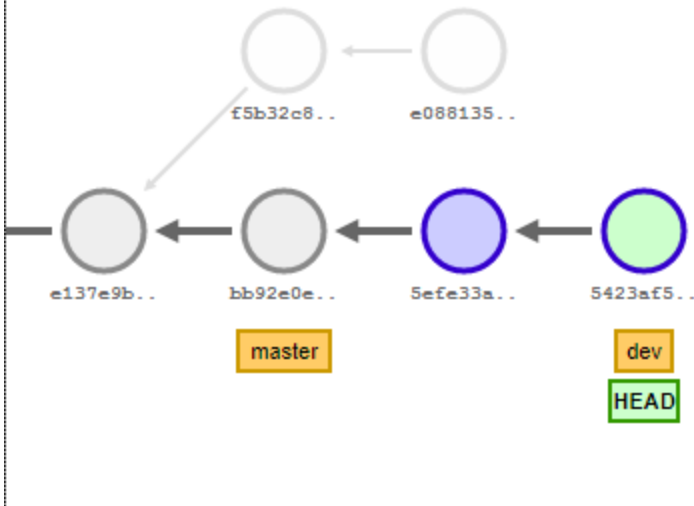# GIT: From Beginner To Fearless

GIT Rebasing Activity:
Moving commits in history

Brian Gorman, Author/Instructor/Trainer

©2019 - MajorGuidanceSolutions

https://www.majorguidancesolutions.com

Introduction

Rebasing is one of the most interesting commands we can do when working with GIT. To rebase or not to rebase – that is the question. Much like the 'tabs vs. spaces' or 'coke vs. pepsi' debates, there are strong camps on both sides of the pulling with and pulling without rebase camps. Just do a quick google search and you'll find many passionate pleas to 'always rebase when you pull' or 'never rebase your public branch.'

So what does a rebase really do, and why is it something we would want to use? To put it quite simply, rebasing is nothing more than changing the parent commit of another commit. To actually describe it would sound something more like 'moving the base commit of a chain of commits so that it appears to have been created in a linear timeline from the most recent commit on the public branch.

A quick look at a rebase shows one common rebasing scenario [from http://onlywei.github.io/explain-git-with-d3/#rebase]

Note that commit f5b32c8 currently has parent e137e9b. When we do a simple rebase, the parent changes to bb92e0e – but we also get a new commit id [this is why rebasing is somewhat dangerous – but nothing to fear – more on that later].

[git rebase master]

MAJOR GUIDANCE
SOLUTIONS

After rebasing, we have a linear commit chain, and it appears that commit 5efe33a started AFTER bb92e0e. In fact, the commit started after e137e9b, but we've changed the history timeline.

We have to be careful -> If other developers are relying on our history to show the commit chain as it was, committing this rebase to public would be a disaster.

Once we have the rebase done, however, we can commit the change into master as a regular merge

[git checkout master]

[git merge dev]
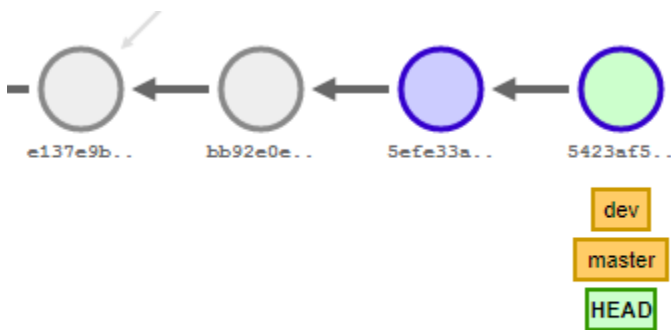
And now everyone can be happy with a history that is linear and public. In this activity, we're going to take a deeper dive into rebasing so we can master the idea of rebasing a commit or commit chain.

Let's gets started!

MAJOR GUIDANCE
S O L U T I O N S

Step 1: Start with any public repository

a) Create a public branch, get it local, make a couple of changes

After creating the public branch, pull it to local, make a couple of changes, and commit to LOCAL HEAD, but don't push to REMOTE

[fork & **clone a simple repo – or create your own with a simple text file**]

[git clone <new_repo_url>]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder
$ git clone https://github.com/majorguidancesolutions/SimpleActivityRepo.git Reb
asingActivity1
Cloning into 'RebasingActivity1'...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 1), reused 8 (delta 1), pack-reused 0
Unpacking objects: 100% (8/8), done.
```

[**git fetch origin**]

[**git pull origin master**] //always make sure to be up-to-date

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git fetch origin

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git pull origin master
From https://github.com/majorguidancesolutions/SimpleActivityRepo
 * branch            master     -> FETCH_HEAD
Already up-to-date.
```
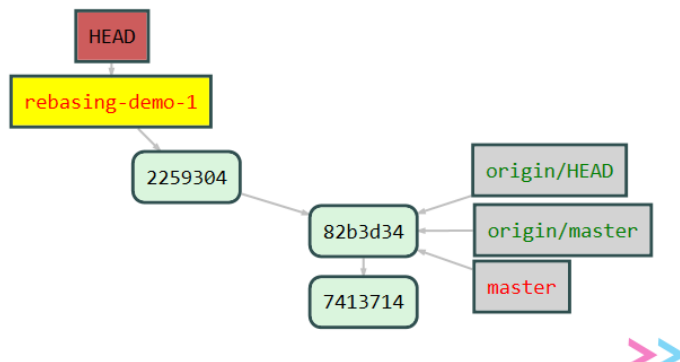
[git checkout -b <branchname>]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/Reba
$ git checkout -b rebasing-demo-1
Switched to a new branch 'rebasing-demo-1'
```

…make some changes…

[code info.txt]

```
≡ info.txt    ✕
   1    This is the first commit in the new SimpleActivityRepo
   2
   3 │  Change #1
```
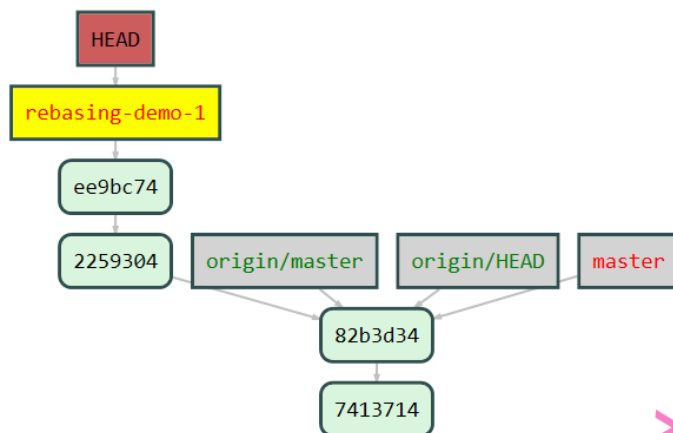
[git commit –am "changes on my local before rebase"]

...make more changes...

```
≡ info.txt      ✕
1    This is the first commit in the new SimpleActivityRepo
2
3 │ Change #1
4 │ Change #2|
```

[git commit –am "more changes on my local branch]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (r
-1)
$ git commit -am "more_changes on my local branch"
[rebasing-demo-1 ee9bc74] more changes_on my local branch
 1 file changed, 2 insertions(+), 1 deletion(-)

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (r
```

```
┌──────────┐
│  HEAD    │
└──────────┘
     │
┌─────────────────┐
│ rebasing-demo-1 │
└─────────────────┘
     │
┌──────────┐
│ ee9bc74  │
└──────────┘
     │
┌──────────┐  ┌───────────────┐  ┌───────────────┐  ┌──────────┐
│ 2259304  │  │ origin/master │  │  origin/HEAD  │  │  master  │
└──────────┘  └───────────────┘  └───────────────┘  └──────────┘
          │          │                   │           │
              ┌──────────┐
              │ 82b3d34  │
              └──────────┘
                   │
              ┌──────────┐
              │ 7413714  │
              └──────────┘
```

`>>` readifv

b) Simulate changes by another developer at the repo

Put in a couple of changes on a branch, the merge it into master
Create the branch

```
GFBTF Demo Repository

Add topics

    🕐 2 commits        ⑂ 1 branch          ⬡ 0 releases

Branch: master ▾   New pull request                    Create nev

Switch branches/tags                    ✕      s-team:master.

AnotherDeveloper|                              b Create info.txt

Branches    Tags

⑂ Create branch: AnotherDeveloper              Initial commit
  from 'master'
                                               Initial commit
  README.md                                    Initial commit

📄 info.txt                         Create info.txt
```
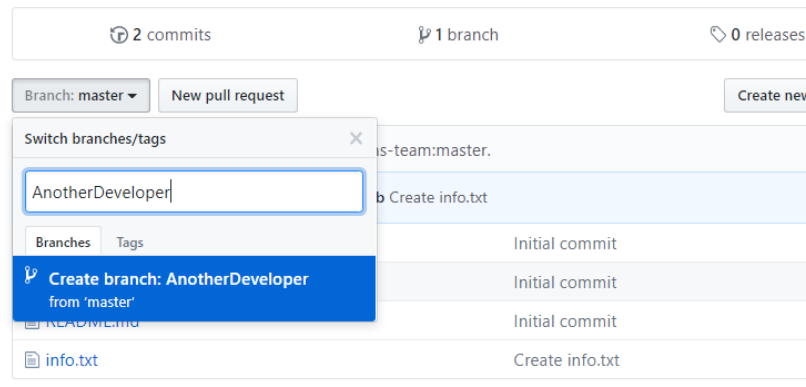
Modify the info.txt file [we are not avoiding conflict]

---

https://www.majorguidancesolutions.com

MAJOR GUIDANCE
S O L U T I O N S

SimpleActivityRepo / info.txt   [icon] or cancel

<> Edit file   ⊙ Preview changes   Spaces ⇕   2

```
1   This is the first commit in the new SimpleActivityRepo
2
3   Developer 2 making critical changes.
```

Commit changes

Update info.txt

Add an optional extended description...

◉ -○- Commit directly to the `AnotherDeveloper` branch.

○ ⑂ Create a **new branch** for this commit and start a pull request. Learn more

Commit changes   Cancel

Repeat to create a second commit, then create pull request with the two commits:

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

☑ Allow edits from maintainers. Learn more   **Create pull request**

Projects   ⚙
None yet

Milestone   ⚙
No milestone

-○- 2 commits   ▤ 1 file changed   💬 0 commit comments   👥 1 contributor

▣ Commits on Sep 23, 2017

-○- 🔲 majorguidancesolutions   Update info.txt   5be242e

-○- 🔲 majorguidancesolutions   Update info.txt   16cea93

Showing 1 changed file with 3 additions and 0 deletions.   Unified | Split

3 ▰▰▰▱▱ info.txt   View ▭ ⌄

```
...  ...  @@ -1 +1,4 @@
1    1    This is the first commit in the new SimpleActivityRepo
     2  +
     3  +Developer 2 making critical changes.
     4  +Developer 2 making more critical changes.
```

MAJOR GUIDANCE
S O L U T I O N S
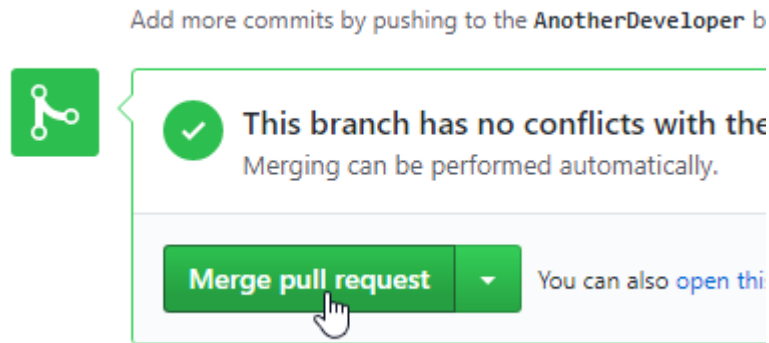
Merge.

Delete branch
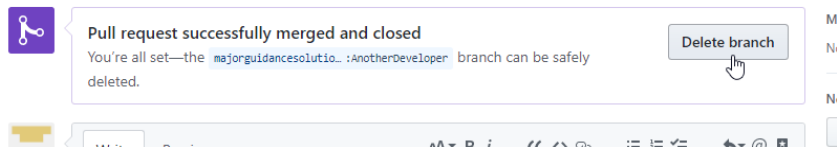
c) Get the latest locally, then rebase locally. Solve the merge conflict on rebase.

First, we need to switch back to master, fetch and pull:

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git fetch origin
gremote: Counting objects: 7, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 5), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (7/7), done.
From https://github.com/majorguidancesolutions/SimpleActivityRepo
   82b3d34..342e3be  master     -> origin/master

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git pull origin master
From https://github.com/majorguidancesolutions/SimpleActivityRepo
 * branch          master     -> FETCH_HEAD
Updating 82b3d34..342e3be
Fast-forward
 info.txt | 3 +++
 1 file changed, 3 insertions(+)
```

Here we see that the origin master has moved ahead three commits – the two for the 'another developer branch' and the one for the merge of the pull request.

https://www.majorguidancesolutions.com

MAJOR GUIDANCE
S O L U T I O N S
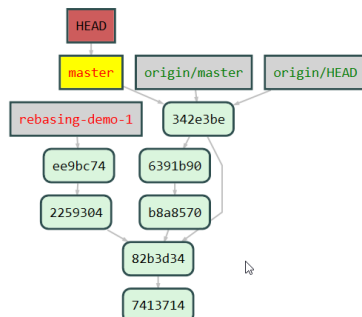
d) Rebase the changes from our branch onto the master

Switch back to our target branch, and rebase master. We'll need to resolve the conflicts with our merge tool as well:

First, make note of our local commit IDs [ee9bc74 and 2259304]

[**git checkout rebasing-demo-1**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git checkout rebasing-demo-1
Switched to branch 'rebasing-demo-1'
```
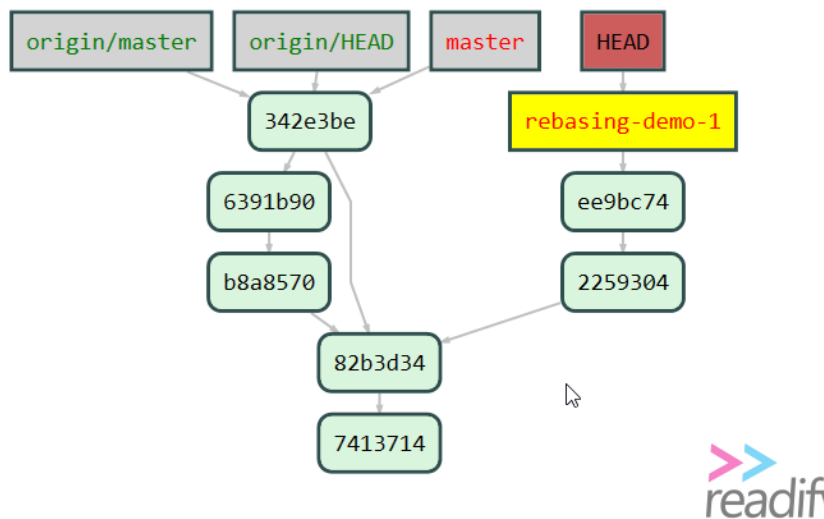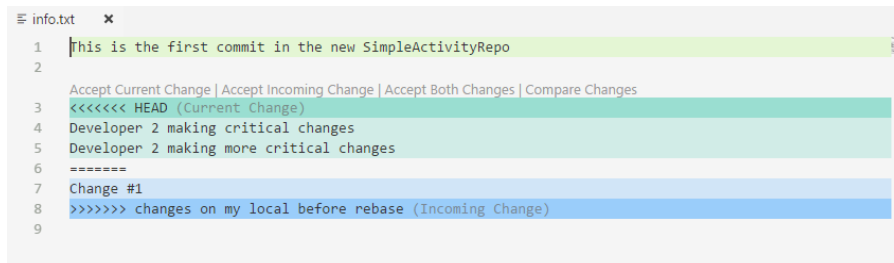

[**git rebase master**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (rebasing-demo
-1)
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: changes on my local before rebase
error: Failed to merge in the changes.
Using index info to reconstruct a base tree...
M       info.txt
Falling back to patching base and 3-way merge...
Auto-merging info.txt
CONFLICT (content): Merge conflict in info.txt
Patch failed at 0001 changes on my local before rebase
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (rebasing-demo
-1|REBASE 1/2)
$
```

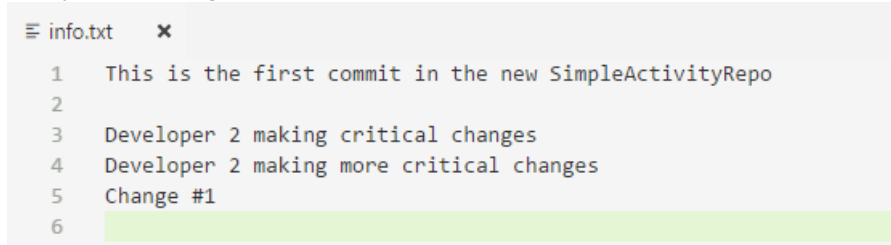https://www.majorguidancesolutions.com

MAJOR GUIDANCE
SOLUTIONS

[**git mergetool**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (reba
-1|REBASE 1/2)
$ git mergetool
Merging:
info.txt

Normal merge conflict for 'info.txt':
  {local}: modified file
  {remote}: modified file
```

```
≡ info.txt    ✕
1    This is the first commit in the new SimpleActivityRepo
2
     Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
3    <<<<<<< HEAD (Current Change)
4    Developer 2 making critical changes
5    Developer 2 making more critical changes
6    =======
7    Change #1
8    >>>>>>> changes on my local before rebase (Incoming Change)
9
```

Accept both changes…

```
≡ info.txt    ✕
1    This is the first commit in the new SimpleActivityRepo
2
3    Developer 2 making critical changes
4    Developer 2 making more critical changes
5    Change #1
6
```

Note in the command line we have to rebase and resolve both commits. So this means we'll see the resolution one more time. [you can see REBASE 1/2 in the command text:

[**git rebase --continue**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (rebasing-demo
-1|REBASE 1/2)
$ git rebase --continue
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (rebasing-demo
-1|REBASE 1/2)
$ git rebase --continue
Applying: changes on my local before rebase
Applying: more changes on my local branch
error: Failed to merge in the changes.
Using index info to reconstruct a base tree...
M       info.txt
Falling back to patching base and 3-way merge...
Auto-merging info.txt
CONFLICT (content): Merge conflict in info.txt
Patch failed at 0002 more changes on my local branch
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (rebasing-demo
-1|REBASE 2/2)
$
```

MAJOR GUIDANCE
S O L U T I O N S
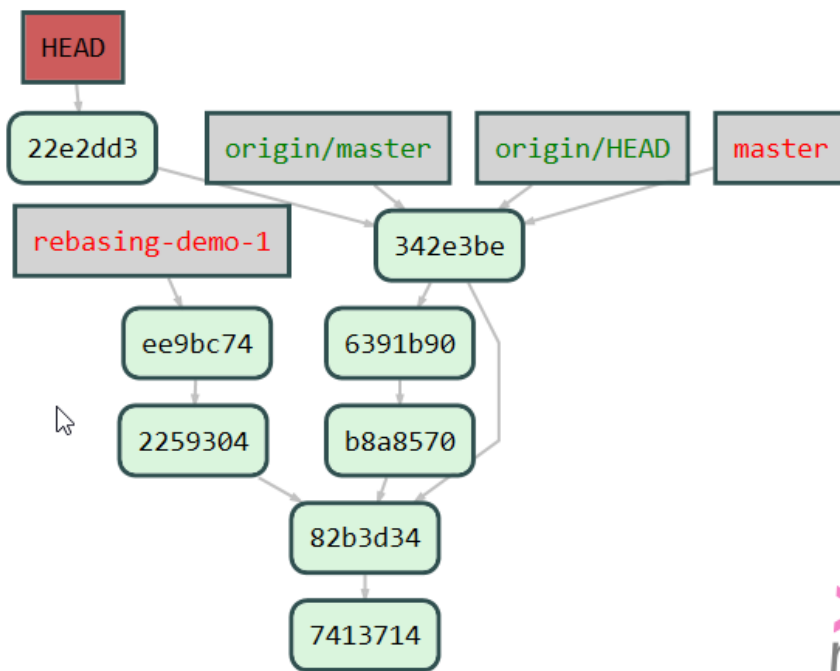
Make a note. We now have a new commit id that is the commit which resolved that first conflict (1 of 2) in the rebase activity. This is going to be our "new" history. This is why it is so critical to not rebase on a public branch. So far no one is dependent on our two commits [ee9... and 22593...] What do you think will happen on the next rebase merge resolution?
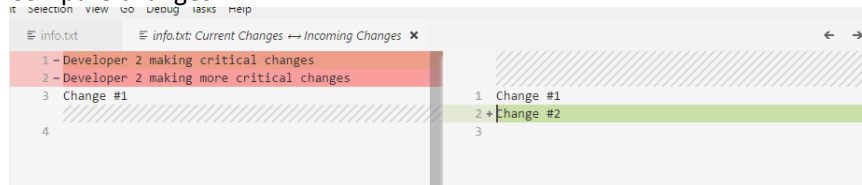
[**git mergetool**] //for our second commit.

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/Reb
-1|REBASE 2/2)
$ git mergetool
Merging:
info.txt

Normal merge conflict for 'info.txt':
  {local}: modified file
  {remote}: modified file
```

Compare changes:

MAJOR GUIDANCE
SOLUTIONS

We need to keep the two lines that are getting removed and we'd be ok, so Accept both changes, and delete the duplicated Change #1 line:

```
≡ info.txt      ✕

  1     This is the first commit in the new SimpleActivityRepo
  2
  3     Developer 2 making critical changes
  4     Developer 2 making more critical changes
  5     Change #1
  6     Change #2
  7
```

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1
-1|REBASE 2/2)
$ git rebase --continue
Applying: more changes on my local branch
```
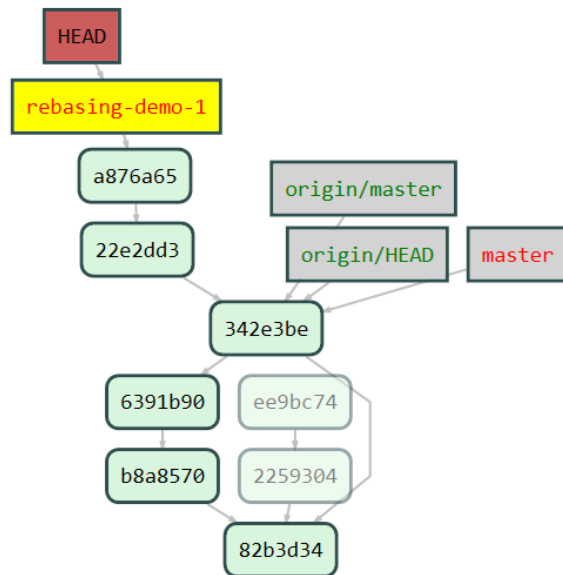

Note the commit id's have changed! Now that we've resolved both, we have two new commits. What happened to ee9bc74 and 2259304? They are kind of grayed out – because they are now in an 'unreachable' state. And that's ok.

Our current changes are in two new commits [22e2dd3 and a876a65]. So now we just need to clean up the repository and push to master.

e) Clean up the unreachable commits

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. I have these commands aliased, but in case you don't and you want to run these [or want the commands for later reference], here they are:

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

And

[**git gc –prune=now**]

MAJOR GUIDANCE SOLUTIONS

And here are my aliases in my global config [check out the aliasing activity for more info about aliasing]:

```
alias.expireunreachablenow=reflog expire --expire-unreachable=now --all
alias.gcunreachablenow=gc --prune=now
fetch.prune=true
```

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/Rebas
-1)
$ git expireunreachablenow

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/Rebas
-1)
$ git gcunreachablenow
Counting objects: 21, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (21/21), done.
Total 21 (delta 11), reused 12 (delta 6)
```
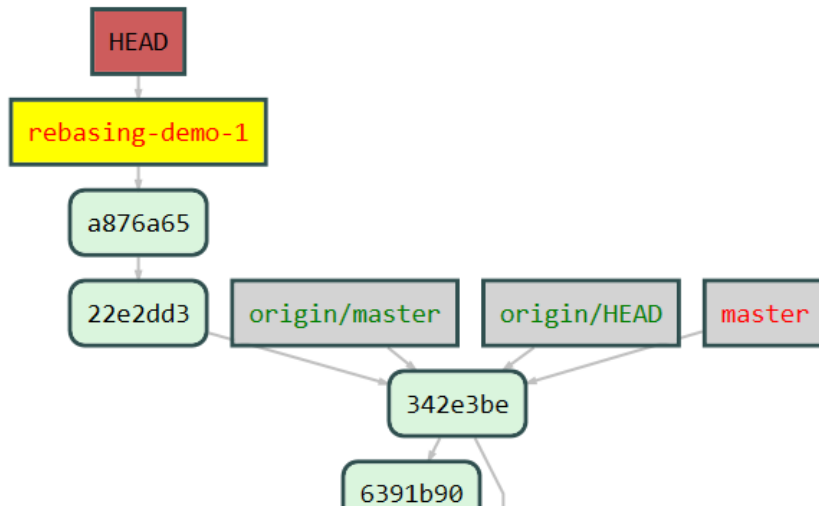

f) Push our changes, pull request, and merge to master
[git push –u origin rebasing-demo-1]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (rebasing-demo
-1)
$ git push -u origin rebasing-demo-1
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 552 bytes | 0 bytes/s, done.
Total 6 (delta 4), reused 4 (delta 2)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/majorguidancesolutions/SimpleActivityRepo.git
 * [new branch]      rebasing-demo-1 -> rebasing-demo-1
Branch rebasing-demo-1 set up to track remote branch rebasing-demo-1 from origin
```

Create a pull request and merge our changes at GitHub

MAJOR GUIDANCE
S O L U T I O N S
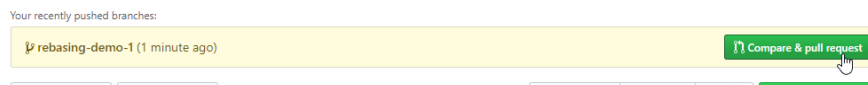
Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ▾ … compare: rebasing-demo-1 ▾   ✓ Able to merge. These branches can be automatically merged.

Rebasing demo 1

| Write | Preview |

AA▾ B *i*   " <> ⊙   ☰ ☰ ☷   ↰▾ @ 🔖

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

📄 Styling with Markdown is supported

Create pull request

Add more commits by pushing to the **rebasing-demo-1** branch on maj...

✓ This branch has no conflicts with the base bra...
Merging can be performed automatically.

Merge pull request ▾   You can also open this in GitHub [

Pull request successfully merged and closed   **Delete branch**
You're all set—the `rebasing-demo-1` branch can be safely deleted.

Now our master has everything and our feature branch is deleted so we need to git local up-to-date and cleaned up

MAJOR GUIDANCE
SOLUTIONS

g) Get up to date on local master branch and delete our feature branch
[**git checkout master**]
[**git fetch origin**]
[**git pull origin master**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (rebasing-
-1)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git fetch origin
From https://github.com/majorguidancesolutions/SimpleActivityRepo
 - [deleted]        (none)     -> origin/rebasing-demo-1
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 1 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
   342e3be..5c552cf  master     -> origin/master

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (mas
$ git pull origin master
From https://github.com/majorguidancesolutions/SimpleActivityRepo
 * branch           master     -> FETCH_HEAD
Updating 342e3be..5c552cf
Fast-forward
 info.txt | 2 ++
 1 file changed, 2 insertions(+)
```
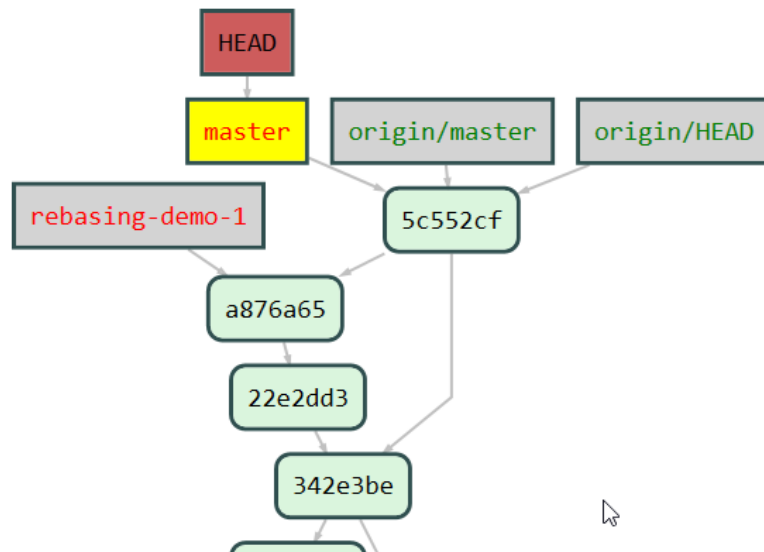

[note: I have my local set to prune on every fetch, so my local has pruned origin/rebasing-demo-1. If you are doing this and see that branch, run [git fetch origin –prune] and it should go away if you have deleted the branch at REMOTE]

Now that master is up to date, the last thing I need to do is get rid of my feature branch.
[git branch –d rebasing-demo-1]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git branch -d rebasing-demo-1
Deleted branch rebasing-demo-1 (was a876a65).
```

MAJOR GUIDANCE
S O L U T I O N S
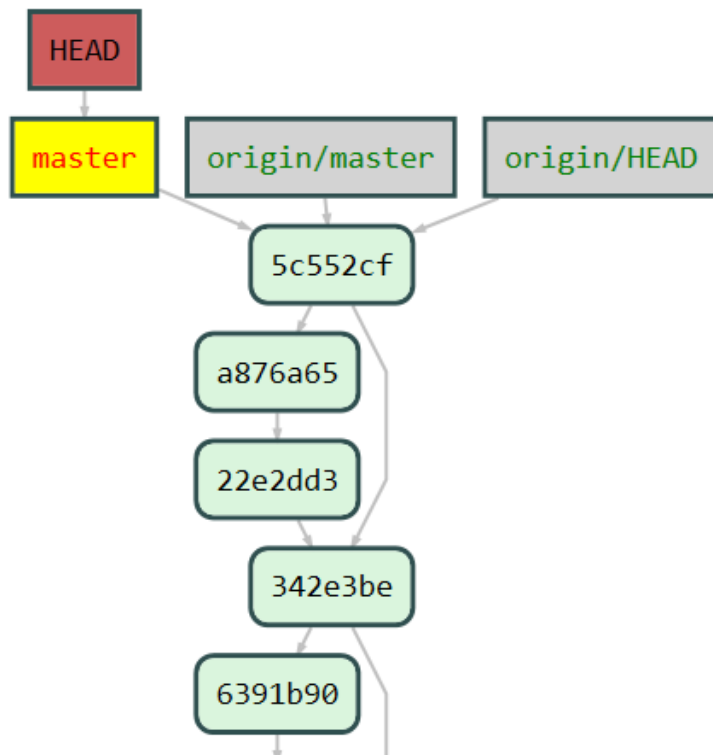
And that is how we do a rebasing operation with conflict resolutions to move our feature branch commits to have a new parent from an updated history after other developers have made changes.

https://www.majorguidancesolutions.com

MAJOR GUIDANCE
S O L U T I O N S

Closing Thoughts

In this activity we worked through a common rebasing scenario, where another developer had made changes on the repository while we were "in progress." The ability to easily rebase makes GIT fairly flexible as to how you want to create merge resolutions. Unlike the traditional route, using the rebase allows us to "change" the order of commits in history. So what had started out as being a couple of commits behind the actual history appears to happen directly after the commits.

In the end, you may never actually need to rebase your work, depending on whether or not you care if your work appears as a straight line with no branching or if you don't mind a few branches with reconnects.

Other scenarios for rebasing do exist. For example, I once had to port a Visual Studio Team System history into GitHub. If I didn't want to keep history, it wouldn't have mattered, of course. However, in order to preserve history, I actually was able to create the repo and then rebase master on top of the original history (I know, I said never to rebase master…to somewhat quote a line from one of my favorite movies "this is where you find out how often [Gorman] does things he says not to do").

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

Notes

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

MAJOR GUIDANCE
S O L U T I O N S