会仇恨你，你的朋友和家人也会嘲笑你，唾弃你。

就我而言。如果 rebase 完后，需要使用 push -f 的话，一定代表该 rebase 操作是不合适。除非你是有意在修改提交历史。

* [2014年03月12日回答](https://segmentfault.com/q/1010000000430041/a-1020000000430874)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/reyesyang)

[reyesyang](https://segmentfault.com/u/reyesyang)870 声望

答案对人有帮助，有参考价值1答案没帮助，是错误的答案，答非所问

不需要push -f啊，如果分支落后就用pull --rebase

* [2015年06月16日回答](https://segmentfault.com/q/1010000000430041/a-1020000002909194)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/yszc)

[yszc](https://segmentfault.com/u/yszc)11 声望

答案对人有帮助，有参考价值0答案没帮助，是错误的答案，答非所问

楼上的答案都很正，个人认为，**除非某个分支只有你自己搞**，你怎么rebase都是没有问题的，但是如果你在**master或者develop**这种分支上来rebase，估计团队里每个人都想拍死你，尤其是对git不熟悉的队友，手足无措是非常正常的表现。

rebase之后 push -f的情况只有一个，就是题主像我一样有强迫症，害怕电脑宕机、系统崩溃这种蛋疼的事情(悲惨的血泪史)，完成一个特性commit之后迅速push到远程**只属于你的分支上**，每天为了拿到develop的新特性，才在自己的分支上rebase develop,重复执行push的操作，这个个人认为是没有问题的，毕竟你只影响了你自己(而你知道这是对的)。

* [2014年03月12日回答](https://segmentfault.com/q/1010000000430041/a-1020000000431174)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/duoduo3_69)

[D咄咄](https://segmentfault.com/u/duoduo3_69)1k 声望

答案对人有帮助，有参考价值0答案没帮助，是错误的答案，答非所问

个人认为， 当你在某个分支进行团队合作的时候， 常用rebase真的是不合理。而且容易出问题。慎用 **push --force**

* [2014年03月18日回答](https://segmentfault.com/q/1010000000430041/a-1020000000439448)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/taobeier)

[TaoBeier](https://segmentfault.com/u/taobeier)562 声望

答案对人有帮助，有参考价值0答案没帮助，是错误的答案，答非所问

git rebase 一般自己一个人开发时使用，用来保持提交记录的整洁。一旦上传到github后，不应该使用git rebase,不然被骂死。

* [2014年03月20日回答](https://segmentfault.com/q/1010000000430041/a-1020000000441214)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/neil_lv)

[Neil\_吕](https://segmentfault.com/u/neil_lv)33 声望

答案对人有帮助，有参考价值0答案没帮助，是错误的答案，答非所问

一旦分支中的提交对象发布到公共仓库，就千万不要对该分支进行衍合操作。 如果你遵循这条金科玉律，就不会出差错。否则，人民群众会仇恨你，你的朋友和家人也会嘲笑你，唾弃你。

[rebase的风险](http://git-scm.com/book/zh/v1/Git-%E5%88%86%E6%94%AF-%E5%88%86%E6%94%AF%E7%9A%84%E8%A1%8D%E5%90%88#%E8%A1%8D%E5%90%88%E7%9A%84%E9%A3%8E%E9%99%A9)

* [2014年11月10日回答](https://segmentfault.com/q/1010000000430041/a-1020000000767929)

* [评论](javascript:void(0);)

* [编辑](javascript:;)

[](https://segmentfault.com/u/poke707)

[poke707](https://segmentfault.com/u/poke707)5 声望

撰写答案

窗体顶端



窗体底端

[](https://sponsor.segmentfault.com/ck.php?oaparams=2__bannerid=65__zoneid=1__cb=6c81d105d9__oadest=http%3A%2F%2Fsegmentfault.com%2Fapp)

广告

热门头条

* [性感与色情有多远——老司机带你走进鉴黄](https://segmentfault.com/p/1210000008525650)

5 赞 | 5 评论

* [笼统的前端](https://segmentfault.com/p/1210000008523754)

10 赞 | 1 评论

* [技术大牛养成指南，一篇不鸡汤的成功学实践](https://segmentfault.com/p/1210000008526301)

4 赞 | 0 评论

* [教你破解隔壁妹子的wifi密码，成功率高达90%](https://segmentfault.com/p/1210000008517323)

17 赞 | 6 评论

* [Android面试题集](https://segmentfault.com/p/1210000008524353)

3 赞 | 0 评论

[更多头条内容→](https://segmentfault.com/news)

相关收藏夹[换一组](javascript:;)

[GitLab](https://segmentfault.com/bookmark/1230000005991984)

8 个条目 | 0 人关注

[github](https://segmentfault.com/bookmark/1230000005006574)

4 个条目 | 0 人关注

[GitHub 相关](https://segmentfault.com/bookmark/1230000000727146)

125 个条目 | 86 人关注

相似问题

* [关于git rebase的问题](https://segmentfault.com/q/1010000004270637) 1 回答
* [git rebase 与 merge](https://segmentfault.com/q/1010000000753512) 1 回答
* [git rebase --onto 中onto的作用](https://segmentfault.com/q/1010000008299360)1 回答 | 已解决
* [git有没有结合 rebase --onto 与 merge --squash 的操作？](https://segmentfault.com/q/1010000004890055) 1 回答
* [git rebase本分支出现的冲突该如何解决？](https://segmentfault.com/q/1010000003938617) 1 回答
* [用git整合分支的时候，大家更常用的是变基操作(git rebase)还是合并操作(git merge)，你们觉得哪个比较好？](https://segmentfault.com/q/1010000007704573)2 回答 | 已解决
* [Git 怎么用 rebase 合并几个 commit](https://segmentfault.com/q/1010000003704970)1 回答 | 已解决
* [git rebase遇到的一个疑问](https://segmentfault.com/q/1010000002876803) 2 回答
* [git log统计某段时间内的提交如何避免rebase的干扰？](https://segmentfault.com/q/1010000000438257)
* [git rebase -i HEAD~2 pick出错。本地提交过的代码丢失](https://segmentfault.com/q/1010000002765965) 1 回答

分享扩散：

* [新浪微博](javascript:void(0);)
* [微信](javascript:void(0);)
* [Twitter](javascript:void(0);)
* [Facebook](javascript:void(0);)

Hi，欢迎来到 SegmentFault 技术社区！⊙▽⊙   
  
在这里，你可以提出编程相关的疑惑，关注感兴趣的问题，对认可的回答投赞同票；  
大家会帮你解决编程的问题，和你探讨技术更新，为你的回答投上赞同票。

立即注册明天提醒我

**网站相关**

[关于我们](https://segmentfault.com/about)

[服务条款](https://segmentfault.com/tos)

[帮助中心](https://segmentfault.com/faq)

[声望与权限](https://segmentfault.com/repu)

[编辑器语法](https://segmentfault.com/markdown)

[每周精选](https://weekly.segmentfault.com/)

[App 下载](https://segmentfault.com/app)

[社区服务中心](https://segmentfault.com/community)

**联系合作**

[联系我们](https://segmentfault.com/contact)

[加入我们](https://segmentfault.com/hiring)

[合作伙伴](https://segmentfault.com/link)

[媒体报道](https://segmentfault.com/press)

[建议反馈](https://board.segmentfault.com/)

**常用链接**

[笔记插件: Chrome](https://chrome.google.com/webstore/detail/segmentfault-%E7%AC%94%E8%AE%B0/pjklfdmleagfaekibdccmhlhellefcfo)

[笔记插件: Firefox](https://addons.mozilla.org/zh-CN/firefox/addon/sf-note-ext/)

订阅：[问答](https://segmentfault.com/feeds) / [文章](https://segmentfault.com/feeds/blogs)

[文档镜像](https://mirrors.segmentfault.com/)

[社区访谈](https://segmentfault.com/blog/interview)

[D-DAY 技术沙龙](https://segmentfault.com/d-day)

[黑客马拉松 Hackathon](https://segmentfault.com/hackathon)

[域名搜索注册](https://namebeta.com/)

[周边店铺](https://shop165859711.taobao.com/)

**关注我们**

[Github](https://github.com/SegmentFault)

[Twitter](https://twitter.com/segment_fault)

[新浪微博](http://weibo.com/segmentfault)

[团队日志](https://segmentfault.com/blog/segmentfault_team)

[产品技术日志](https://segmentfault.com/blog/segmentfault)

[社区运营日志](https://segmentfault.com/blog/community_admin)

[市场运营日志](https://segmentfault.com/blog/segmentfault_news)

**内容许可**

除特别说明外，用户内容均采用 [知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议](https://creativecommons.org/licenses/by-nc-nd/4.0/) 进行许可

本站由 [又拍云](https://www.upyun.com/?utm_source=segmentfault&utm_medium=link&utm_campaign=upyun&md=segmentfault) 提供 CDN 存储服务

Copyright © 2011-2017 SegmentFault. 当前呈现版本 17.02.05  
[浙ICP备 15005796号-2](http://www.miibeian.gov.cn/)   [浙公网安备 33010602002000号](http://www.beian.gov.cn/portal/registerSystemInfo?recordcode=33010602002000)

[回顶部](https://segmentfault.com/q/1010000000430041)