GIT: From Beginner To Fearless

GIT Commit with Amend Activity:
A simple exercise using git commit with amend

Brian Gorman, Author/Instructor/Trainer

Introduction

Git makes it possible for us to "change" a commit. Sometimes we want to change the commit message. Other times we might want to actually make a small change, perhaps to fix a bug or remove some commented code that we don't want in the repo that we forgot to remove before committing.

In the case where we want to fix something and rewrite the commit history, we can do so with the git amend command.

One thing about rewriting commit history that we need to be very clear of before we begin. Once we have made our commit changes public, we should try to avoid rewriting history unless we are certain it will not affect anyone else. For example, we wouldn't want to merge our code to master, have another developer then branch from that merge, then amend the merge commit. If that happened, the other developer would have to do some work to get their code to merge back to master since their history will not contain the common commit id that it is expecting to point to.

Therefore, while we can perform commits that amend, we should limit this to our own private commit chains as much as possible.

Keeping all that in mind, let's gets started!

Step 1: Amend a commit to change the commit message.

a) Make sure you have any repository up to date and are working on any branch.

[**clone repo**]

Or

[**git checkout master**]

[**git fetch origin**]

[**git pull origin master**]

[**git checkout -b GitAmendDemo**]

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

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

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (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/DemoFold
$ git checkout -b GitAmendDemo
Switched to a new branch 'GitAmendDemo'
```

Make a small change, add and commit.

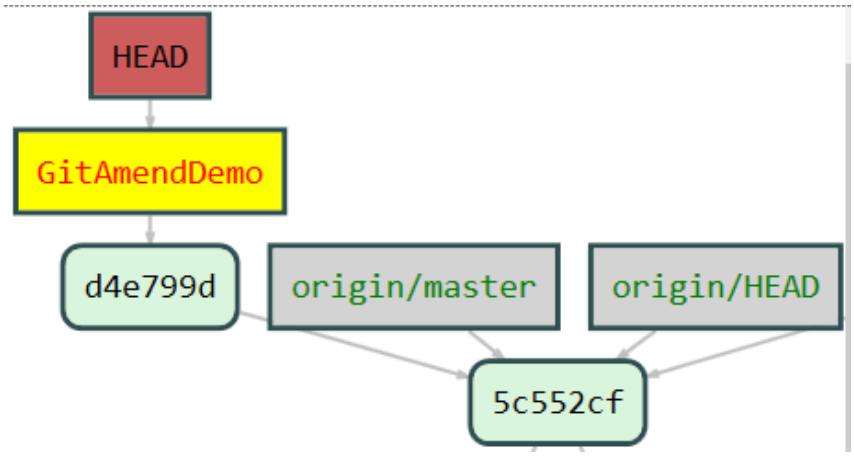
[**code info.txt**]

[**git status –s**]

[**git commit -am "This is my commit messag"**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo
$ code info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo
$ git status -s
 M info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo
$ git commit -am "This is my commit messag"
[GitAmendDemo d4e799d] This is my commit messag
 1 file changed, 2 insertions(+)

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo
```

Notes

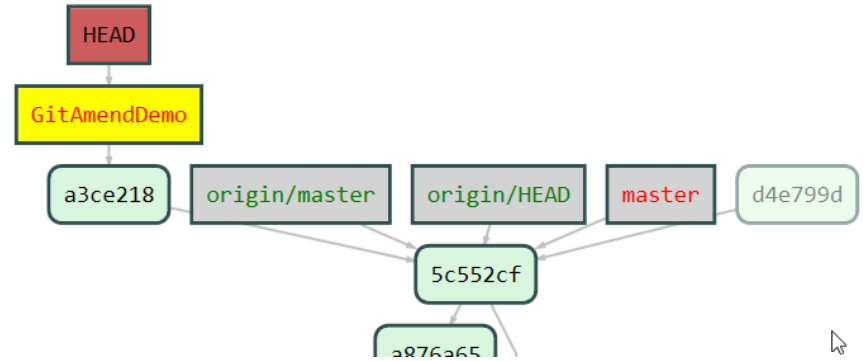


Note the commit id [**d4e799d**] – yours will be different, of course

b) The message had a typo, let's fix it.

[**git commit --amend -m "This is an important change to the code"**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (GitAmendDemo)
$ git commit --amend -m "This is an important change to the code"
[GitAmendDemo a3ce218] This is an important change to the code
 Date: Sat Sep 23 21:45:21 2017 -0500
 1 file changed, 2 insertions(+)
```



Note that the commit id is different, so we would need to be careful on a public branch if we are using an amend. We can also see the old commit is still showing as unreachable. Therefore, we could do a quick cleanup.

Step 2: Clean up the unreachable commits.

a) Use reflog to expire unreachable and then cleanup with the garbage collector

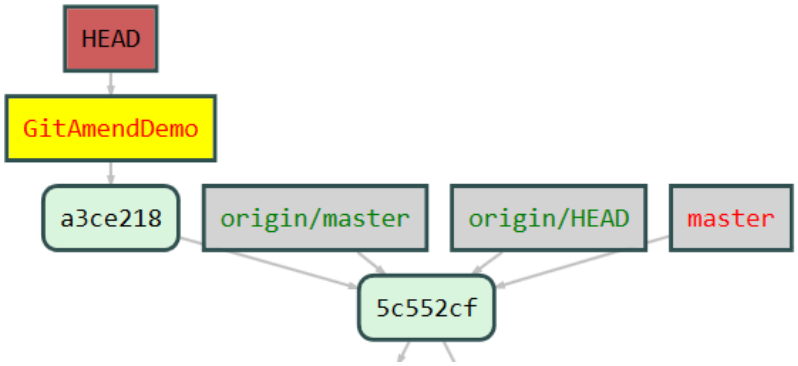
To clean up the commits we just need to make sure we have the reflog set to expire our commits and then run the garbage collector.

[**git reflog expire --expire-unreachable=now –all**]

And

[**git gc --prune=now**]

```
blgor@CORSAIRONE MINGW64 /d/Data/Test_Run/DefaultWeb-m
itAmendDemo)
$ git reflog expire --expire-unreachable=now --all

blgor@CORSAIRONE MINGW64 /d/Data/Test_Run/DefaultWeb-m
itAmendDemo)
$ git gc --prune=now
Enumerating objects: 67, done.
Counting objects: 100% (67/67), done.
Delta compression using up to 8 threads
Compressing objects: 100% (55/55), done.
Writing objects: 100% (67/67), done.
Total 67 (delta 22), reused 0 (delta 0)
```



Forcing the garbage collector to run is a good way to clean up the unreachable commits.

Step 3: Amend with some actual changes.

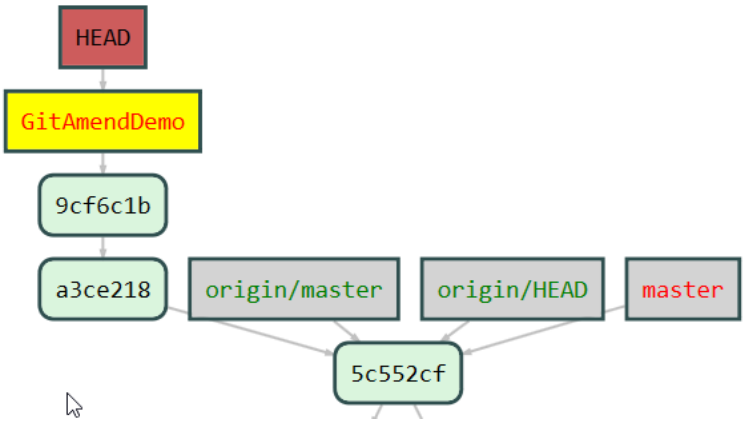
a) Create some changes to info.txt and commit

[code info.txt]

[git status –s]

[git commit -am "Some changes for the next release"]

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

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (Git
$ git status -s
 M info.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (Git
$ git commit -am "Some changes for the next release"
[GitAmendDemo 9cf6c1b] Some changes for the next release
 1 file changed, 2 insertions(+), 1 deletion(-)
```



Make note of the commit id [**9cf6c1b**] – yours will be different, of course

b) Add a new file to the repo

[**touch readme.txt**]

[**code readme.txt**]

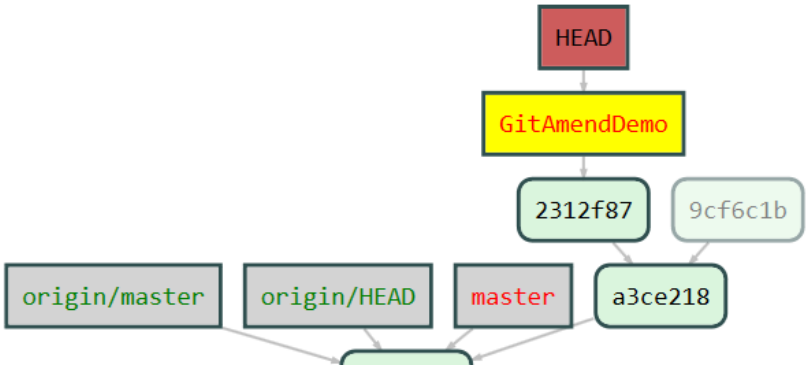
[**git status -s**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/De
$ touch readme.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/De
$ code readme.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/De
$ git status -s
?? readme.txt
```

c) Commit with amend to put the new file into the same commit as the changes for info.txt

[**git commit --amend -m "Changed info.txt and added readme.txt"**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (GitAmendDemo)
$ git commit --amend -m "Changed info.txt and added readme.txt"
[GitAmendDemo 2312f87] Changed info.txt and added readme.txt
 Date: Sat Sep 23 22:33:01 2017 -0500
 1 file changed, 2 insertions(+), 1 deletion(-)
```

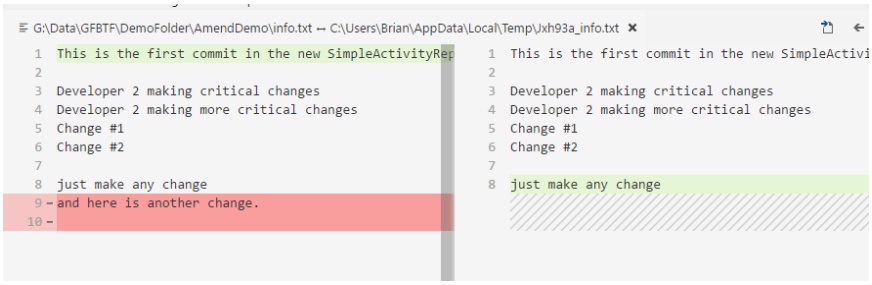


[**git log --oneline**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (GitAmendDemo)
$ git log --oneline
2312f87 (HEAD -> GitAmendDemo) Changed info.txt and added readme.txt
a3ce218 This is an important change to the code
5c552cf (origin/master, origin/HEAD, master) Merge pull request #2 from m
dancesolutions/rebasing-demo-1
a876a65 more changes on my local branch
```

[**git difftool 2312f87 a3c3218**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFol
$ git difftool 2312f87 a3ce218
```



Using the difftool I noticed I forgot to add the new file!!!

[**git status -s**]

```
Brian@SENTINEL MINGW64 /g/Dat
$ git status -s
?? readme.txt
```

Alternatively, I could have run [**git show --name-status**] to see that only the one file was included in the last commit

d) First add, then commit with amend

[**git add .**]

[**git commit --amend -m "Changed info.txt and added readme.txt"**]

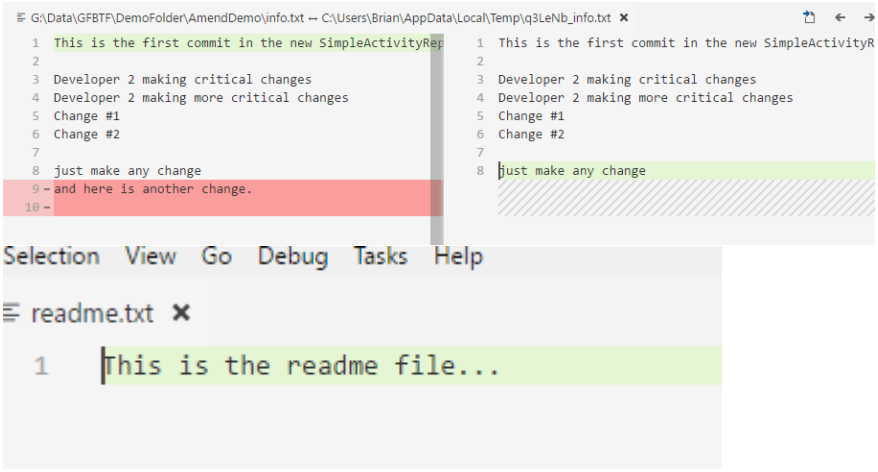
[**git status -s**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (G
$ git add .

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (G
$ git commit --amend -m "Changed info.txt and added readme.t
[GitAmendDemo 0cac76b] Changed info.txt and added readme.txt
 Date: Sat Sep 23 22:33:01 2017 -0500
 2 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 readme.txt

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (G
$ git status -s
```

[**git log –oneline**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/A
$ git log --oneline
0cac76b (HEAD -> GitAmendDemo) Changed info.txt a
a3ce218 This is an important change to the code
5c552cf (origin/master, origin/HEAD, master) Merg
dancesolutions/rebasing-demo-1
```

[**git difftool 0cac76b a3ce218**]



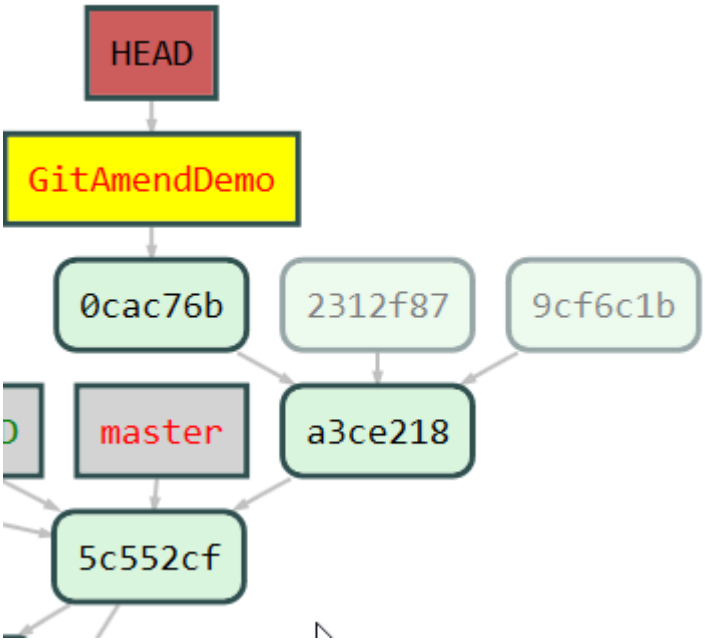
Now we are on track! Alternatively, [**git show --name-status**] shows both files as well:

```
$ git show --name-status
commit 0109f5d1793e6295abd44065651f26db8a392d4f (HEAD -> GitAmendDemo)
Author: Brian L. Gorman <blgorman@gmail.com>
Date:   Sun Jul 14 21:25:05 2019 -0500

    Changed info.txt and added readme.txt

M       info.txt
A       readme.txt
```

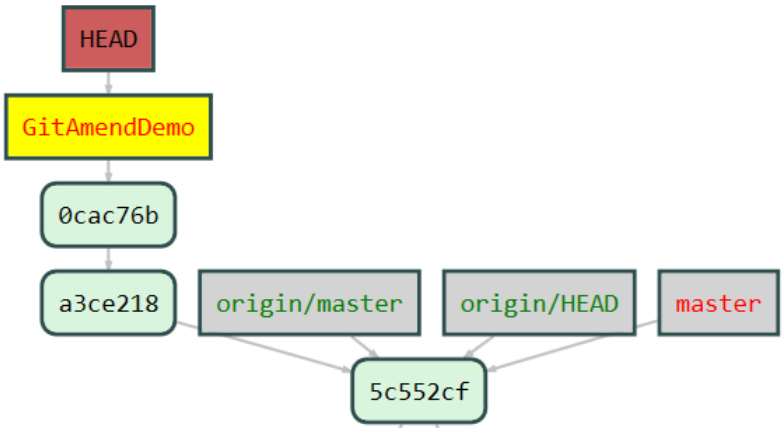
Step 3: [Optional] - Cleanup the local unreachable commits, push to GitHub and merge changes into master.

a) Repeat Step 2 for cleanup.

Before:



After:



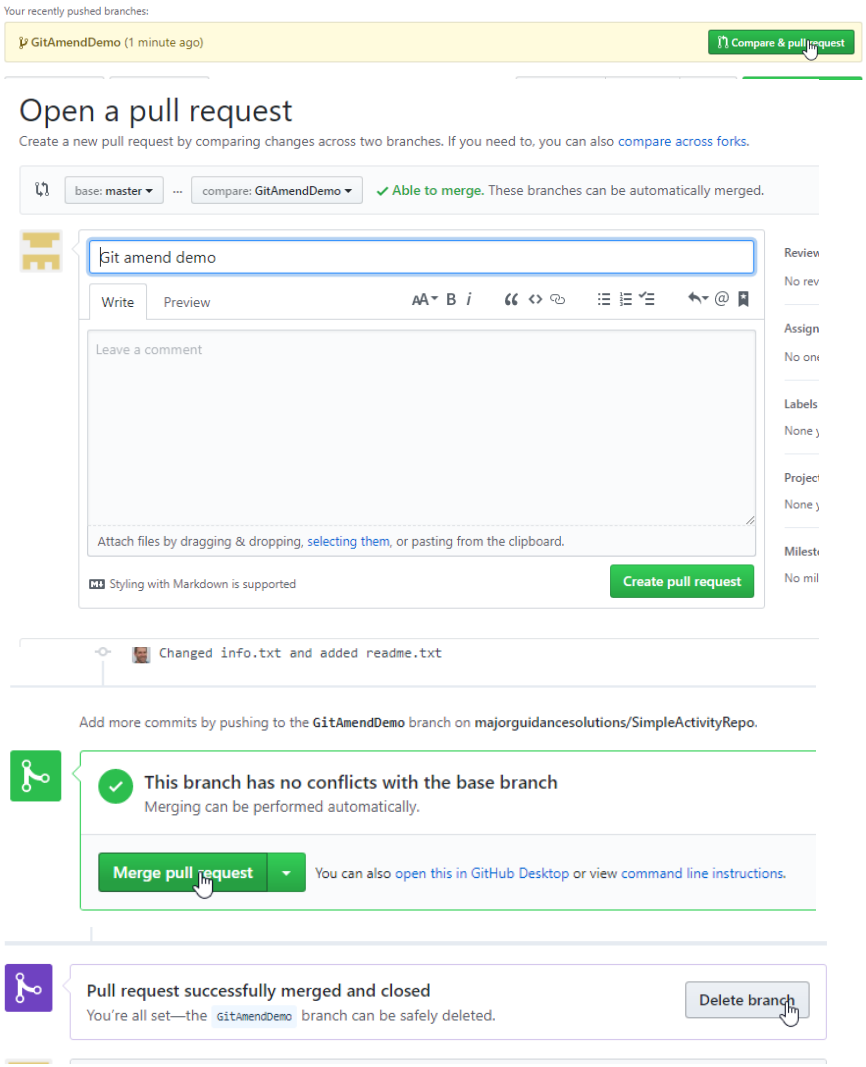
[**git reflog expire --expire-unreachable=now -all**]
[**git gc --prune=now**]
Note: This is entirely optional. Not cleaning up will not harm anything. Cleaning up like this might actually cause you some pain as well, so you may choose not to do this, ever. The garbage collector will clean up your repo for you at around 90 days. Not cleaning allows you to potentially recover some lost changes in the event something does go wrong.

b) Push to GitHub

[git push -u origin GetAmendDemo]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (GitAmendDemo)
$ git push -u origin GitAmendDemo
Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 662 bytes | 0 bytes/s, done.
Total 7 (delta 4), reused 5 (delta 2)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/majorguidancesolutions/SimpleActivityRepo.git
 * [new branch]      GitAmendDemo -> GitAmendDemo
Branch GitAmendDemo set up to track remote branch GitAmendDemo from origin.
```

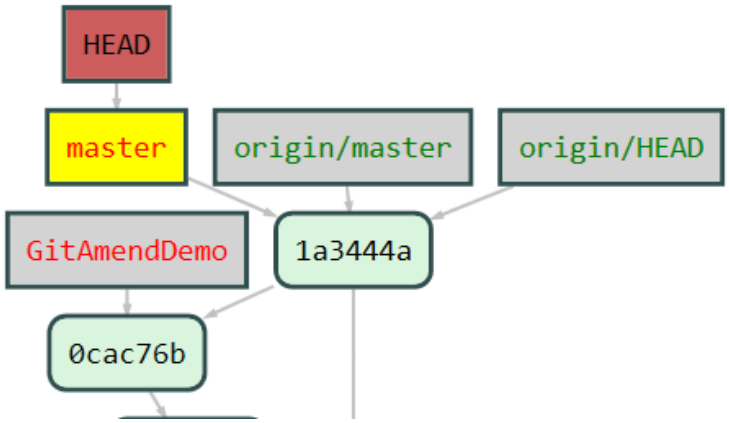
Create a Pull Request and Merge at GitHub



c) Get the latest from github into the local repo

[git checkout master]
[git fetch origin]
[git pull origin master]

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

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (
$ git fetch origin
From https://github.com/majorguidancesolutions/SimpleActivi
 - [deleted]         (none)     -> origin/GitAmendDemo
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 1 (delta 0), pack-reused
Unpacking objects: 100% (1/1), done.
   5c552cf..1a3444a  master     -> origin/master

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/AmendDemo (
$ git pull origin master
From https://github.com/majorguidancesolutions/SimpleActivi
 * branch            master     -> FETCH_HEAD
Updating 5c552cf..1a3444a
Fast-forward
 info.txt   | 3 +++
 readme.txt | 1 +
 2 files changed, 4 insertions(+)
 create mode 100644 readme.txt
```



[**git branch -d GitAmendDemo**]

[**git status**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/Ame
$ git branch -d GitAmendDemo
Deleted branch GitAmendDemo (was 0cac76b).

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/Ame
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

This concludes our GIT Amend Activity.

Closing Thoughts

In this activity we saw a couple of scenarios where we were able to change the commit history and amend changes into the most recent commit.

The first amend was a simple change to the commit message. The second change was a full commit with added files amended into the commit.

With the amend option, we are able to change the most recent commit, but once again we would want to keep the history intact if this was a public branch. Since this is a private branch, we were able to do what we wanted without fear of any issues.

The other nice thing about the amend options is that all our changes only made one combined commit into the repository, rather than two or three commits.

We also saw that we can run a couple of commands that allow us to wipe our unreachable commits from memory. While this is entirely possible and makes our GitVIS look a lot nicer, we may want to be wary of using this in the real world as there could be scenarios where we want to get one of the "unreachable" commits back. Additionally, with the fact that it doesn't hurt anything to keep the unreachables around and the GC will automatically clean up around 90 days anyway, we should probably just leave that operation off in most cases.

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

Notes