GIT: From Beginner To Fearless

GIT Revert Activity:
Undoing a public commit using revert

Brian Gorman, Author/Instructor/Trainer

Introduction

Sometimes we have made a commit and we need to undo the commit. In many cases, when the commit is private, we can simply 'reset' back to where we want to go. However, there are times when a commit has been made public. In these cases, one of the safest ways to undo the commit is to use the revert command.

In essence, a revert command simply reverses the original commit(s) and gets the repository back to the state where it was before the reverted commit(s). The system then records a new commit that manages the "undo" operations, and allows us to publish a public commit that will allow all other dependent users to be able to easily get the latest changes, which include the revert commit with changes reverted. The public commit allows for history to remain in tact. As with any merge, reset, rebase, and/or pull operation, a revert requires any conflicts to be resolved for the operation to complete successfully.

Let's gets started!

Step 1: Make sure you have a valid working repository, where a pull request was just closed on master.

a) Clone the repo with a commit to revert or get the latest on your existing repo.

[**git clone <repo> <folder>**]

[cd <folder>]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder
$ git clone https://github.com/majorguidancesolutions/SimpleActivityRepo.git Gi
RevertDemo
Cloning into 'GitRevertDemo'...
remote: Counting objects: 30, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 30 (delta 15), reused 23 (delta 8), pack-reused 0
Unpacking objects: 100% (30/30), done.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder
$ cd GitRevertDemo/

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$
```

--if getting latest:

[git checkout master]

[git fetch origin]

[git pull origin master]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$ git checkout master
Already on 'master'
Your branch is up-to-date with 'origin/master'.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$ git fetch origin

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$ git pull origin master
From https://github.com/majorguidancesolutions/SimpleActivityRepo
 * branch            master     -> FETCH_HEAD
Already up-to-date.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$
```

b) Create a new branch for changes

[git checkout -b GitRevertDemo]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDem
$ git checkout -b GitRevertDemo
Switched to a new branch 'GitRevertDemo'
```

Notes

Step 2: Create a simple commit chain, then revert one commit:

a) Make a small change

[code info.txt]

[git status]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (GitRevertDemo)
$ code info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (GitRevertDemo)
$ git status
On branch GitRevertDemo
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   info.txt
```

b) Add and commit the change

[git commit -am "I want to revert this change"]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevert
(GitRevertDemo)
$ git commit -am "I want to revert this change"
[GitRevertDemo 2b482c4] I want to revert this change
 1 file changed, 2 insertions(+)
```

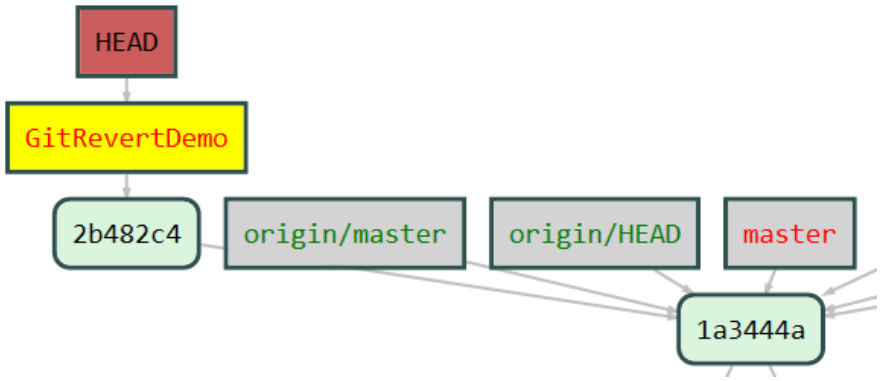
c) View the log – find the commit to revert to, revert it.

[git log --oneline]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo
(GitRevertDemo)
$ git log --oneline
2b482c4 (HEAD -> GitRevertDemo) I want to revert this change
1a3444a (origin/master, origin/HEAD, master) Merge pull reque
t #3 from majorguidancesolutions/GitAmendDemo
0cac76b Changed info.txt and added readme.txt
```

d) Revert the commit.

NOTE: it is important to NOT enter the commit you want to revert TO, but only the commit you want to revert. IF there are other commits, they would be reverted. For example, on this chain:



If I stated [git revert 1a3444a], I would actually be trying to revert BOTH 2b482c4 and 1a3444a from where I am at, and that would be a mess. Instead, I want to revert 2b482c4 -> which essentially should get my repo to be the same as 1a3444a, just with both commits.

[git revert 2b482c4]

COMMIT_EDITMSG

```
Revert "I want to revert this change"

This reverts commit 2b482c4db28ea811a6109ad7f6d883f3c123a48e.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch GitRevertDemo
# Changes to be committed:
#   modified:   info.txt
#
```

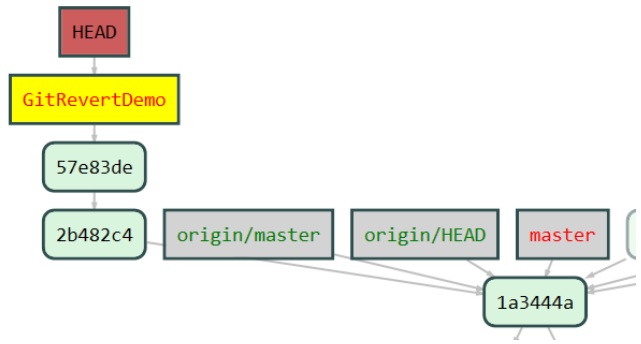
I just need to enter a message to perform the revert. I changed it to 'reverted a bad change'

COMMIT_EDITMSG

```
Reverted a bad change

This reverts commit 2b482c4db28ea811a6109ad7f6d883f3c123a48e.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit
```

Save and Exit:

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRev
(GitRevertDemo)
$ git revert 2b482c4

[GitRevertDemo 57e83de] Reverted a bad change
 1 file changed, 2 deletions(-)

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRev
(GitRevertDemo)
$
```



[git log --oneline]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRever
(GitRevertDemo)
$ git log --oneline
57e83de (HEAD -> GitRevertDemo) Reverted a bad change
2b482c4 I want to revert this change
1a3444a (origin/master, origin/HEAD, master) Merge pull
t #3 from majorguidancesolutions/GitAmendDemo
```

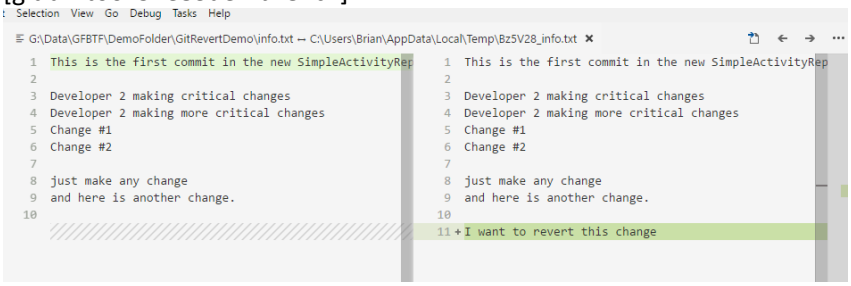
e) See the differences

[git difftool 57e83de 1a3444a]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/
(GitRevertDemo)
$ git difftool 57e83de 1a3444a

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/
(GitRevertDemo)
$
```

//as expected, no difference, difftool not launched

[git difftool 57e83de 2b482c4]



[git difftool 2b482c4 1a3444a]

f) If you want to go further

If you want a bigger challenge, make a couple of commits and revert the first commit in the chain (you'll need to revert everything back that you did)

If you want an even bigger challenge, revert a merge commit. That will require you being able to determine parents. Here are a couple of hints.

1) You can always see merge commit parents by using the command:
   [git show --pretty=raw <commit>]
2) You can dive into the commit parent differences with the following commands
   [git show <commit>^1] //parent 1
   [git show <commit>^2] //parent 2

g) Delete your branch that essentially has no changes [no need to merge]

[git branch –D <branchname>] //need to force it since has unmerged commits

This concludes our GIT Revert Activity.

Closing Thoughts

Using Git Revert is one way we can 'undo' a commit while keeping the commit history in tact. For this demo, we took a quick look into the command and hit a pretty easy revert scenario. Much more difficult revert scenarios exist, however this is beyond the detail I wanted to spend on our first encounter with the command.

The nice thing about a revert operation is that as long as there were no conflicts to resolve [such as when reverting a merge commit], the system will auto-revert for us and we just need to enter a message.

The main thing to remember is that we don't want to enter the commit id of the commit we want to get back to, but rather, we want to enter the commit id of the first commit (and any of its descendants) that we want to undo.

The last part gives a couple of challenges if you want to go deeper with the revert command.

Take a few minutes to make some notes about the various commands we've learned about in this activity, and practice using them.

Notes