Git commit-tree parlor tricks, Part 8: I just rebased my branch, but now I realize that I should have merged

devblogs.microsoft.com/oldnewthing/20200921-00

September 21, 2020

Raymond Chen

Suppose you created a feature branch in git and made some changes. Meanwhile, the main branch has also made some changes:

```
A    M1    M2    master

     F1    F2    feature
```

From a common ancestor commit A, we create a feature branch and make two commits, F1 and F2. Meanwhile, the master branch has received two commits M1 and M2.

You decide to rebase your topic branch onto the main branch. Many merge conflicts later, you finish with this:

```
A    M1    M2                    master

                 F1′    F2′      feature
```

The resulting graph is now linear, with the original commits A1, M1 and M2, followed by new commits F1′ and F2′.

And then you realize that what you really meant to do was merge, not rebase. Is there a way to convert the rebase into a merge without having to go back and deal with all those merge conflicts again?

Indeed there is.

We've seen something very similar before, when we retroactively converted a squash to a merge. This is pretty much the same thing: We have a final result, and we want to manufacture a merge that has the same final result.

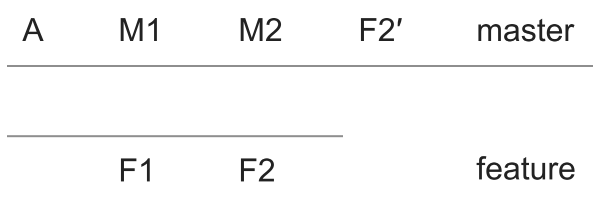
```
git commit-tree HEAD^{tree} -p M2 -p F2 -m comment
```

Note: If using the Windows `cmd` command prompt, you need to type

```
git commit-tree HEAD^^{tree} -p M2 -p F2 -m comment
```

for reasons discussed earlier.

What we did was manufacture a new commit that contains the same results as F2′, but assigned it the parents M2 and F2. The first parent is the branch you want to pretend that you are merging *to*, and the second parent is the branch you want to pretend that you are merging *from*.



The output of the `git commit-tree` command is a commit hash. You can now reset to that commit, and all will be forgiven.

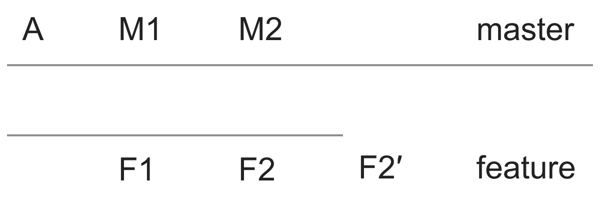
```
git reset --soft ⟨hash⟩
```

Resetting in soft mode preserves all the changes you may have staged. Those staged changes are still valid because the starting point hasn't changed: The commit you are resetting to has the same tree as the commit you are moving from.

Bonus chatter: If we had swapped the two parent commits, like this:

```
git commit-tree HEAD^{tree} -p F2 -p M2 -m comment
```

then the result would have been



Raymond Chen

Follow