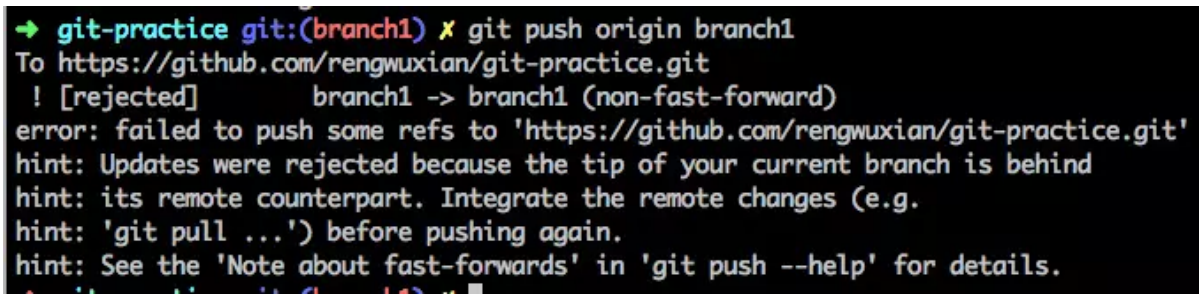
高级 6：代码已经 push 上去了才发现写错？

有的时候，代码 `push` 到了中央仓库，才发现有个 `commit` 写错了。这种问题的处理分两种情况：

1. 出错的内容在你自己的 branch

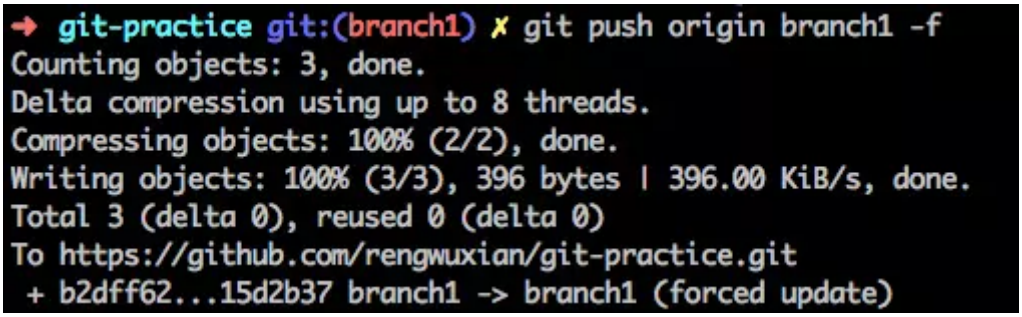
假如是某个你自己独立开发的 `branch` 出错了，不会影响到其他人，那没关系用前面几节讲的方法把写错的 `commit` 修改或者删除掉，然后再 `push` 上去就好了。不过……



由于你在本地对已有的 `commit` 做了修改，这时你再 `push` 就会失败，因为中央仓库包含本地没有的 `commit` s。但这个和前面讲过的情况不同，这次的冲突不是因为同事 `push` 了新的提交，而是因为你刻意修改了一些内容，这个冲突是你预料到的，你本来就希望用本地的内容覆盖掉中央仓库的内容。那么这时就不要乖乖听话，按照提示去先 `pull` 一下再 `push` 了，而是要选择「强行」 `push` ：

```shell
git push origin branch1 -f
```

`-f` 是 `--force` 的缩写，意为「忽略冲突，强制 `push` 」。



这样，在本地修改了错误的 `commit` s，然后强制 `push` 上去，问题就解决了。

2. 出错的内容已经合并到 master

这就不能用上面那招了。同事的工作都在 `master` 上，你永远不知道你的一次强制 `push` 会不会洗掉同事刚发上去的新提交。所以除非你是人员数量和行为都完全可控的超小团队，可以和同事做到无死角的完美沟通，不然一定别在 `master` 上强制 `push` 。

在这种时候，你只能退一步，选用另一种策略：增加一个新的提交，把之前提交的内容抹掉。例如之前你增加了一行代码，你希望撤销它，那么你就做一个删掉这行代码的提交；如果你删掉了一行代码，你希望撤销它，那么你就做一个把这行代码还原回来的提交。这种事做起来也不算麻烦，因为 Git 有一个对应的指令： `revert` 。

它的用法很简单，你希望撤销哪个 `commit` ，就把它填在后面：

```shell
git revert HEAD^
```

上面这行代码就会增加一条新的 `commit` ，它的内容和倒数第二个 `commit` 是相反的，从而和倒数第二个 `commit` 相互抵消，达到撤销的效果。

在 `revert` 完成之后，把新的 `commit` 再 `push` 上去，这个 `commit` 的内容就被撤销了。它和前面所介绍的撤销方式相比，最主要的区别是，这次改动只是被「反转」了，并没有在历史中消失掉，你的历史中会存在两条 `commit` ：一个原始 `commit` ，一个对它的反转 `commit` 。

小结

这节的内容是讲当错误的 `commit` 已经被 `push` 上去时的解决方案。具体的方案有两类：

1. 如果出错内容在私有 `branch` ：在本地把内容修正后，强制 `push` （ `push -f` ）一次就可以解决；
2. 如果出错内容在 `master` ：不要强制 `push` ，而要用 `revert` 把写错的 `commit` 撤销。