Revert pushed merge

Tuesday, May 1, 2018 2:34 PM

first checkout the master branch:

```
git checkout master
```

then run a git log and get the id of the merge commit.

```
git log
```

then revert to that commit:

```
git revert -m 1 <merge-commit>
```

With '-m 1' we tell git to revert to the first parent of the mergecommit on the master branch. -m 2 would specify to revert to the first parent on the develop branch where the merge came from initially.

Now commit the revert and push changes to the remote repo and you are done.

From <https://www.christianengvall.se/undo-pushed-merge-git/>

How to revert a merge commit in Git

Fri 09 December 2016 by Johannes Bornhold in tools Tags git

Reverting a commit - git revert

Within this article I focus on the command git revert. It can be used to create a *new* commit, which will undo the changes introduced by other commits.

It is important to understand that it will always add a new commit to undo the change. Typically this is the right thing to do when a change has been introduced in the past and shall be removed again, e.g. because it turned out to introduce some unnoticed negative side-effects.

If you want to remove something from the history instead, then revert is not the right tool for the job. In such a case better look at commands to edit the history, but be warned that editing the history usually causes a good chunk of confusion when applied to shared repositories.

Revert a merge

A merge in Git is a commit which has at least two parents. It brings together multiple lines of development. In a work-flow where features are developed in branches and then merged into a mainline the merge commits would typically have two parents.

Just running git revert MERGE_HASH will not work as expected:

```
$ git revert 603c1333339bc9b5ad4d8b864e948d4bd950bf05
error: Commit 603c1333339bc9b5ad4d8b864e948d4bd950bf05 is a merge but no -m option was given.
fatal: revert failed
```

The reason is that a merge commit has multiple parents and revert needs additional information to decide which parent of the merge shall be considered as the mainline. The parameter -m has to be used in this case.

When looking at the merge commit, you can see the parents:

```
$ git show 603c1333339bc9b5ad4d8b864e948d4bd950bf05
commit 603c1333339bc9b5ad4d8b864e948d4bd950bf05
Merge: d595b4a 8cf6231
Author: Johannes Bornhold <johannes@bornhold.name>
Date:   Sun Sep 6 11:31:16 2015 +0200
Merge remote-tracking branch 'origin/master'
```

In the following example, I am using -m 1 to select the first parent as the mainline:

```
git revert 603c1333339bc9b5ad4d8b864e948d4bd950bf05 -m 1
Waiting for Emacs...
[spacemacs-layer 2606e9c] Revert "Merge remote-tracking branch 'origin/master'"
1 file changed, 1 deletion(-)
```

Suddenly it works out.

Revert the revert - when continuing the change

If work shall be continued in the old branch, then you have to prepare a bit more, since the old commits are still recorded as being merged. Remember from the first section that revert adds a new commit on top of the current one, the original merge commit is still in the history. This means that there is nothing left to be merged from the old branch into your mainline.

One option to approach this is to add a "revert of the revert" to start a fresh branch for continuing the work on the problematic feature.

From <https://www.johbo.com/2016/how-to-revert-a-merge-commit-in-git.html>