

Rescuing your GIT Repository

How to move forward when you commit and feel like you got pulled over by the cops!



Who am I?

Brian is an experienced speaker, author, trainer, and .Net developer with

MCT (x4) / MVP (x2)/MSLearn Expert
MCSD: App Builder
Azure Fundamentals
Azure Data Fundamentals
Azure SCI Fundamentals
Azure Administrator Associate
Azure Security Associate
Azure Developer Associate
Azure IOT Developer Specialization
Azure Cosmos DB Developer Specialization
Azure Database Administrator Associate
Azure Solutions Architect Expert
Azure DevOps Engineer Expert

Brian has a masters of science degree in computer information systems, and a bachelor of science degree in computer science. Additionally, Brian has over ten years of experience instructing college courses online in SQL databases, C#/VB .Net programming, Java programming, and Microsoft Office.

Brian has also created many online video courses, available on various platforms. Recently, Brian published the second edition of his first book: Practical Entity Framework Core 6 with APress.

Brian runs a company named Major Guidance Solutions, which does contract and custom training and curriculum development. If you are interested in becoming Azure certified, make sure to connect with me.

Connect with Brian on Twitter and/or Linked In

@blgorman

<https://www.linkedin/in/brianlgorman>

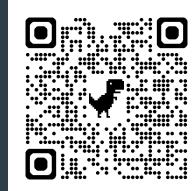


Award Categories:
Azure Application PaaS,
Azure Infrastructure as Code

Number of MVP Awards:
4 years in the program

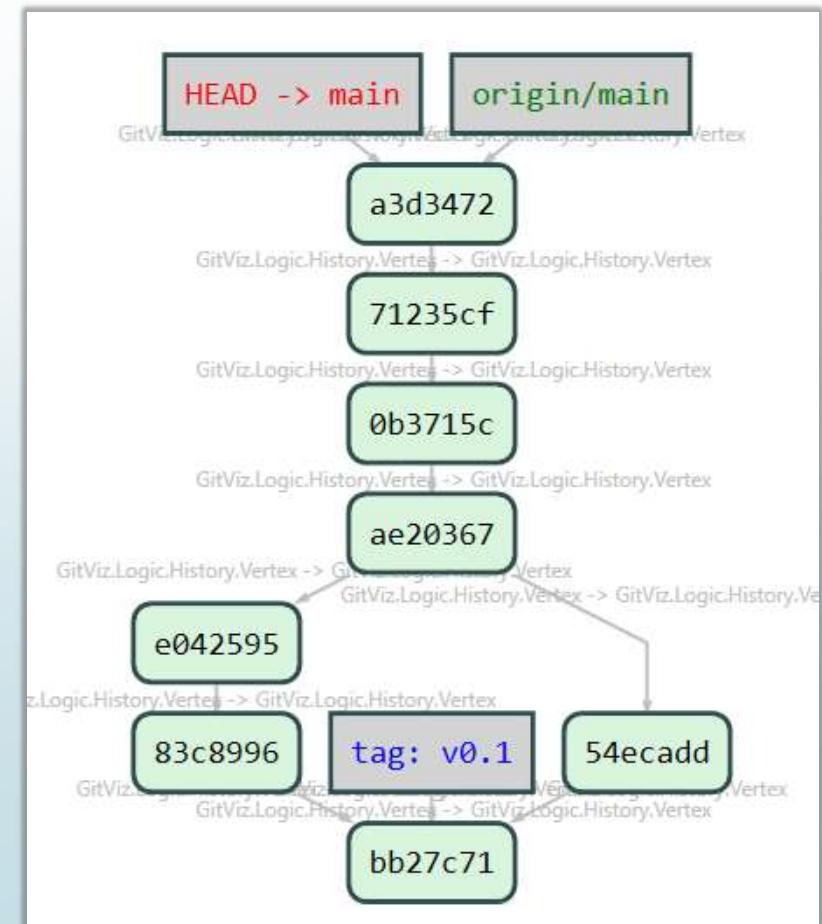
Hands-on Preparation and Practice for Microsoft Azure Certification
Practical Entity Framework Core 6
Database Access for Enterprise Applications
Second Edition
Brian L. Gorman
Apress®

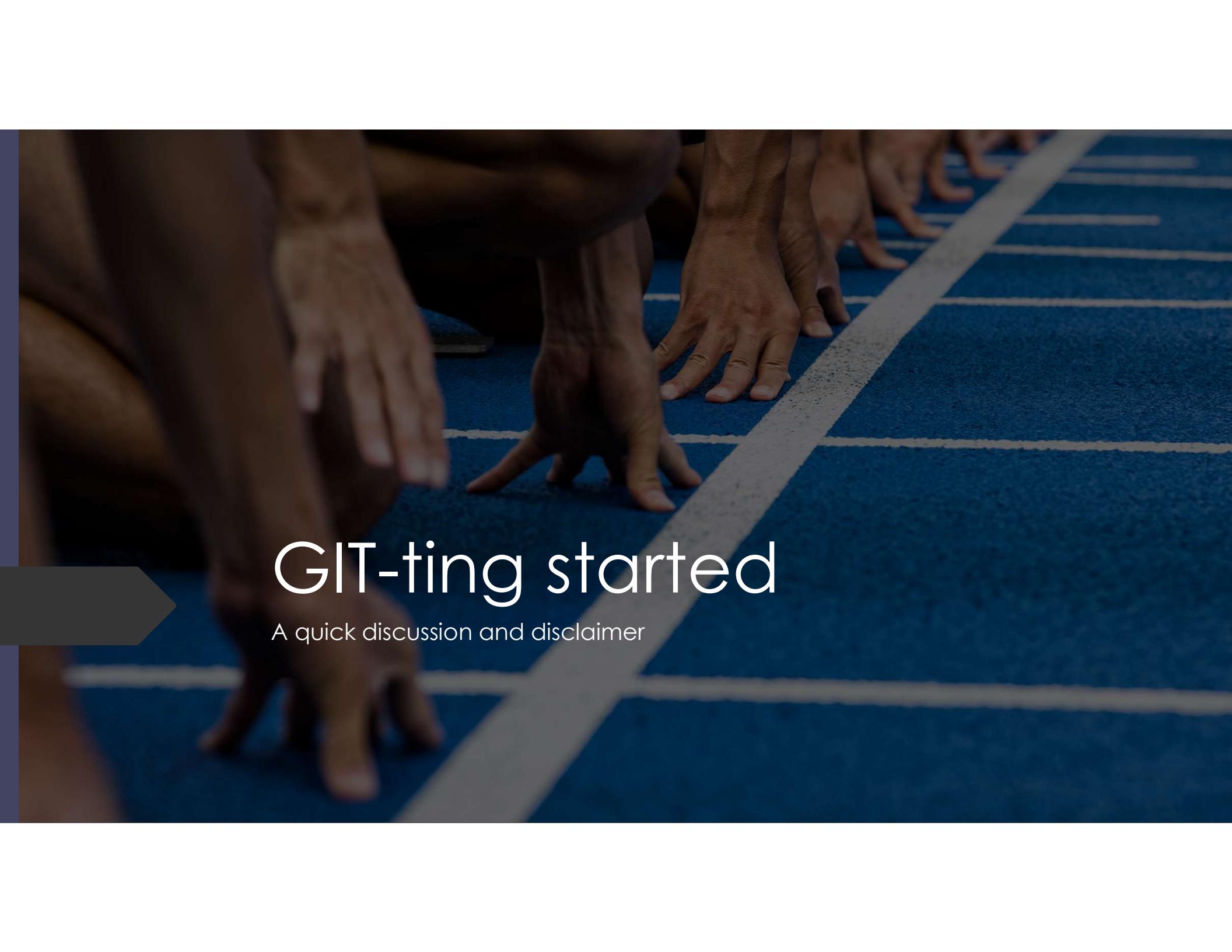
Developing Solutions for Microsoft Azure Certification Companion
Practical Entity Framework Core 6
Database Access for Enterprise Applications
Second Edition
Brian L. Gorman
Apress®



Agenda

- ▶ GIT-ting started
- ▶ Linear/Non-Linear/Squashing
- ▶ Scenarios
 - ▶ 1) Rebase/lost work
 - ▶ 2) Merged two features
We need only one
 - ▶ 3) Committed to remote main
Fix the mess I made
 - ▶ 4) Committed a Secret
Make it go away in all history
- ▶ Conclusion
 - ▶ Links
 - ▶ Connect



A photograph showing a row of people's hands and forearms resting on a blue running track. The hands are positioned as if they are about to start a race. The track has white lane markings.

GIT-ting started

A quick discussion and disclaimer

Requirements and a few notes



- ▶ You should be familiar with these commands
 - ▶ clone/push/fetch/pull/add/commit/branch/checkout/switch
- ▶ The way I'm going to show how to do things is not the only way
 - ▶ Follow your team standards
 - ▶ Let me know if you have a better way (I'd love to hear about it)
 - ▶ Yes, some of this is contrived for demonstration purposes



Interesting Global Settings

```
[alias]
    lol = log --oneline --graph --decorate --all
    expireunreachablenow = reflog expire --expire-unreachable=now --all
    gcunreachablenow = gc --prune=now

[merge]
    tool = code
[mergetool "code"]
    cmd = code --wait --merge $REMOTE $LOCAL $BASE $MERGED
[mergetool]
    prompt = false
    keepbackup = false
```

Thank the sponsors and the team

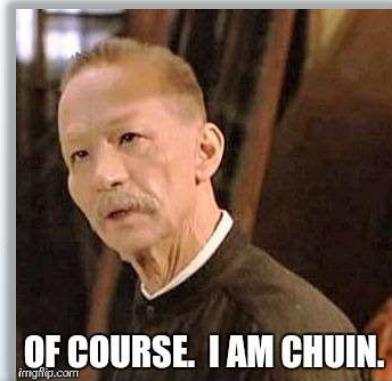


businessgrowthmentor

- 1) There's someone out there who's way less qualified than you - living the life you want.

Simply because they took action.

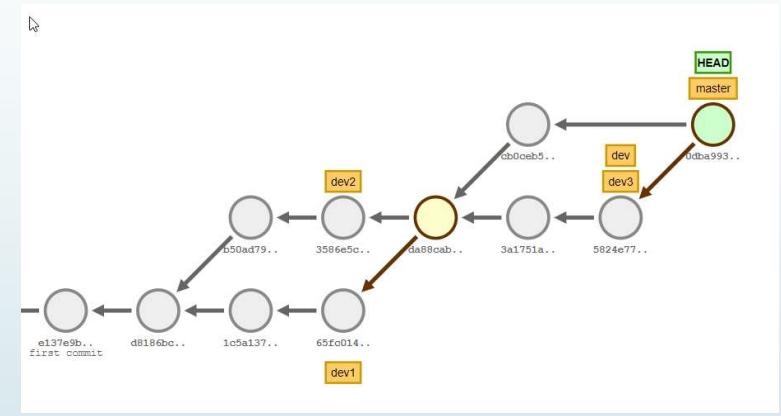
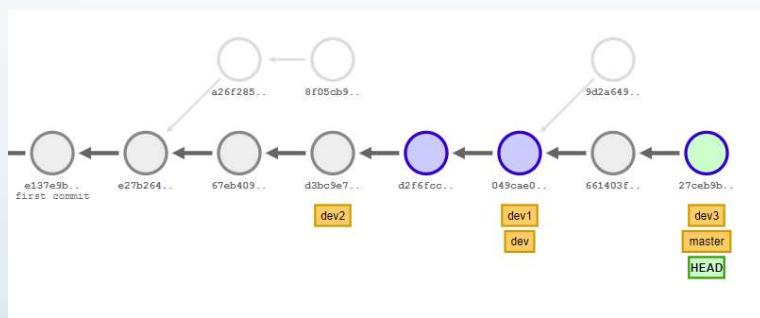
- ▶ Conferences and Code camps are hard work!
- ▶ Be excellent to each other
- ▶ Become a producer and not just a consumer



NO!

Linear and Non-Linear

Which do you prefer?



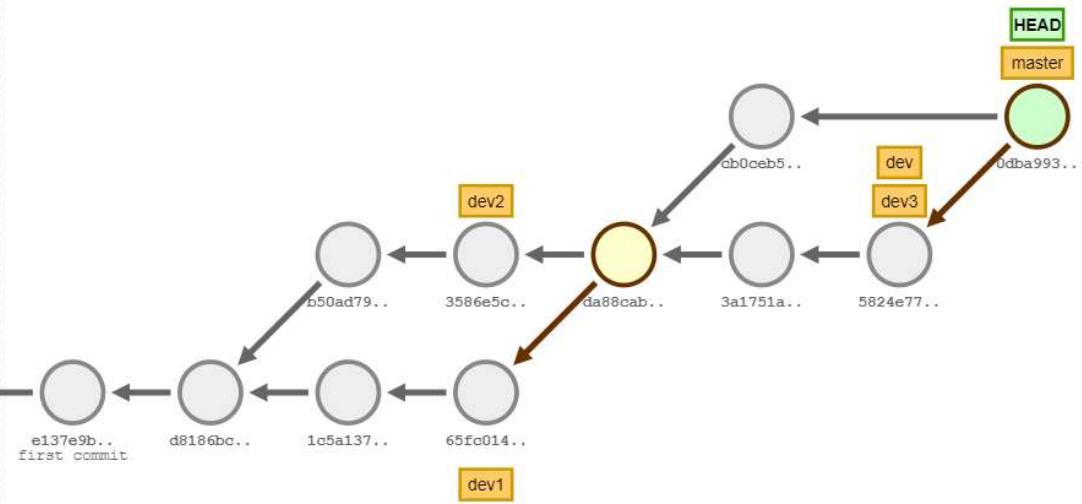
Traditional Merging

- ▶ Any branch strategy
- ▶ Non-linear commit history
- ▶ Super easy to integrate team members with little or no GIT experience
- ▶ Managed easily with pull requests
- ▶ You can always know the exact state of your codebase when you started your development



```
$ git checkout -b dev
$ git commit
$ git checkout -b dev1
$ git commit
$ git commit
$ git checkout dev
$ git checkout -b dev2
$ git commit
$ git commit
$ git checkout dev
$ git merge dev2
You have performed a fast-forward
merge.
$ git merge dev1
$ git checkout master
$ git merge dev
You have performed a fast-forward
merge.
$ git checkout dev
$ git checkout -b dev3
$ git commit
$ git commit
$ git checkout dev
$ git merge dev3
You have performed a fast-forward
merge.
$ git checkout master
$ git commit
$ git merge dev
```

Local Repository
Current Branch: master



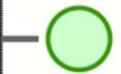
Demo: A traditional merge strategy

Multiple “branch” commits are merged with fast-forward strategy, and this leads to a non-linear history

```
Have fun.
```

Local Repository

Current Branch: master

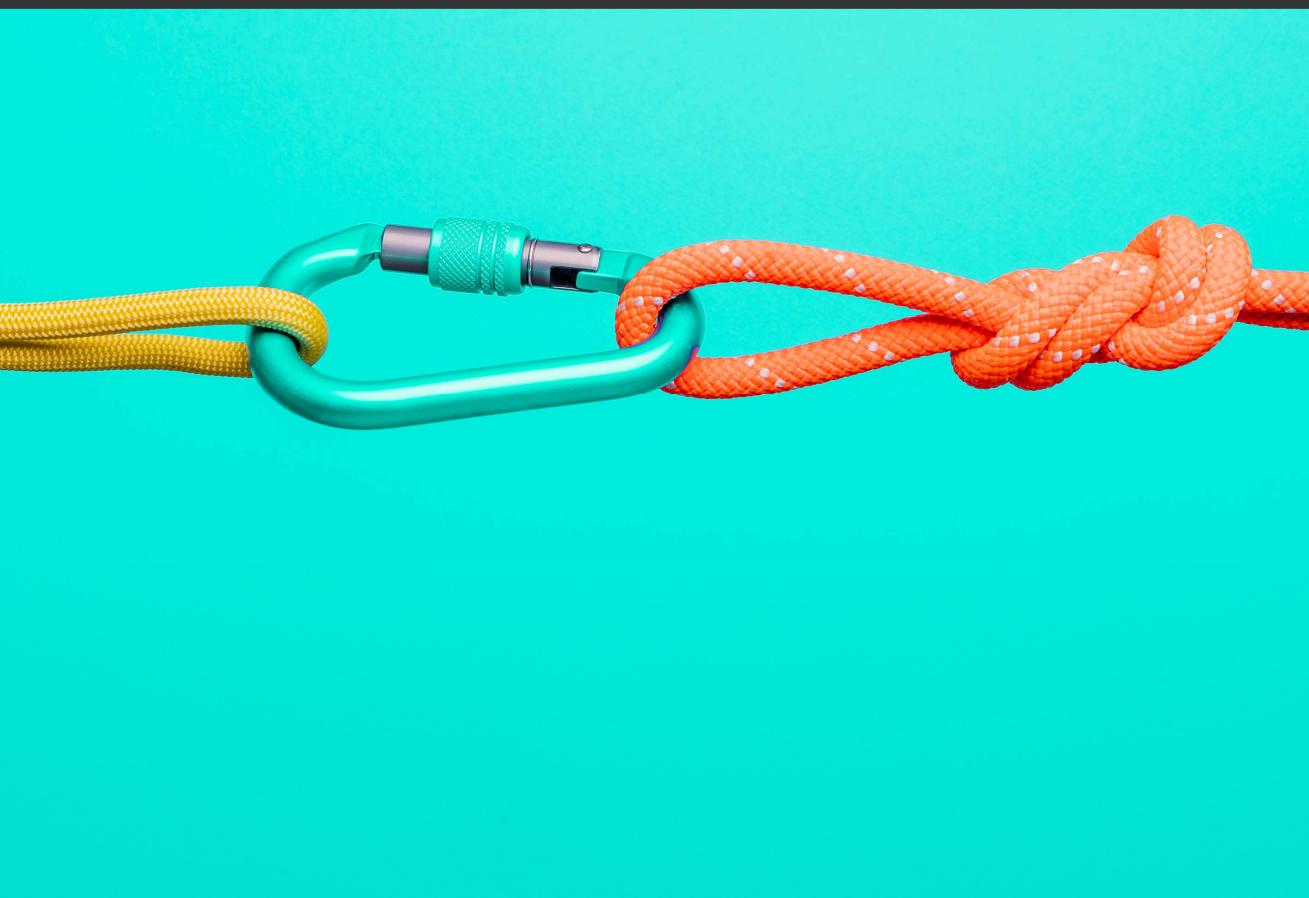


e137e9b..
first commit

master
HEAD

```
$ enter git command
```

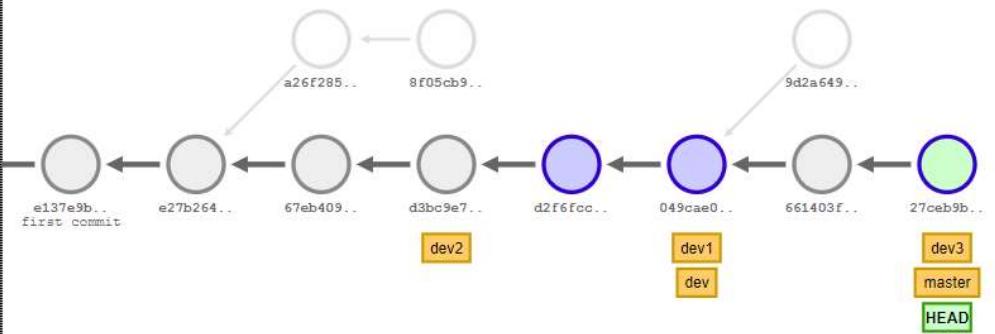
Linear Commit History



- ▶ Any branch strategy
- ▶ Linear commit history
- ▶ Much better to perform advanced commands such as pick and reset
- ▶ Managed with pull requests
- ▶ Can be combined with squashing for a very-succinct commit history

```
$ Have fun.  
$ git checkout -b dev  
$ git commit  
$ git checkout -b dev1  
$ git commit  
$ git commit  
$ git checkout dev  
$ git checkout -b dev2  
$ git commit  
$ git commit  
$ git checkout dev  
$ git merge dev2  
You have performed a fast-forward  
merge.  
$ git checkout dev1  
$ git rebase dev  
$ git checkout dev  
$ git merge dev1  
You have performed a fast-forward  
merge.  
$ git checkout master  
$ git merge dev  
You have performed a fast-forward  
merge.  
$ git checkout dev3  
Cannot find commit: dev3  
$ git checkout -b dev3  
$ git commit  
$ git checkout master  
$ git commit  
$ git checkout dev3  
$ git rebase master  
$ git checkout master  
$ git merge dev3  
You have performed a fast-forward  
merge.
```

Local Repository
Current Branch: master



Demo: Linear Commit History

Can squash commits to make it even more concise

Have fun.

Local Repository

Current Branch: master



e137e9b..
first commit

master

HEAD

\$ enter git command



Scenario #1 – Rebase Fear

I messed something up with my rebase and I think I lost work

Some scary things

- ▶ Your branch and origin/branch has diverged and have N and M different commits each, respectively
- ▶ Might have to resolve conflicts at every commit that is being rebased, which can result in you making a mistake if you don't resolve your conflict correctly, or the feeling that you are doing the same thing multiple times.

```
b1gor@Voyager2 MINGW64 /c/  
ourGitRepo (keep-dev2-changes)  
$ git status  
On branch keep-dev2-changes  
Your branch and 'origin/keep-dev2-changes' have diverged,  
and have 7 and 4 different commits each, respectively.  
(use "git pull" to merge the remote branch into yours)  
nothing to commit, working tree clean
```



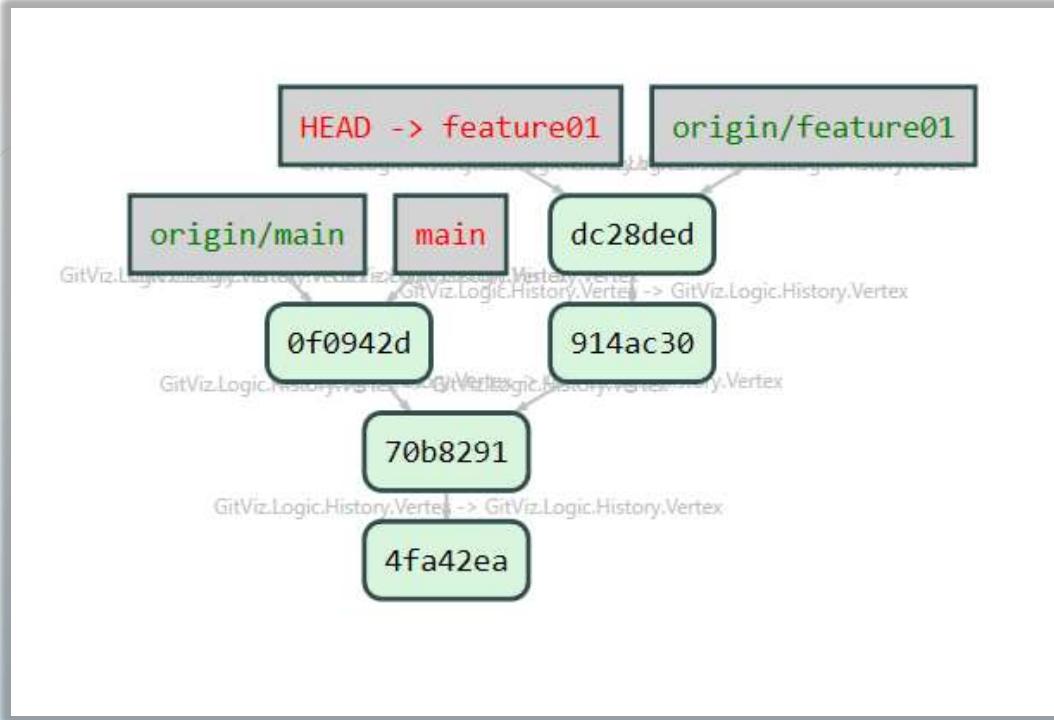
Rebase and Force push

- ▶ Rebase locally, can be based on remote or local branch
- ▶ Orphaned commits
- ▶ Mismatched history so can't push
- ▶ Force Push (with lease)
- ▶ Rebase on PR (don't create a merge commit)

Rebase and Force Push (with lease)

Your local history is correct, but remote won't let you push. Use the force, Luke!

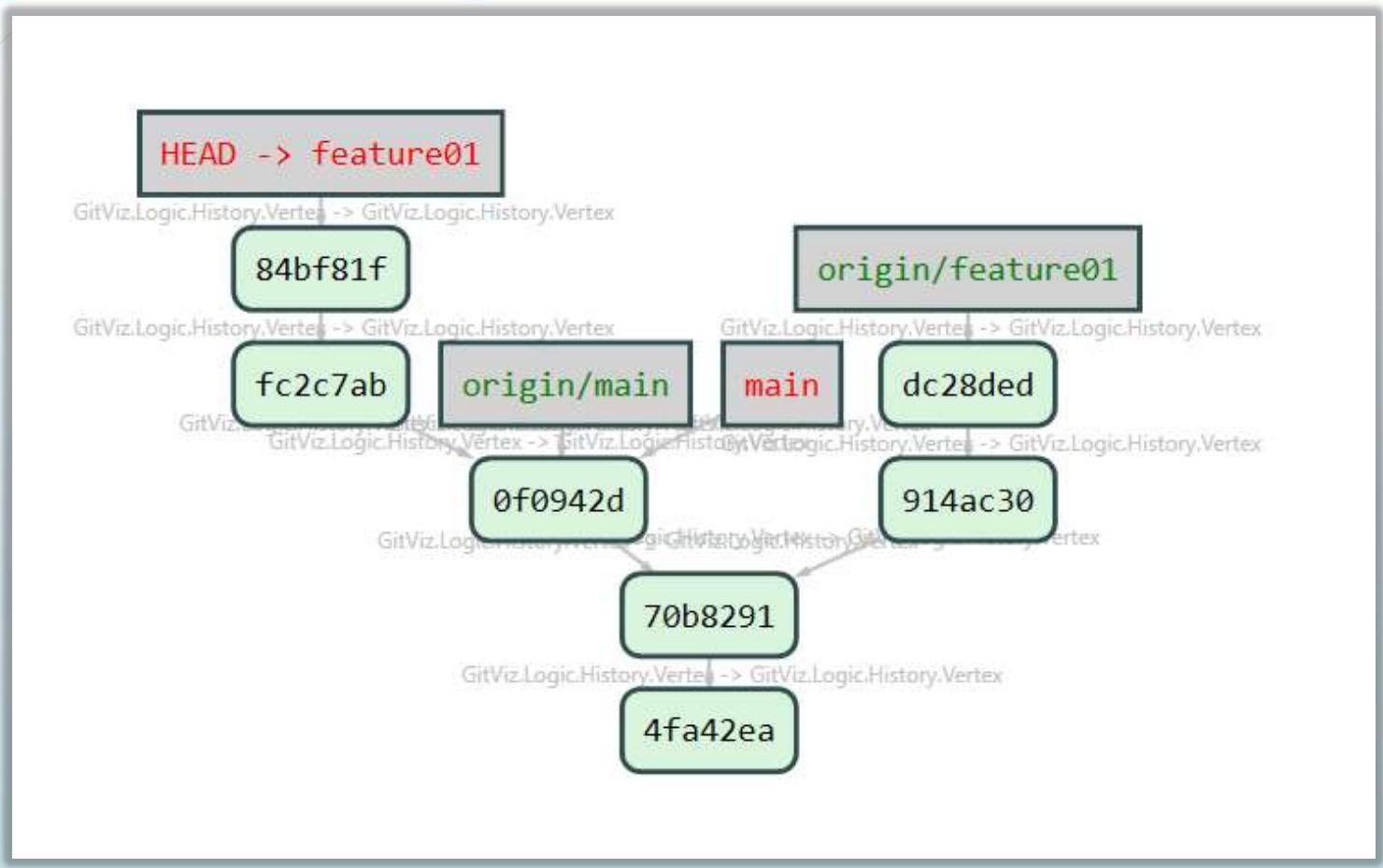




Rebase and Force Push

Another developer got changes in before you did, so you have to rebase and then force push

git rebase origin/main



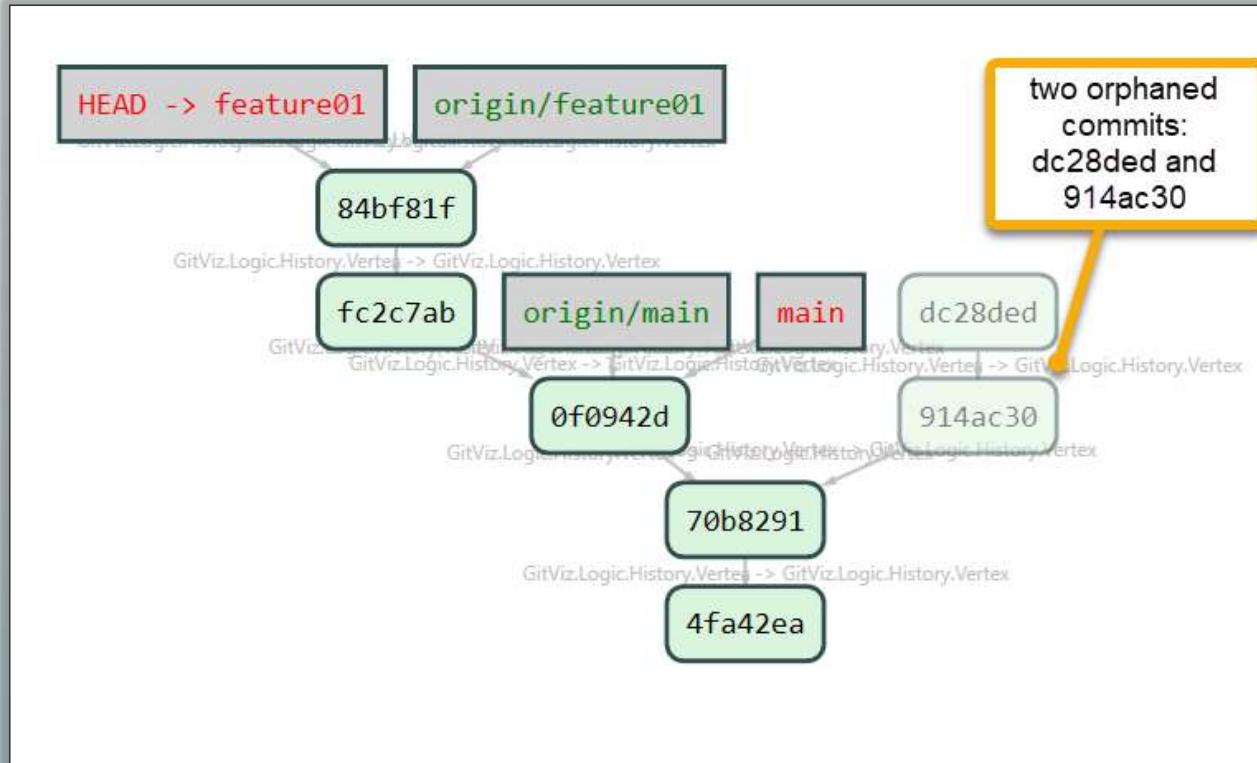
git status/git push/git push --force-with-lease

```
blgor@Voyager2 MINGW64 /c/          'RescueYourGitRepo (feature01)
$ git status
On branch feature01
Your branch and 'origin/feature01' have diverged,
and have 3 and 2 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)
```

```
blgor@Voyager2 MINGW64 /c/          /RescueYourGitRepo (feature01)
$ git push
To https://github.com/blgorman/RescueYourGitRepo.git
 ! [rejected]      feature01 -> feature01 (non-fast-forward)
error: failed to push some refs to 'https://github.com/blgorman/RescueYourGitRepo.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo (feature01)
$ git push --force-with-lease
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 641 bytes | 641.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/blgorman/RescueYourGitRepo.git
 + dc28ded...84bf81f feature01 -> feature01 (forced update)
```

git push --force-with-lease



A screenshot of a developer's workspace showing Visual Studio Code and a GitHub repository page.

Visual Studio Code:

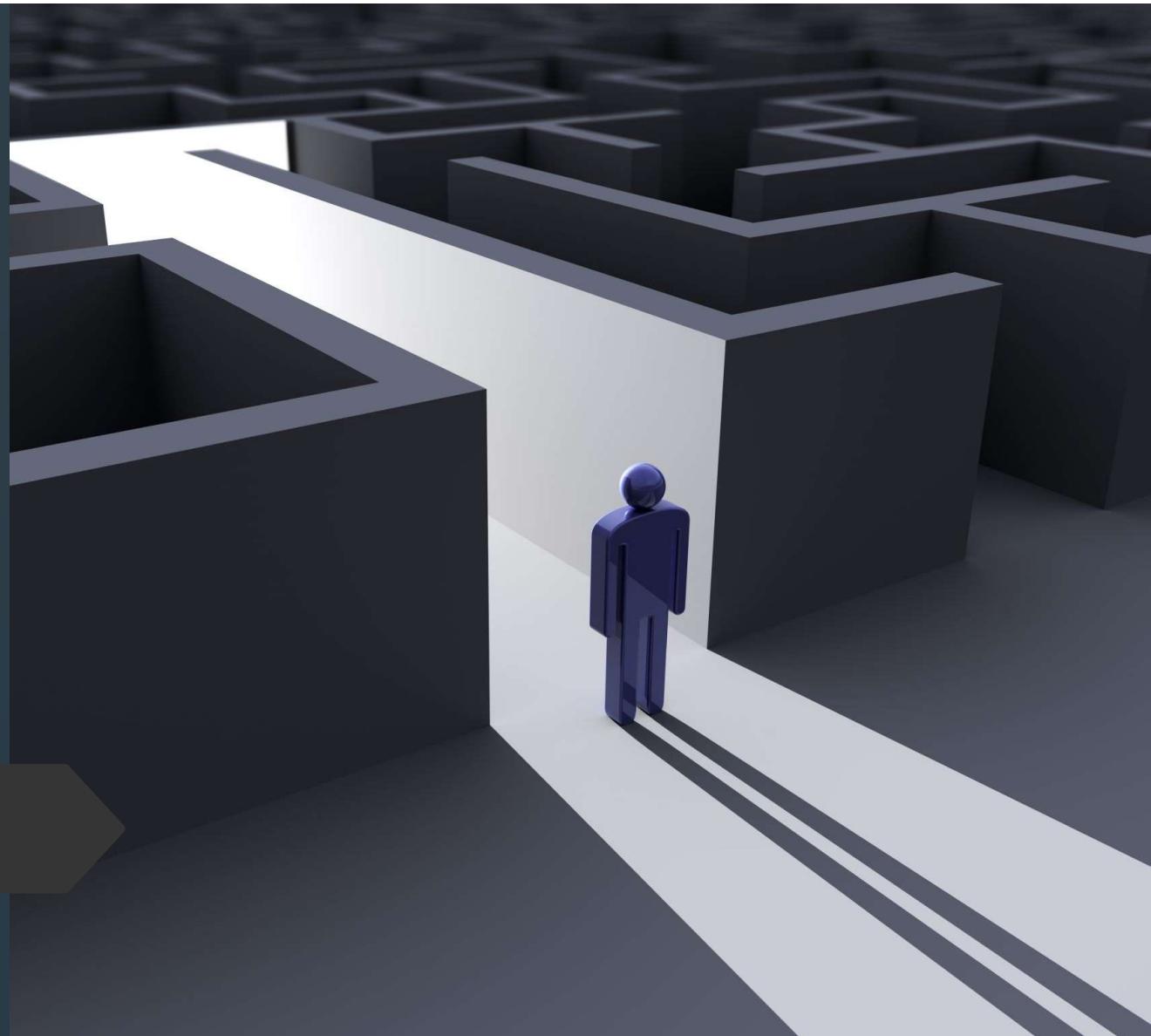
- Left Sidebar:** Shows icons for File, Edit, Search, Find, Replace, and Settings.
- File Explorer:** Displays files: featurechanges.txt, mainfile.txt, and featurechanges.txt.
- Code Editor:** Shows the content of featurechanges.txt.
- Terminal:** Shows the command line output:
blgor@Voyager2 MINGW64 /c/Business
oriantanboba_github/RescueYourGitRepo
\$
- Bottom Bar:** Includes tabs for main, Git Graph, and various file formats (Spaces: 4, UTF-8, CRLF, Plain Text), along with Spell Check and a search icon.

GitHub Repository Page (boba-fet/RescueYourGitRepo):

- Header:** Shows the repository name, boba-fet/RescueYourGitRepo, is public, and was forked from blgorman/RescueYourGitRepo.
- Navigation:** Includes links for Pull requests, Issues, Codespaces, Marketplace, and Explore.
- Branch Summary:** Shows 1 branch and 0 tags.
- Commit History:** Lists 17 commits by blgorman, starting with "updates for instructions after dry run" at 46ab49a (38 minutes ago) and ending with "update for ex3" at 0f0942d (2 days ago). Commit details include:
 - 01_TraditionalMerge: updates for gif demos in presentation (yesterday)
 - 02_LinearStrategy: updates for gif demos in presentation (yesterday)
 - 03_RebaseForcePush: updates for instructions after dry run (38 minutes ago)
 - 04_RecoverOrphaned...: some touchups to the files (2 days ago)
 - 05_MultipleFeaturesKe...: updates for instructions after dry run (38 minutes ago)
 - 06_MultipleFeaturesKe...: updates for instructions after dry run (38 minutes ago)
 - 07_AccidentallyPushed...: update for ex3 (2 days ago)
- About Section:** Describes the repository as the "Home base for the Rescue your Git Repo talk". It includes links for Readme, Stars (0), Watching (0), and Forks (2).
- Releases Section:** States "No releases published" and "Create a new release".
- Packages Section:** States "No packages published" and "Publish your first package".

Finding Lost Commits after history rewrite

When you rewrite your history but
need to get it back



git reflog --all

```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo (feature01)
$ git reflog --all
84bf81f (HEAD -> feature01, origin/feature01) refs/remotes/origin/feature01@{0}: update by push
84bf81f (HEAD -> feature01, origin/feature01) refs/heads/feature01@{0}: rebase (finish): refs/heads/feature01 onto 0f0942d73bbefdfdac2e
3cb39d8f6
84bf81f (HEAD -> feature01, origin/feature01) HEAD@{0}: rebase (finish): returning to refs/heads/feature01
84bf81f (HEAD -> feature01, origin/feature01) HEAD@{1}: rebase (pick): updates for feature branch
fc2c7ab HEAD@{2}: rebase (pick): Added code changes for feature 01
0f0942d (origin/main, main) HEAD@{3}: rebase (start): checkout origin/main
dc28ded refs/remotes/origin/feature01@{1}: update by push
dc28ded refs/heads/feature01@{1}: commit: updates for feature branch
dc28ded HEAD@{4}: commit: updates for feature branch
914ac30 refs/heads/feature01@{2}: branch: Created from refs/remotes/origin/feature01
914ac30 HEAD@{5}: checkout: moving from main to feature01
0f0942d (origin/main, main) refs/heads/main@{0}: pull: Fast-forward
0f0942d (origin/main, main) HEAD@{6}: pull: Fast-forward
914ac30 refs/remotes/origin/feature01@{2}: fetch: storing head
0f0942d (origin/main, main) refs/
70b8291 refs/remotes/origin/main@... orphaned commits show up in reflog
70b8291 refs/heads/main@{1}: commit: updates for force push demo walkthrough
```

Checkout your lost commits in a detached state

```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo (feature01)
$ git checkout dc28ded
Note: switching to 'dc28ded'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

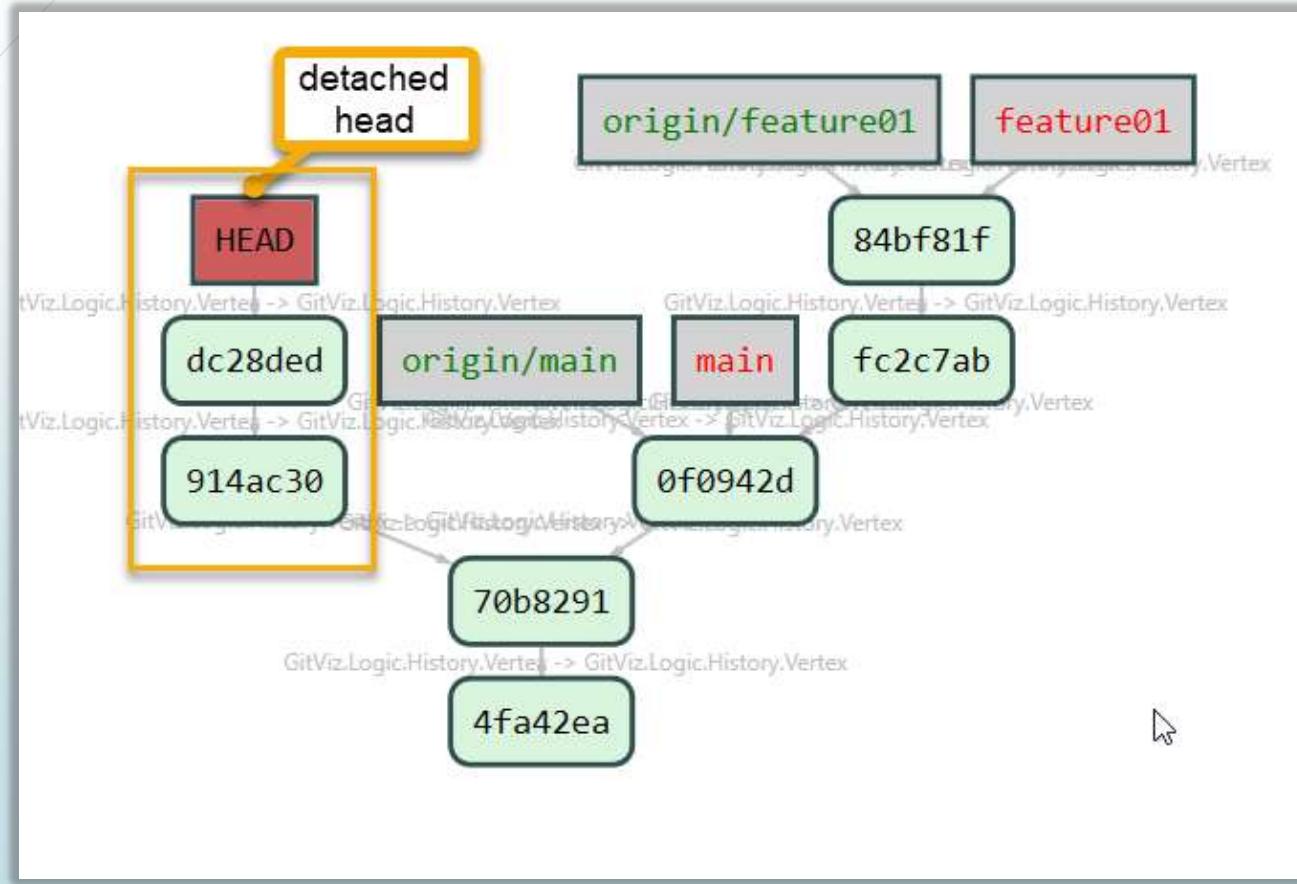
```
git switch -
```

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at dc28ded updates for feature branch

```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo ((dc28ded...))
$
```

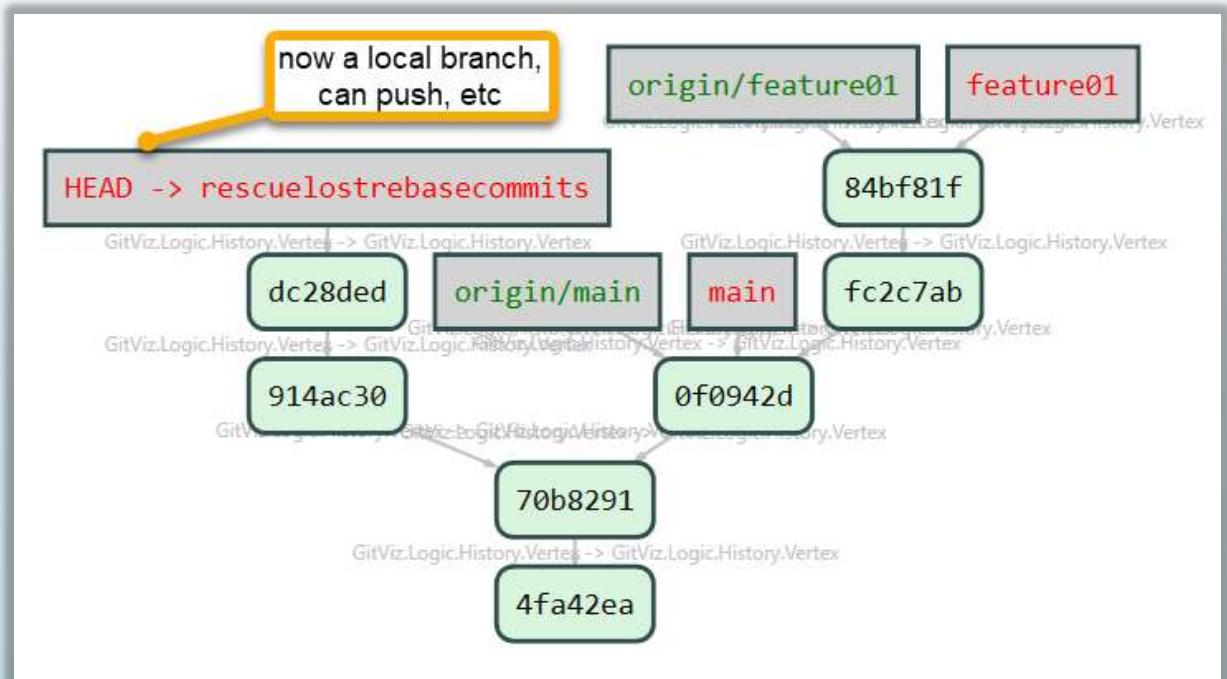
The current state (visually)

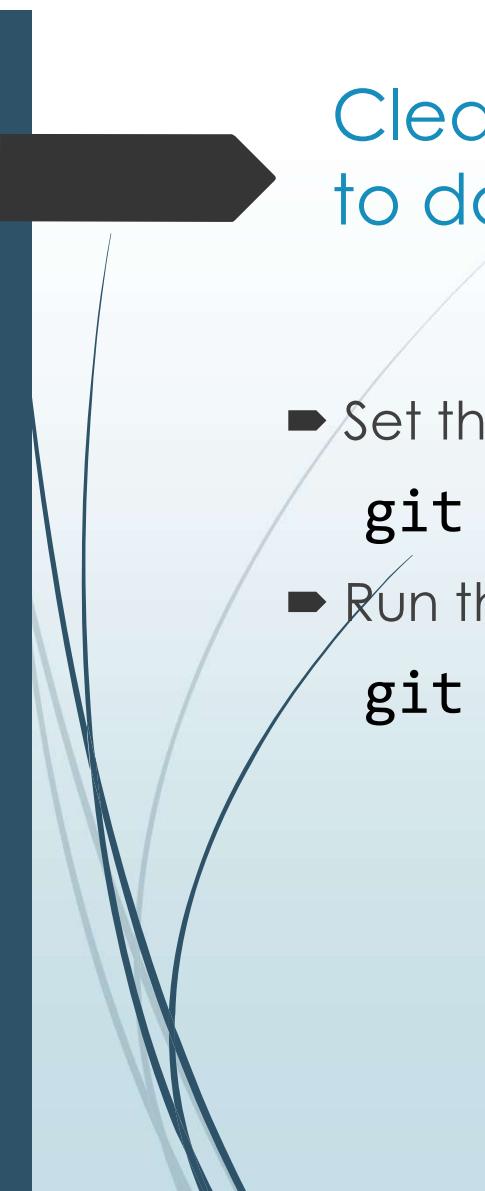


git checkout -b <new-branch-name> or git switch -c <new-branch-name>

```
blgor@Voyager2 MINGW64 /c/          /RescueYourGitRepo ((dc28ded...))  
$ git switch -c rescuelostrebasecommits  
Switched to a new branch 'rescuelostrebasecommits'
```

```
blgor@Voyager2 MINGW64 /c/          /RescueYourGitRepo (rescuelostrebasecommits)  
$
```





Clear your local Cache (you don't ever need to do this, and probably shouldn't ever do it)

- ▶ Set the reflog to expire all current unreachable commits

```
git reflog expire --expire-unreachable=now --all
```

- ▶ Run the garbage collector

```
git gc --prune=now
```

The screenshot shows a Visual Studio Code interface with several panes:

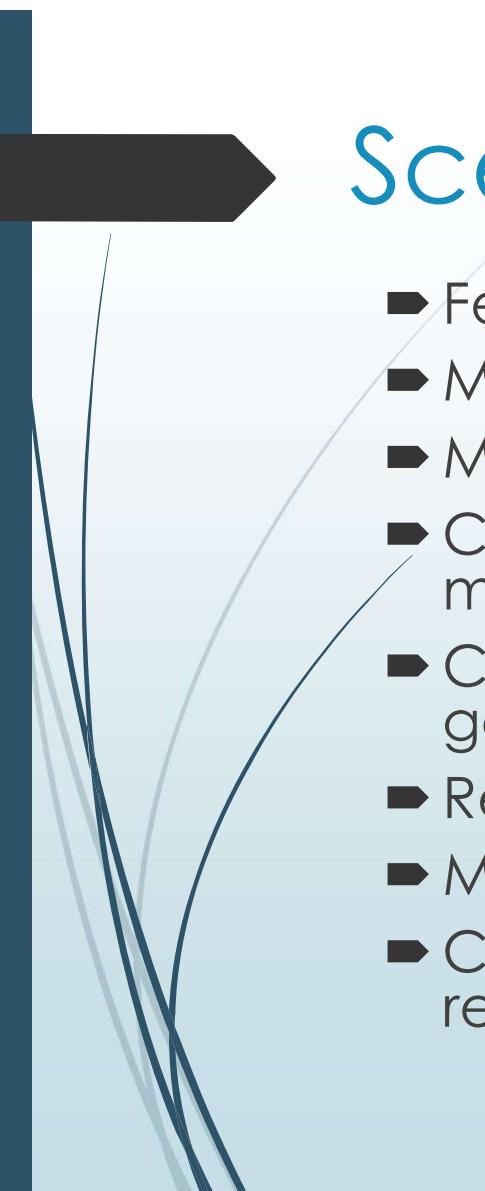
- Left Sidebar:** Includes icons for file operations, search, and other development tools.
- Top Bar:** Shows the file "featurechanges.txt - RescueYourGitRepo - Visual Studio Code".
- Editor:** Displays two files: "featurechanges.txt" and "mainfile.txt". "featurechanges.txt" contains code related to a feature branch simulation.
- Terminal:** Shows a command-line session:

```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandal  
orianboba_github/RescueYourGitRepo (main)  
$ git branch -D myfeature  
Deleted branch myfeature (was b4011ea).  
  
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandal  
orianboba_github/RescueYourGitRepo (main)  
$
```
- Bottom Bar:** Includes icons for main, refresh, search, Git Graph, and various text settings like Spaces: 4, UTF-8, CRLF, Plain Text, Spell Check, and a gear icon with a '1'.
- Right Panel:** A "GitViz (Beta) - RescueYourGitRepo" window showing a visual representation of the repository's history. It features a grid of commit hash nodes and three header nodes: "HEAD -> main" (red), "origin/main" (green), and "origin/HEAD" (green).



Scenario #2 – Multiple Merged Features Ready for Production

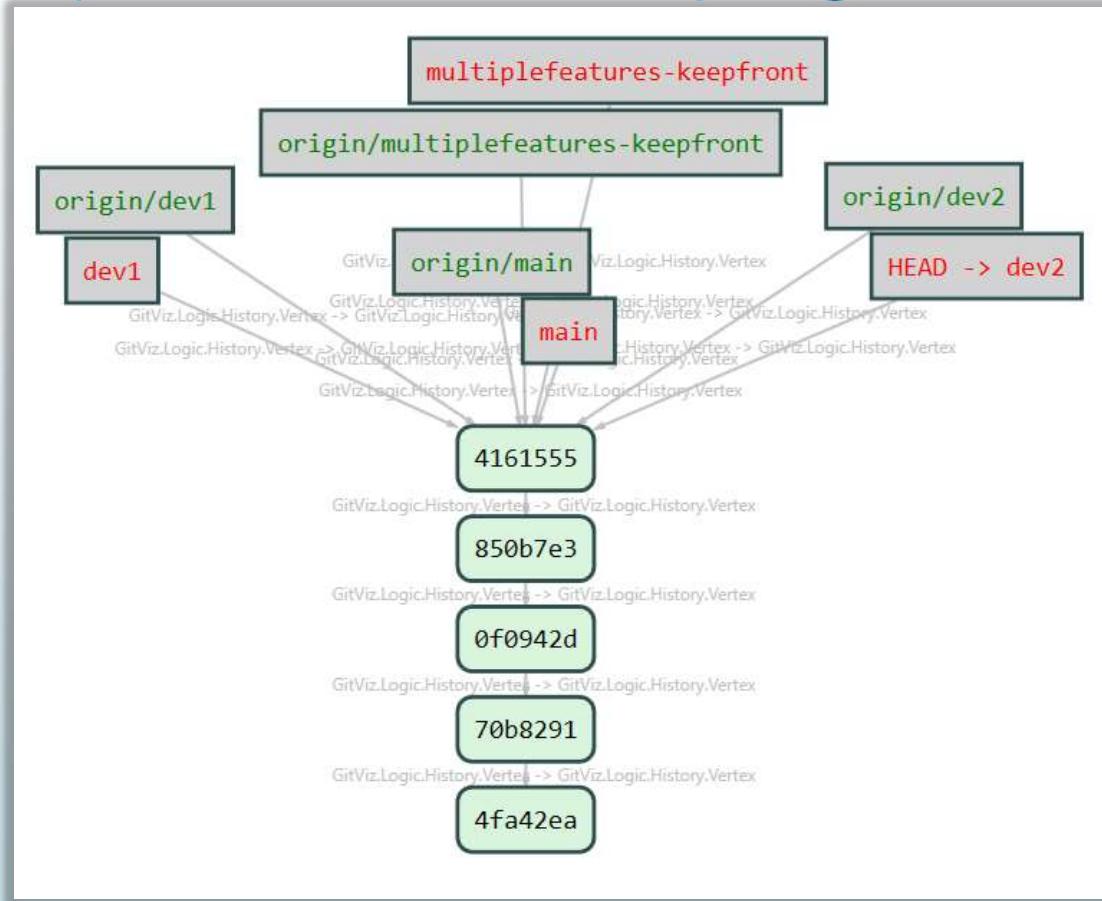
We've merged multiple features, but we only need one to go live right now



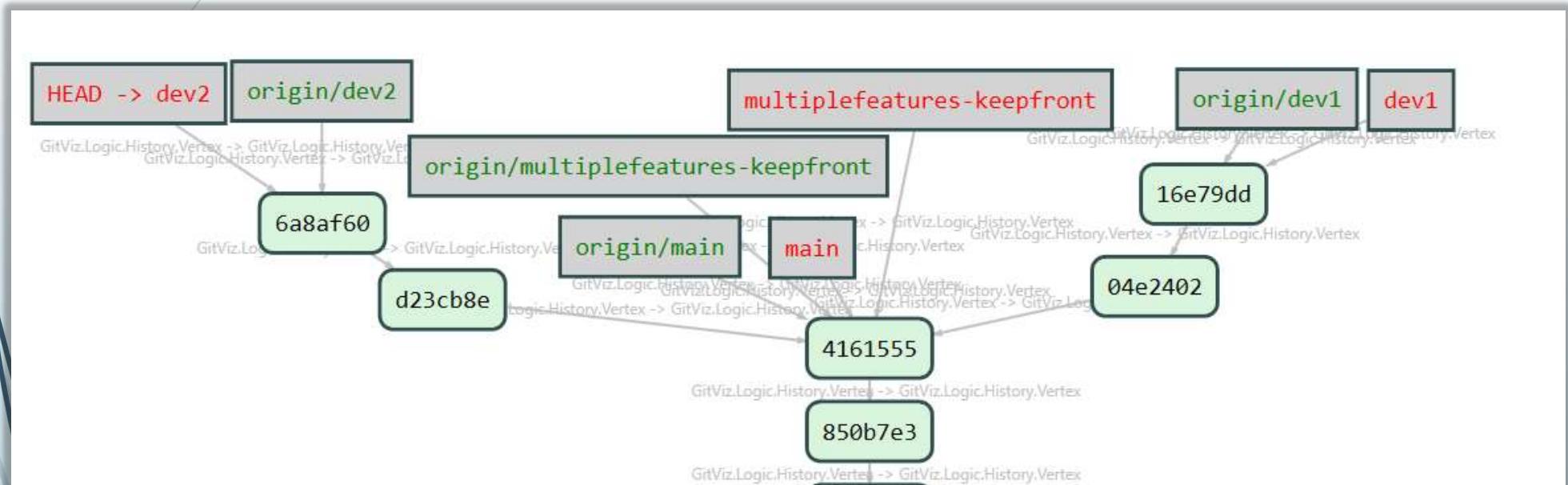
Scenario/Set up

- ▶ Feature/flow-based commit strategy
- ▶ Multiple devs/teams creating features
- ▶ Multiple features are merged
- ▶ Commit position can be directly after last release on main
- ▶ Commit position can be after the feature that is not getting promoted
- ▶ Requires a git reset command
- ▶ May also require a cherry-pick (not the first feature)
- ▶ Consider using feature flag release strategy to avoid rewriting history

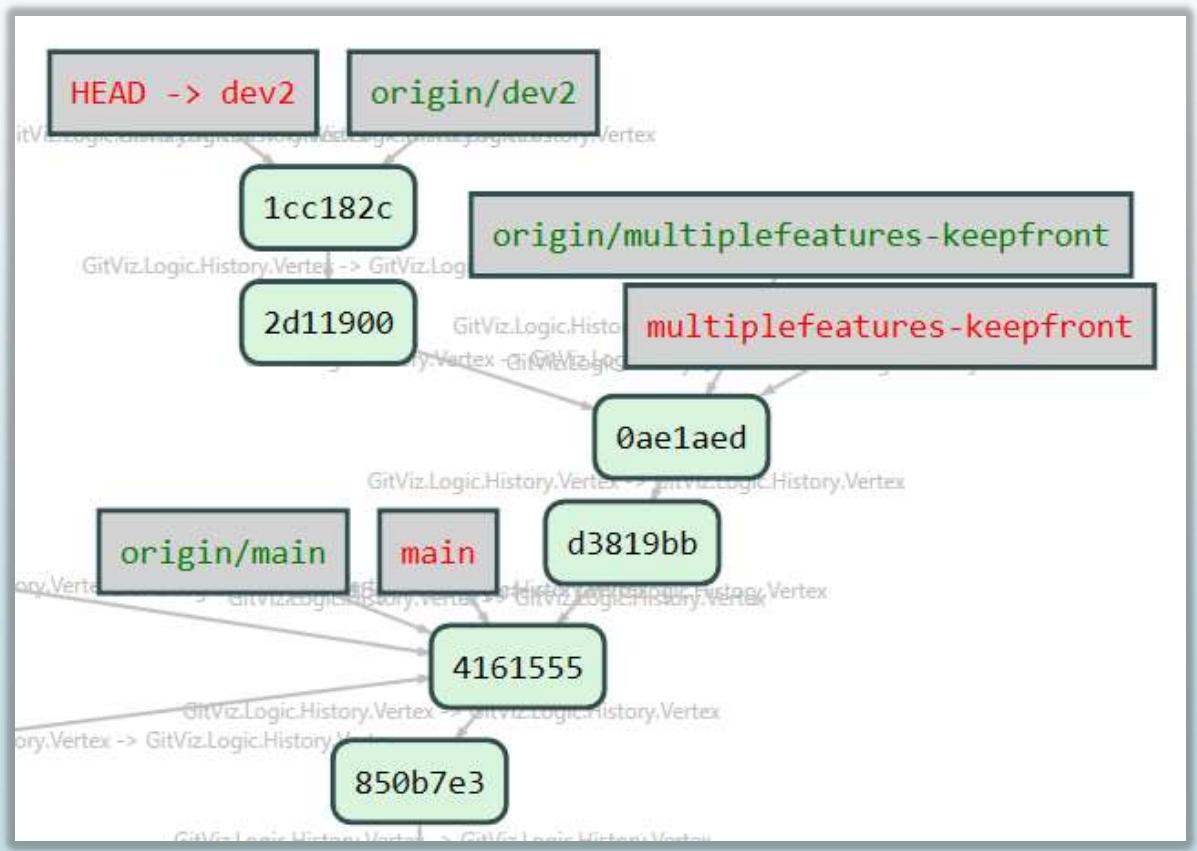
Multiple features keeping the front



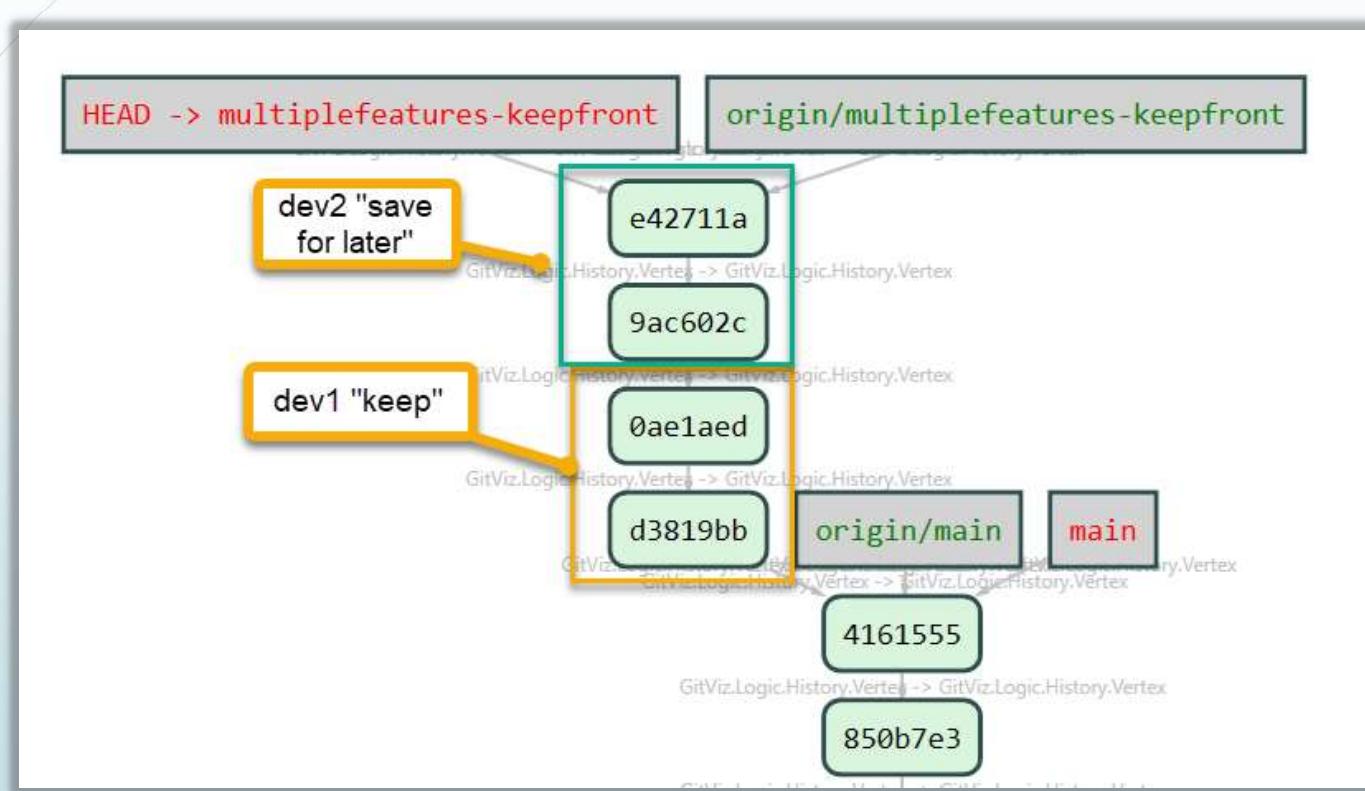
Move both dev branches forward a couple of commits



Create one linear tree with all commits using rebase



Rebase merge the dev2 branch



The screenshot shows a Visual Studio Code interface with a Git history visualization and a terminal window.

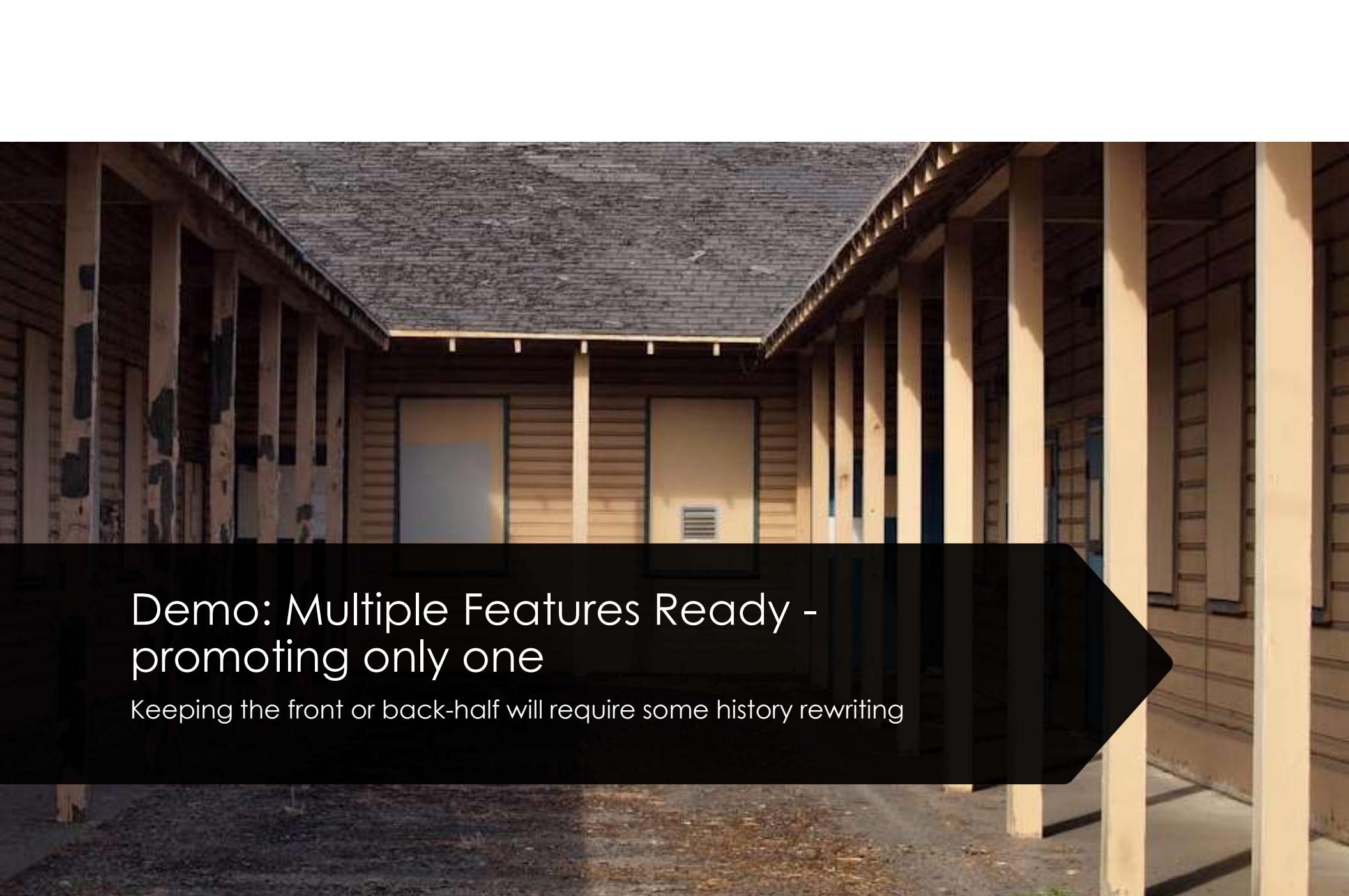
Git History Visualization: The right-hand panel displays a vertical list of commit history. At the top, three boxes indicate the current state: **HEAD -> main** (red), **origin/main** (green), and **origin/HEAD** (green). Below these are 18 commit entries, each consisting of a hex commit hash in a blue rounded rectangle, followed by the author, date, and message. The commits are arranged from oldest at the bottom to newest at the top.

```
HEAD -> main
origin/main
origin/HEAD
1f53b8e
d8ca4de
acafb7c
5864681
1c6ff15
9de1570
dcaa5bc
46ab49a
6e1842f
83e712d
f2561f4
e7c48a0
9da1497
286b771
751bb97
```

Terminal: The bottom-left panel shows a terminal window with the following output:

```
b1gor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (main)
$ [ ]
```

Status Bar: The bottom bar shows the current file is **dev1.txt**, the repository is **RescueYourGitRepo**, and the commit count is **0**. It also includes icons for Git Graph, Spell Check, and other settings.

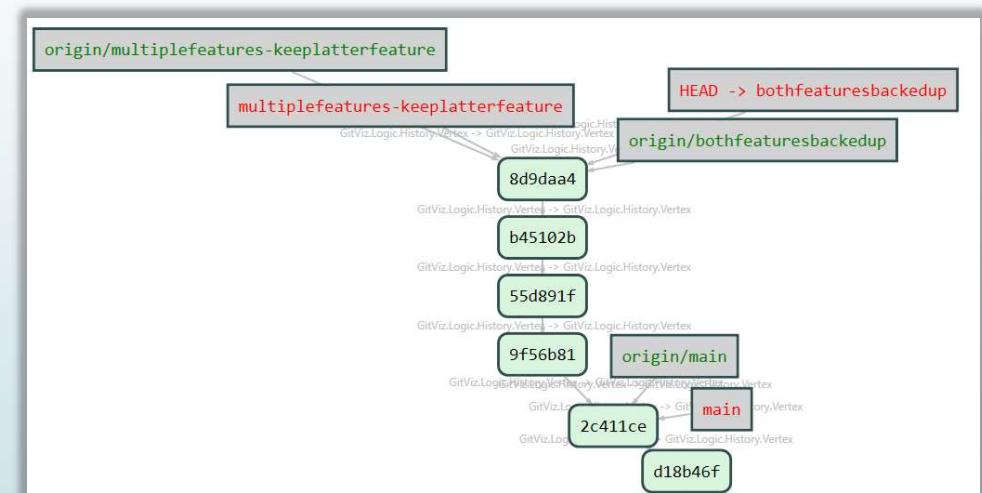


Demo: Multiple Features Ready -
promoting only one

Keeping the front or back-half will require some history rewriting

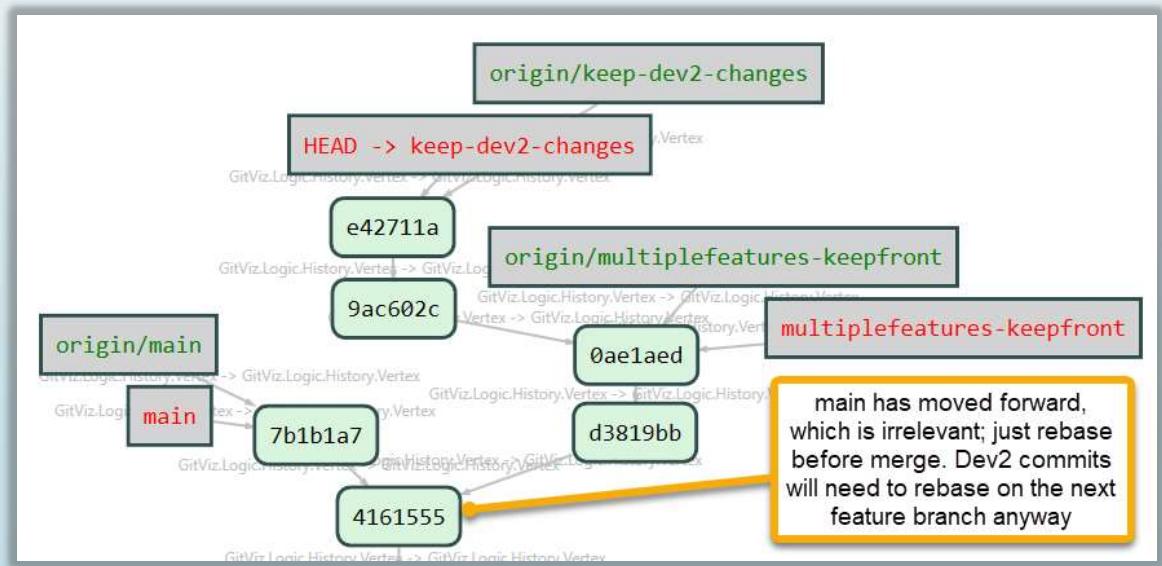
The setup is the same for either walkthrough

- ▶ Set up as before
- ▶ Create a backup branch



Keeping the front half

- ▶ Create a branch to keep the back half safe
- ▶ Reset the feature branch to the last commit you want to keep
- ▶ Create a new feature branch and pick the remaining commits to promote in the next cycle



Reset the feature, then pick commits

The screenshot shows a terminal window and a GitViz visualization side-by-side.

Terminal Output:

```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo (multiplefeatures-keeplatterfeature)
$ git cherry-pick b45102b
[multiplefeatures-keeplatterfeature 274700a] dev2-1
Date: Tue Apr 4 07:04:00 2023 -0500
1 file changed, 3 insertions(+), 1 deletion(-)

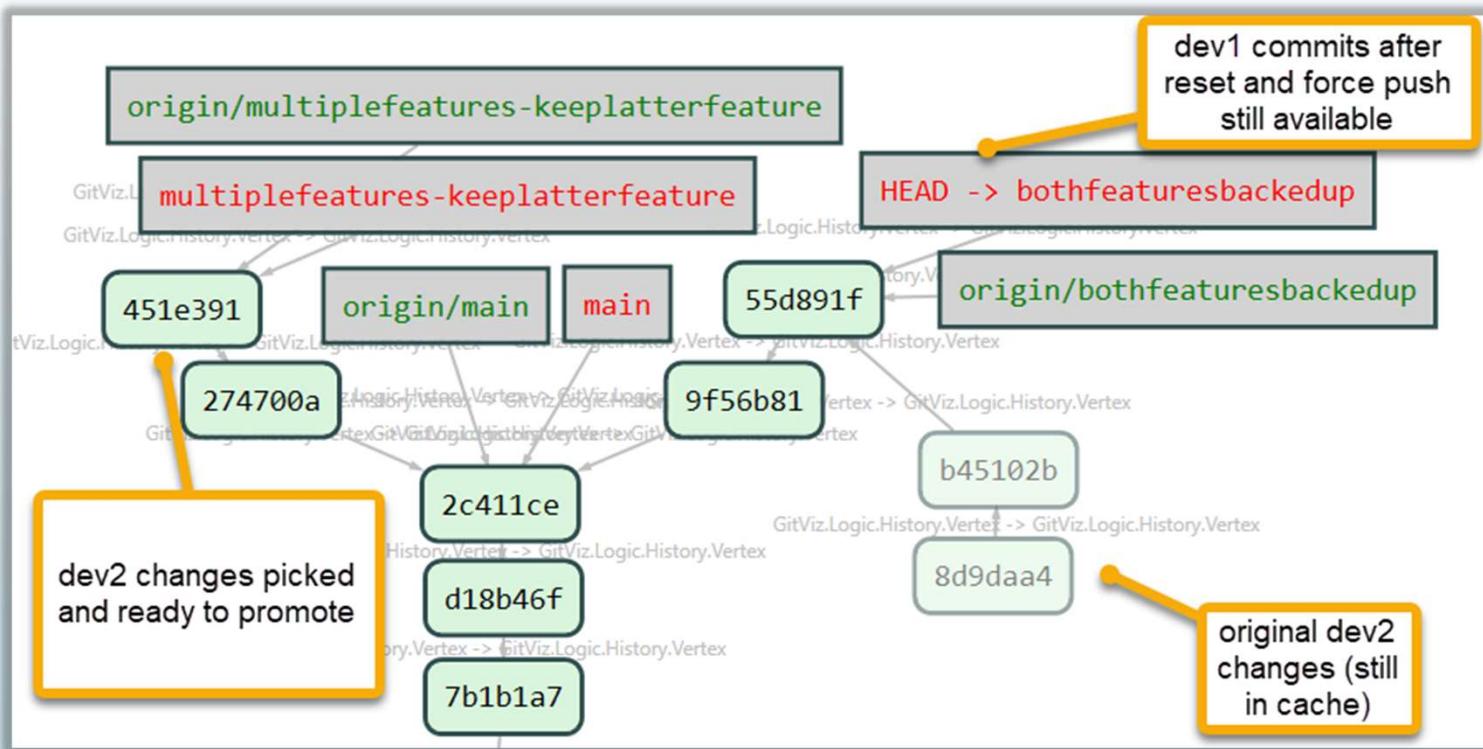
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo (multiplefeatures-keeplatterfeature)
$ git cherry-pick 8d9daa4
[multiplefeatures-keeplatterfeature 451e391] dev2-2
Date: Tue Apr 4 07:04:22 2023 -0500
1 file changed, 2 insertions(+), 1 deletion(-)
```

GitViz Visualization:

A graph showing commit history. Commits are represented by colored boxes: green for 'origin/bothfeaturesbackup' and red for 'bothfeaturesbackup'. The 'HEAD' commit is red and labeled **HEAD -> multiplefeatures-keeplatterfeature**. Other commits shown include 8d9daa4, b45102b, 55d891f, 9f56b81, 2c411ce, d18b46f, 7b1b1a7, 4161555, and 850b7e3. A red box highlights the 'HEAD' commit, and another red box highlights the 'main' commit.

Reset backup branch to remove f2

- Remove Feature 2 from the backup branch to keep only feature one



The screenshot shows a Visual Studio Code interface with several panes:

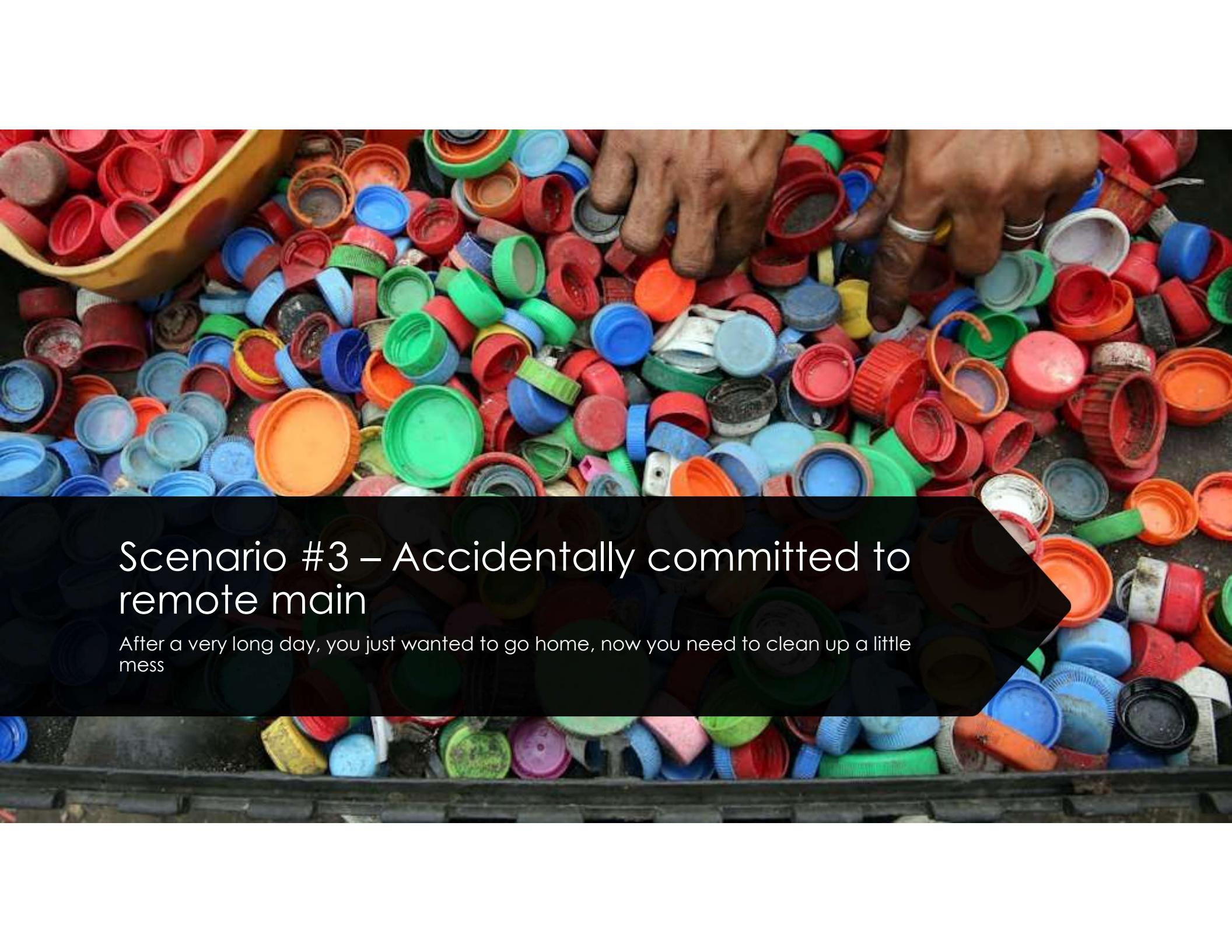
- Left Sidebar:** Includes icons for file operations (New, Open, Save, Find, Replace, Undo, Redo), a search bar, and a pinned icon.
- Top Bar:** Shows the title "dev1.txt - RescueYourGitRepo - Visual Studio Code" and standard window controls.
- Editor Area:** Displays two files: "dev1.txt" and "dev2.txt". "dev1.txt" contains the following content:

```
You, 5 minutes ago | 1 author (You)
1 # I am dev1. Hear me roar!
2
3
4 Commit 1 again for dev 1
5 Commit 2 again for dev 1
```
- Terminal:** Shows a command-line session:

```
-keeplatter)
$ git expireunreachablenow

blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (multiplefeatures-keeplatter)
$ git gcn unreachablenow
Enumerating objects: 120, done.
Counting objects: 100% (120/120), done.
Delta compression using up to 16 threads
Compressing objects: 100% (62/62), done.
Writing objects: 100% (120/120), done.
Total 120 (delta 45), reused 106 (delta 39), pack-reused
0

blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (multiplefeatures-keeplatter) [
```
- Bottom Status Bar:** Shows tabs for "multiplefeatures-keeplatter" and "Plain Text", along with "Git Graph" and other status indicators.
- Right Panel:** A "Git Graph" visualization showing the commit history. It features a tree structure with commits represented as rounded rectangles. Labels include "multiplefeatures-keeplatter", "origin/multiplefeatures-keeplatter", "c562445", "5848db7", "61b1a63", "0cebf03", "origin/main", "origin/HEAD", "main", "1f53b8e", "d8ca4de", "acafb7c", "5864681", "1c6ff15", "9de1570", "dcaa5bc", "46ab49a", "6e1842f", "83e712d", and "f2561f4". Some labels are in red (e.g., "multiplefeatures-keeplatter") and others in green (e.g., "origin/main").



Scenario #3 – Accidentally committed to remote main

After a very long day, you just wanted to go home, now you need to clean up a little mess

How can I fix this?

- ▶ Pick your commit to a new branch
- ▶ Reset main
- ▶ Revert the change to keep history



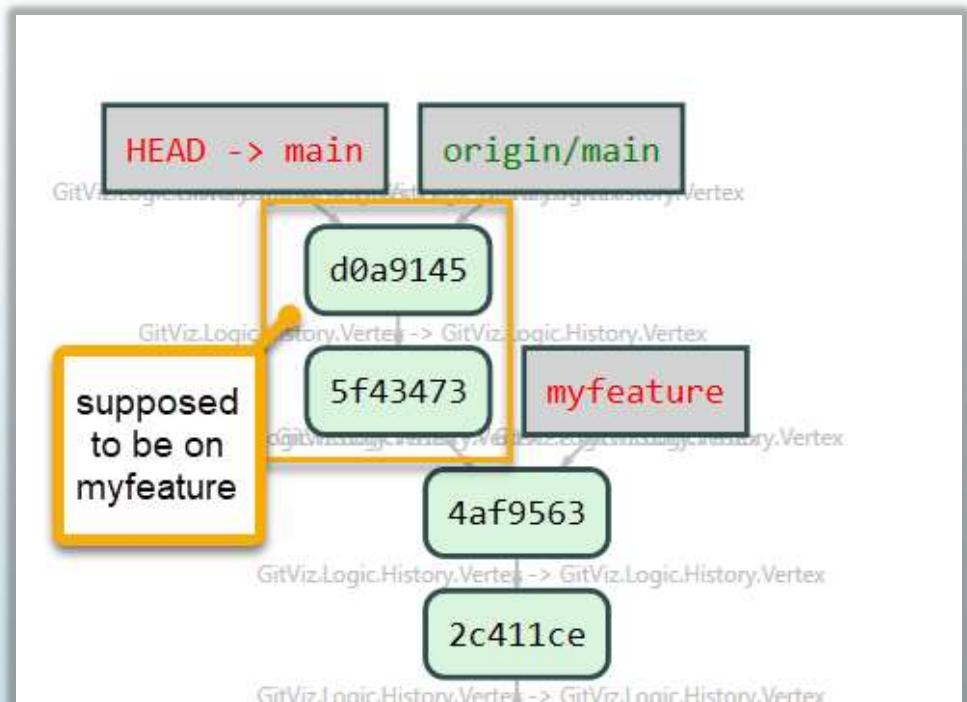


Demo: Direct push to remote main

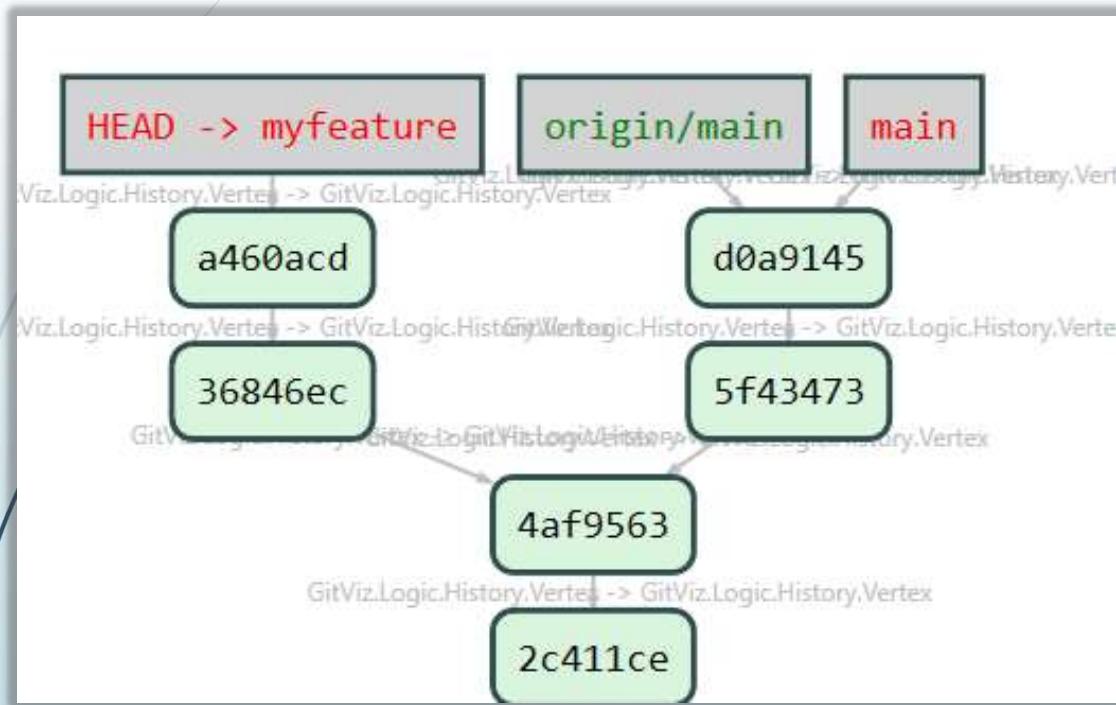
I accidentally pushed my changes to remote main, how can I fix this?

Get Set Up

- ▶ Create a feature branch and/or dev branch to simulate your work
- ▶ Forget to switch to the branch
- ▶ Move your branch forward a couple of commits (oops....it's on main, but you don't notice)
- ▶ Blindly push your changes assuming you are on your branch



Pick the commits to your feature branch

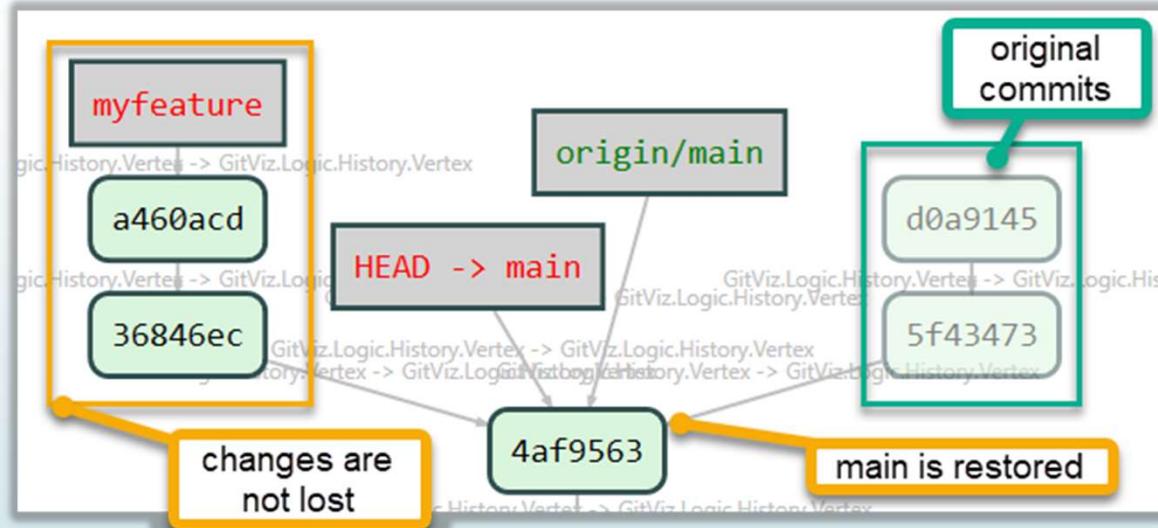


git switch myfeature

git cherry-pick 5f43473

git cherry-pick d0a9145

Switch to main, reset, and force push



```
git switch main  
git reset --hard 4af9563  
git push --force-with-lease  
git pull --prune
```

The screenshot shows a Visual Studio Code interface with the following components:

- File Explorer:** On the left, showing files `dev1.txt`, `dev2.txt`, and `dev1.txt`.
- Editor:** The main area displays `dev1.txt` with the following content:

```
You, 12 minutes ago | 1 author (You)
1 # I am dev1. Hear me roar!
2
3
4 Commit 1 again for dev 1
5 Commit 2 again for dev 1
```
- Terminal:** Shows a command-line session:

```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main

blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (main)
$
```
- Git Graph:** A visualization of the repository's commit history. It shows a tree structure with the following commits:
 - `HEAD -> main` (red)
 - `origin/main` (green)
 - `origin/HEAD` (green)
 - `7147d09`
 - `4fa8f1b`
 - `57a9135`
 - `09c6d62`
 - `1f53b8e`
 - `d8ca4de`
 - `acafb7c`
 - `5864681`
 - `1c6ff15`
 - `9de1570`
 - `dcaa5bc`
 - `46ab49a`
 - `6e1842f`
 - `83e712d`
 - `f2561f4`
- Bottom Bar:** Includes icons for file operations, a gear icon with a '1' (notifications), and tabs for `main`, `Git Graph`, `Spaces: 4`, `UTF-8`, `CRLF`, `Plain Text`, `Spell`, and a search icon.



Your Team Members

- ▶ They might `hate` you right now
- ▶ If they are out of sync and have your bad commits:
 - ▶ Hard reset from origin

git switch main

git fetch

git reset --hard origin/main

- ▶ If they are out of sync and also made this error:
 - ▶ Follow the same procedure to pick their commits to their feature
 - ▶ Hard reset main from origin

```
39 19 18832 932 1816 5 0.7 0.1 0:00:46 matrix -b
39 19 4508 1580 1476 5 0.7 0.2 0:00:37 /bin/bash /usr/bin/
39 19 40940 12196 4728 5 0.8 1.2 0:00:39 /usr/bin/python
20 0 3568 2884 1772 5 0.9 0.3 0:01:24 /usr/bin/less -d
20 0 2838 364 28 5 0.9 0.9 0:00:37 /usr/bin/lshw -v
20 0 3568 2884 1772 5 0.9 0.3 0:02:58 /usr/bin/less -d
20 0 4688 4360 1972 5 0.9 0.4 0:01:35 /usr/bin/bash /usr/bin/
39 19 19720 2964 2828 5 0.9 0.3 0:00:31 /usr/bin/lshw -v
20 0 2938 364 28 5 0.9 0.9 0:00:35 /usr/bin/bash /usr/bin/
39 19 19704 2878 2368 5 0.9 0.2 0:00:37 less -s -show-all
39 19 33016 3368 3000 5 0.9 0.4 0:00:28 /usr/bin/lshw -v
20 0 4688 4360 1972 5 0.9 0.2 0:01:37 /usr/bin/python -i
20 0 1888 2064 1736 5 0.9 0.2 0:00:37 /usr/bin/python -i
20 0 6550 24280 10940 5 0.9 2.4 0:00:39 /usr/bin/python -i
39 19 4508 1584 1476 5 0.9 0.2 0:00:39 /usr/bin/python -i
20 0 4508 4744 3988 5 0.9 0.5 0:00:33 /usr/bin/bash /usr/bin/
20 0 6428 3568 508 5 0.9 0.3 0:00:34 /usr/bin/python -i
20 0 19644 2892 2628 5 0.9 0.2 0:00:36 /usr/bin/python -i
39 19 4776 1928 1532 5 0.9 0.2 0:00:32 /usr/bin/python -i
39 19 4508 1816 1428 5 0.9 0.2 0:00:35 /usr/bin/python -i
39 19 4508 1592 1488 5 0.9 0.2 0:00:32 /usr/bin/python -i
39 19 4508 1596 1472 5 0.9 0.7 0:00:32 /usr/bin/python -i
39 19 4508 1596 1472 5 0.9 0.2 0:00:32 /usr/bin/python -i
20 0 5858 7398 5440 5 0.9 0.2 0:00:37 /usr/bin/python -i
39 19 4508 1620 1468 5 0.9 0.2 0:00:37 /usr/bin/python -i
39 19 5538 13884 4784 5 0.9 0.9 0:00:37 /usr/bin/python -i
```

ERROR: 12 Cannot allocate memory

Screen 0001: speedometer 2.0

Screen 0002:

Screen 0003: less -s -show-all

Screen 0004: less -s -show-all

The running environment contains many more processes. To prevent them

from appearing in the history, use

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

history -c

history -z

history -d

history -r

history -s

history -w

history -x

history -y

history -z

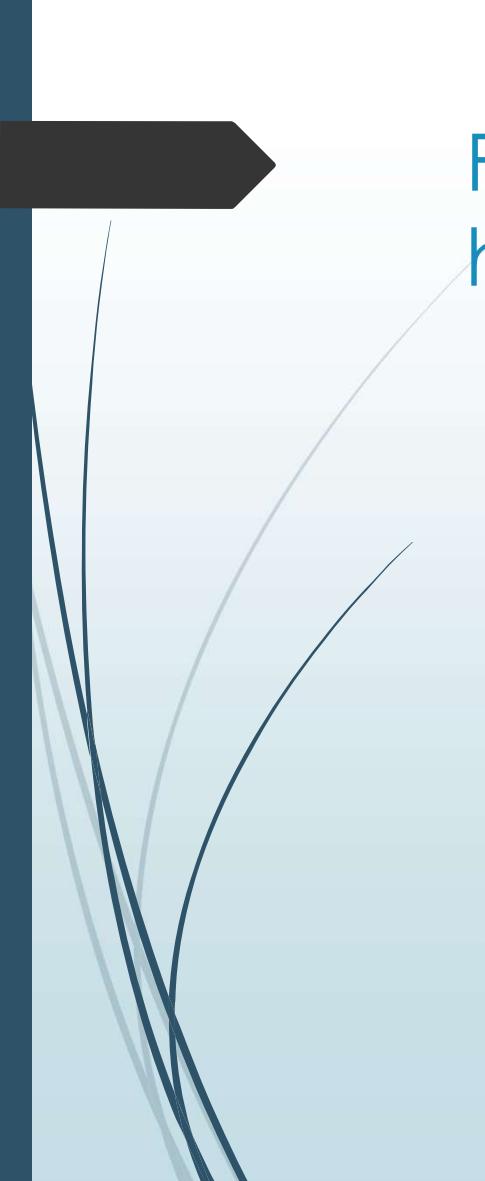
history -c

history -z

history -d

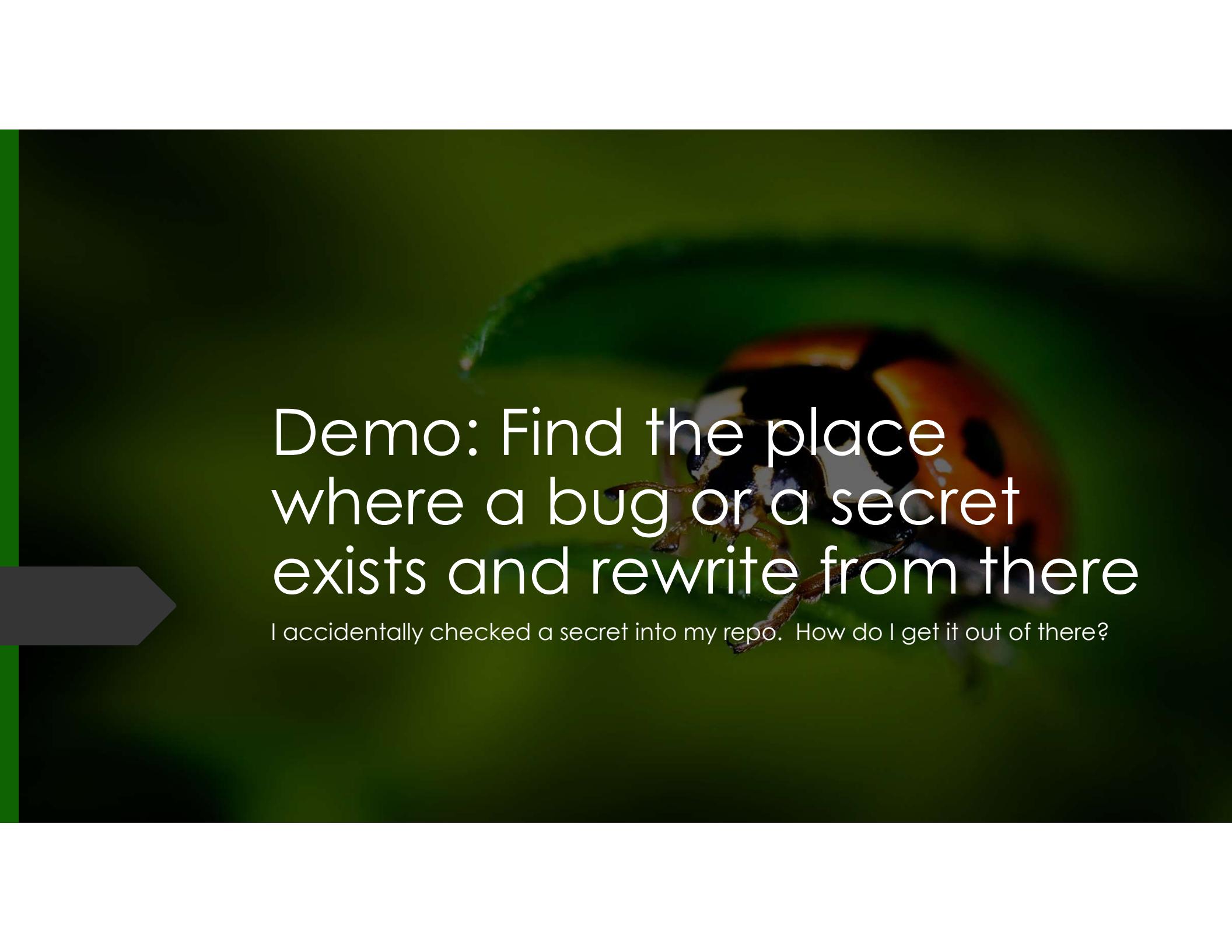
history -r

history -s</p



Find the first `bad` commit, then rewrite history

- ▶ FREEZE Development -> history is about to change
- ▶ Use `git bisect` to find the first commit where the secret is in the repo [or do a log search]
- ▶ Once the commit is located, the fix will be to rewrite the history from that point with the secret removed.
- ▶ After history rewrite, you can restore any dev/feature branches
- ▶ You can also pick any commits that are no longer based on the trunk after this operation [dev work in progress]



Demo: Find the place where a bug or a secret exists and rewrite from there

I accidentally checked a secret into my repo. How do I get it out of there?

Review Commit History

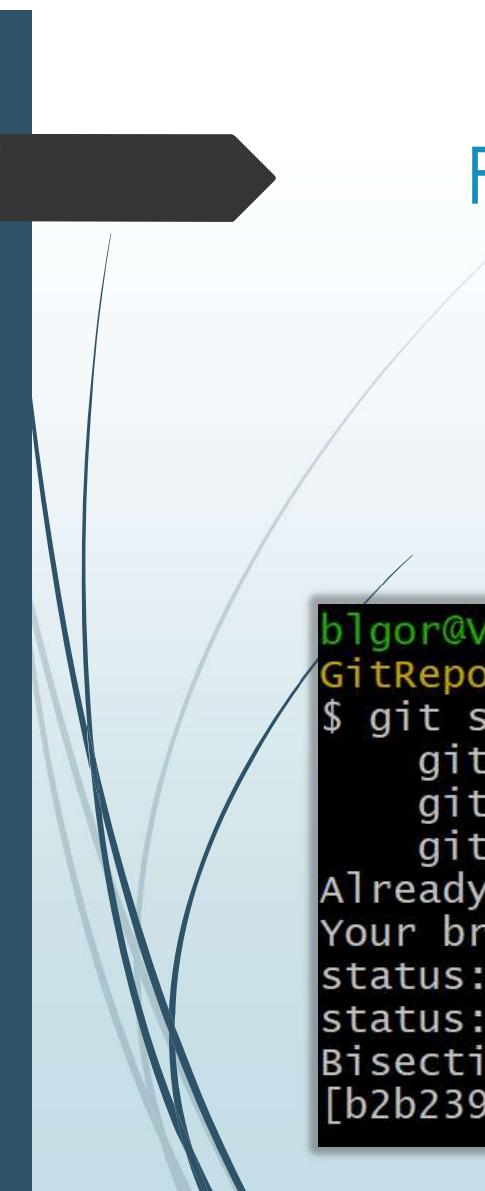
```
$ git log --oneline
9728ac3 (HEAD -> main, origin/main, origin/HEAD) uncommen
ting
5d704e8 updated stock image
56c7d07 added todo info
8da34cb delete working
1c8f7ba fixed a small bug
d8b5272 delete blob
6d6381a list and download
f5a892c upload blob
b30bf63 got the storage container working
444157c working on blob storage code
d0829ce working on container storage code
b2b2398 container code
ff7b8ce started working on the container code
ff997b4 Added configurationfile and loaded it
543491e Created the initial project
4850171 Initial commit
```

Remember commit b2b2398 as this half-way commit will be the first check in bisect

bad

error was introduced somewhere between a known good and bad commit

good



Find the first bad commit

```
git switch main  
git bisect start  
git bisect good 4850171  
git bisect bad f5a892c
```

```
b1gor@Voyager2 MINGW64 /c/  
GitRepo-RemoveASecret (main)  
$ git switch main  
    git bisect start  
    git bisect good 4850171  
    git bisect bad f5a892c  
Already on 'main'  
Your branch is up to date with 'origin/main'.  
status: waiting for both good and bad commits  
status: waiting for bad commit, 1 good commit known  
Bisecting: 3 revisions left to test after this (roughly 2 steps)  
[b2b2398467a82c55774ea901d6bac7e414400a5d] container code
```



Review the file to validate if it is good or bad

```
b1gor@Voyager2 MINGW64 /c/ /Rescue  
GitRepo-RemoveASecret ((b2b2398...)|BISECTING)  
$ cat SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json  
{  
  "Storage": {  
    "ConnectionString": "theconnectionstring"  
  }  
}
```



this commit is `good`

Mark commit as good, review next commit.
Mark as bad. Repeat until finished

```
$ git bisect good
Bisecting: 1 revision left to test after this (roughly 1 step)
[444157c75946a004cab25425ba53776b69f27108] working on blob storage code

blgor@Voyager2 MINGW64 /c/ GitRepo-RemoveASecret ((444157c...)|BISECTING)
$ cat SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json
{
    // "Storage": {
    //     "ConnectionString": "DefaultEndpointsProtocol=https;AccountName=simpl
    estor20251231blg;AccountKey=d2505129a437fb435fd83ec%7a9f762a3b51==;Endpoint
    Suffix=core.windows.net"
    //}
}
```

this commit is 'bad'

Bisect identifies the first bad commit

```
b1gor@Voyager2 MINGW64 /c/ /RescueY  
GitRepo-RemoveASecret ((d0829ce...)|BISECTING)  
$ cat SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json  
{  
    "Storage": {  
        "ConnectionString": "DefaultEndpointsProtocol=https;AccountName=simp  
tor20251231blg;AccountKey=d2505129a437fb435fd83ec%7a9f762a3b51==;Endpo  
nfix=core.windows.net"  
    }  
}  
  
b1gor@Voyager2 MINGW64 /c/ /RescueY  
GitRepo-RemoveASecret ((d0829ce...)|BISECTING)  
$ git bisect bad  
d0829cebbecc8fc799b81a9da3a90eefa412b177 is the first bad commit  
commit d0829cebbecc8fc799b81a9da3a90eefa412b177  
Author: Brian L. Gorman <  
Date:   Wed Apr 5 14:26:08 2023 -0500  
  
working on container storage code
```



Review the log with a search

```
git log -S
```

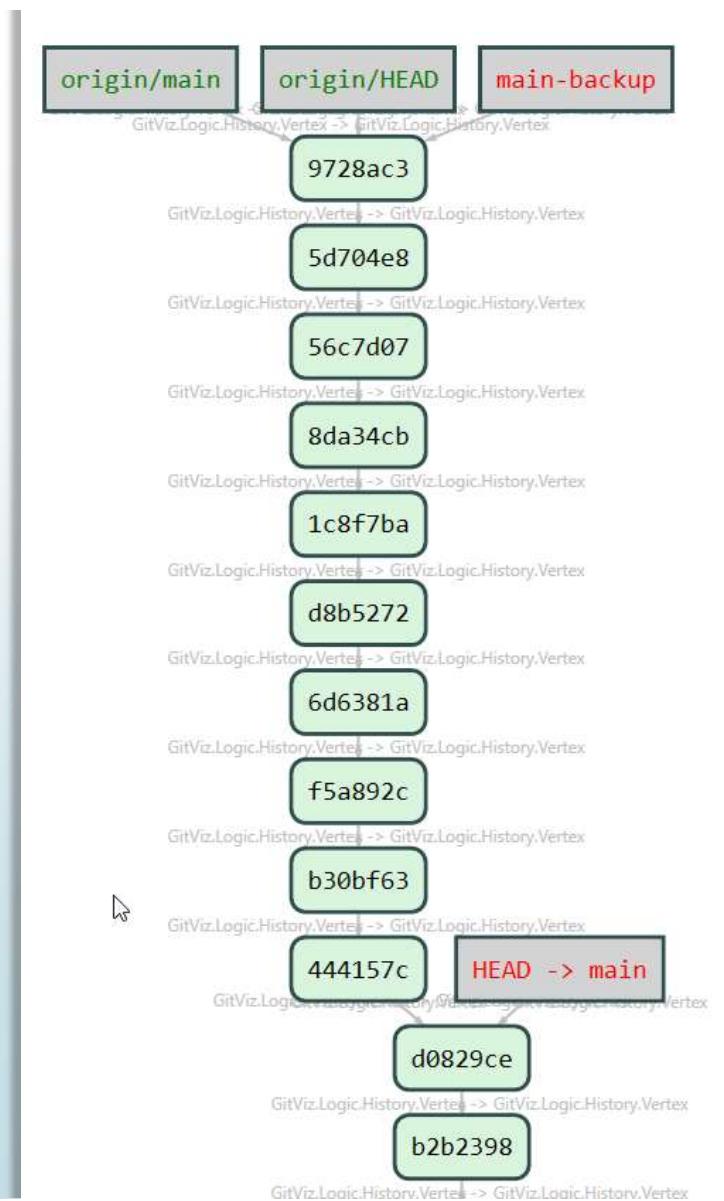
```
"AccountKey=d2505129a437fb435fd83ec%7a9f762a3b51=="
```

```
$ git log -S "AccountKey=d2505129a437fb435fd83ec%7a9f762a3b51=="  
commit d0829cebbecc8fc799b81a9da3a90eefa412b177  
Author: Brian L. Gorman <[REDACTED]>  
Date:   Wed Apr 5 14:26:08 2023 -0500  
  
        working on container storage code
```



Create a backup and
reset local main to
the first bad commit

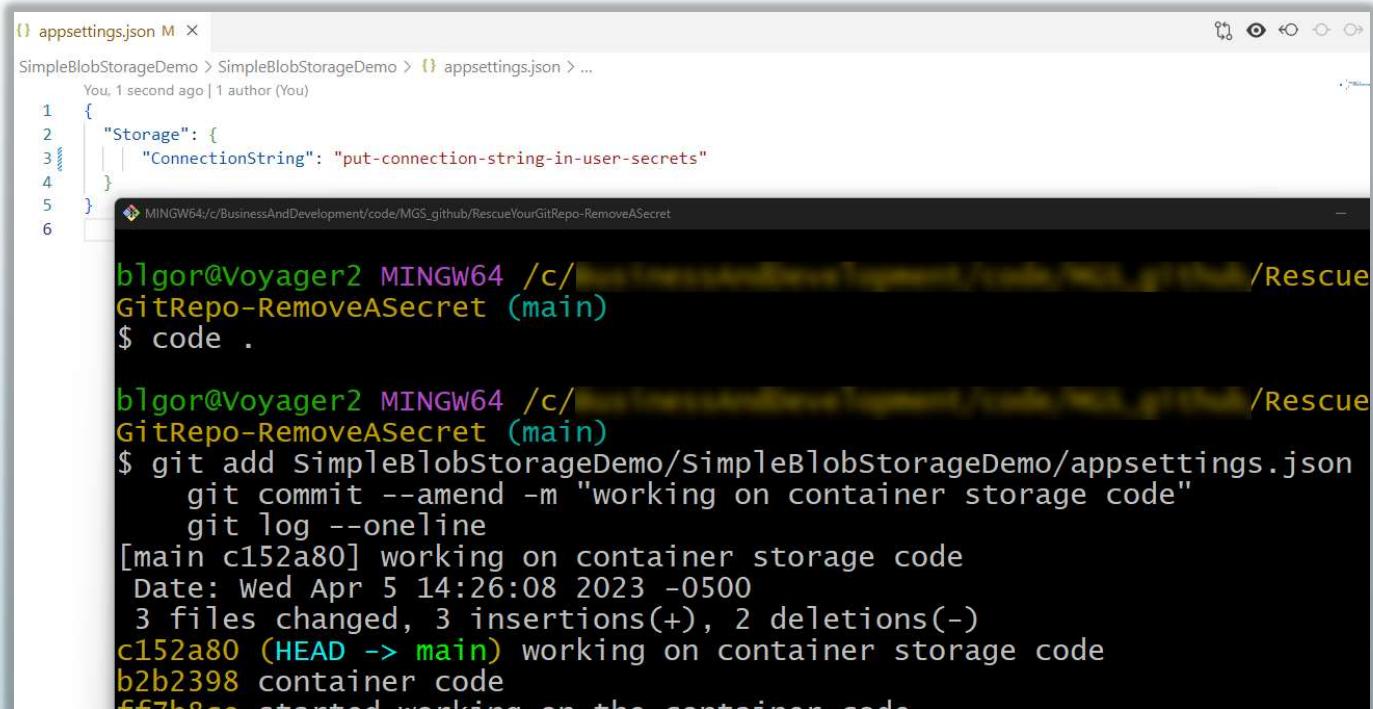
```
git switch -c main-backup  
git switch main  
git reset --hard d0829ce  
git log --oneline
```



Fix the bad file and amend the commit

code .

fix the json



The screenshot shows a terminal window with a code editor overlay. The code editor displays a JSON file named 'appsettings.json' with the following content:

```
1 {
2     "Storage": {
3         | "ConnectionString": "put-connection-string-in-user-secrets"
4     }
5 }
```

The terminal window shows the following command history:

```
b1gor@Voyager2 MINGW64 /c/ [REDACTED] /Rescue  
GitRepo-RemoveASecret (main)  
$ code .  
  
b1gor@Voyager2 MINGW64 /c/ [REDACTED] /Rescue  
GitRepo-RemoveASecret (main)  
$ git add SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json  
      git commit --amend -m "working on container storage code"  
      git log --oneline  
[main c152a80] working on container storage code  
  Date: Wed Apr 5 14:26:08 2023 -0500  
  3 files changed, 3 insertions(+), 2 deletions(-)  
c152a80 (HEAD -> main) working on container storage code  
b2b2398 container code  
ff7b9cc started working on the container code
```

```
git add SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json  
git commit --amend -m "working on container storage code"  
git log --oneline
```

Cherry-pick the rest of the commits

```
$ git cherry-pick d0829ce..9728ac3  
[main 8f2b62c] working on blob storage code  
Date: Wed Apr 5 14:26:30 2023 -0500  
1 file changed, 3 insertions(+), 3 deletions(-)
```

```
Auto-merging SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json  
CONFLICT (content): Merge conflict in SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json  
error: could not apply 9728ac3... uncommited connection string  
hint: After resolving the conflicts, mark them with  
hint: "git add/rm <pathspec>", then run  
hint: "git cherry-pick --continue".  
hint: You can instead skip this commit with "git cherry-pick --skip".  
hint: To abort and get back to the state before "git cherry-pick",  
hint: run "git cherry-pick --abort".
```

```
b1gor@Voyager2 MINGW64 /c/  
GitRepo-RemoveASecret (main|CHERRY-PICKING) I  
$ |
```

Resolve the conflict and complete the cherry-pick operation

```
$ git mergetool
Merging:
SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json

Normal merge conflict for 'SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json':
{local}: modified file
{remote}: modified file
```

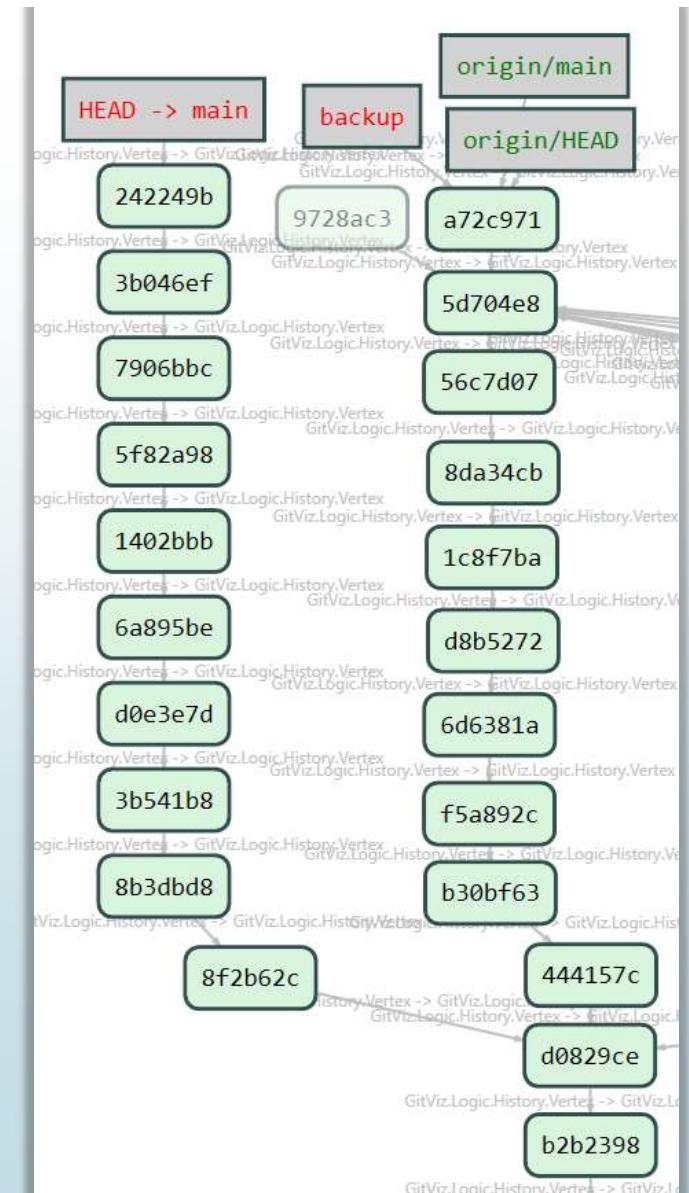
```
b1gor@Voyager2 MINGW64 /c/ /RescueYou
GitRepo-RemoveASecret (main|CHERRY-PICKING)
$ git add SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json
```

```
$ git cherry-pick --continue
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

git commit --allow-empty
```

Force your changes onto remote main

git push --force-with-lease



The screenshot shows a Visual Studio interface with the following components:

- EXPLORER**: Shows the project structure with `SimpleBlobStorageDemo` selected. Inside, there are `bin`, `images`, and `obj` folders, along with files `Program.cs` (7 problems), `appsettings.json`, `.gitignore`, `LICENSE`, and `README.md`.
- appsettings.json**: The current file in the editor, containing configuration for storage.
- TERMINAL**: Shows a command-line session:

```
b1gor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo-RemoveASecret (main)
$
```
- Git Graph**: A separate window titled "GitViz (Beta) - RescueYourGitRepo-RemoveASecret" showing the local repository path `C:\BusinessAndDevelopment\code\mandalorianboba_github\RescueYourGitRepo-RemoveASecret`. It displays a history of commits from `HEAD -> main` down to `543491e`.

```
SimpleBlobStorageDemo > SimpleBlobStorageDemo > appsettings.json
Brian L. Gorman, 2 days ago | 1 author (Brian L. Gorman)
1 {
2   "Storage": {
3     "ConnectionString": "DefaultEndpointsProtocol=https;AccountName=...;AccountKey=...;ContainerName=..."
4   }
5 }
```

```
b1gor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo-RemoveASecret (main)
$
```

Commit Hash	Author	Date	Message
HEAD -> main			
9159268			
5d704e8			
56c7d07			
8da34cb			
1c8f7ba			
d8b5272			
6d6381a			
f5a892c			
b30bf63			
444157c			
d0829ce			
b2b2398			
ff7b8ce			
ff997b4			
543491e			



Summary

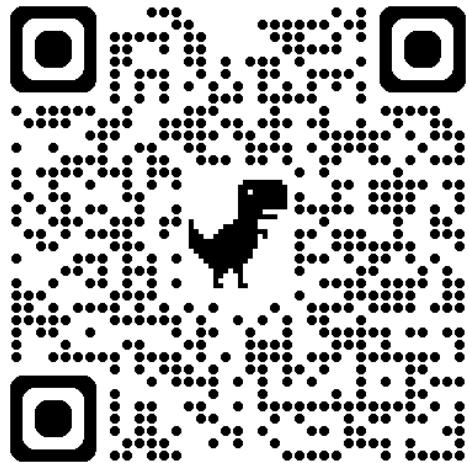
- ▶ Consider a linear history for best results
- ▶ Consider squash/interactive rebase for single commit features
- ▶ Nothing is ever ruined (it's very difficult to mess up a GIT repo)
- ▶ Remember that you have a local cache of commits
- ▶ Use caution and rework your history with confidence
- ▶ When in doubt, create a JIC branch
- ▶ Get good with GIT commands (you'll be the hero of your team)



Questions?

- ▶ Email:
 - ▶ brian@majorguidancesolutions.com
 - ▶ blgorman@gmail.com
- ▶ X
 - ▶ @blgorman
- ▶ Linked In
 - ▶ <https://www.linkedin/in/brianlgorman>
- ▶ Code:
 - ▶ <https://github.com/blgorman/RescueYourGitRepo>
- ▶ Join an MS Learn Room:
 - ▶ <https://techcommunity.microsoft.com/t5/custom/page/page-id/learn>
- ▶ Bluesky
 - ▶ <https://bsky.app/profile/blgorman.bsky.social>

Conclusion



Practical Entity Framework Core 6

Developing Solutions for Microsoft Azure Certification Companion

This slide shows two Amazon product listings side-by-side. On the left is 'Practical Entity Framework Core 6' by Brian L. Gorman, Second Edition, published by Apress. On the right is 'Developing Solutions for Microsoft Azure Certification Companion' by Brian L. Gorman, part of the 'CERTIFICATION STUDY COMPANION SERIES'. Both books are priced at \$47.99 and have 5 ratings. The 'Developing Solutions' book includes a 'Look inside' button and a green shield icon indicating it's a certified product.

@blgorman

<https://www.linkedin.com/in/brianlgorman>

Thanks for having me today!

Brian Gorman

3 monthly listeners

FOLLOW

Popular

- 1 Wake Me Up
- 2 Abba Father
- 3 Fully Alive

A screenshot of Brian Gorman's Spotify profile. It features a large black and white portrait of him on the right. His name, 'Brian Gorman', is prominently displayed in white text. Below his name, it says '3 monthly listeners'. There is a 'FOLLOW' button with a green play icon. Under the 'Popular' section, three songs are listed with their covers and titles: 'Wake Me Up', 'Abba Father', and 'Fully Alive'.

Azure Fundamentals and Developer Technologies - blgorman

This room will focus on Azure development and DevOps as well as the AZ-900, DP-900, and SC-900 fundamentals exams.

A screenshot of Brian Gorman's LinkedIn profile. It features a circular profile picture of him smiling. To the right of the picture, the text 'Azure Fundamentals and Developer Technologies - blgorman' is displayed. Below that, a description reads: 'This room will focus on Azure development and DevOps as well as the AZ-900, DP-900, and SC-900 fundamentals exams.'