Stop cherry-picking, start merging, Part 6: Replacing the temporary fix with the permanent fix

devblogs.microsoft.com/oldnewthing/20180319-00

March 19, 2018

Raymond Chen

Continuing our exploration of using merges as a replacement for cherry-picking, here's another scenario you can now solve:

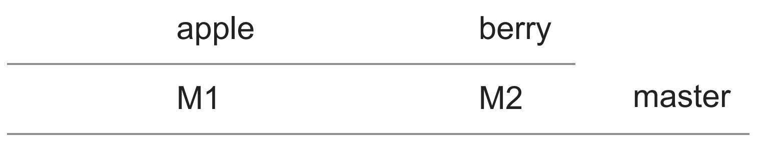
How do I make a change in the master branch that will be automatically reverted when the feature branch merges in?

For this scenario, you have a starting commit A from which you create a feature branch. Both the master branch and feature branch add additional commits M1 and F1, respectively, and then you discover a problem that is common to both branches. You want to apply a temporary fix to the master branch while you work on a permanent fix in the feature branch. How do you arrange so that the temporary fix in the master branch will be reverted automatically when the permanent fix merges in?

Okay, first the thing that doesn't work: Make the temporary fix in the master branch, cherry-pick it into the feature branch, and then revert it in the feature branch. This doesn't work because you have essentially created the ABA problem.

One way to solve this problem is to use the patch branch technique to apply the temporary fix to both branches, and then immediately revert it in the feature branch. This is basically explicitly creating an ABA situation, but doing so in a way that will cause the A to be the winner during the eventual merge.

Another way to solve the problem is to skip the patch+revert step and replace it with an ignore (which is basically what the patch+revert is). Create the patch branch as usual, merge it into the master branch as usual, but instead of performing a normal merge into the feature branch, ignore the change.



apple		berry		
A		P		patch
	F1		F2	feature
	apple		apple	

We create a patch branch from the common commit A and apply the temporary fix to it (changing `apple` to `berry`), producing commit P. We merge that branch into the master branch (producing commit M2 with the temporary fix), and we also merge it into the feature branch with `-s ours` to indicate that we don't want any code changes from this merge; we are creating it only for bookkeeping purposes. The line remains `apple` in the feature branch, so that you can continue debugging the problem and developing a permanent fix.

When the two branches merge, the patch branch will be used as the merge base, and relative to that choice of merge base, the master branch didn't change anything, whereas the feature branch "changed" it to `apple`. The merge will therefore change the line back to `apple`. Presumably you won't perform this merge until you have a permanent fix in the feature branch!

If you use an online service to manage your pull requests, and that online service doesn't provide a way to override the merge algorithm, then you can use the pre-merge technique described earlier.

What if I already made a change in the master branch that I want to be automatically reverted when the feature branch merges in? Can I create a patch branch retroactively?

Yes, you can still create a patch branch retroactively, using the same technique we have been using throughout this series. Create a patch branch with a copy of the temporary fix, merge it into the master branch (where it will do nothing because the temporary fix is already in the master branch), and also merge it into the feature branch with `-s ours` to indicate that you want no code change, just bookkeeping.

We continue with another question next time.

Raymond Chen

Follow