在上一节我们看到了，多人在同一个分支上协作时，很容易出现冲突。即使没有冲突，后push的童鞋不得不先pull，在本地合并，然后才能push成功。

每次合并再push后，分支变成了这样：

$ git log --graph --pretty=oneline --abbrev-commit

\* d1be385 (HEAD -> master, origin/master) init hello

\* e5e69f1 Merge branch 'dev'

|\

| \* 57c53ab (origin/dev, dev) fix env conflict

| |\

| | \* 7a5e5dd add env

| \* | 7bd91f1 add **new** env

| |/

\* | 12a631b merged bug fix 101

|\ \

| \* | 4c805e2 fix bug 101

|/ /

\* | e1e9c68 merge with no-ff

|\ \

| |/

| \* f52c633 add merge

|/

\* cf810e4 conflict fixed

总之看上去很乱，有强迫症的童鞋会问：为什么Git的提交历史不能是一条干净的直线？

其实是可以做到的！

Git有一种称为rebase的操作，有人把它翻译成“变基”。



先不要随意展开想象。我们还是从实际问题出发，看看怎么把分叉的提交变成直线。

在和远程分支同步后，我们对hello.py这个文件做了两次提交。用git log命令看看：

$ git log --graph --pretty=oneline --abbrev-commit

\* 582d922 (HEAD -> master) add author

\* 8875536 add comment

\* d1be385 (origin/master) init hello

\* e5e69f1 Merge branch 'dev'

|\

| \* 57c53ab (origin/dev, dev) fix env conflict

| |\

| | \* 7a5e5dd add env

| \* | 7bd91f1 add new env

...

注意到Git用(HEAD -> master)和(origin/master)标识出当前分支的HEAD和远程origin的位置分别是582d922 add author和d1be385 init hello，本地分支比远程分支快两个提交。

现在我们尝试推送本地分支：

$ git push origin master

To github.com:michaelliao/learngit.git

! [rejected] master -> master (fetch first)

error: failed to push some refs to 'git@github.com:michaelliao/learngit.git'

hint: Updates were rejected because the remote contains work that you **do**

hint: **not** have locally. This **is** usually caused **by** another repository pushing

hint: **to** the same ref. You may want **to** **first** integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' **in** 'git push --help' **for** details.

很不幸，失败了，这说明有人先于我们推送了远程分支。按照经验，先pull一下：

$ git pullremote: Counting objects: 3, done.remote: Compressing objects: 100% (1/1), done.remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0Unpacking objects: 100% (3/3), done.From github.com:michaelliao/learngit

d1be385..f005ed4 master -> origin/master

\* [new tag] v1.0 -> v1.0Auto-merging hello.pyMerge made by the 'recursive' strategy.

hello.py | 1 +

1 file changed, 1 insertion(+)

再用git status看看状态：

$ git status

On branch master

Your branch is ahead of 'origin/master' by 3 commits.

(use "git push" to publish your local commits)

nothing to **commit**, working tree clean

加上刚才合并的提交，现在我们本地分支比远程分支超前3个提交。

用git log看看：

$ git log *--graph --pretty=oneline --abbrev-commit*

\* e0ea545 (HEAD -> master) Merge branch 'master' of github.com:michaelliao/learngit

|\

| \* f005ed4 (origin/master) **set** exit=1

\* | 582d922 **add** author

\* | 8875536 **add** comment

|/

\* d1be385 init hello

...

对强迫症童鞋来说，现在事情有点不对头，提交历史分叉了。如果现在把本地分支push到远程，有没有问题？

有！

什么问题？

不好看！

有没有解决方法？

有！

这个时候，rebase就派上了用场。我们输入命令git rebase试试：

$ git rebaseFirst, rewinding head to replay your work on top of it...Applying: add commentUsing index info to reconstruct a base tree...M hello.pyFalling back to patching base **and** 3-way merge...Auto-merging hello.pyApplying: add authorUsing index info to reconstruct a base tree...M hello.pyFalling back to patching base **and** 3-way merge...Auto-merging hello.py

输出了一大堆操作，到底是啥效果？再用git log看看：

$ git log *--graph --pretty=oneline --abbrev-commit*

\* 7e61ed4 (HEAD -> master) add author

\* 3611cfe add comment

\* f005ed4 (origin/master) **set** exit=1

\* d1be385 init hello

...

原本分叉的提交现在变成一条直线了！这种神奇的操作是怎么实现的？其实原理非常简单。我们注意观察，发现Git把我们本地的提交“挪动”了位置，放到了f005ed4 (origin/master) set exit=1之后，这样，整个提交历史就成了一条直线。rebase操作前后，最终的提交内容是一致的，但是，我们本地的commit修改内容已经变化了，它们的修改不再基于d1be385 init hello，而是基于f005ed4 (origin/master) set exit=1，但最后的提交7e61ed4内容是一致的。

这就是rebase操作的特点：把分叉的提交历史“整理”成一条直线，看上去更直观。缺点是本地的分叉提交已经被修改过了。

最后，通过push操作把本地分支推送到远程：

Mac:~/learngit michael$ git push origin masterCounting objects: 6, done.Delta compression using up to 4 threads.Compressing objects: 100% (5/5), done.Writing objects: 100% (6/6), 576 bytes | 576.00 KiB/s, done.Total 6 (delta 2), reused 0 (delta 0)remote: Resolving deltas: 100% (2/2), completed with 1 local object.To github.com:michaelliao/learngit.git

f005ed4..7e61ed4 master -> master

再用git log看看效果：

$ git log *--graph --pretty=oneline --abbrev-commit*

\* 7e61ed4 (HEAD -> master, origin/master) add author

\* 3611cfe add comment

\* f005ed4 **set** exit=1

\* d1be385 init hello

...

远程分支的提交历史也是一条直线。

### 小结

rebase操作可以把本地未push的分叉提交历史整理成直线；

rebase的目的是使得我们在查看历史提交的变化时更容易，因为分叉的提交需要三方对比。