How can I create a git feature branch that can merge into multiple other branches?

devblogs.microsoft.com/oldnewthing/20230315-00

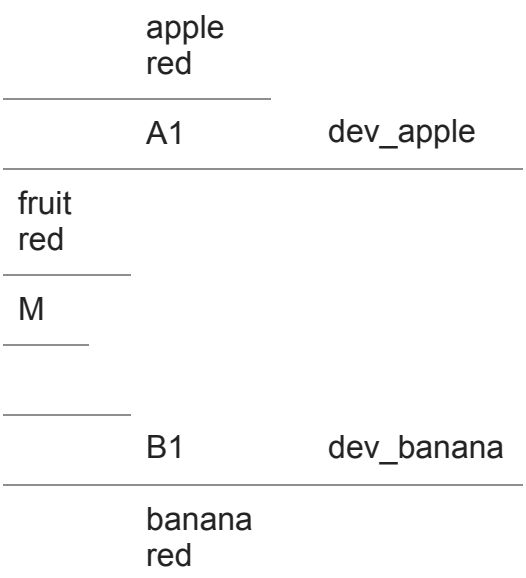
March 15, 2023

Raymond Chen

A customer had a git repo that had two branches, let's call them `dev_apple` and `dev_banana`. These branches were very similar, differing only in a choice of fruit. The customer wanted to create a new branch, call it `feature`, in which they could develop a feature that was not fruit-dependent. When finished, they could then create two pull requests, one to merge `feature` into `dev_apple` and another to merge `feature` into `dev_banana`. Is this possible?

Yes, it's possible, and we already explored how it works as a side effect of an earlier investigation into merging as a substitute for cherry-picking.

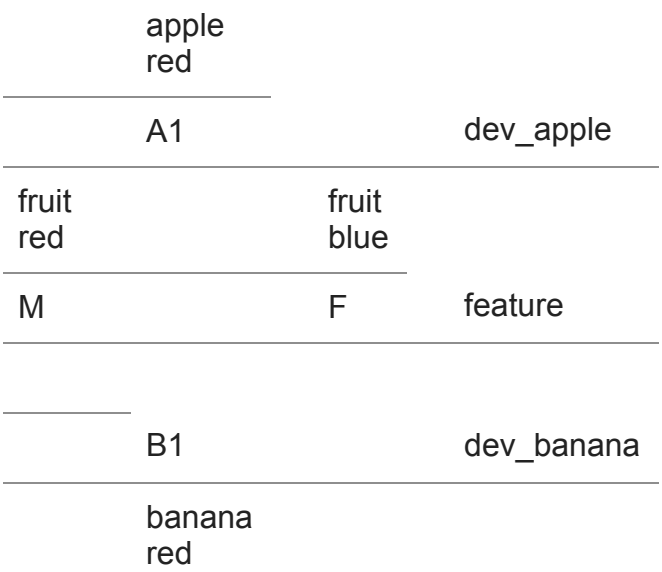
Suppose the repo looks like this:



At some starting commit `M`, the code said "fruit red". From the commit, the `dev_apple` branch commited a change to change the code to read "apple red". Meanwhile, from the same starting commit, the `dev_banana` branch commit a change to change the code to "banana red".

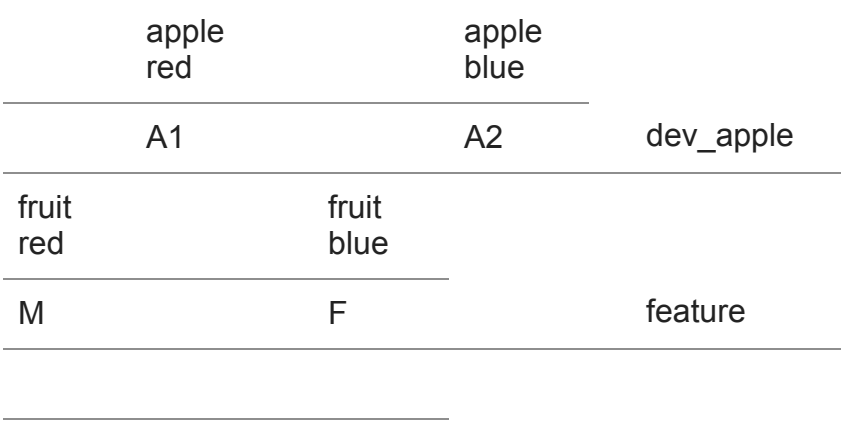
Your goal is to create some branch `feature` that changes `red` to `blue`, and then merge that branch into both the `dev_apple` and `dev_banana` branches. The expected result of the merge is that `red` changes to `blue`, but the fruit in each branch remains unchanged. Merging into `dev_apple` produces "apple blue", and merging into `dev_banana` produces "banana blue".

You already know how to do this: Create a patch branch that starts at the common ancestor of the two `dev` branches, which in our example is the initial commit `M`.



In this scenario, the patch branch is what we're calling the `feature` branch. In that branch, we can make a commit that makes the changes we want to apply to both of the `dev` branches, namely, changing the code to "fruit blue".

Once we're happy with the work we've done in the `feature` branch (which could consists of several commits), we can create pull requests to merge the changes into both of the `dev` branches.



B1	B2	dev_banana
banana red	banana blue	

The result of these merges is that the `dev_apple` branch says “apple blue” and the `dev_banana` branch says “banana blue”, as desired.

Note that this trick assumes that the most recent common ancestor of the `dev_apple` and `dev_banana` branches is not too old, or at least not so old that the code you want to change isn’t even present. In our case, we’re in good shape because the common ancestor commit `M` does have the word “red” that we want to change to “blue”.

The customer was satisfied with this recommendation. My guess is that the split into `dev_apple` and `dev_banana` was relatively recent and temporary, and the expectation was that the two `dev` branches would eventually merge back together once the fruit discrepancy was resolved.