I did that merge-as-cherry-pick thing, but my change still didn't merge correctly

devblogs.microsoft.com/oldnewthing/20220929-00

September 29, 2022

Raymond Chen

A colleague used the patch-branch technique of merging a single commit to another branch, but found that the subsequent merge of the complete branch produced the wrong results. They asked me to investigate what went wrong.

After untangling the changes to the main and feature branches, I was able to reconstruct a simplified version of what happened.

We start with this:

	apple	apple	
	M1	M2	main
apple			
A			

	F1	F2	feature
	apple	berry	

They created a feature branch and did some work: Commit F1 didn't affect the file, but commit F2 changed "apple" to "berry".

They then realized that this "berry" change was something they wanted in the main branch early, so they could do some preliminary integration work before the rest of the feature work was done. They created a second berries-only branch that contained just the "berry" change and merged it into both the main branch (which delivers the payload to the main branch) and

into the feature branch (which has no net code effect, but records that the change as have already been incorporated so it doesn't get counted as payload when the feature branch merges back up to the main branch).

	apple	apple	berry	
	M1	M2	M3	main
apple		berry		
A		P		patch

	F1	F2	F3	feature
	apple	berry	berry	

The integration validation didn't turn out so great, so they reverted the change in both the main and feature branches.

	apple	apple	berry	apple	
	M1	M2	M3	M4	main
apple		berry			
A		P			patch

	F1	F2	F3	F4	feature
	apple	berry	berry	apple	

The team continued working on the feature, and this time they felt that they got the "berry" thing right, so they made a commit in their feature branch to change "apple" back to "berry", this time with feeling.

	apple	apple	berry	apple	
	M1	M2	M3	M4	main
apple		berry			
A		P			patch

	F1	F2	F3	F4	F5	feature
	apple	berry	berry	apple	berry	

All the tests were passing, so they got the green light to merge the feature into the main branch. And that's where something strange happened.

	apple	apple	berry	apple		apple	
	M1	M2	M3	M4		M5	main
apple		berry					
A		P					patch

	F1	F2	F3	F4	F5		feature
	apple	berry	berry	apple	berry		

The result of the merge into the main branch didn't carry the final "berry" change. The file remained "apple" in the main branch.

As a result, the main branch was broken.

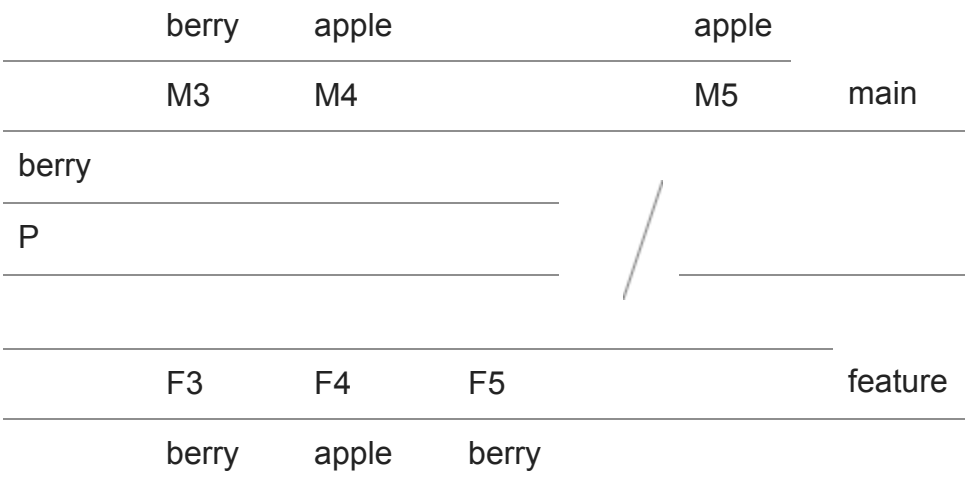
What happened?

The merge of the "only berries" branch into both the main and feature branches established "berry" as the baseline for the next merge. In the main branch, "berry" changed to "apple". In the feature branch, "berry" underwent some turmoil but emerged unchanged. The three-way merge algorithm therefore saw that the main branch changed "berry" to "apple", and the feature branch made no (net) change. Therefore, the result of the merge is "apple". (apple + 0 = apple).

The fatal error was the dual revert.

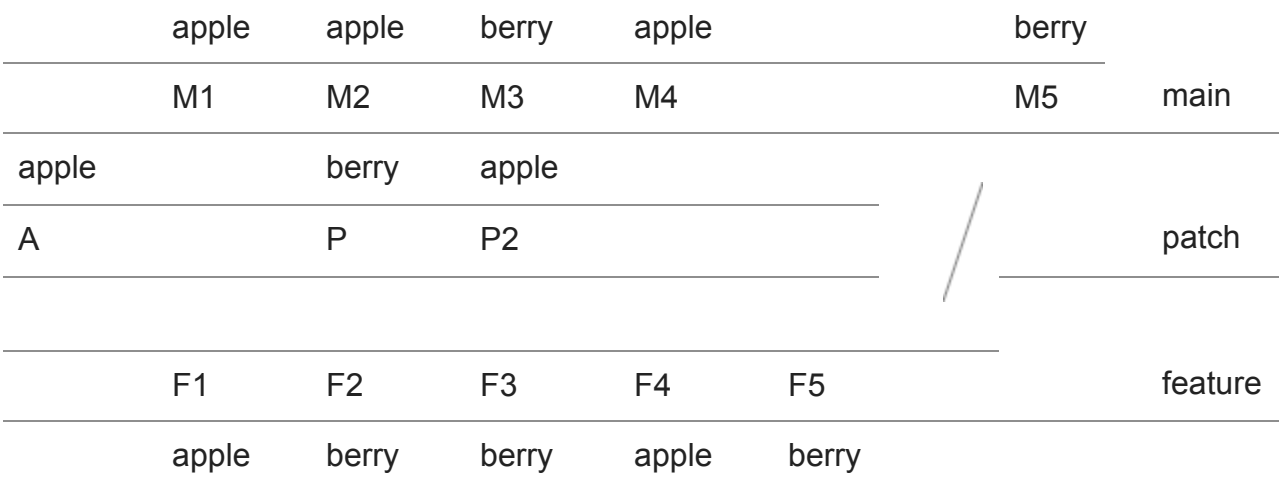
These reverts were independent and therefore git did not consider them to be related to each other. But really, you wanted them to be considered the same revert, so that the feature branch could un-revert it.

Basically, once you merged the "only berries" branch, the diagram (from the apple/berry's point of view) simplified to this:



It's as if the feature branch was created from the "only-berries" branch, and the main branch continued by reverting "berries" back to "apple", whereas the feature branch underwent some soul-searching and ultimately made no change to "berries". Naturally, the result of this merge is that "berries" is reverted to "apple".

What my colleague should have done was to perform the revert in a separate branch, possibly just extending the "only-berries" branch so it is now a "changed-my-mind-about-berries" branch. Merge that branch into both the main and feature branches, thereby advancing the baseline forward to the shared revert.



By making the revert on the patch branch and merging it into the main and feature branches (instead of reverting separately on the main and feature branches), git now understands that the revert is part of the shared history of the two branches. This time, when the final merge occurs, git sees that the main branch made no changes, and the feature branch did an unrevert, so the result of the merge is the unrevert, and "berry" makes it into the main branch.

Raymond Chen

Follow