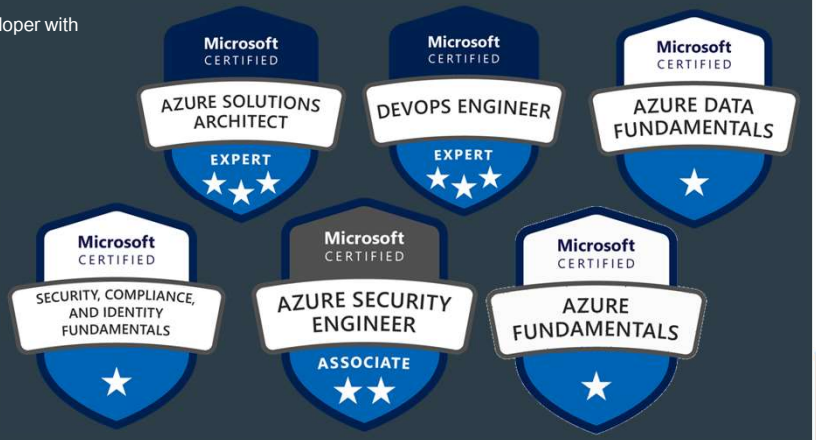
Rescuing your GIT Repository

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.

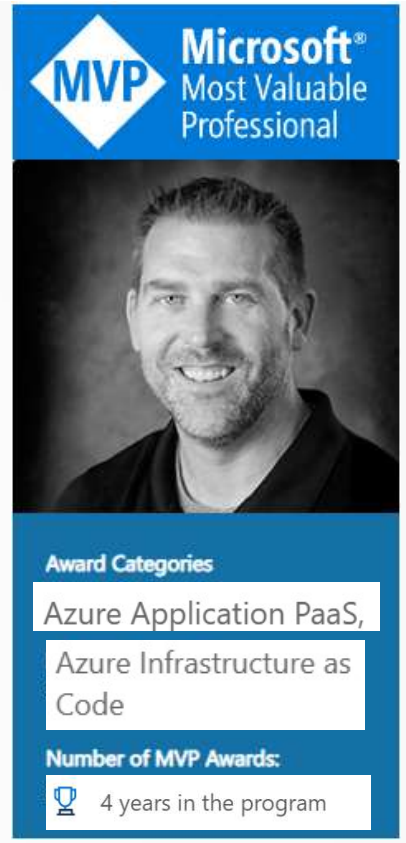
Brian has also created many online video courses, available on various platforms. Recently, Brian published the second edition of his first book: Practical Entity Framework Core 6 with APress.

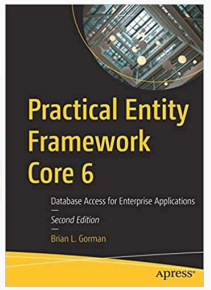
Brian runs a company named Major Guidance Solutions, which does contract and custom training and curriculum development. If you are interested in becoming Azure certified, make sure to connect with me.

Connect with Brian on Twitter and/or Linked In

@blgorman

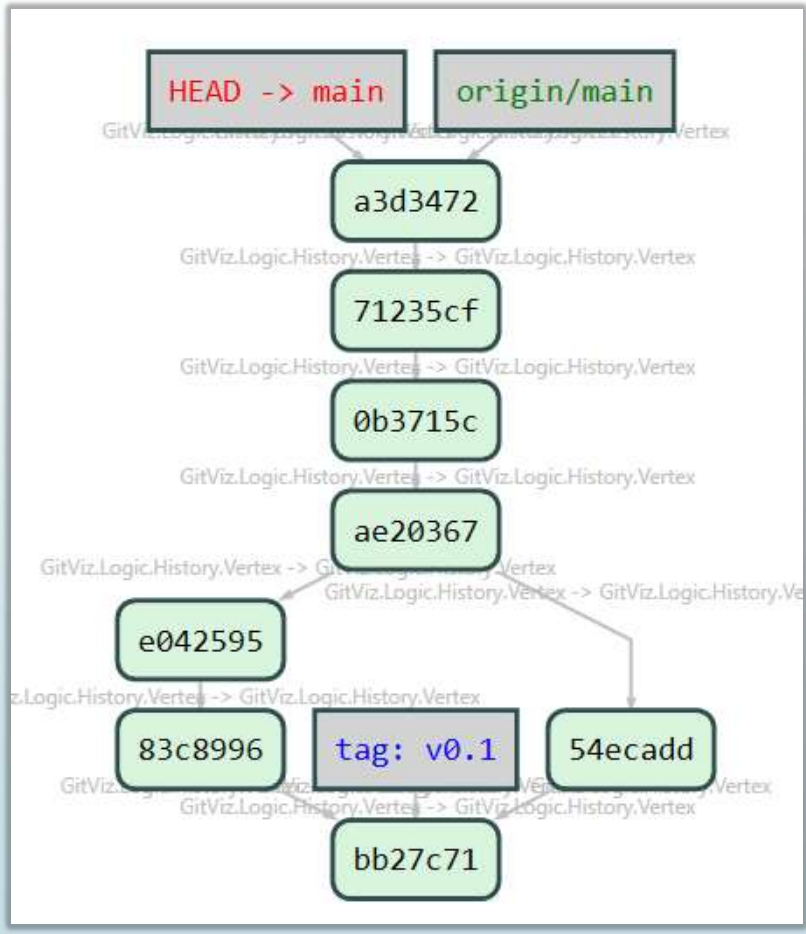
https://www.linkedin/in/brianlgorman

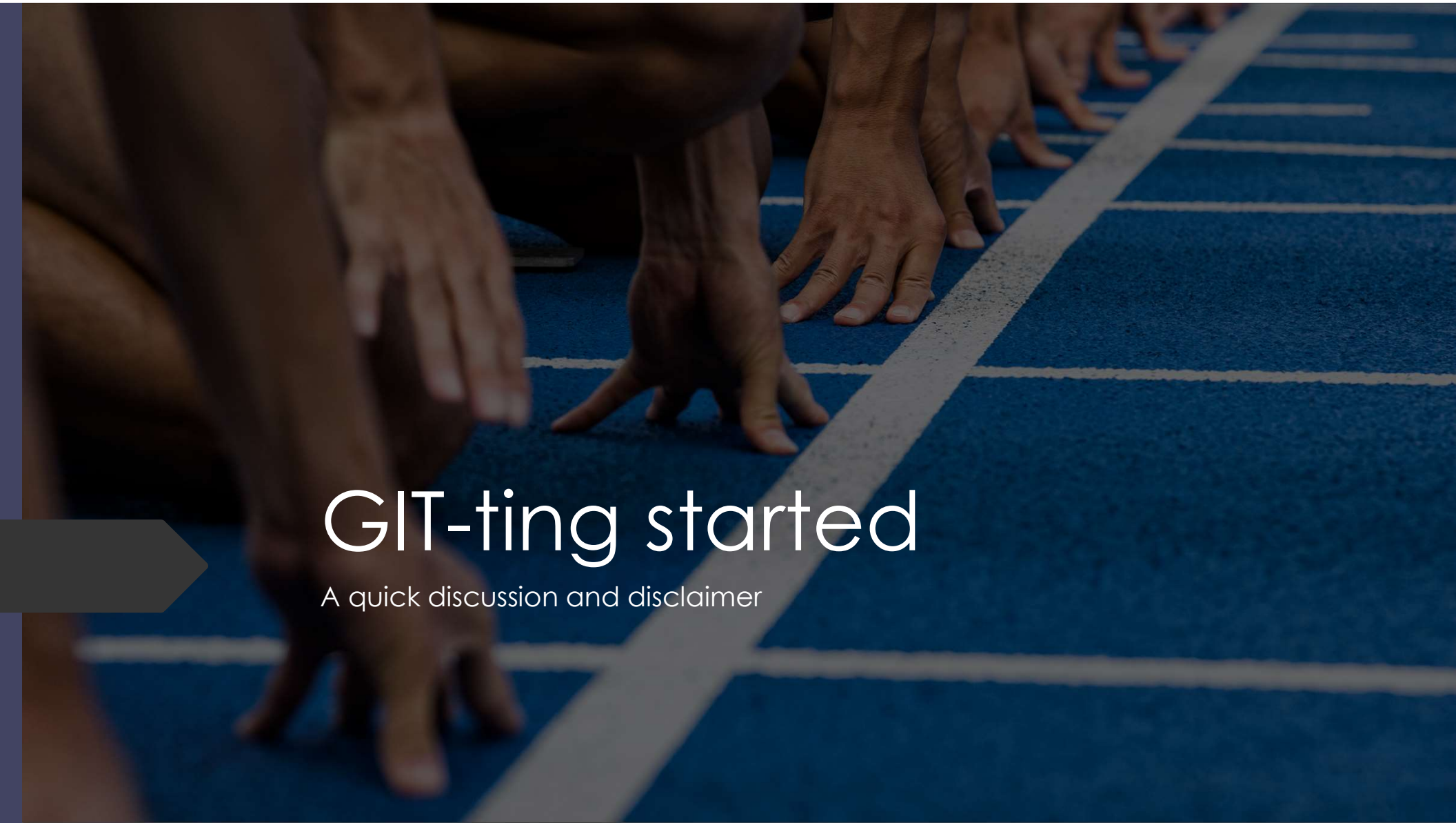




Agenda

- 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





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
```

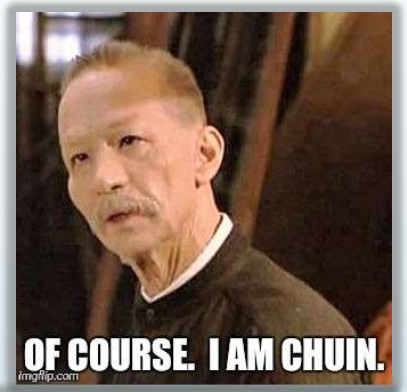
Thank the sponsors and the team

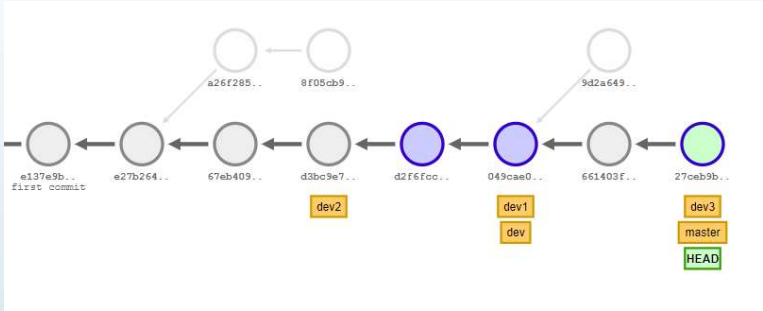
businessgrowthmentor

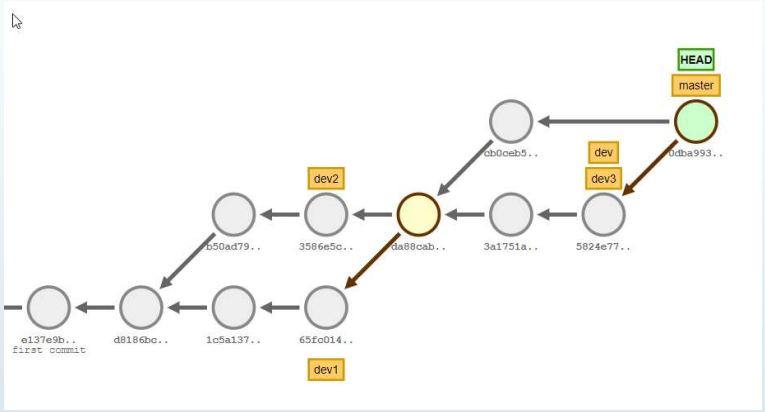
1) There's someone out there who's way less qualified than you - living the life you want.

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







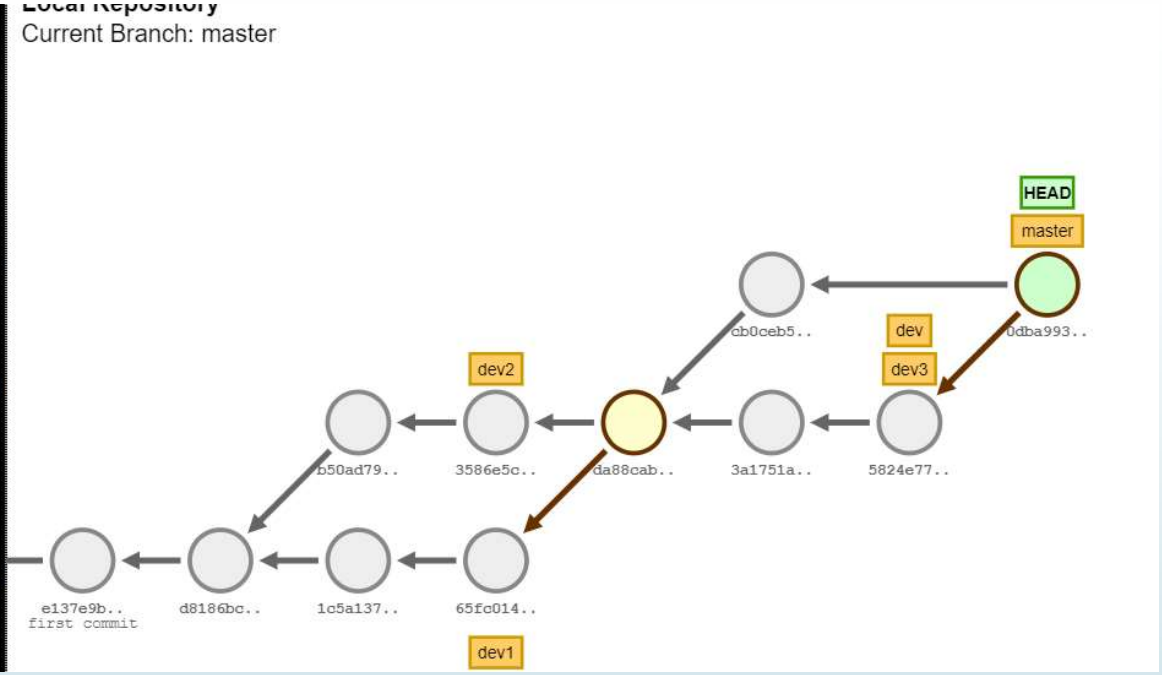
Linear and Non-Linear

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
```



Demo: A traditional merge strategy

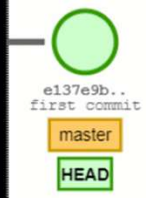
Multiple "branch" commits are merged with fast-forward strategy, and this leads to a non-linear history

Have fun.

$ enter git command

Local Repository

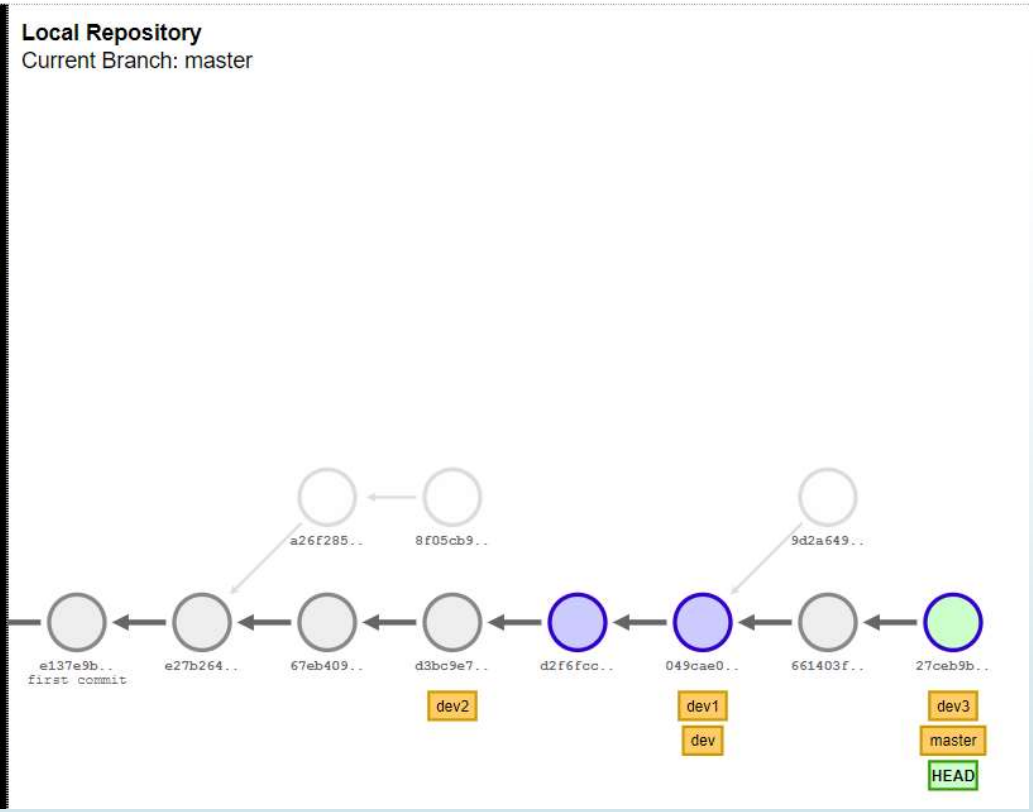
Current Branch: master



Linear Commit History

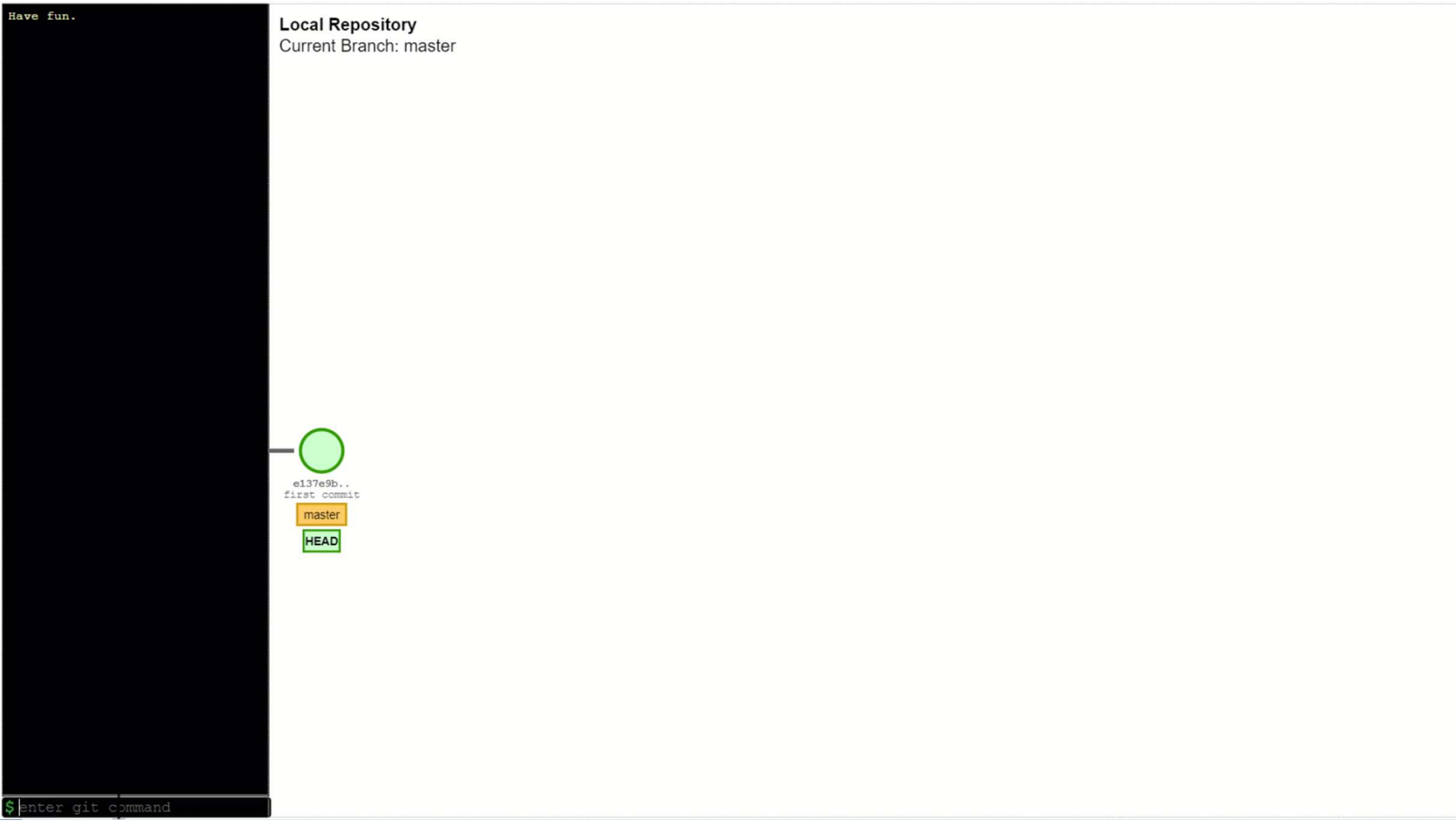
- Any branch strategy
- Linear commit history
- Much better to perform advanced commands such as pick and reset
- Managed with pull requests
- Can be combined with squashing for a very-succinct commit history

```
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.
```



Demo: Linear Commit History

Can squash commits to make it even more concise

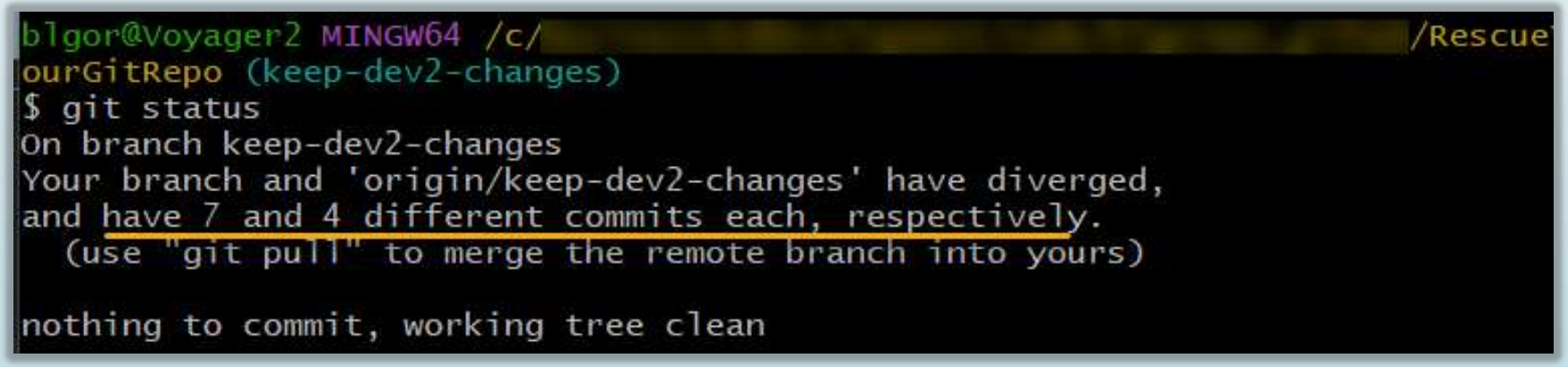

Have fun.
Local Repository
Current Branch: master
e137e9b..
first commit
master
HEAD
$ enter git command

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.

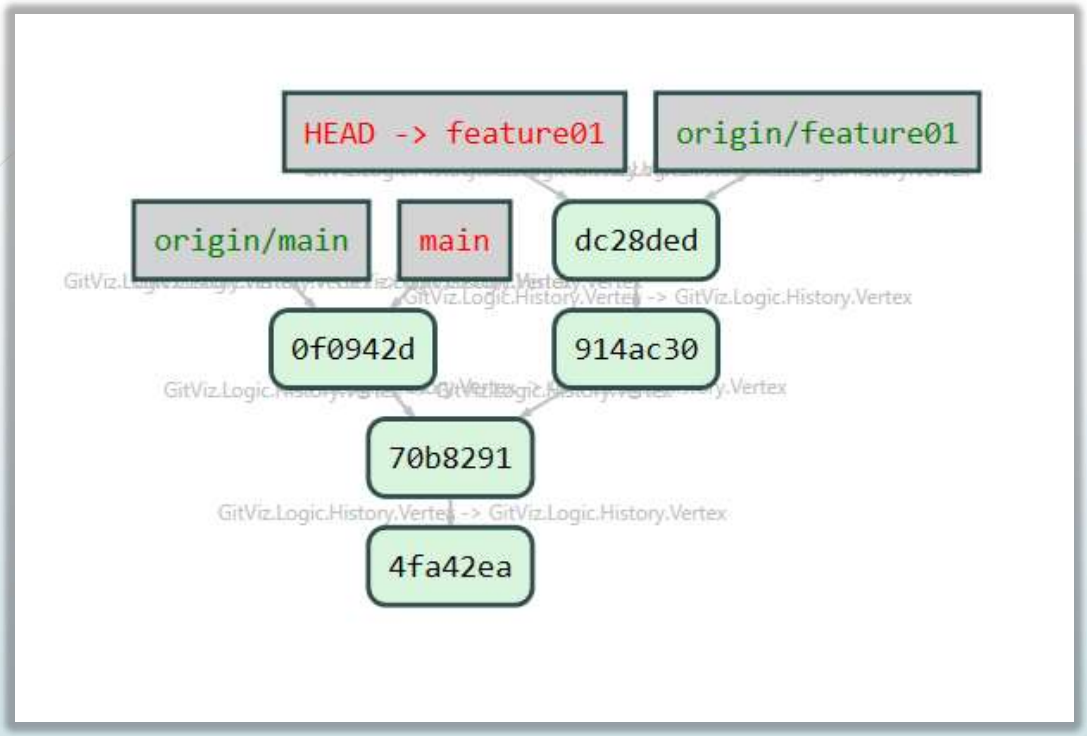


Rebase and Force push

- 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)

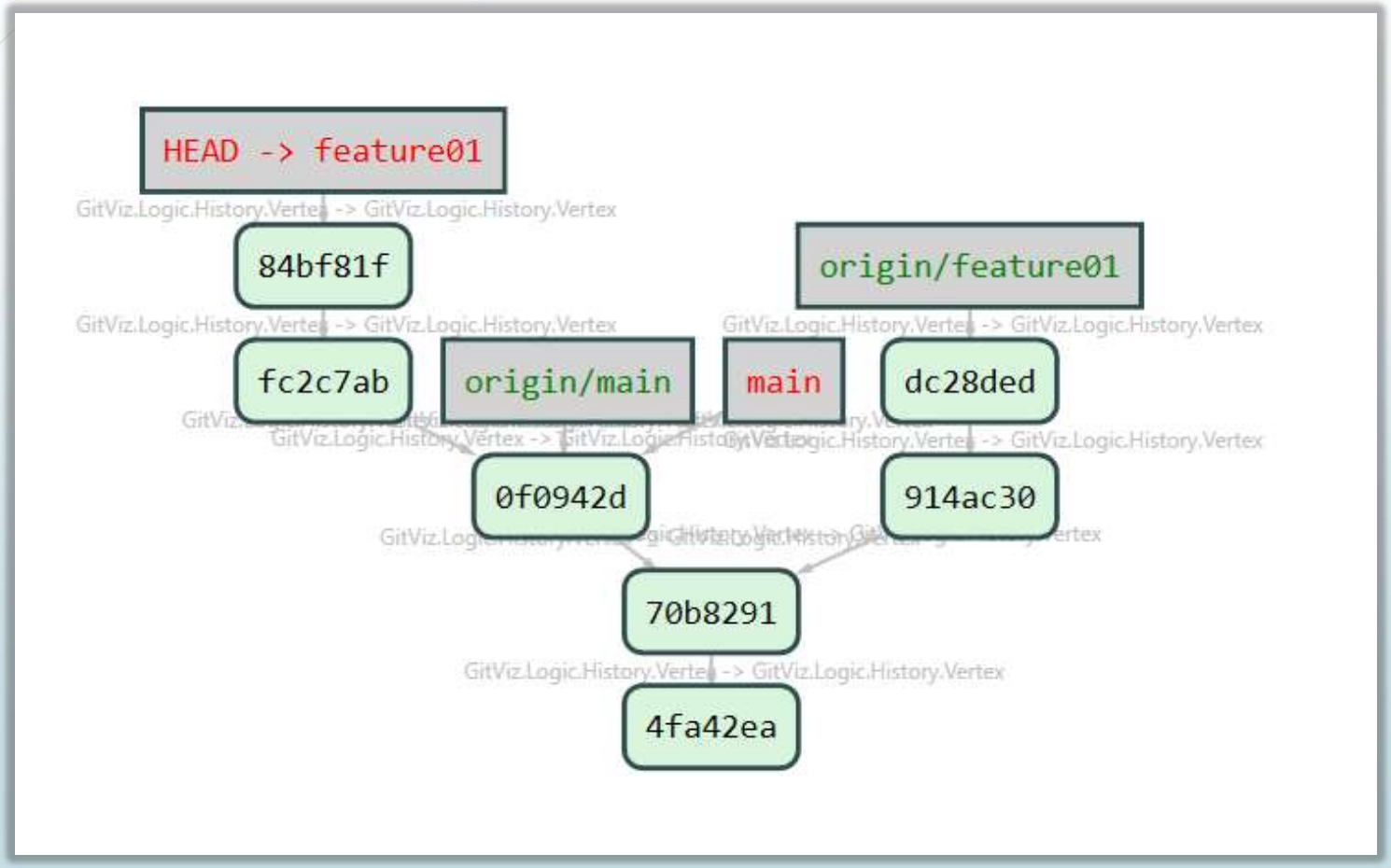
Your local history is correct, but remote won't let you push. Use the force, Luke!



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



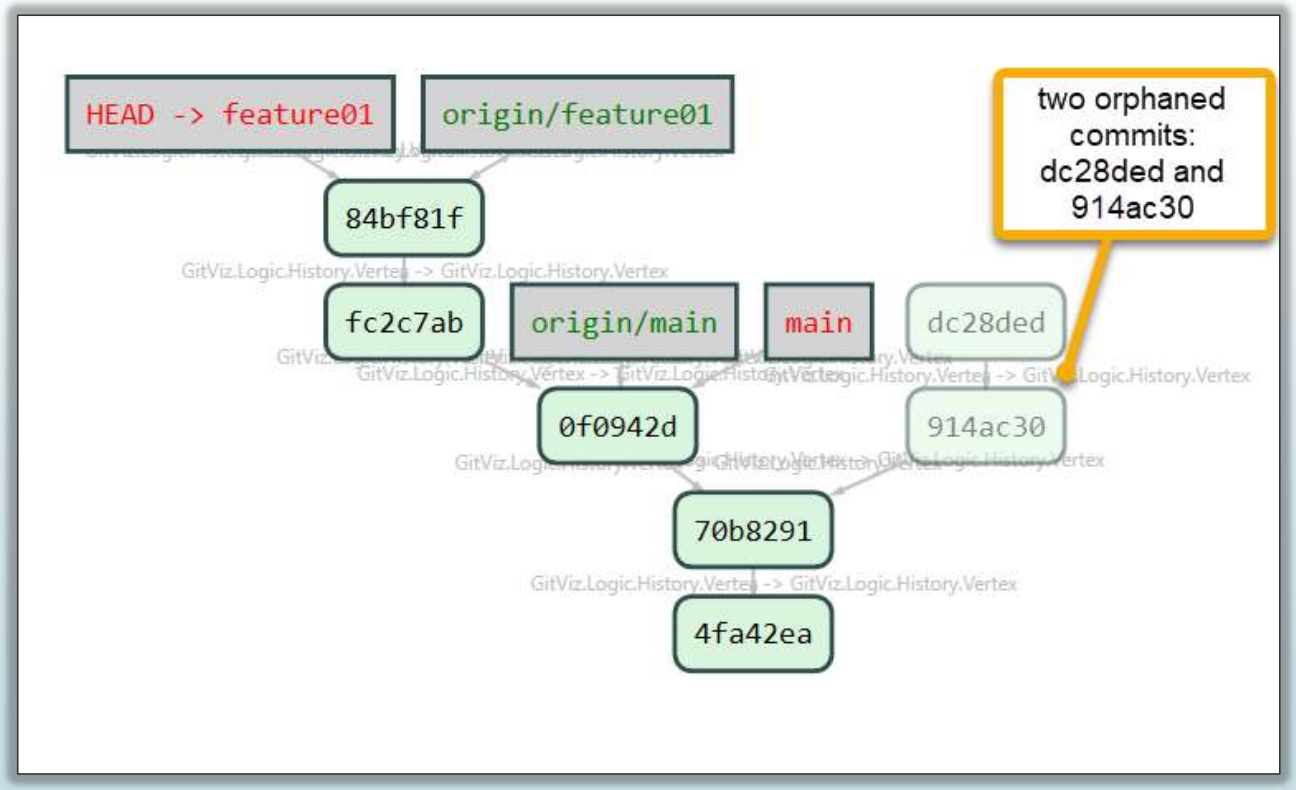
git status/git push/git push --force-with-lease

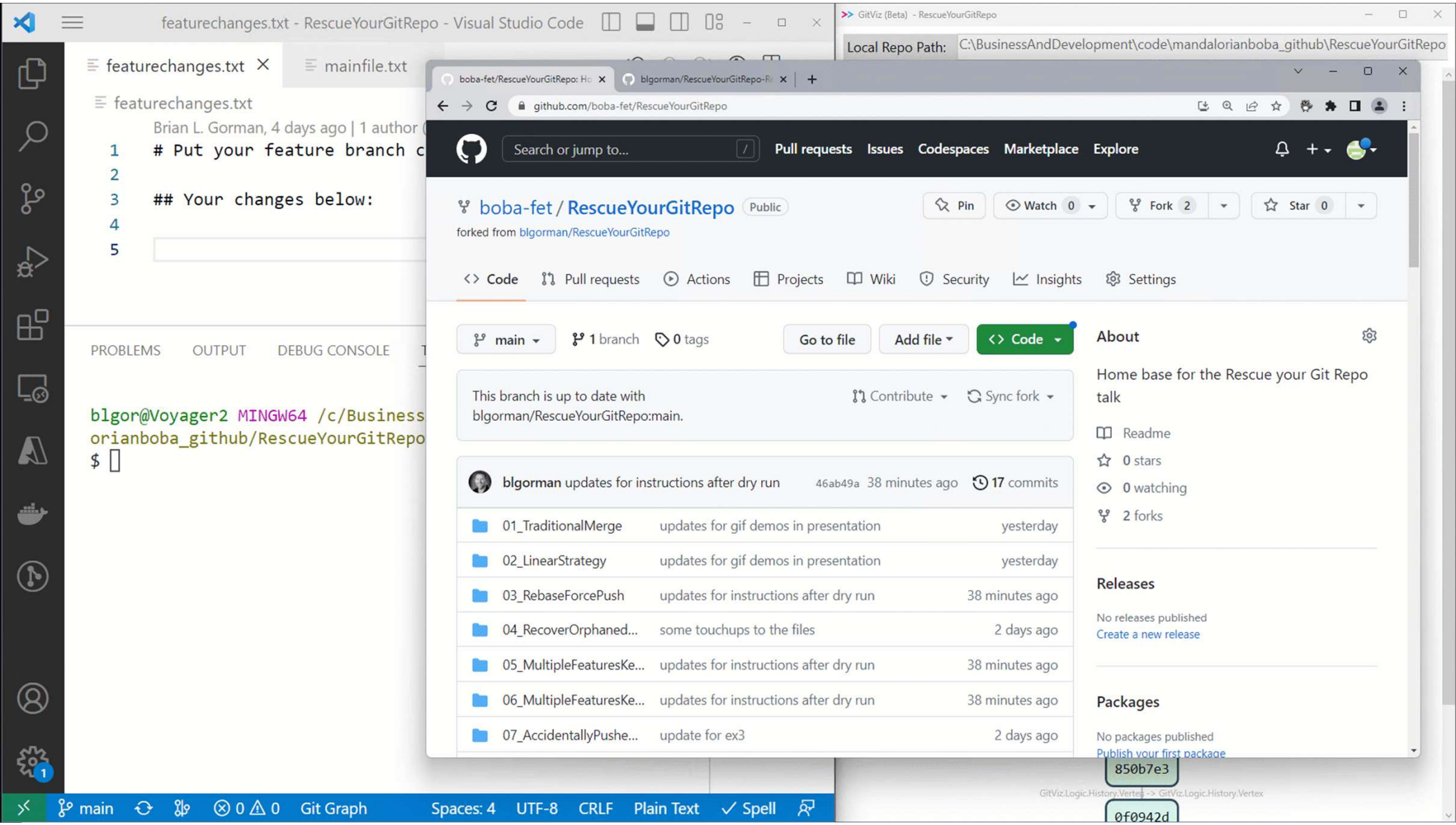
```
blgor@Voyager2 MINGW64 /c/                                    'RescueYourGitRepo (feature01)
$ git status
On branch feature01
Your branch and 'origin/feature01' have diverged,
and have 3 and 2 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
```

```
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)
```

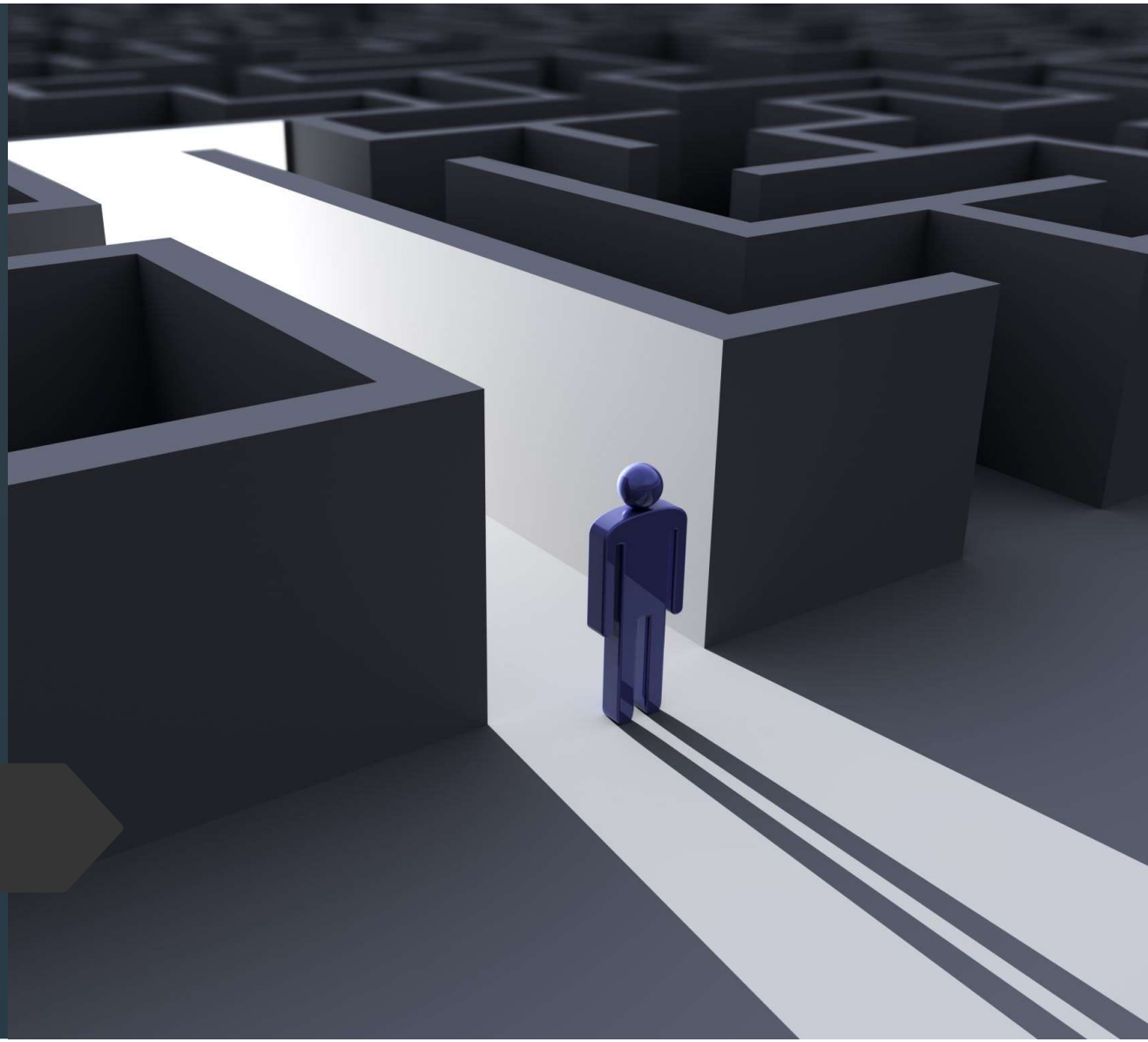
git push --force-with-lease




featurechanges.txt - RescueYourGitRepo - Visual Studio Code
featurechanges.txt
mainfile.txt
featurechanges.txt
Brian L. Gorman, 4 days ago | 1 author
1 # Put your feature branch c
2
3 ## Your changes below:
4
5
PROBLEMS
OUTPUT
DEBUG CONSOLE
blgor@Voyager2 MINGW64 /c/Business
orianboba_github/RescueYourGitRepo
$
main
0 0
Git Graph
Spaces: 4
UTF-8
CRLF
Plain Text
Spell
GitViz (Beta) - RescueYourGitRepo
Local Repo Path:
C:\BusinessAndDevelopment\code\mandalorianboba_github\RescueYourGitRepo
boba-fet/RescueYourGitRepo: Ho
blgorman/RescueYourGitRepo-R
github.com/boba-fet/RescueYourGitRepo
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
boba-fet / RescueYourGitRepo
Public
forked from blgorman/RescueYourGitRepo
Pin
Watch 0
Fork 2
Star 0
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
This branch is up to date with blgorman/RescueYourGitRepo:main.
Contribute
Sync fork
blgorman updates for instructions after dry run
46ab49a
38 minutes ago
17 commits
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_AccidentallyPushe... update for ex3 2 days ago
About
Home base for the Rescue your Git Repo talk
Readme
0 stars
0 watching
2 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
850b7e3
GitViz.Logic.History.Vertex -> GitViz.Logic.History.Vertex
0f0942d

Finding Lost Commits after history rewrite

When you rewrite your history but need to get it back



git reflog --all

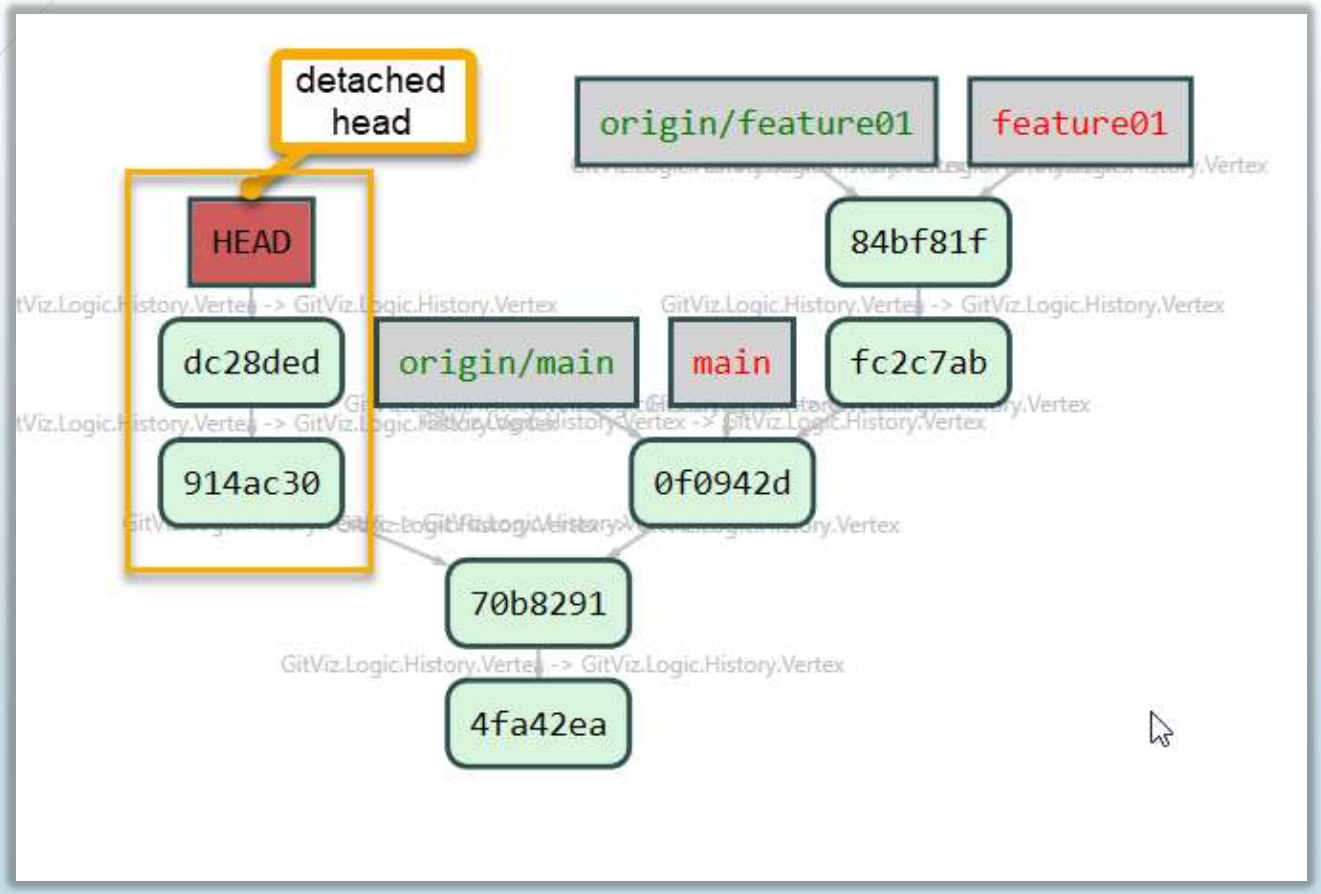
```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo (feature01)
$ git reflog --all
84bf81f (HEAD -> feature01, origin/feature01) refs/remotes/origin/feature01@{0}: update by push
84bf81f (HEAD -> feature01, origin/feature01) refs/heads/feature01@{0}: rebase (finish): 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@
70b8291 refs/heads/main@{1}: commit: updates for force push demo walkthrough
```

orphaned commits show up in reflog

Checkout your lost commits in a detached state

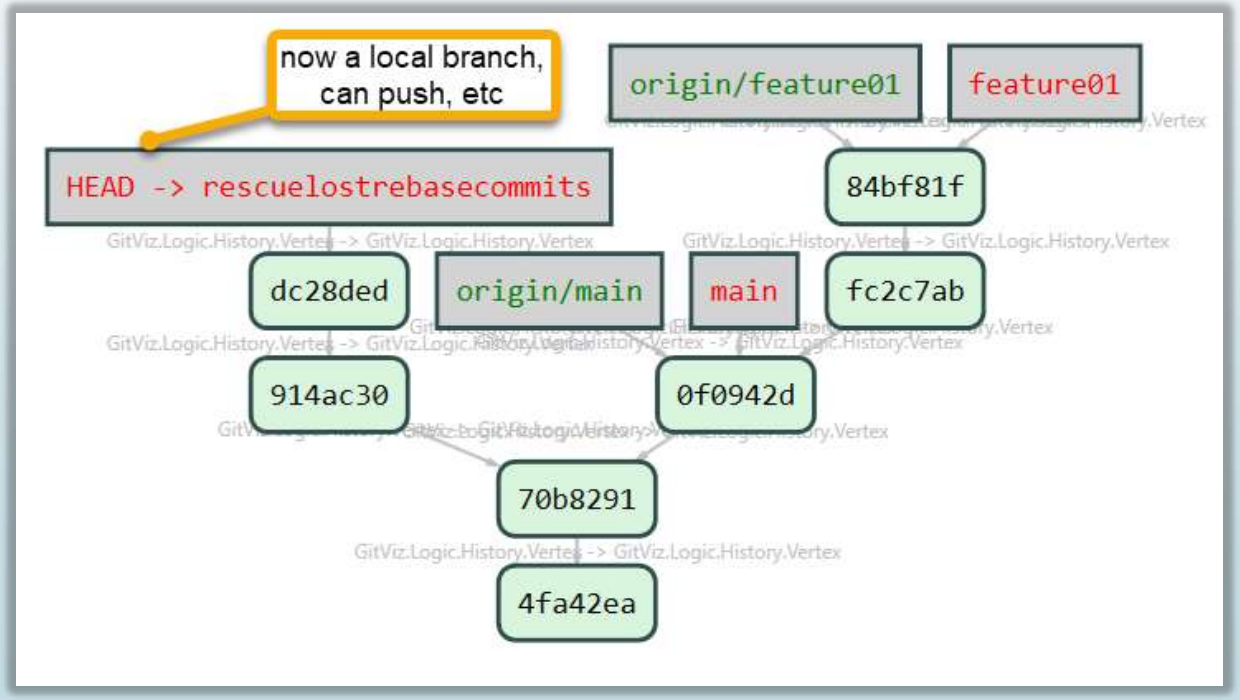
```
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/blgorman_github/RescueYourGitRepo (feature01)
$ git checkout dc28ded
Note: switching to 'dc28ded'.

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/[redacted]/RescueYourGitRepo ((dc28ded...))
$ git switch -c rescuelostrebasecommits
Switched to a new branch 'rescuelostrebasecommits'

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



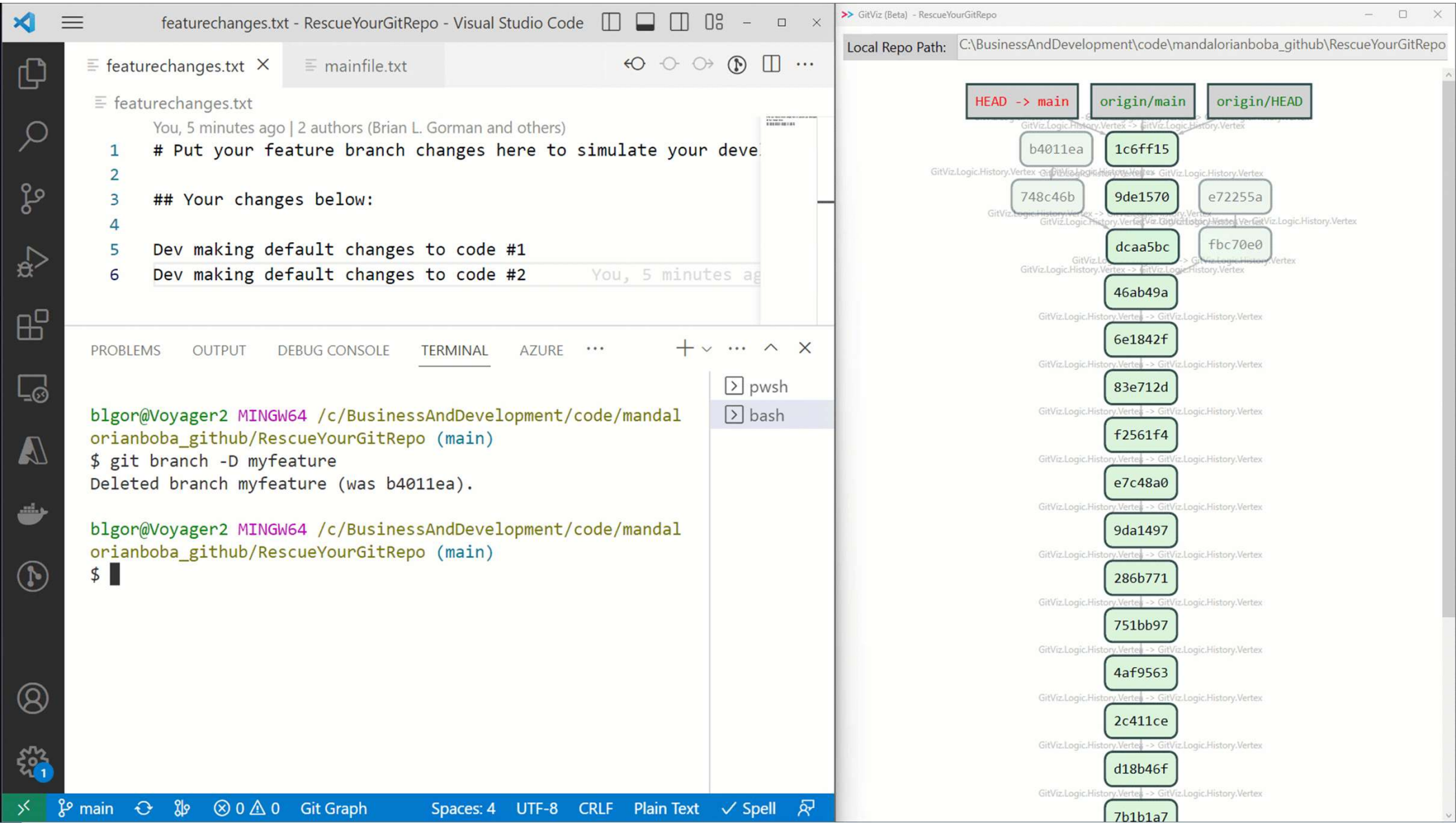
Clear your local Cache (you don't ever need to do this, and probably shouldn't ever do it)

- Set the reflog to expire all current unreachable commits

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

- Run the garbage collector

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


featurechanges.txt - RescueYourGitRepo - Visual Studio Code
featurechanges.txt
mainfile.txt
featurechanges.txt
You, 5 minutes ago | 2 authors (Brian L. Gorman and others)
1 # Put your feature branch changes here to simulate your deve
2
3 ## Your changes below:
4
5 Dev making default changes to code #1
6 Dev making default changes to code #2 You, 5 minutes ag
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
AZURE
pwsh
bash
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)
$
main
0 0
Git Graph
Spaces: 4
UTF-8
CRLF
Plain Text
Spell
GitViz (Beta) - RescueYourGitRepo
Local Repo Path:
C:\BusinessAndDevelopment\code\mandalorianboba_github\RescueYourGitRepo
HEAD -> main
origin/main
origin/HEAD
b4011ea
1c6ff15
748c46b
9de1570
e72255a
dcaa5bc
fbc70e0
46ab49a
6e1842f
83e712d
f2561f4
e7c48a0
9da1497
286b771
751bb97
4af9563
2c411ce
d18b46f
7b1b1a7

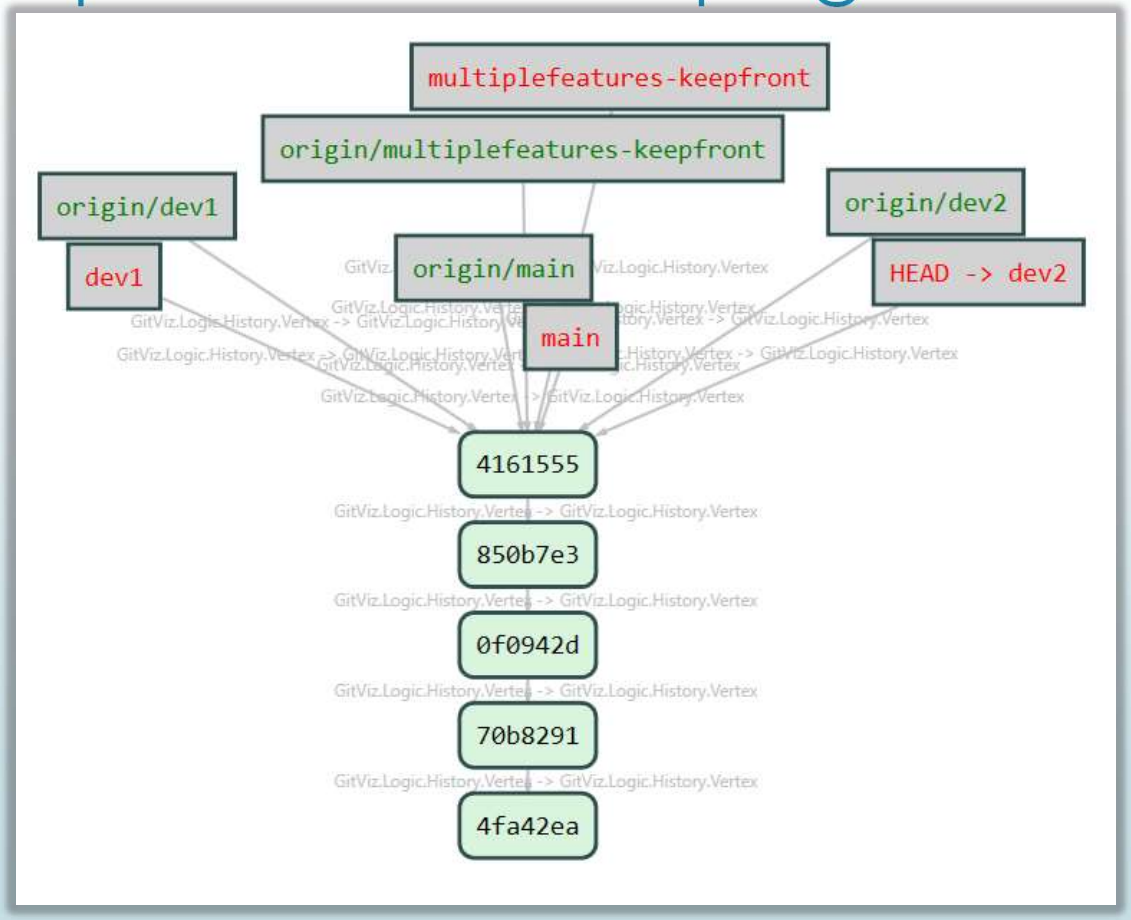
Scenario #2 – Multiple Merged Features Ready for Production

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

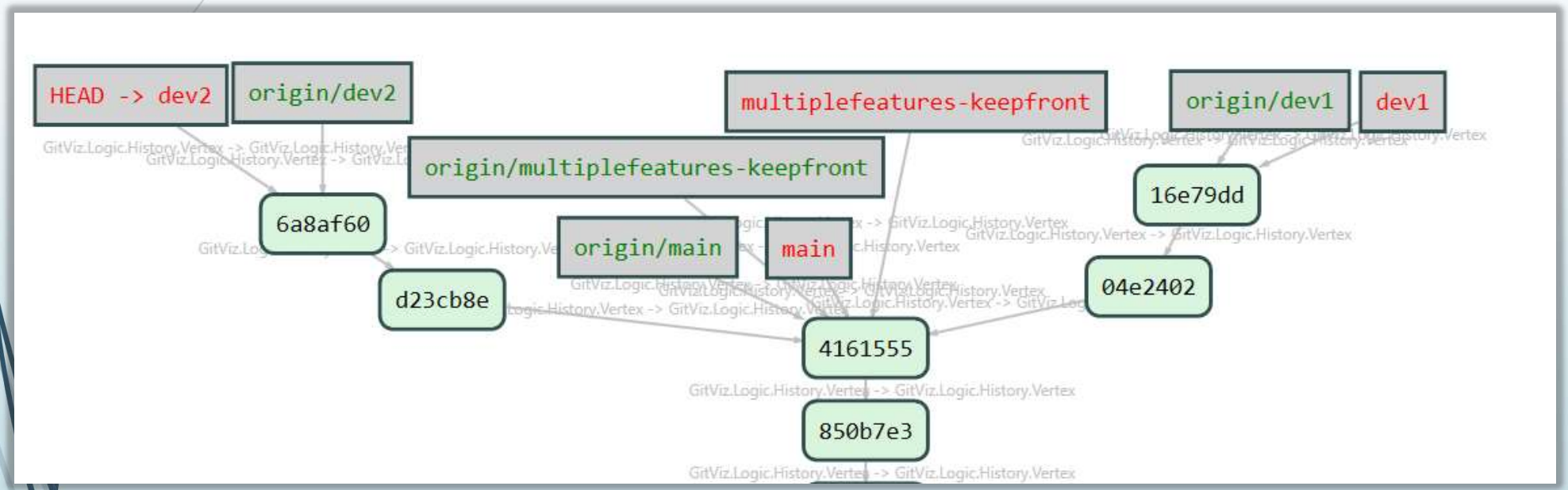
Scenario/Set up

- 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

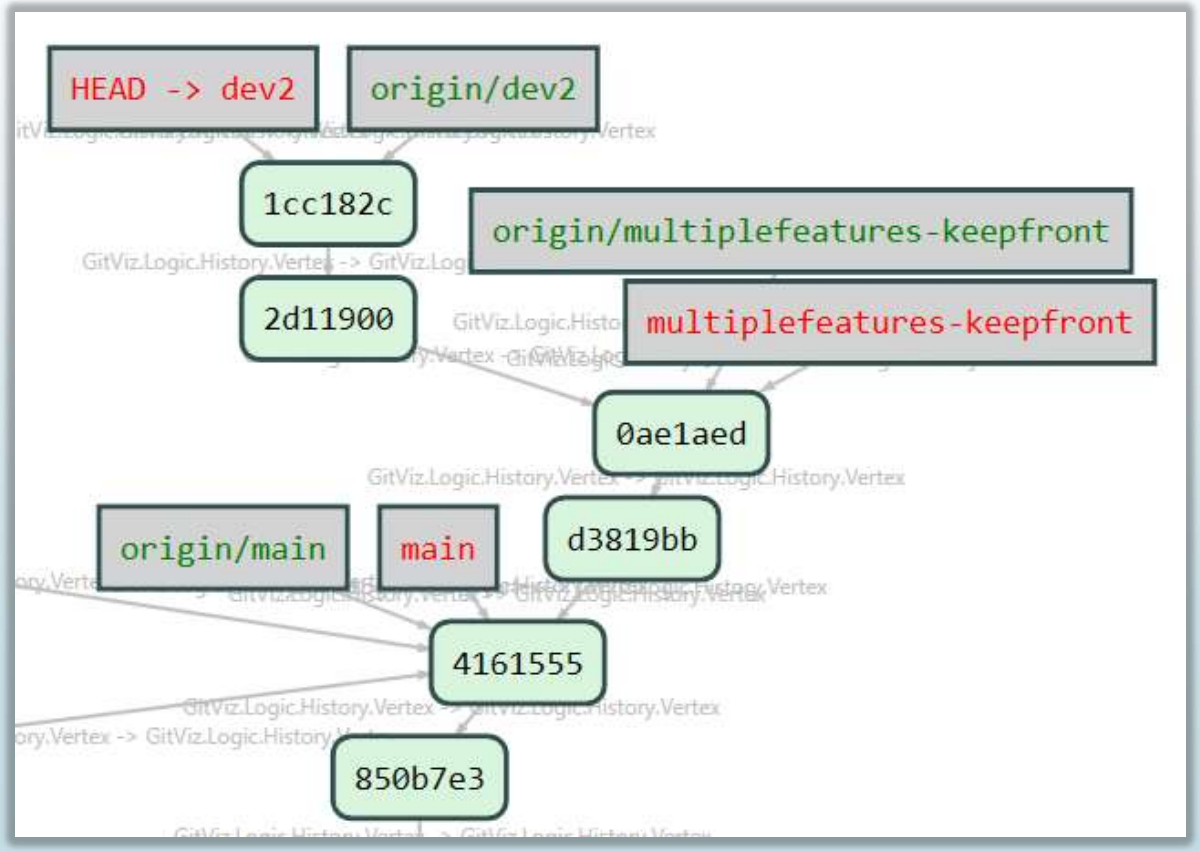
Multiple features keeping the front



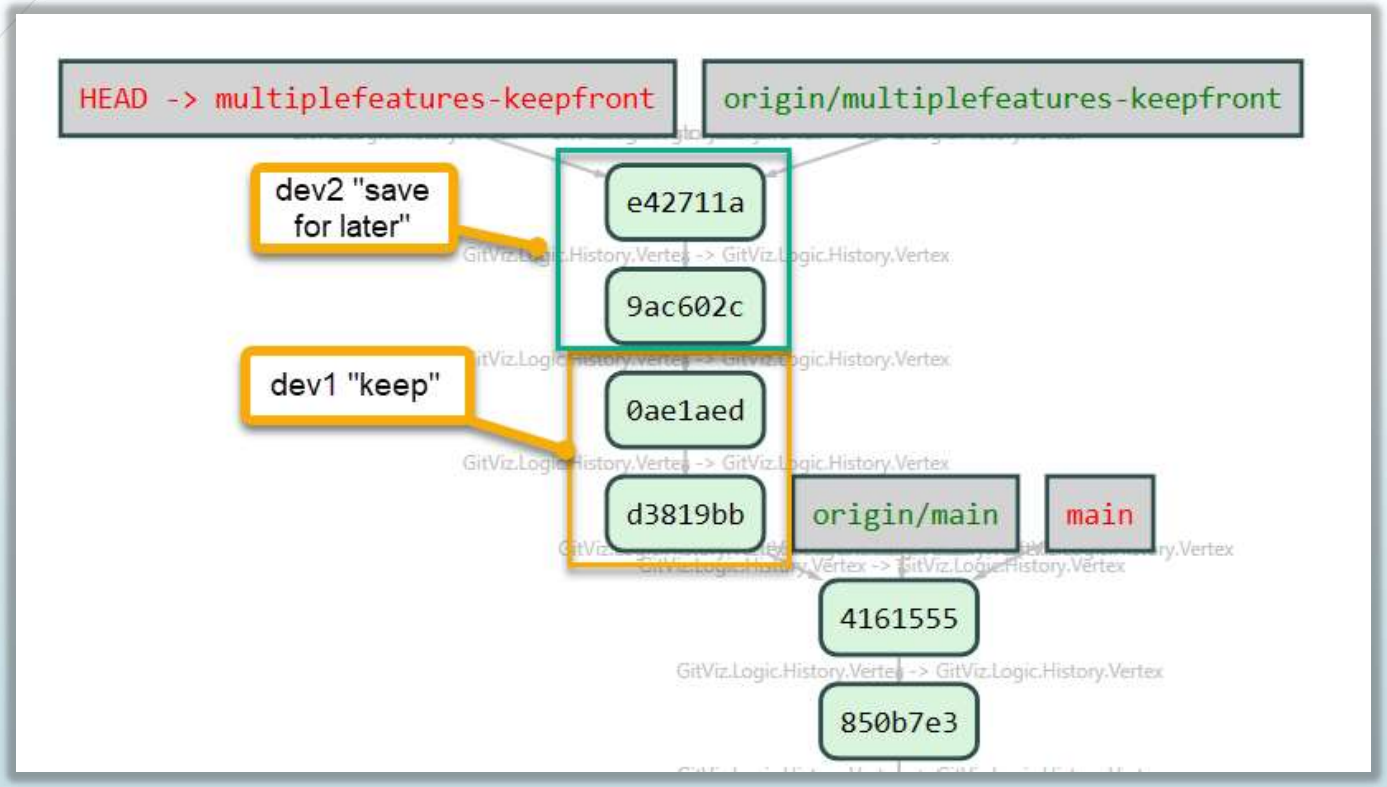
Move both dev branches forward a couple of commits

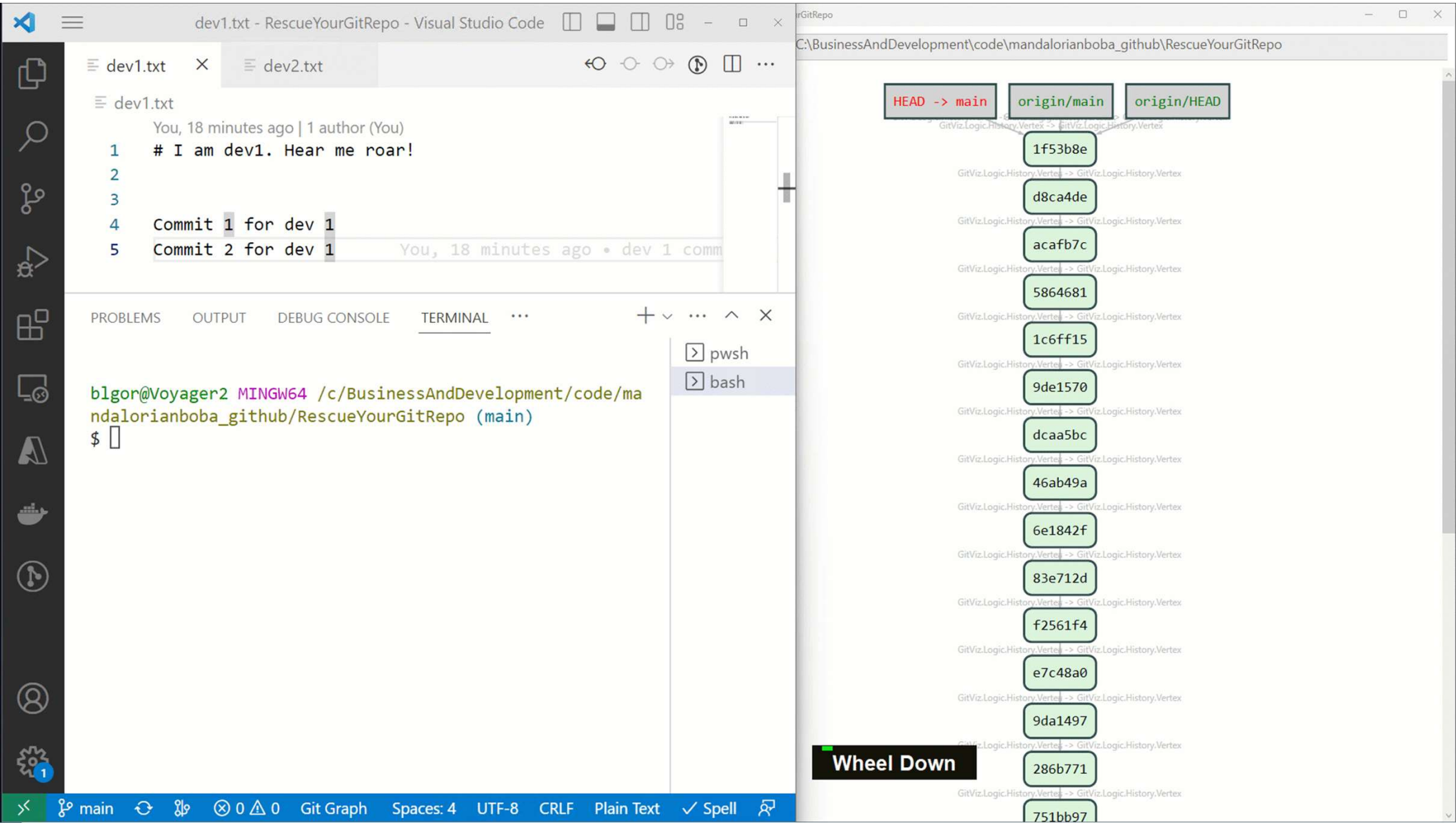


Create one linear tree with all commits using rebase



Rebase merge the dev2 branch



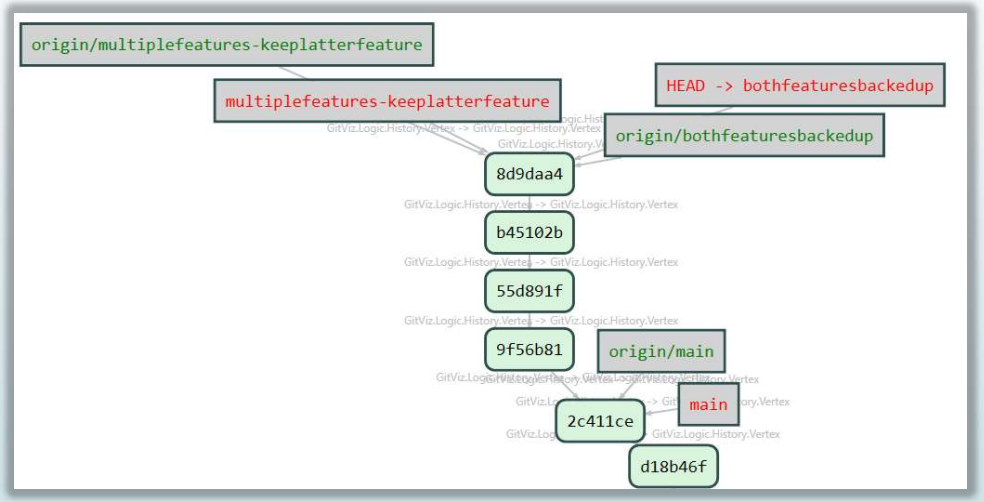

dev1.txt - RescueYourGitRepo - Visual Studio Code
dev1.txt
dev2.txt
dev1.txt
You, 18 minutes ago | 1 author (You)
1 # I am dev1. Hear me roar!
2
3
4 Commit 1 for dev 1
5 Commit 2 for dev 1 You, 18 minutes ago • dev 1 comm
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
pwsh
bash
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (main)
$
main
0 0
Git Graph
Spaces: 4
UTF-8
CRLF
Plain Text
Spell
C:\BusinessAndDevelopment\code\mandalorianboba_github\RescueYourGitRepo
HEAD -> main
origin/main
origin/HEAD
1f53b8e
d8ca4de
acafb7c
5864681
1c6ff15
9de1570
dcaa5bc
46ab49a
6e1842f
83e712d
f2561f4
e7c48a0
9da1497
286b771
751bb97
Wheel Down

Demo: Multiple Features Ready - promoting only one

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

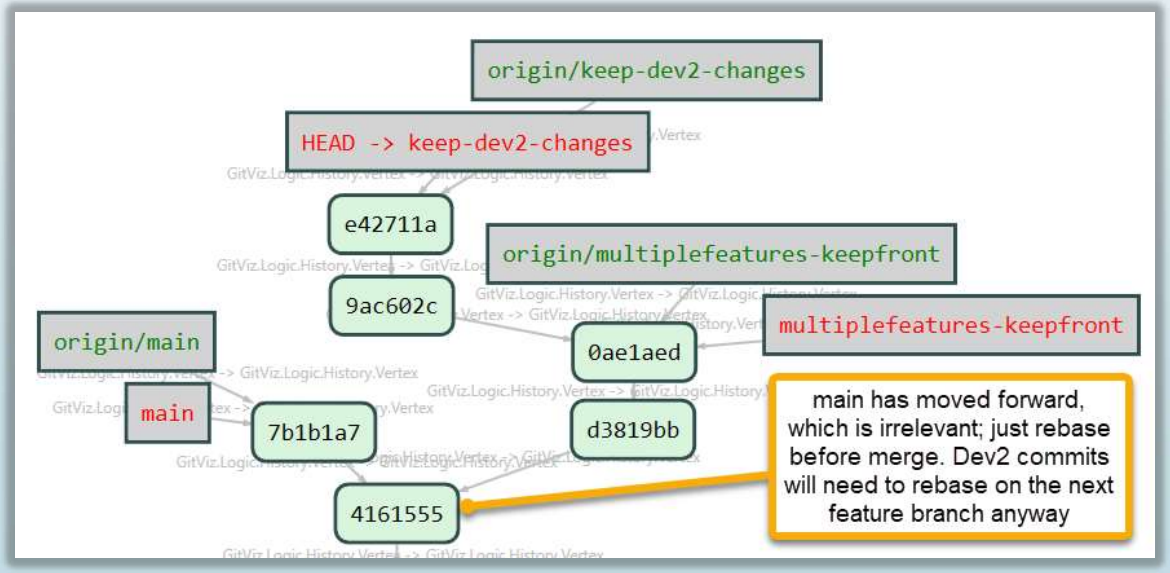
The setup is the same for either walkthrough

- Set up as before
- Create a backup branch

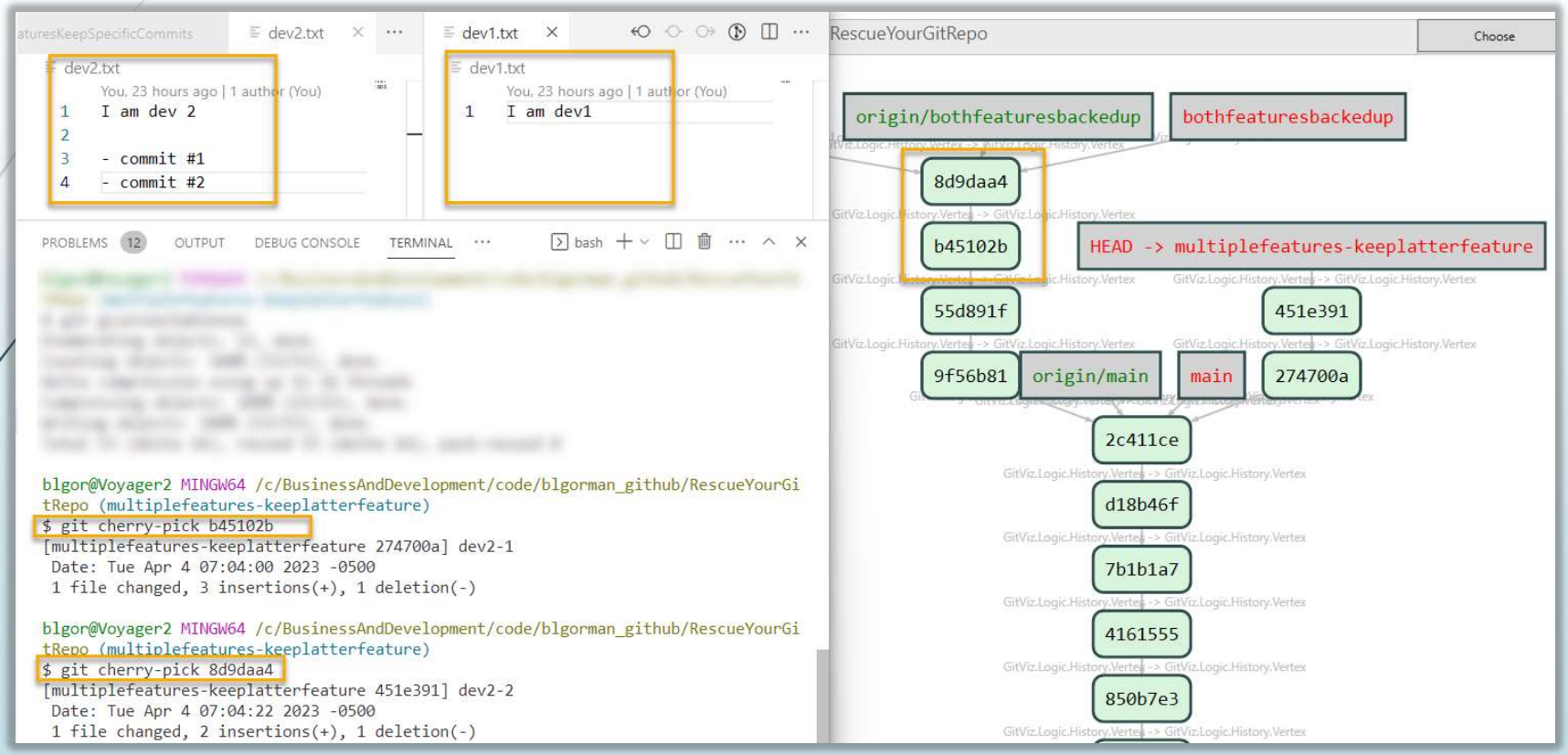


Keeping the front half

- 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

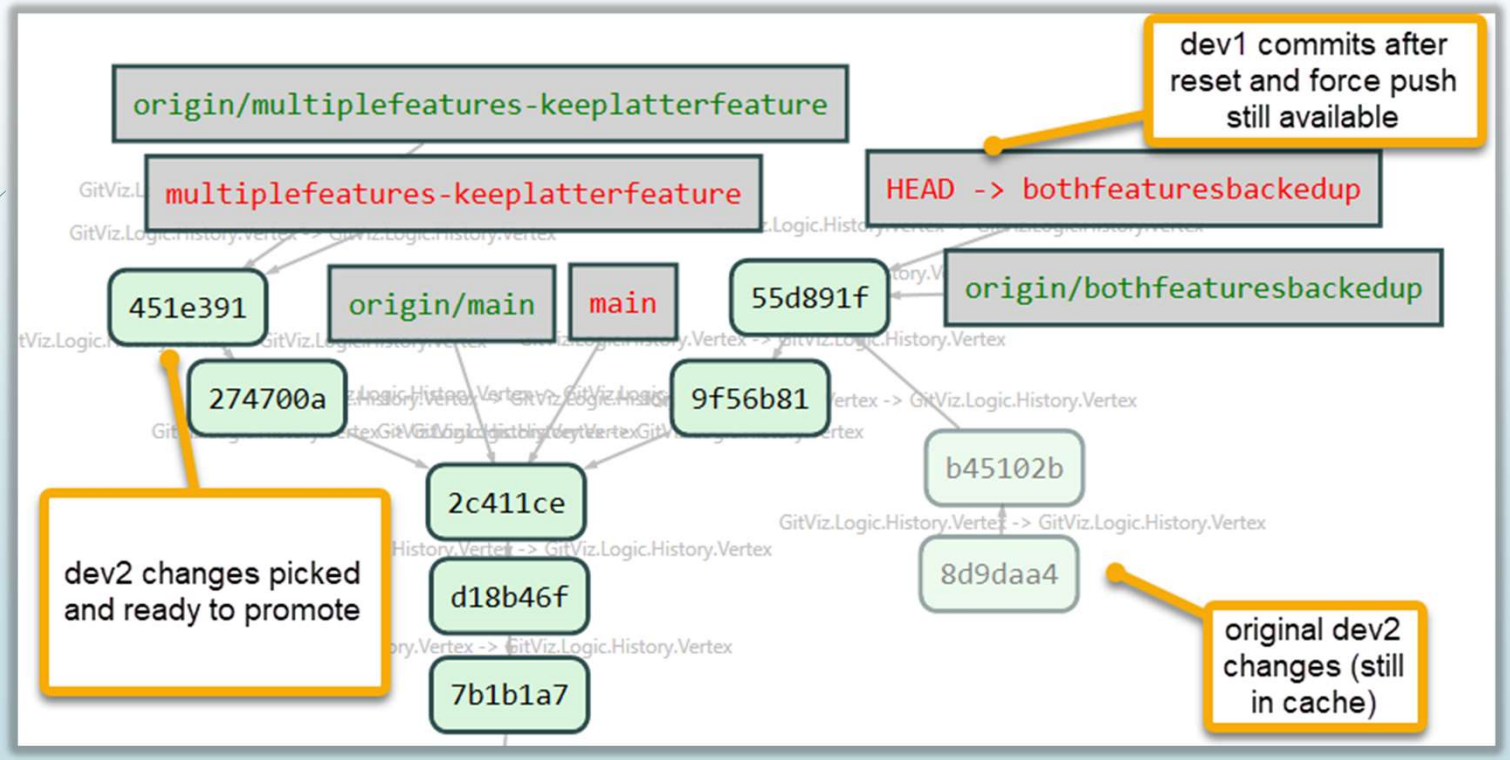


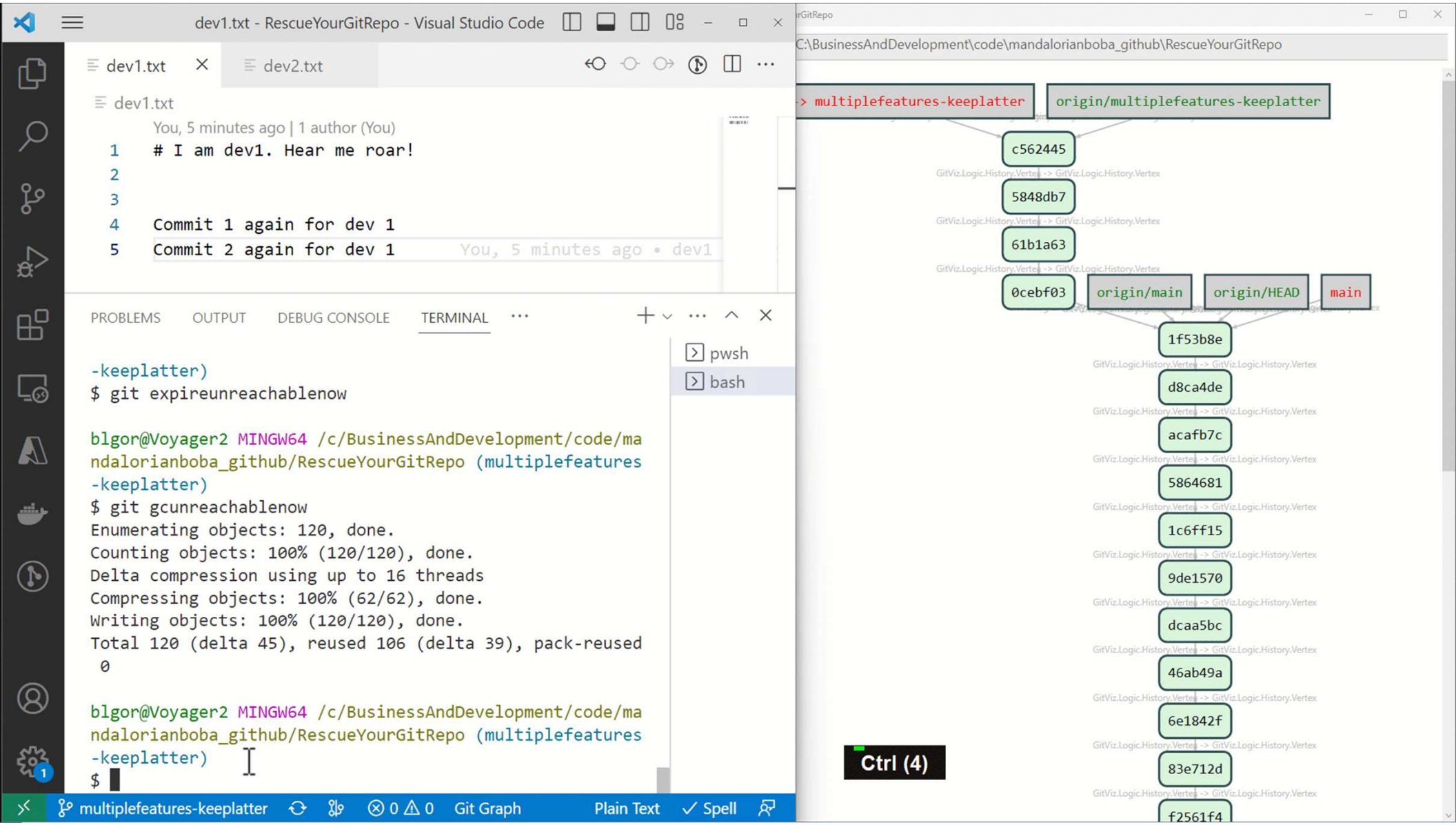
Reset the feature, then pick commits

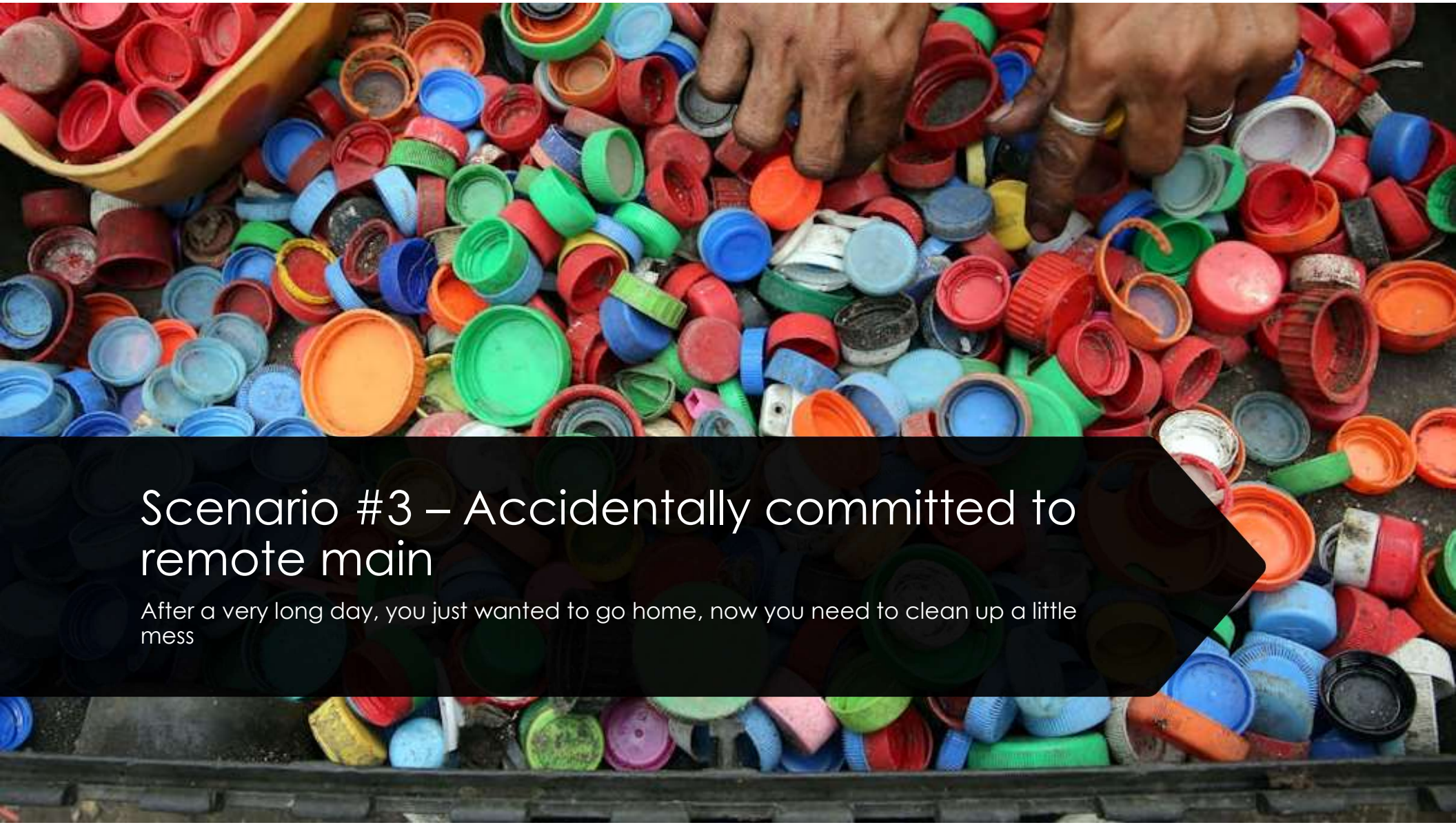


Reset backup branch to remove f2

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



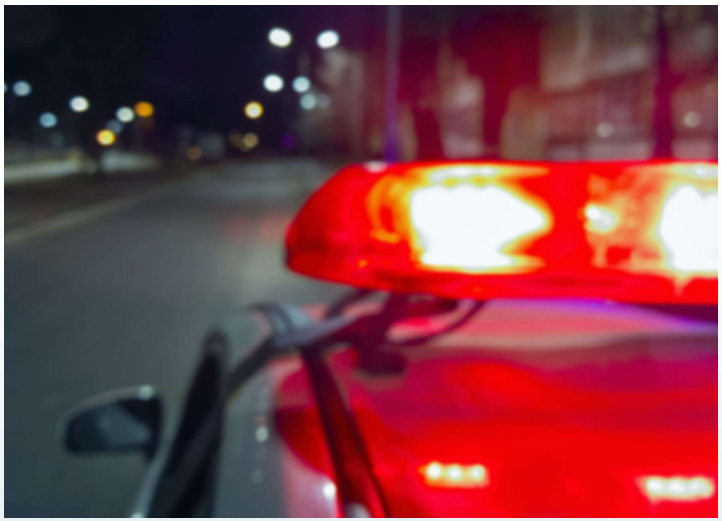

dev1.txt - RescueYourGitRepo - Visual Studio Code
dev1.txt
dev2.txt
dev1.txt
You, 5 minutes ago | 1 author (You)
1 # I am dev1. Hear me roar!
2
3
4 Commit 1 again for dev 1
5 Commit 2 again for dev 1 You, 5 minutes ago • dev1
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
pwsh
bash
-keeplatter)
$ git expireunreachablenow
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (multiplefeatures-keeplatter)
$ git gcunreachablenow
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)
$
multiplefeatures-keeplatter
0 0
Git Graph
Plain Text
Spell
C:\BusinessAndDevelopment\code\mandalorianboba_github\RescueYourGitRepo
> multiplefeatures-keeplatter
origin/multiplefeatures-keeplatter
c562445
5848db7
61b1a63
0cebf03
origin/main
origin/HEAD
main
1f53b8e
d8ca4de
acafb7c
5864681
1c6ff15
9de1570
dcaa5bc
46ab49a
6e1842f
83e712d
f2561f4
Ctrl (4)



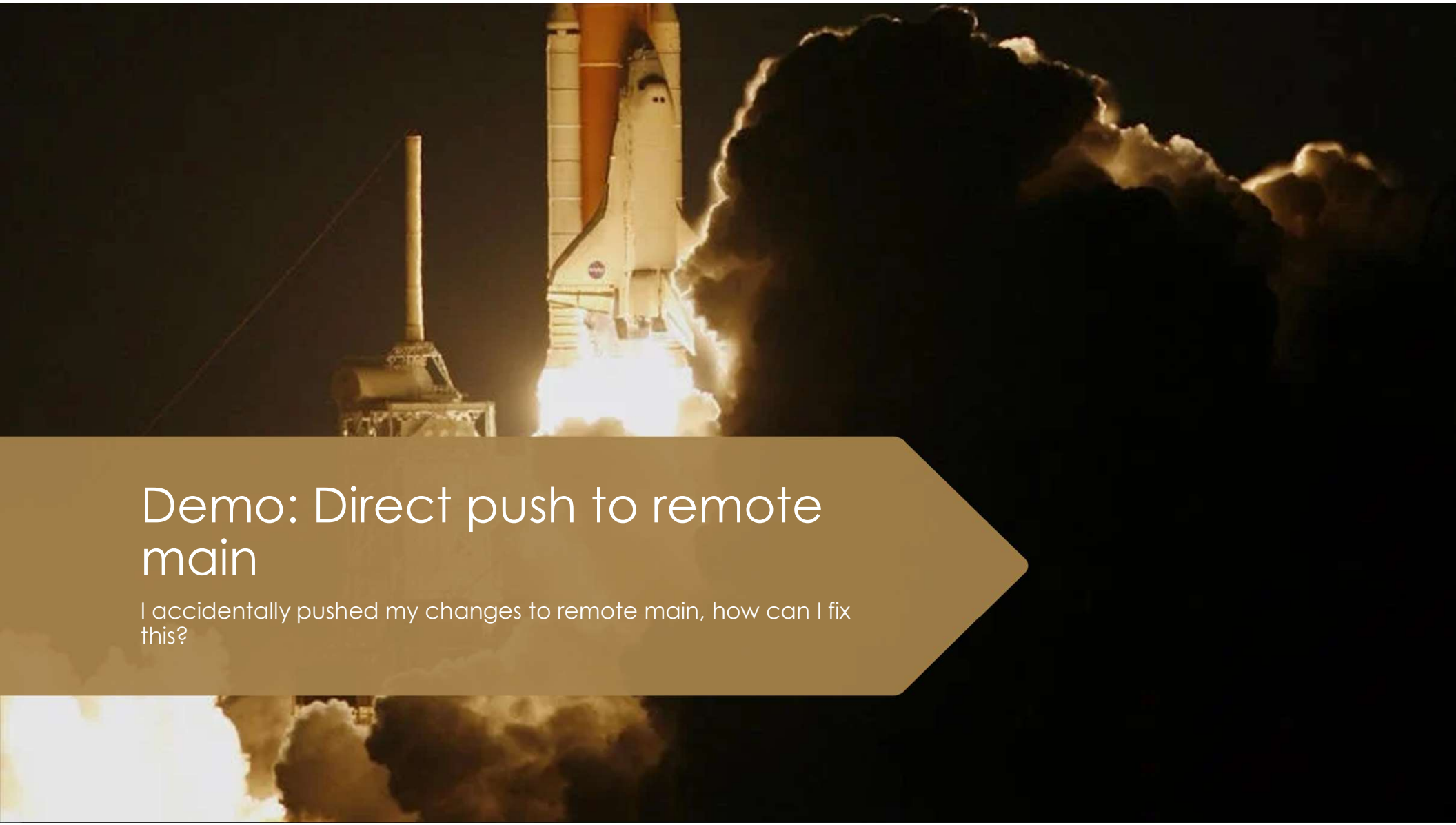
Scenario #3 – Accidentally committed to remote main

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

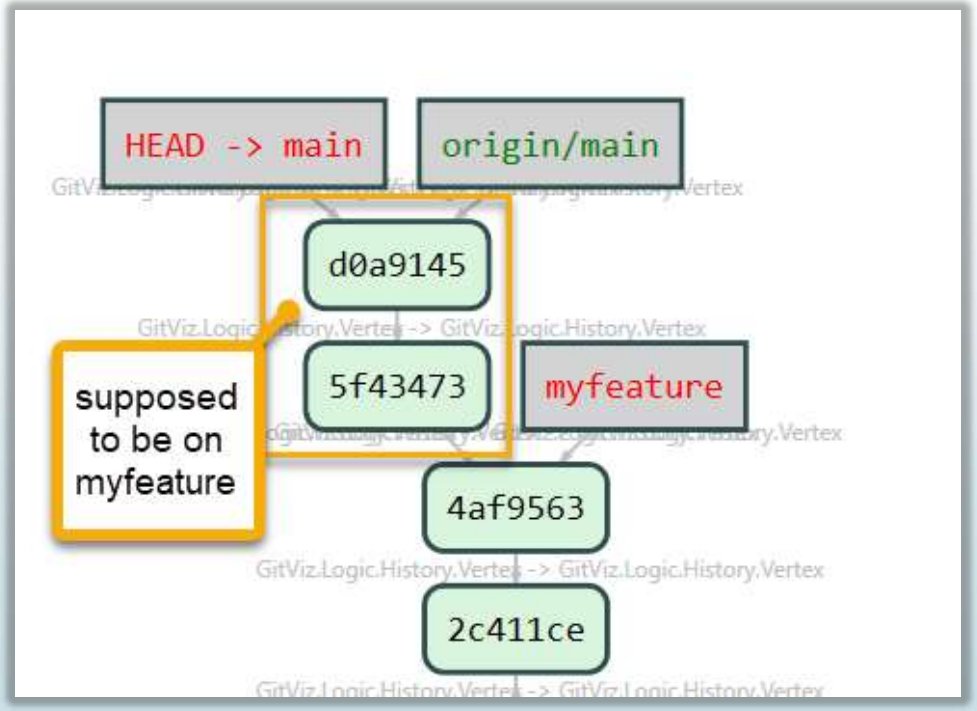


Demo: Direct push to remote main

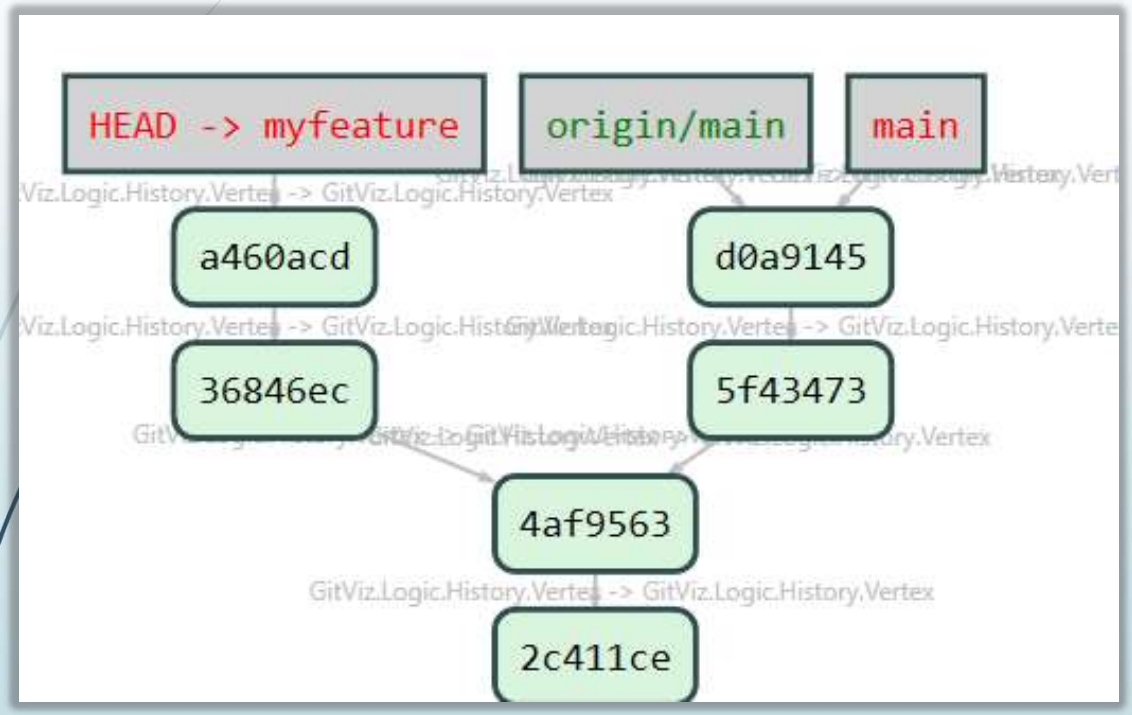
I accidentally pushed my changes to remote main, how can I fix this?

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

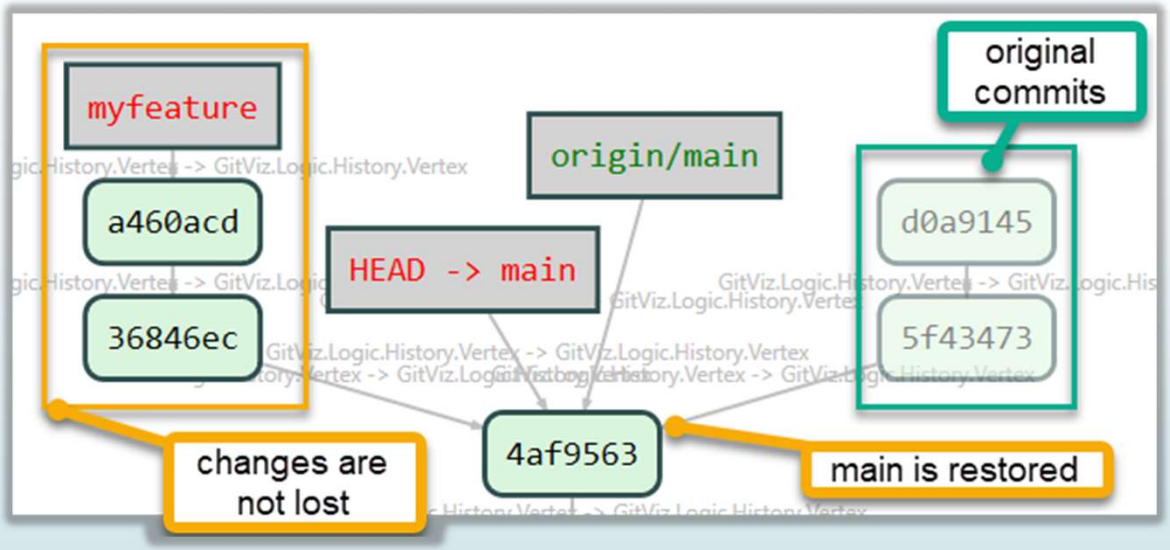


git switch myfeature

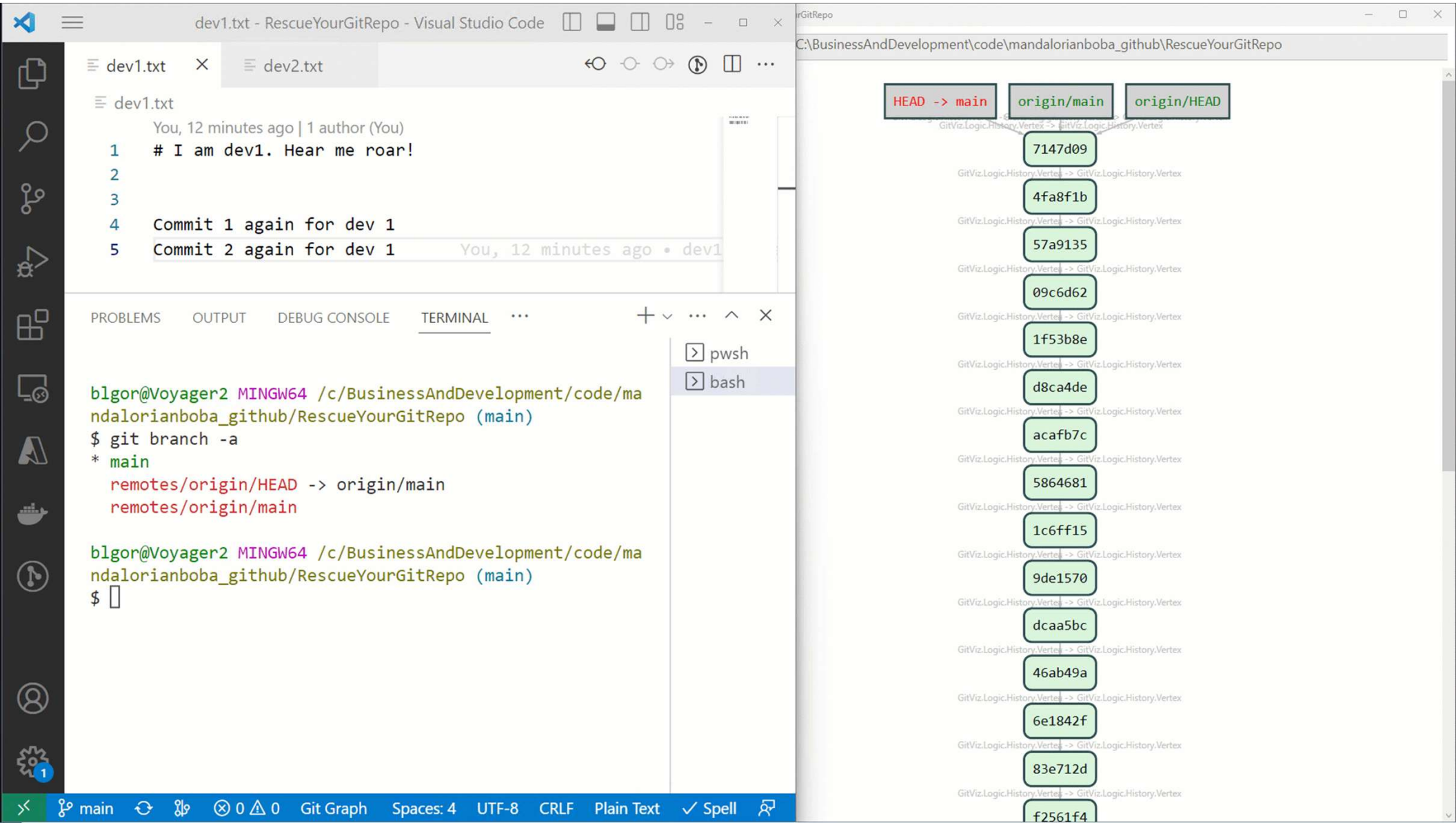
git cherry-pick 5f43473

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
```

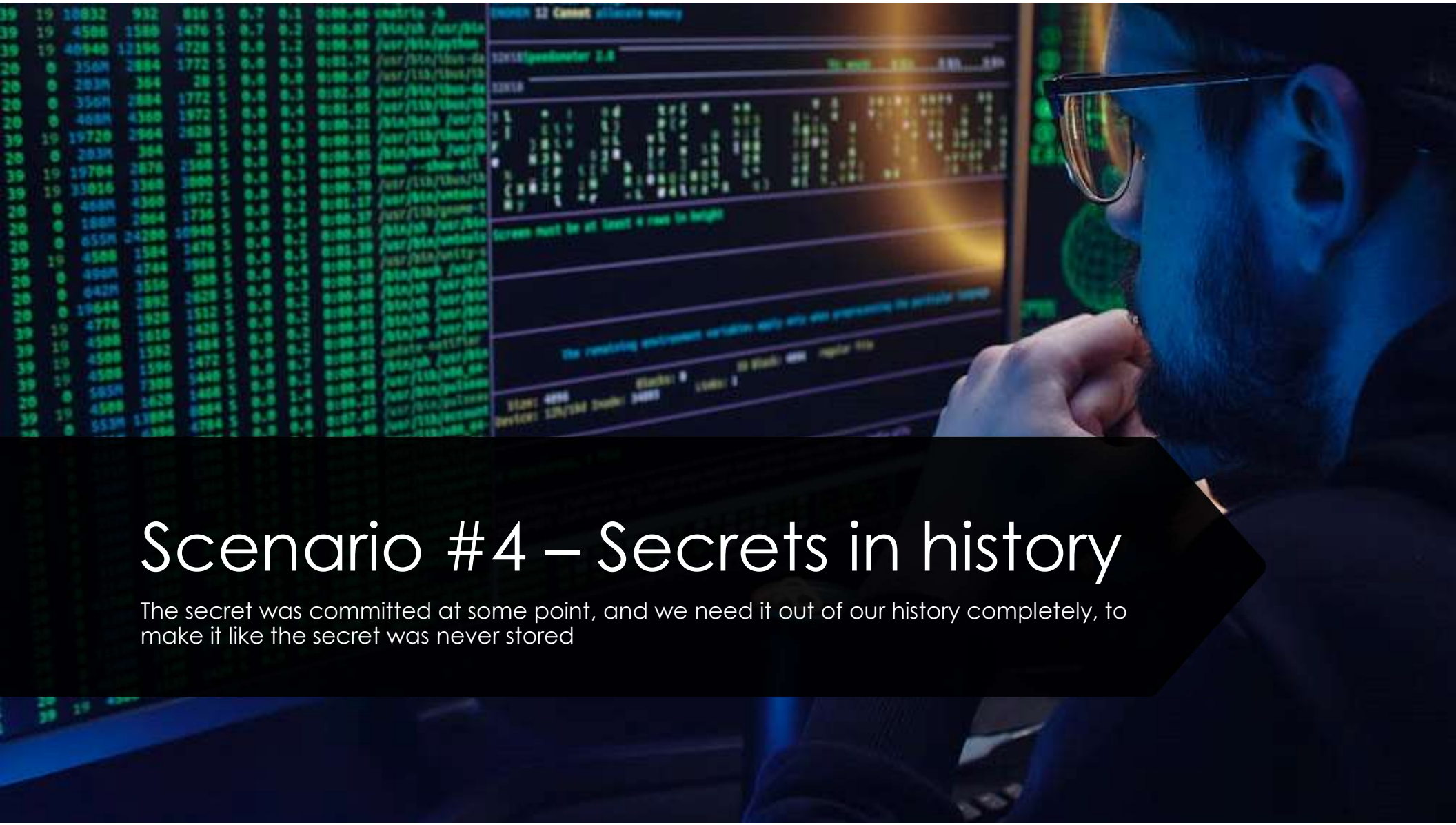

dev1.txt - RescueYourGitRepo - Visual Studio Code
dev1.txt
dev2.txt
dev1.txt
You, 12 minutes ago | 1 author (You)
1 # I am dev1. Hear me roar!
2
3
4 Commit 1 again for dev 1
5 Commit 2 again for dev 1 You, 12 minutes ago • dev1
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
pwsh
bash
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (main)
$ git branch -a
* main
remotes/origin/HEAD -> origin/main
remotes/origin/main
blgor@Voyager2 MINGW64 /c/BusinessAndDevelopment/code/mandalorianboba_github/RescueYourGitRepo (main)
$
main
0 0
Git Graph
Spaces: 4
UTF-8
CRLF
Plain Text
Spell
C:\BusinessAndDevelopment\code\mandalorianboba_github\RescueYourGitRepo
HEAD -> main
origin/main
origin/HEAD
7147d09
4fa8f1b
57a9135
09c6d62
1f53b8e
d8ca4de
acafb7c
5864681
1c6ff15
9de1570
dcaa5bc
46ab49a
6e1842f
83e712d
f2561f4

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



Scenario #4 – Secrets in history

The secret was committed at some point, and we need it out of our history completely, to make it like the secret was never stored

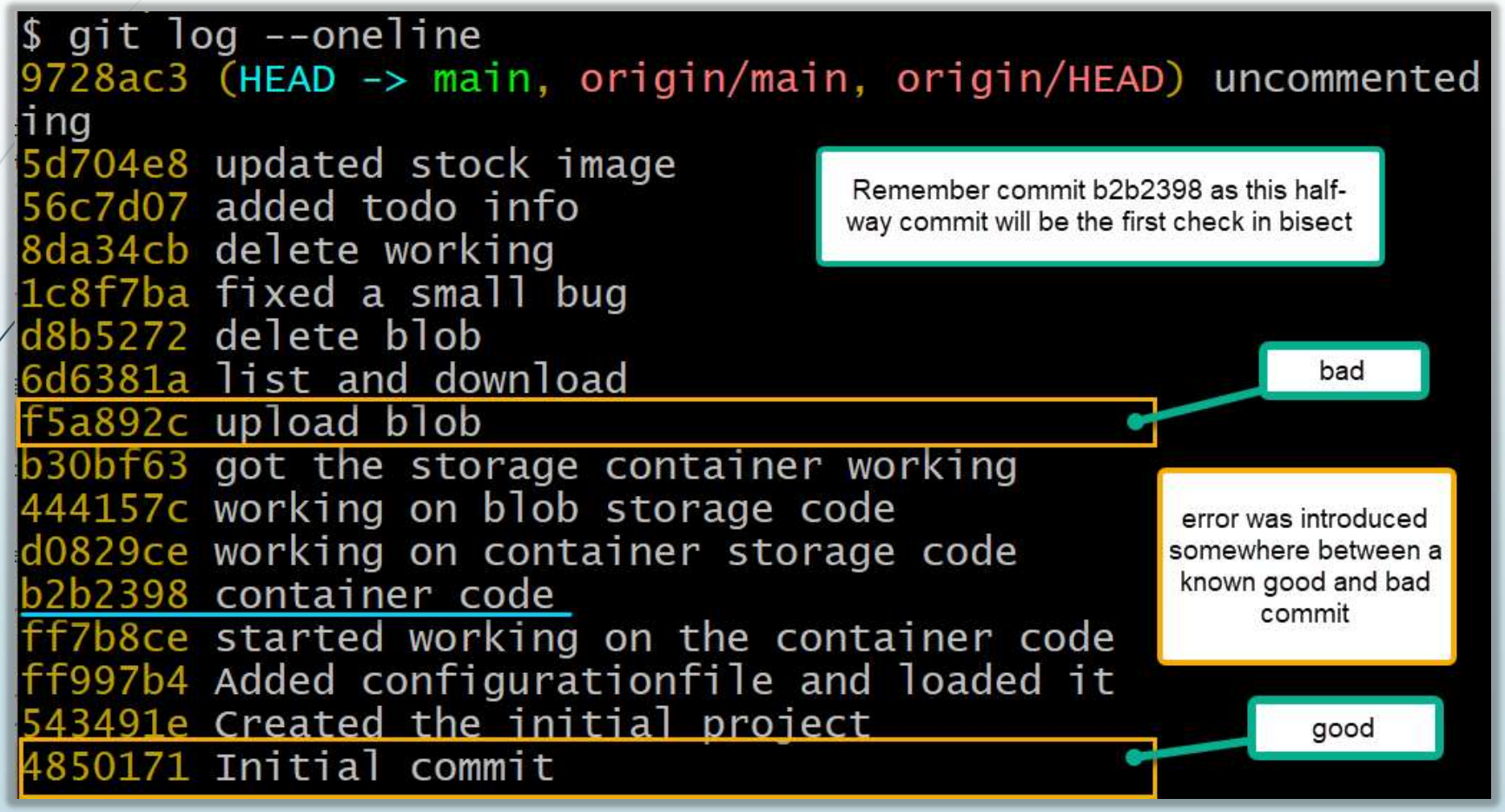
Find the first `bad` commit, then rewrite history

- 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

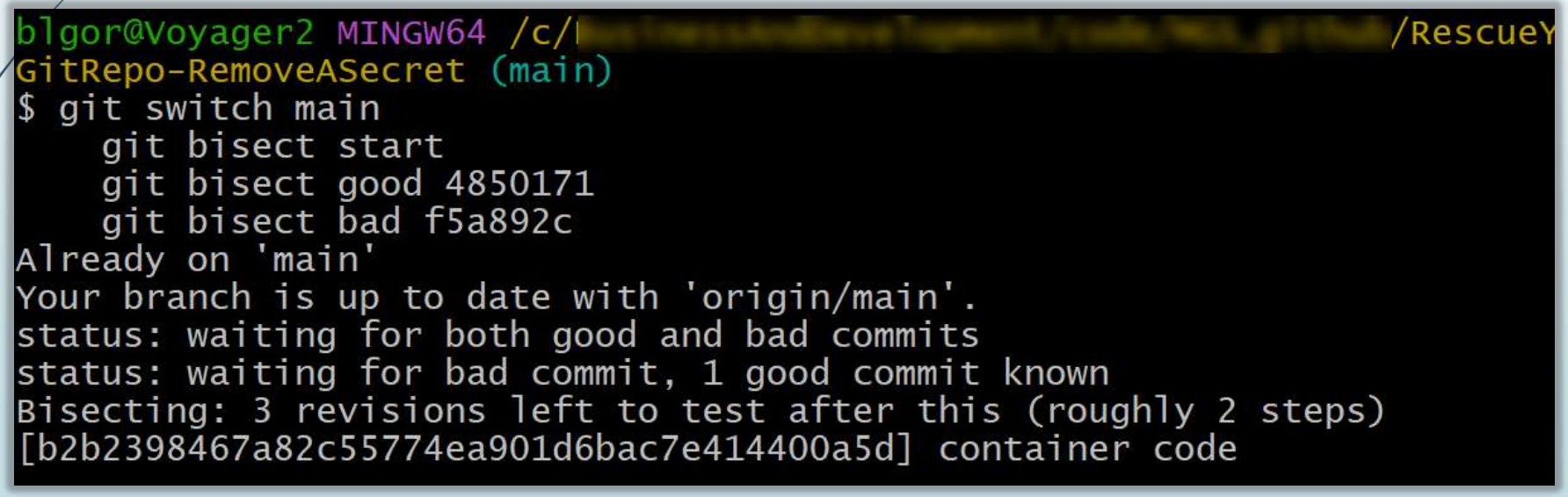
I accidentally checked a secret into my repo. How do I get it out of there?

Review Commit History

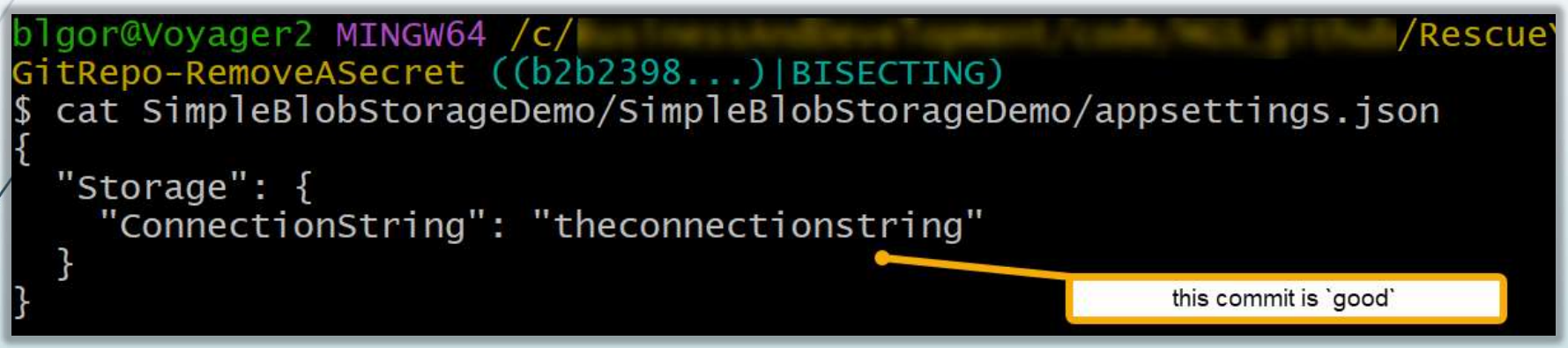


Find the first bad commit

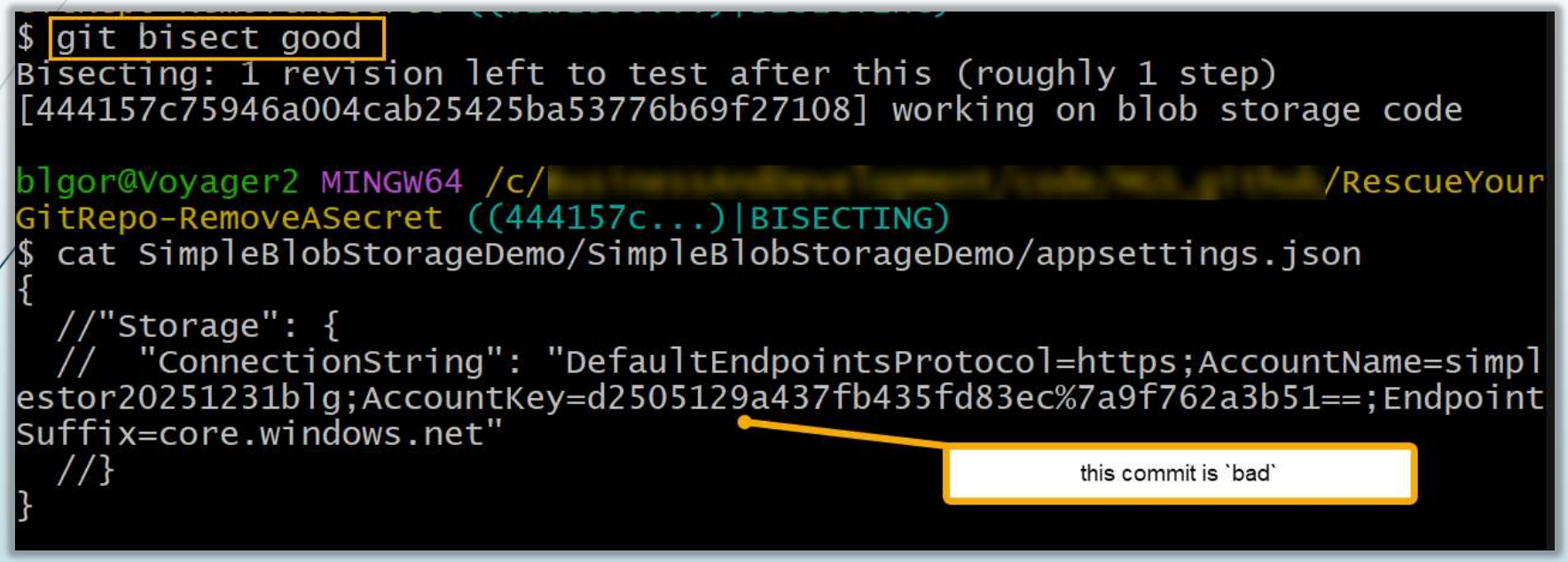
```
git switch main
git bisect start
git bisect good 4850171
git bisect bad f5a892c
```



Review the file to validate if it is good or bad



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

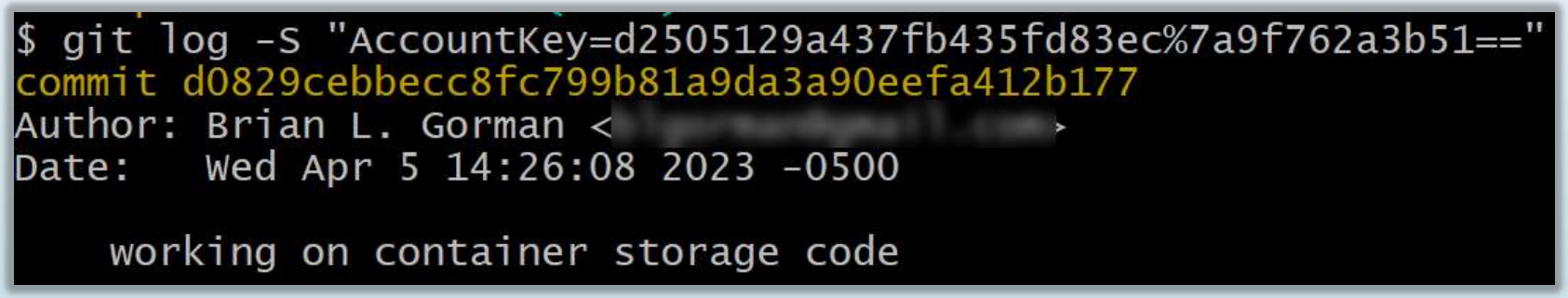


Bisect identifies the first bad commit

```
blgor@Voyager2 MINGW64 /c/[redacted]/RescueY
GitRepo-RemoveASecret ((d0829ce...)|BISECTING)
$ cat SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json
{
  "Storage": {
    "ConnectionString": "DefaultEndpointsProtocol=https;AccountName=simp
tor20251231blg;AccountKey=d2505129a437fb435fd83ec%7a9f762a3b51==;Endpoin
ffix=core.windows.net"
  }
}

blgor@Voyager2 MINGW64 /c/[redacted]/RescueY
GitRepo-RemoveASecret ((d0829ce...)|BISECTING)
$ git bisect bad
d0829cebbecc8fc799b81a9da3a90eefa412b177 is the first bad commit
commit d0829cebbecc8fc799b81a9da3a90eefa412b177
Author: Brian L. Gorman <[redacted]>
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=="
```

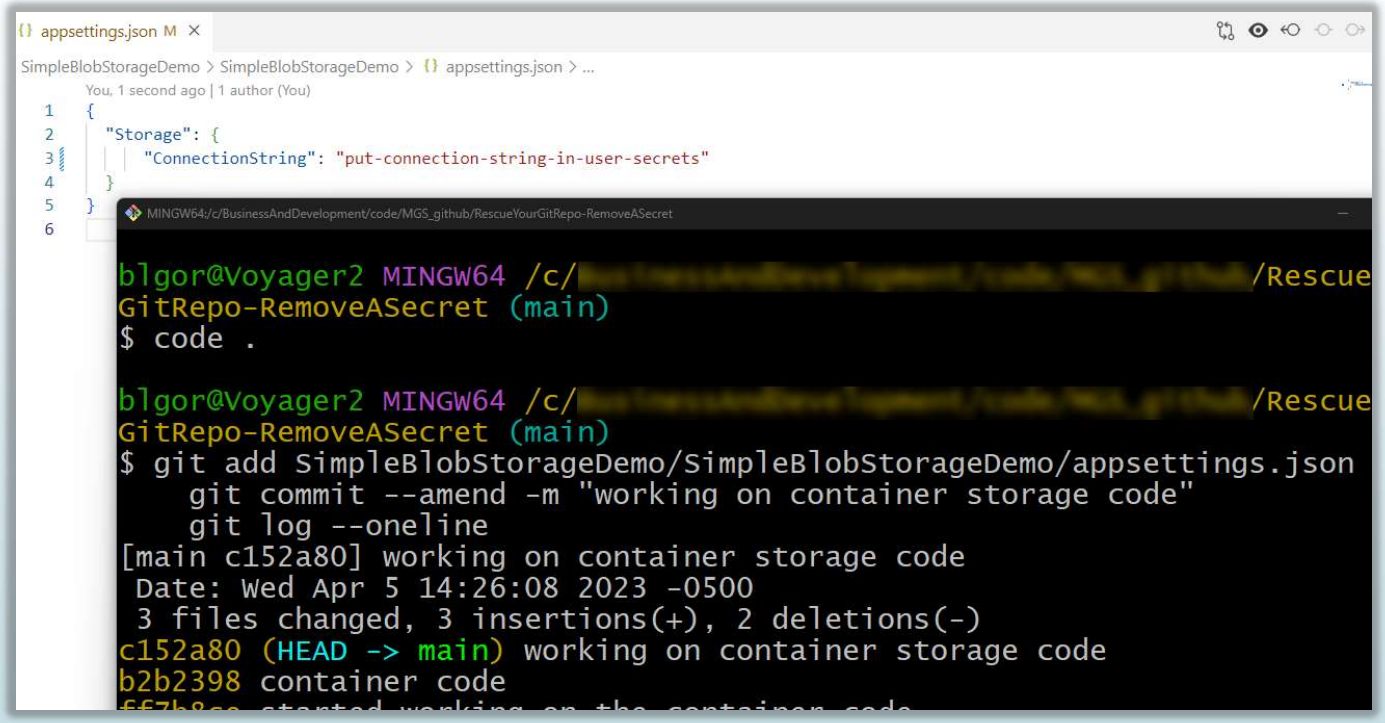


Create a backup and reset local main to the first bad commit

```
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




```
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/SimpleBlobStora
geDemo/appsettings.json
error: could not apply 9728ac3... uncommented 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".

blgor@Voyager2 MINGW64 /c/[redacted]/RescueYour
GitRepo-RemoveASecret (main|CHERRY-PICKING)
$
```

Resolve the conflict and complete the cherry-pick operation

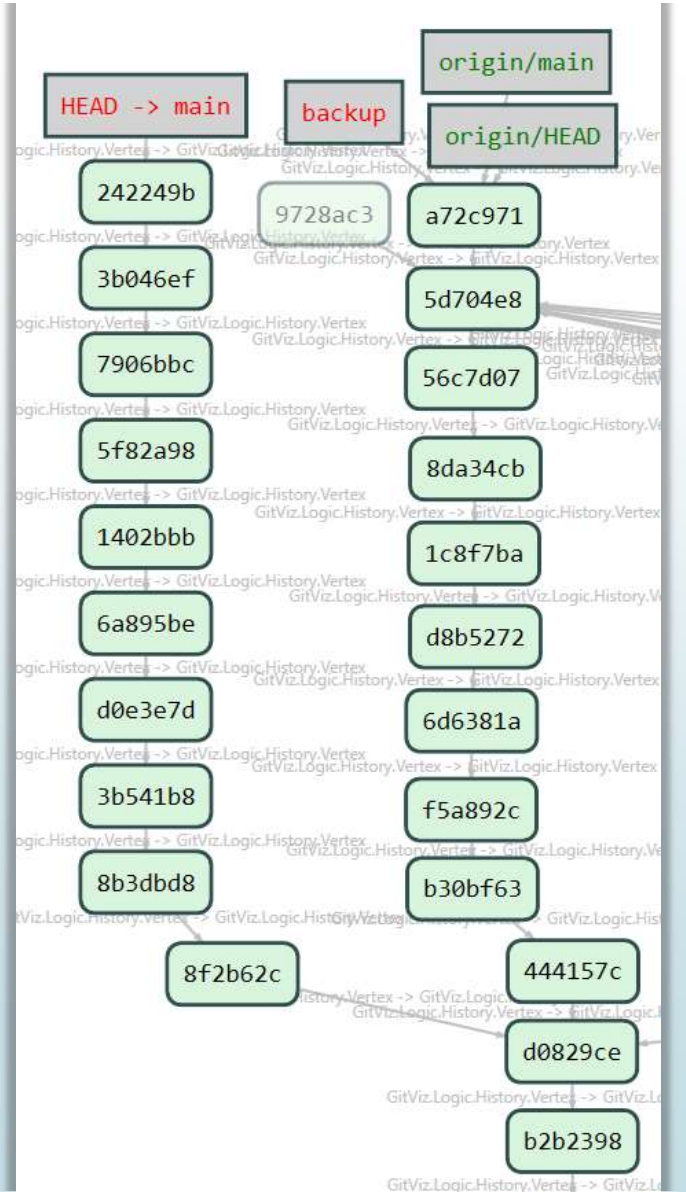
```
$ git mergetool
Merging:
SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json

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

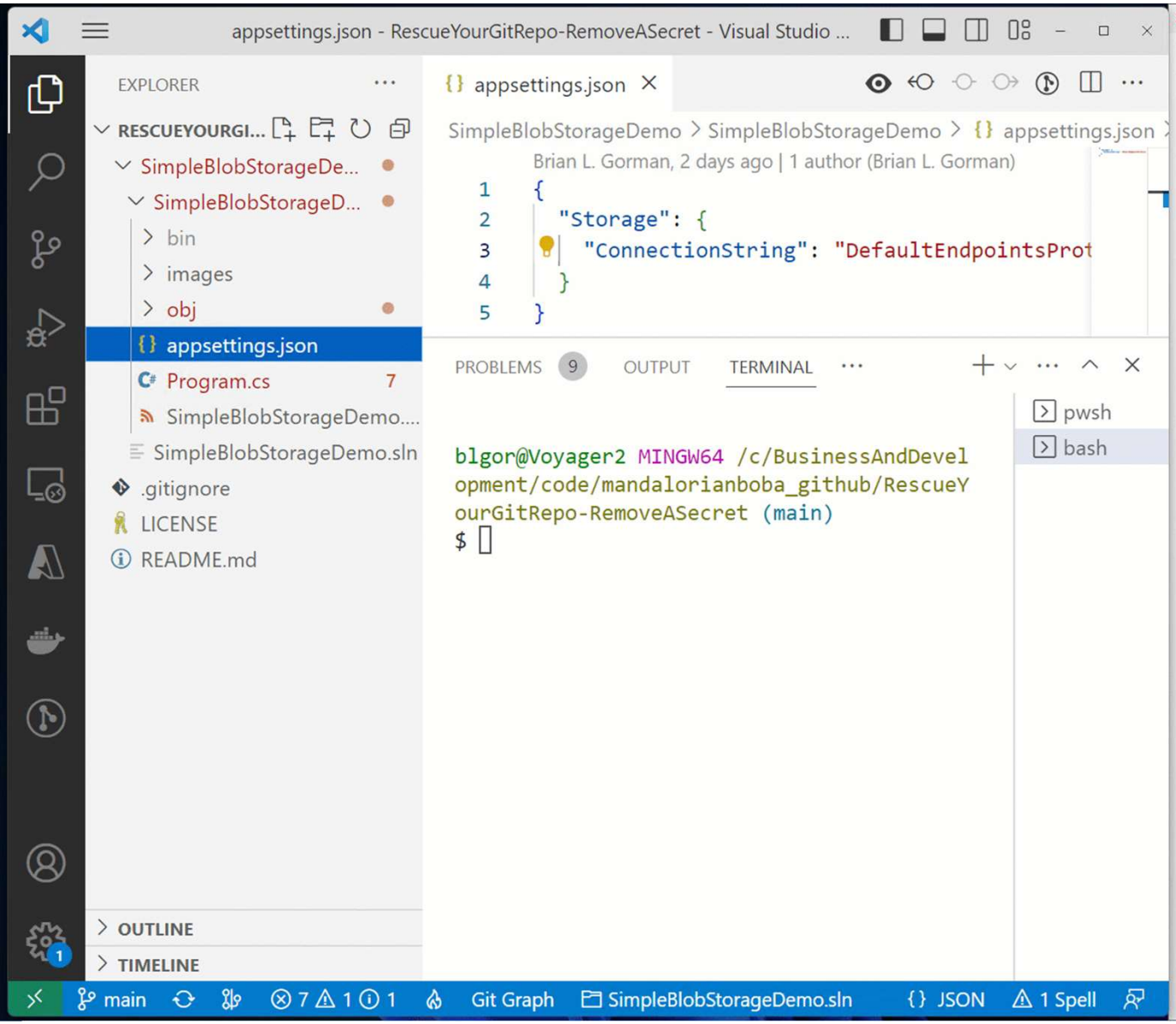
```
blgor@Voyager2 MINGW64 /c/[redacted]/RescueYou
GitRepo-RemoveASecret (main|CHERRY-PICKING)
$ git add SimpleBlobStorageDemo/SimpleBlobStorageDemo/appsettings.json
```

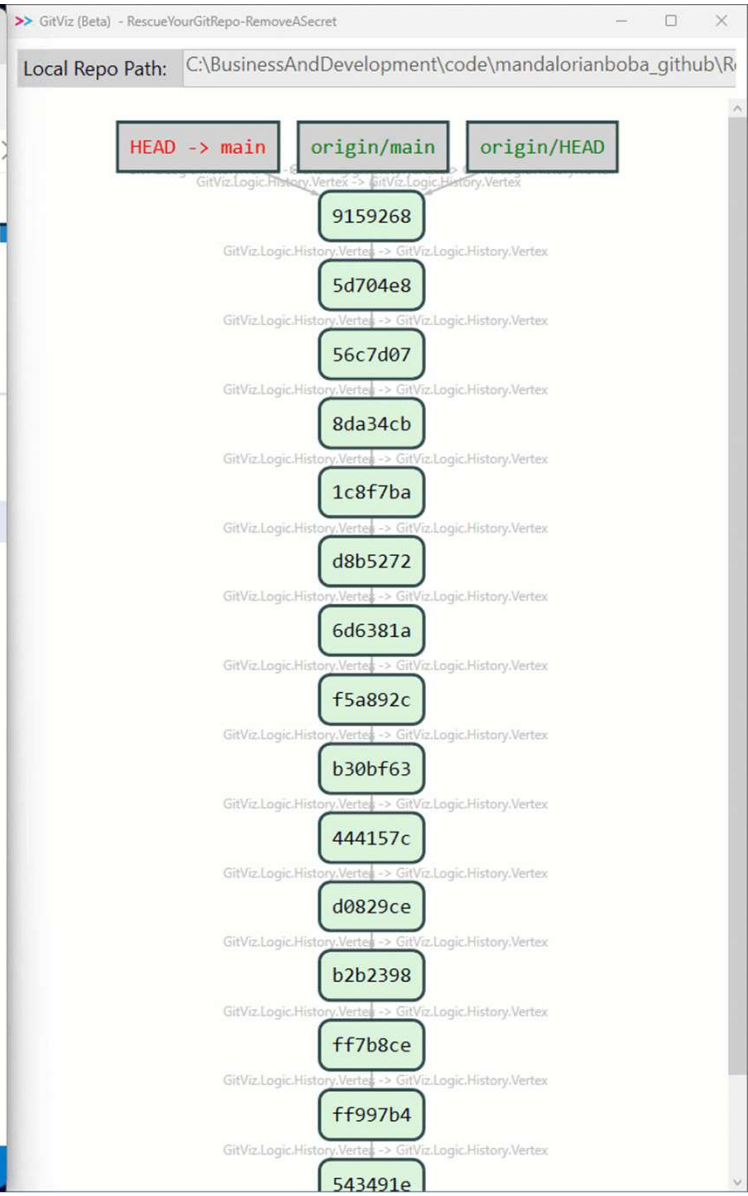
```
$ git cherry-pick --continue
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty
```

Force your changes onto remote main

```
git push --force-with-lease
```




appsettings.json - RescueYourGitRepo-RemoveASecret - Visual Studio ...
EXPLORER
RESCUEYOURGI...
SimpleBlobStorageDe...
SimpleBlobStorageD...
bin
images
obj
appsettings.json
Program.cs 7
SimpleBlobStorageDemo....
SimpleBlobStorageDemo.sln
.gitignore
LICENSE
README.md
OUTLINE
TIMELINE
appsettings.json
SimpleBlobStorageDemo > SimpleBlobStorageDemo > appsettings.json
Brian L. Gorman, 2 days ago | 1 author (Brian L. Gorman)
1 {
2 "Storage": {
3 "ConnectionString": "DefaultEndpointsProt
4 }
5 }
PROBLEMS 9
OUTPUT
TERMINAL
pwsh
bash
blgor@Voyager2 MINGW64 /c/BusinessAndDevel
opment/code/mandalorianboba_github/RescueY
ourGitRepo-RemoveASecret (main)
$
main
7 1 1
Git Graph
SimpleBlobStorageDemo.sln
JSON
1 Spell


GitViz (Beta) - RescueYourGitRepo-RemoveASecret
Local Repo Path:
C:\BusinessAndDevelopment\code\mandalorianboba_github\R
HEAD -> main
origin/main
origin/HEAD
9159268
5d704e8
56c7d07
8da34cb
1c8f7ba
d8b5272
6d6381a
f5a892c
b30bf63
444157c
d0829ce
b2b2398
ff7b8ce
ff997b4
543491e

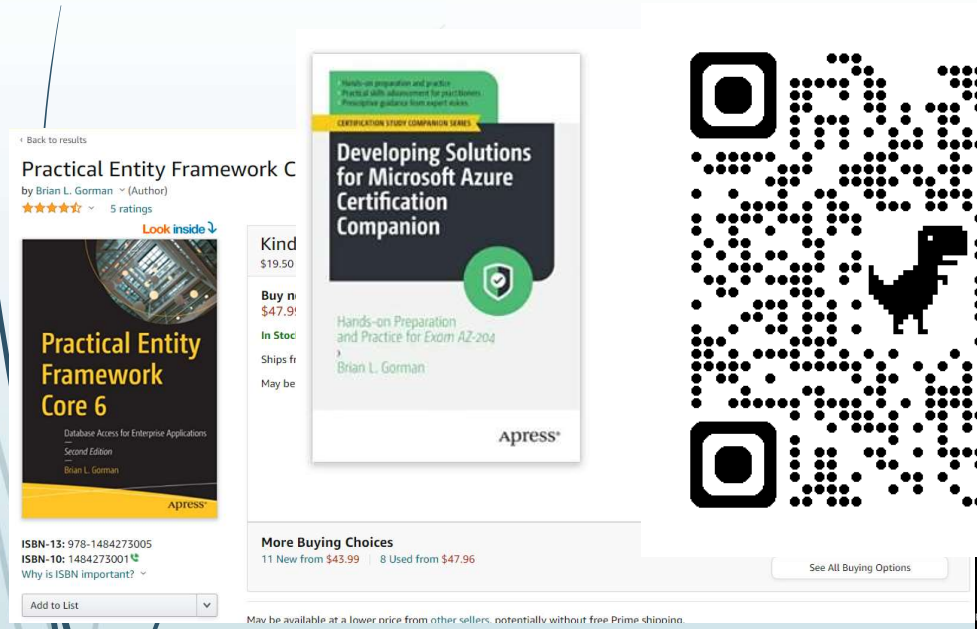
Summary

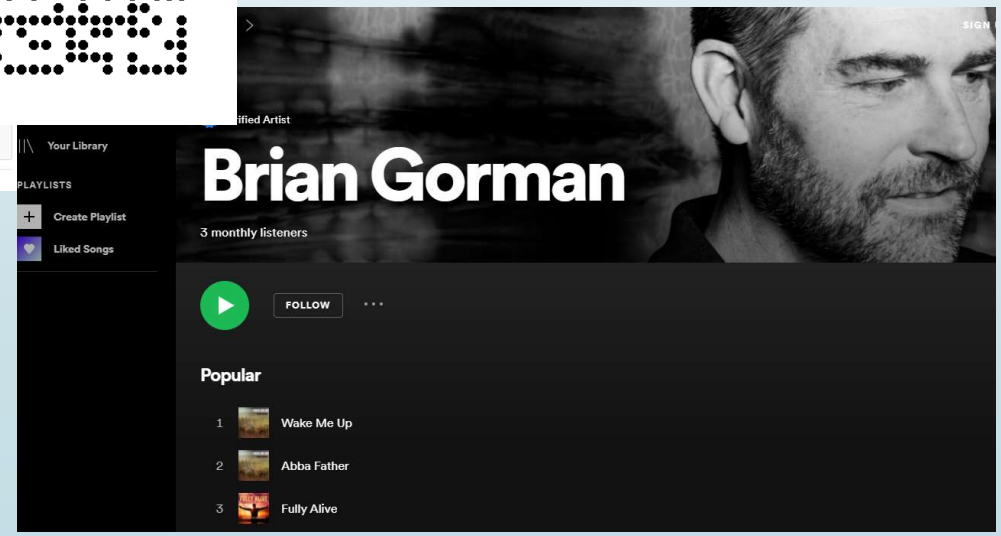
- Consider a linear history for best results
- Consider squash/interactive rebase for single commit features
- Nothing is ever ruined (it's very difficult to mess up a GIT repo)
- Remember that you have a local cache of commits
- Use caution and rework your history with confidence
- When in doubt, create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





@blgorman

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

Thanks for having me today!