Alternative to Squash Commit

If we have three commits that we want to squash as below, we can simply use git reset, add and commit to squash those changes into one commit.

<pre>

commit edba792d5ea2aec40f413a23bf539fa25270da65

Author: mdh

Date:   Thu Feb 27 09:00:16 2020 +1100

    Updated model3

commit 8ea10ef480c8cde59b2d78ae0fbe5367f877e59d

Author: mdh

Date:   Thu Feb 27 08:59:51 2020 +1100

    Updated model 2

commit d5da1aed0b45d546e95e9bc01207d73a115ef337

Author: mdh

Date:   Thu Feb 27 08:59:20 2020 +1100

    updated model

commit 7bd22703c418ffa9bd92cbbb06fb92926776211e

Author: mdh

Date:   Thu Feb 27 08:58:36 2020 +1100

    initial commit

</pre>

First, we reset to the initial commit. Reset unstage the change, but the change still remains.

<pre>

$ git reset 7bd22703c418ffa9bd92cbbb06fb92926776211e

Unstaged changes after reset:

M       CsvProcessor/Models/UniversityRankingModel.cs

M       CsvProcessor/Program.cs

</pre>

Then, we can add and commit as usual. It will create a single commit with all the changes.

<pre>

commit dc41d5ae2950b962c5cc5313caa1160429db8d25 (HEAD -> squash-test)

Author: mdh

Date:   Thu Feb 27 09:21:25 2020 +1100

    model update

commit 7bd22703c418ffa9bd92cbbb06fb92926776211e

Author: mdh

Date:   Thu Feb 27 08:58:36 2020 +1100

    initial commit

</pre>

Of course, you can try to rebase -i on the number of commit or sha

<pre>

> git rebase -i HEAD~3 (this means rebasing in the last 3 commits)

or

> git rebase -i [SHA]

or simply rebase to the master to see all the branch commit

> git rebase -i master

</pre>

On the vim editor we can make the beginning of the commit as pick and the rest as squash. As rebasing changed the commit history, you need to force push to the remote branch.

<pre>

pick 5641b70 model update1

squash 3c5fe33 model update2

squash 83b7e45 model update3

# Rebase 7bd2270..83b7e45 onto 7bd2270 (3 commands)

#

# Commands:

# p, pick <commit> = use commit

# r, reword <commit> = use commit, but edit the commit message

# e, edit <commit> = use commit, but stop for amending

# s, squash <commit> = use commit, but meld into previous commit

# f, fixup <commit> = like "squash", but discard this commit's log message

# x, exec <command> = run command (the rest of the line) using shell

# d, drop <commit> = remove commit

# l, label <label> = label current HEAD with a name

# t, reset <label> = reset HEAD to a label

# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]

# .       create a merge commit using the original merge commit's

# .       message (or the oneline, if no original merge commit was

# .       specified). Use -c <commit> to reword the commit message.

</pre>

Once wq on the vim editor, the rebase gets completed. Then, we will have a nice commit histroy.

<pre>

commit 9f8b36612acc169886fd8c5c9285888e3a763d63 (HEAD -> squash-test)

Author: Takahiro Honda <takahiro.honda@open.edu.au>

Date:   Thu Feb 27 10:12:11 2020 +1100

    model update1

    model update2

    model update3

commit 7bd22703c418ffa9bd92cbbb06fb92926776211e

Author: Takahiro Honda <takahiro.honda@open.edu.au>

Date:   Thu Feb 27 08:58:36 2020 +1100

    initial commit

</pre>

For squash commit, check the git documentation: <https://github.com/wprig/wprig/wiki/How-to-squash-commits>