Pull in changes to feature branch from master

Say you branch off from the main branch and are currently working on a feature in Git. Suddenly, your co-worker pushes changes to the main branch (this is entirely possible), while you are still working on the feature branch, and you would like to have that new changes that your co-worker have pushed to main incorporated into your current feature branch that you are working on.

How do you do it? There are two ways of handling it:

1. Merge master into feature/branch

This way is the recommended way since it doesn't require force pushing the commit flow into the repository.

So how it works is that you will first `git fetch` in the changes that your co-worker have done from the remote repository. This will sync up the remote-tracking branches like origin/main in your repository. Remember these branches reflect the branch state in the remote repository thus their name.

Then you can just run `git merge origin/main` to merge in the changes that your co-worker has done.

> If you want to sync up the main branch in your local repository first then you would have to run `git checkout main` then run `git merge` to pull in the hcanges from your co-worker. Then finally you can run `git merge main` to merge from your local main branch instead of remote-tracking branch. Both way works it is just whether or not you want to sync up your local main branch as well.

2. Rebase feature/branch on master

This way will make your git history look cleaner without all the merge commits, but without the track that it was merged into the branch. This is less visible for tracking commit histories. However, this will require a **force push** if you have commits in the feature branch already and is pushed to the remote.

So again fetch the changes from the remote repository by running `git fetch`.

Then you would just run `git rebase origin/main` on the feature branch to rebase your branch on the latest remote-tracking branch changes. Or you can sync up the local main then rebase it on main, it works the same way.

> Why exactly do you need a force push? This is because after you rebase the feature branch to master (incorporating those changes that your co-worker have been making) the existing commits that you have made will be **rehashed!** The changes will remain the same, but because those commits have different parents now the hashes will be different. Hence after you rebase to incorporate the changes and pushes to main after you will need a force push since the remote branch's history is completely different now compared to the local one. Doesn't mean it is wrong, but yea a force push is required.

A force push isn't needed if you didn't make any commits to the branch yet (so it is even with main, and so is the remote branch), it will just add in the changes from main, that is all. You can simply `git push` to the remote feature branch to make the feature branch stay up to date with the main.

Summary

Either way they both achieve pulling in your co-worker's change but in different ways. So pick your poison, if you like cleaner git history then go with the second method, but if you want visibility of when someone merged in what then first way would be the way to go. At the end of the day, it is just preferences.

> This may or may not require you to force push

Revision #2
Created 2023-04-01 01:12:58 UTC by Tamarine
Updated 2023-04-01 03:59:23 UTC by Tamarine