Roman Petrenko

15 Followers About Follow

Roman Petrenko Mar 21, 2018 · 3 min read

When working with Git you will find that sometimes commits need to be removed as they have introduced a bug or need to be reworked.

If it is the last commit this is very straight forward. Simply run:

```
git reset HEAD
```

This will pop off the latest commit but leave all of your changes to the files intact.

If you need to delete more than just the last commit there are two methods you can use. The first is using **rebase** this will allow you to remove one or more consecutive commits the other is **cherry-pick** which allows you to remove non consecutive commits.

Example git log

NumberHashCommit MessageAuthor12c6a45b(HEAD) Adding public method to access protected methodTom2ae45fabUpdates to database interfaceContractor 1377b9b82Improving database interfaceContractor 243c9093cMerged develop branch into masterTom5b3d92c5Adding new Event CMS ModulePaul67feddbbAdding CMS class and filesTom7a809379Adding project to GitTom

Using Rebase

Using the git log above we want to remove the following commits; 2 & 3 (ae45fab & 77b9b82). As they are consecutive commits we can use rebase.

```
git rebase --onto <branch name>~<first commit number to remove> <branch name>~<first commit to be kept> <branch name>
```

e.g to remove commits 2 & 3 above

```
git rebase --onto repair~3 repair~1 repair
```

Using Cherry Pick

Step 1: Find the commit before the commit you want to remove `git log`

Step 2: Checkout that commit `git checkout <commit hash>`

Step 3: Make a new branch using your current checkout commit `git checkout -b <new branch>`

Step 4: Now you need to add the commit after the removed commit `git cherry-pick <commit hash>`

Step 5: Now repeat Step 4 for all other commits you want to keep.

Step 6: Once all commits have been added to your new branch and have been commited. Check that everything is in the correct state and working as intended. Double check everything has been commited: `git status`

Step 7: Switch to your broken branch `git checkout <broken branch>`

Step 8: Now perform a hard reset on the broken branch to the commit prior to the one your want to remove `git reset --hard <commit hash>`

Step 9: Merge your fixed branch into this branch `git merge <branch name>`

Step 10: Push the merged changes back to origin. **WARNING: This will overwrite the remote repo!** `git push --force origin <branch name>`

You can do the process without creating a new branch by replacing **Step 2 & 3** with **Step 8** then not carry out **Step 7 & 9.**

Example

Say we want to remove commits 2 & 4 from the repo.

1. `git checkout b3d92c5` Checkout the last usable commit.
2. `git checkout -b repair` Create a new branch to work on.
3. `git cherry-pick 77b9b82` Run through commit 3.
4. `git cherry-pick 2c6a45b` Run through commit 1.
5. `git checkout master` Checkout master.
6. `git reset --hard b3d92c5` Reset master to last usable commit.
7. `git merge repair` Merge our new branch onto master.
8. `git push --hard origin master` Push master to the remote repo.

Final note

Git rebase & cherrypick are dangerous but powerful solutions that should only be used as a last option and only be undertaken by someone who knows what they are doing. Beware that both solutions could have adverse effects on other users who are working on the same repository / branch.

Finally remember to be careful and good luck!

original blog you can find on : https://www.clock.co.uk/insight/deleting-a-git-commit

Git