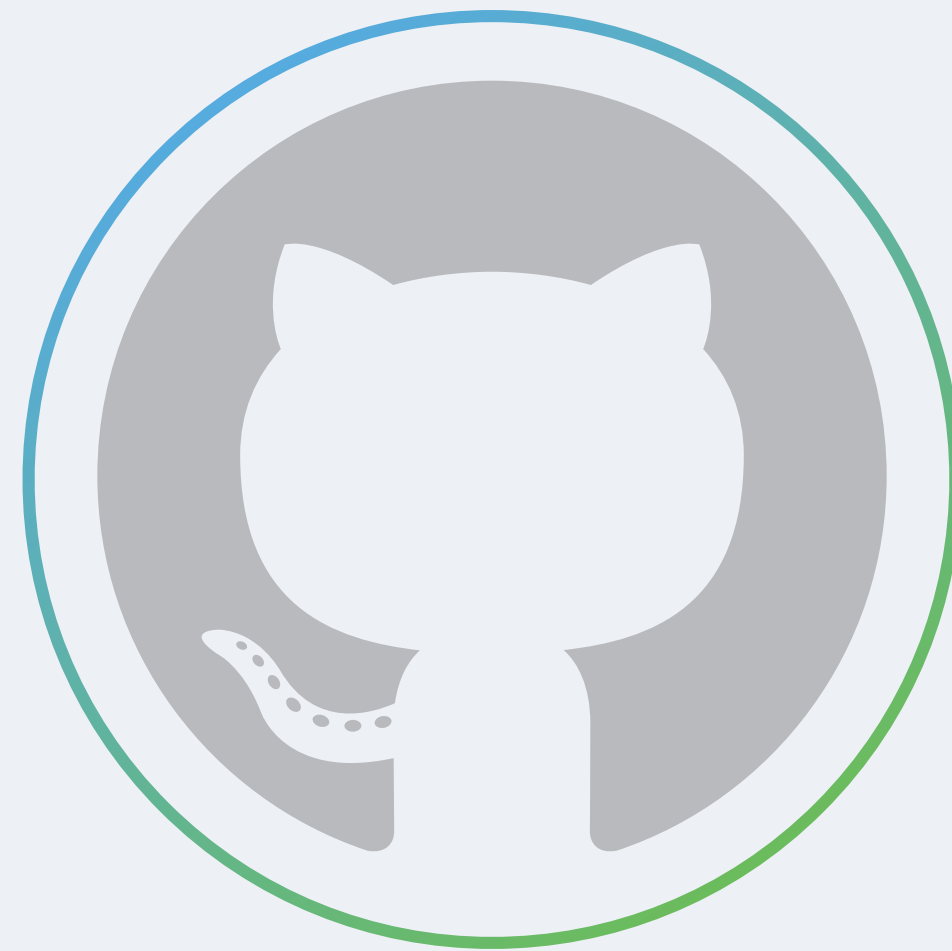
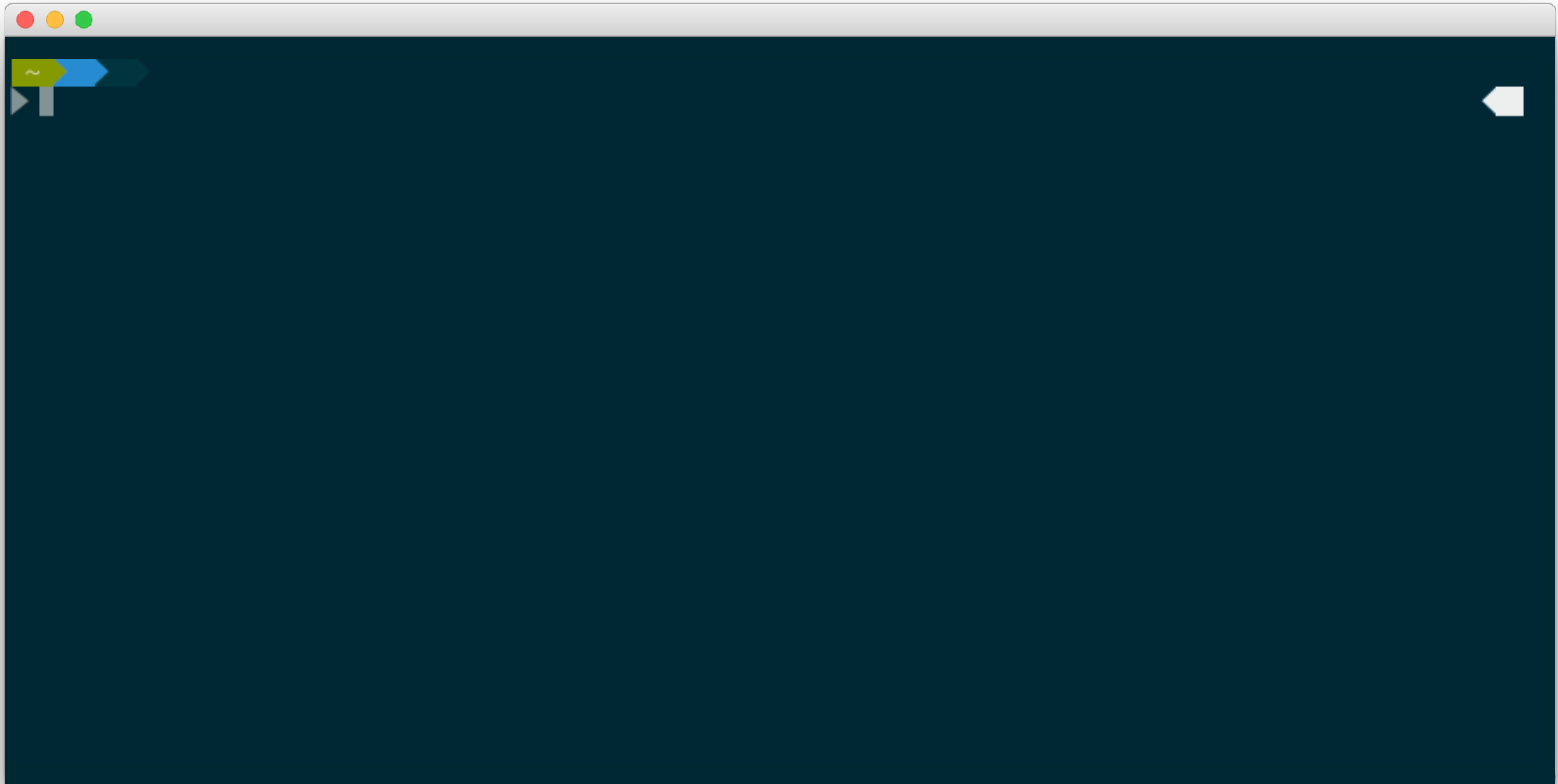


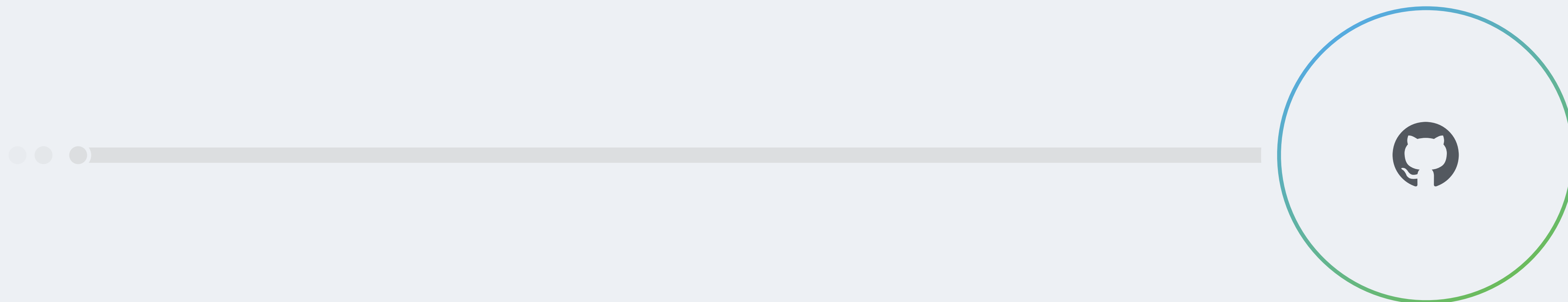
**Git and the
Terrible,
Horrible,
No Good,
Very Bad Day**



@hectorsector
Trainer, GitHub

github.com/hectorsector/git-and-the-bad-day

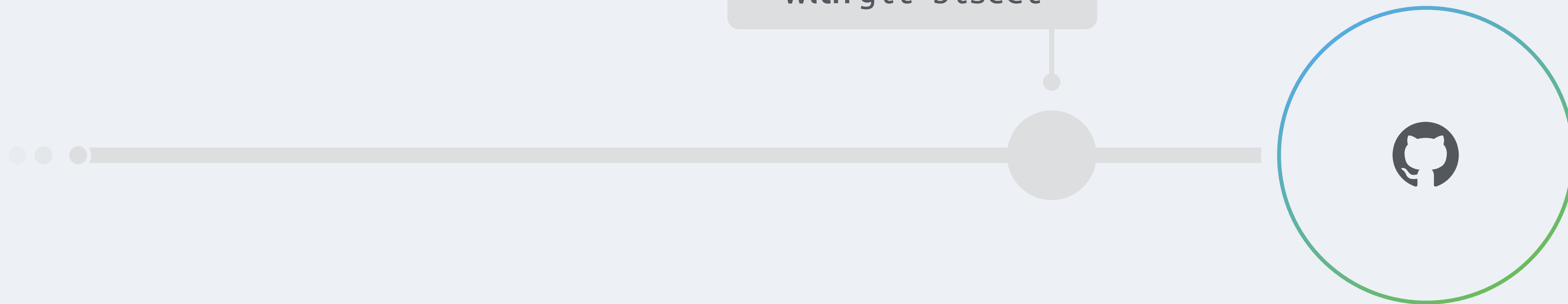




Find naughty commits with `git bisect`

```
~
▶ git clone https://github.com/hectorsector/git-and-the-bad-day/
Cloning into 'git-and-the-bad-day'...
remote: Counting objects: 18, done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 18 (delta 5), reused 18 (delta 5), pack-reused 0
Unpacking objects: 100% (18/18), done.
~
▶ cd git-and-the-bad-day
~/git-and-the-bad-day master 2h31m
▶ git log --oneline
c2ebae1 add conclusion
815de43 add section on BFG
7434ff2 add section on filter-branch
0097846 add section on reset
0d37556 add section on git revert
06f84ba add section on git bisect
~/git-and-the-bad-day master 2h31m
▶
```

Find naughty commits
with `git bisect`



Safely undo commits
with `git revert`

But before bed I decided to give it another shot. I was successful with `git filter-branch` I was happy with myself, but then I realized a local backup of what I cleaned up was still in my local git directory.

I pushed my changes up to GitHub, and my push was rejected. It's because I altered commit history so I had to force push. Cynthia then told me that GitHub caches commits that may still have my credentials. So I had to contact Support and they removed the cached commits from GitHub.

~/git-and-the-bad-day ➤ 7434ff2...bisect ➤ 2h34m

▶ git bisect good

815de431b994a5fe6e7af032a3ae0b4aa4423db9 is the first bad commit

commit 815de431b994a5fe6e7af032a3ae0b4aa4423db9

Author: Hector Alfaro <hectorsector@github.com>

Date: Thu Feb 2 01:43:57 2017 +0100

add section on BFG

:100644 100644 833b06b37142e60febfcdb8a2826a4a796dea 754eda8da37069f912795382dc3ea13110a78aee M git-and-the-bad-day.md

~/git-and-the-bad-day ➤ bisect/good-7434ff282e17737d36398773b7f82e5a756a5631bisect ➤ 2h34m

▶ git bisect reset

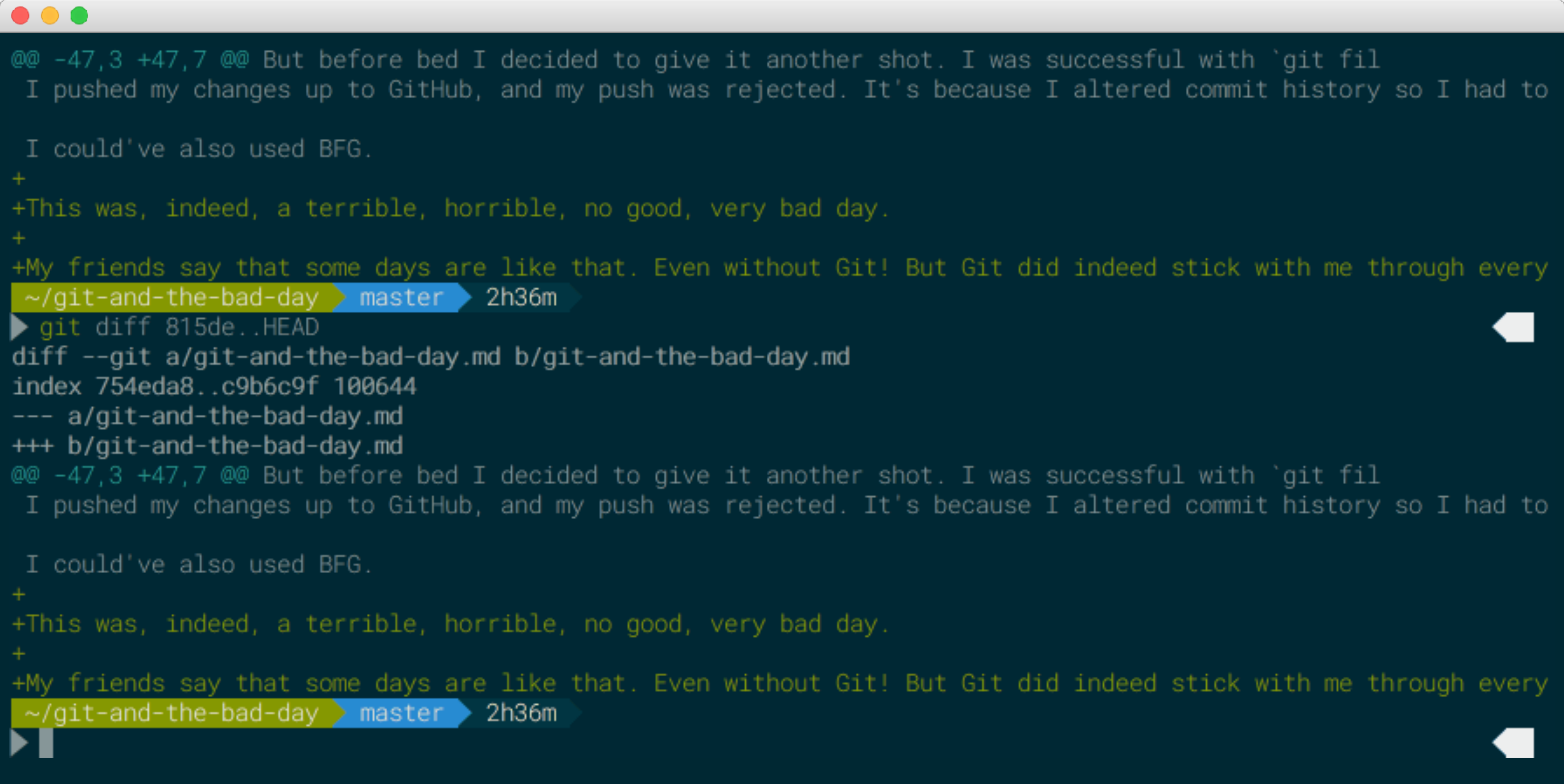
Previous HEAD position was 7434ff2... add section on filter-branch

Switched to branch 'master'

Your branch is up-to-date with 'origin/master'.

~/git-and-the-bad-day ➤ master ➤ 2h34m

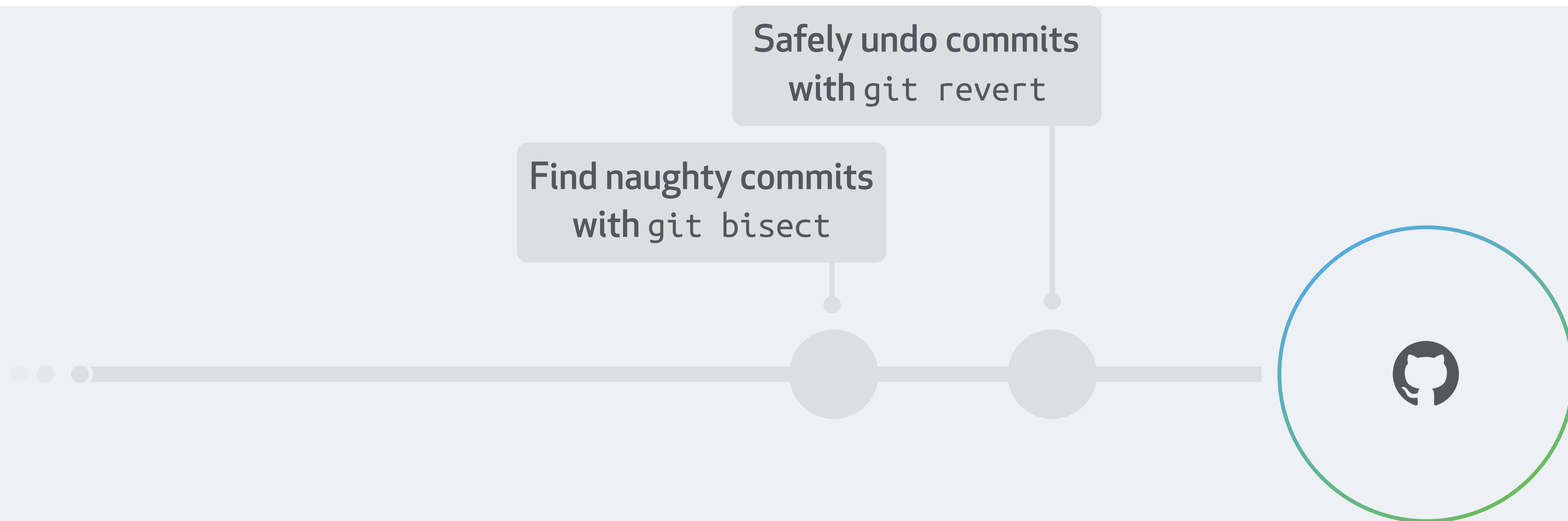
▶



```
@@ -47,3 +47,7 @@ But before bed I decided to give it another shot. I was successful with `git fil
I pushed my changes up to GitHub, and my push was rejected. It's because I altered commit history so I had to

I could've also used BFG.
+
+This was, indeed, a terrible, horrible, no good, very bad day.
+
+My friends say that some days are like that. Even without Git! But Git did indeed stick with me through every
~/git-and-the-bad-day master 2h36m
git diff 815de..HEAD
diff --git a/git-and-the-bad-day.md b/git-and-the-bad-day.md
index 754eda8..c9b6c9f 100644
--- a/git-and-the-bad-day.md
+++ b/git-and-the-bad-day.md
@@ -47,3 +47,7 @@ But before bed I decided to give it another shot. I was successful with `git fil
I pushed my changes up to GitHub, and my push was rejected. It's because I altered commit history so I had to

I could've also used BFG.
+
+This was, indeed, a terrible, horrible, no good, very bad day.
+
+My friends say that some days are like that. Even without Git! But Git did indeed stick with me through every
~/git-and-the-bad-day master 2h36m
```

**Move commits to
a new branch with
git reset**

```
#
# Changes to be committed:
#   modified:   git-and-the-bad-day.md
#
# Untracked files:
#   git-and-the-bad-day.md~
#
```

File /Users/halfaro1/git-and-the-bad-day/.git/COMMIT_EDITMSG not changed so no update needed

[master e34274e] Revert "add section on BFG"

1 file changed, 2 deletions(-)

~/git-and-the-bad-day master ● 0m

▶ git log --oneline

e34274e Revert "add section on BFG"

c2ebae1 add conclusion

815de43 add section on BFG

7434ff2 add section on filter-branch

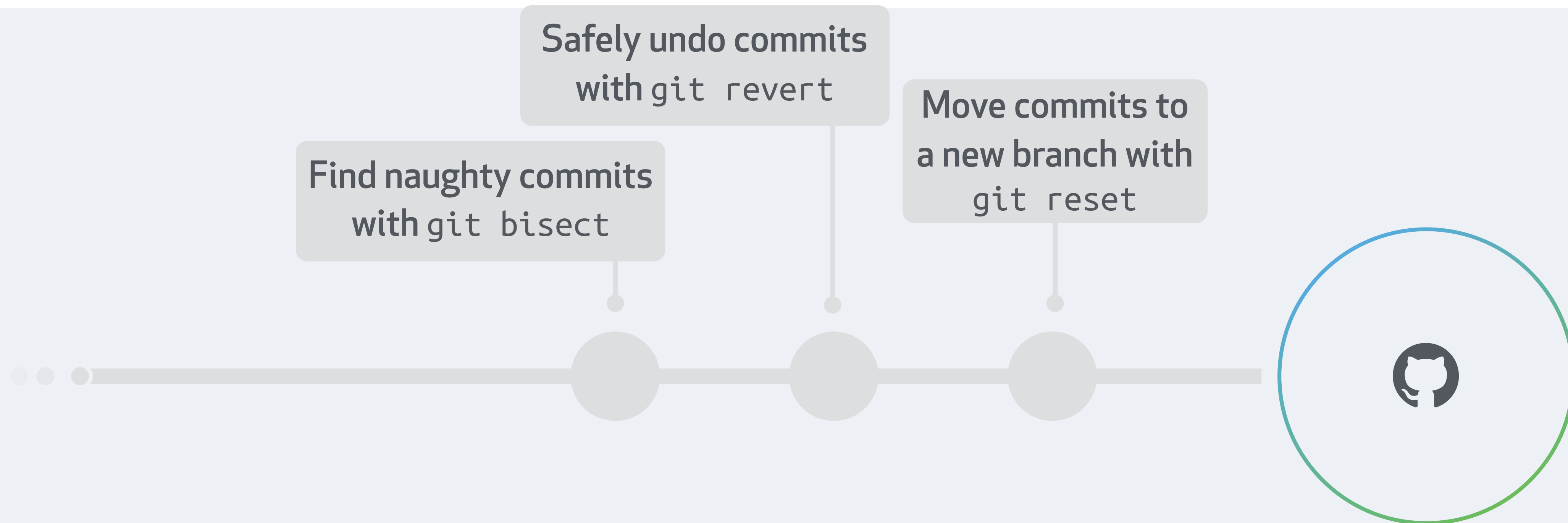
0097846 add section on reset

0d37556 add section on git revert

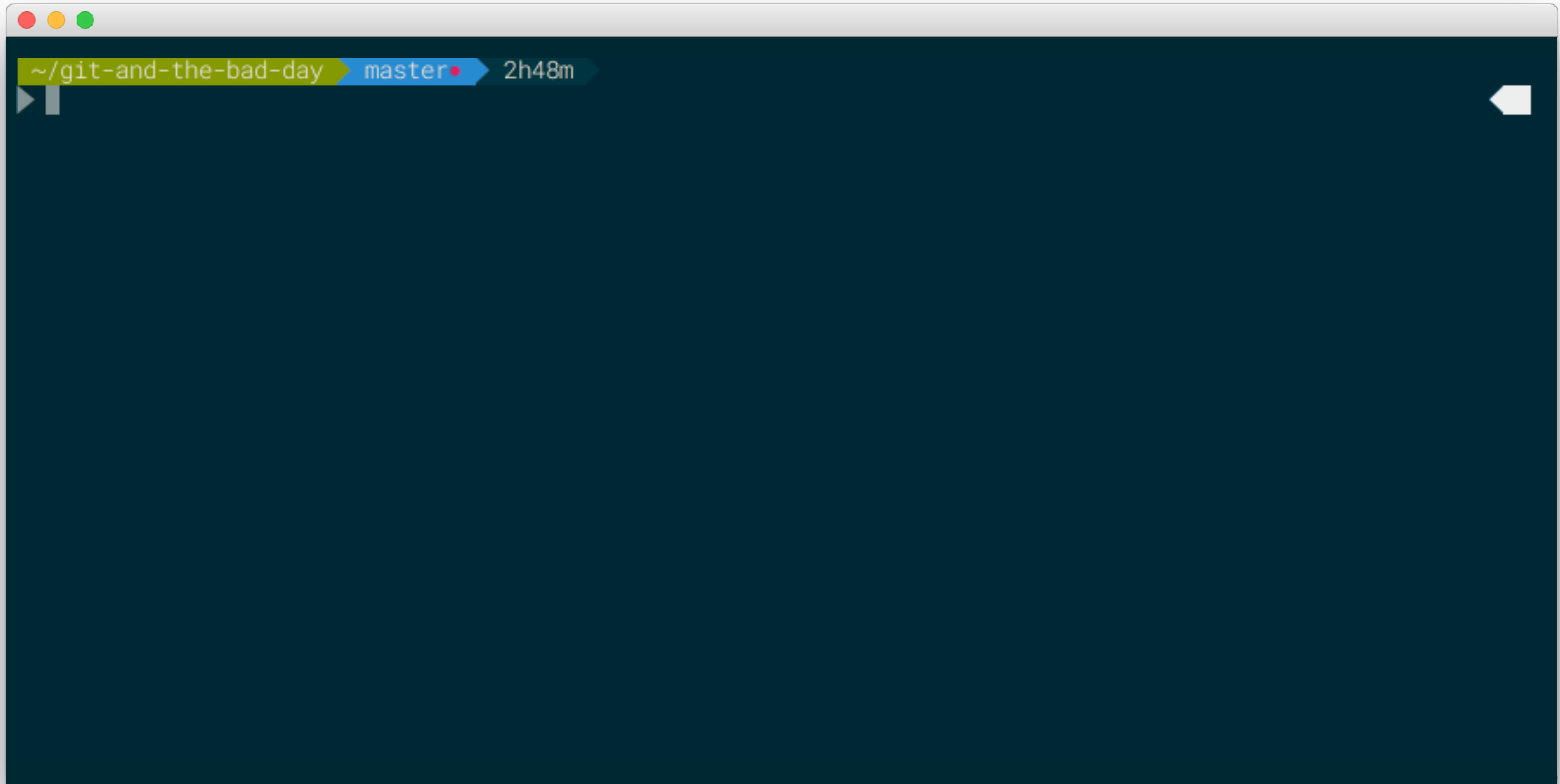
06f84ba add section on git bisect

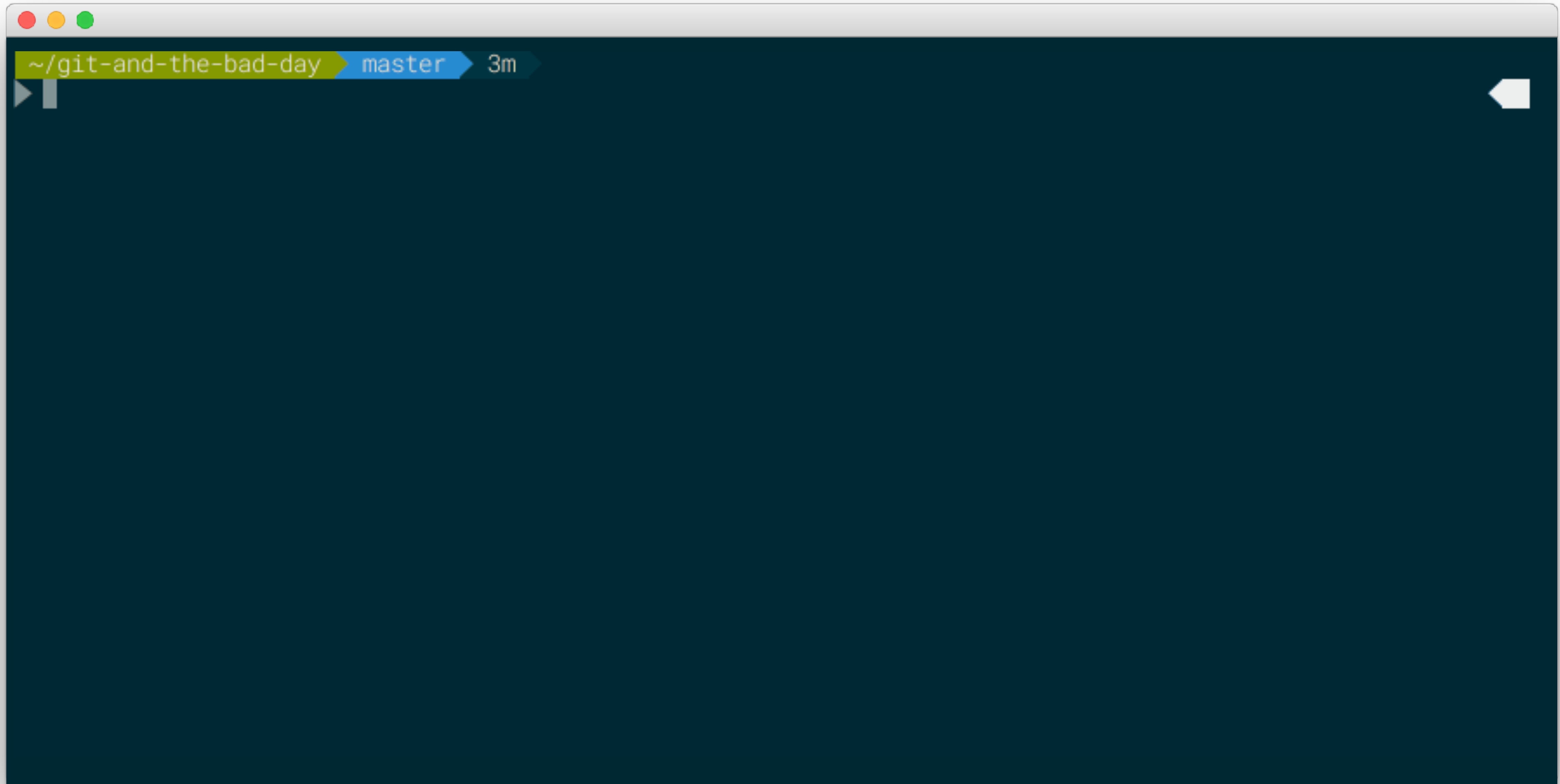
~/git-and-the-bad-day master ● 0m





Remove a
binary file with
`git filter-branch`



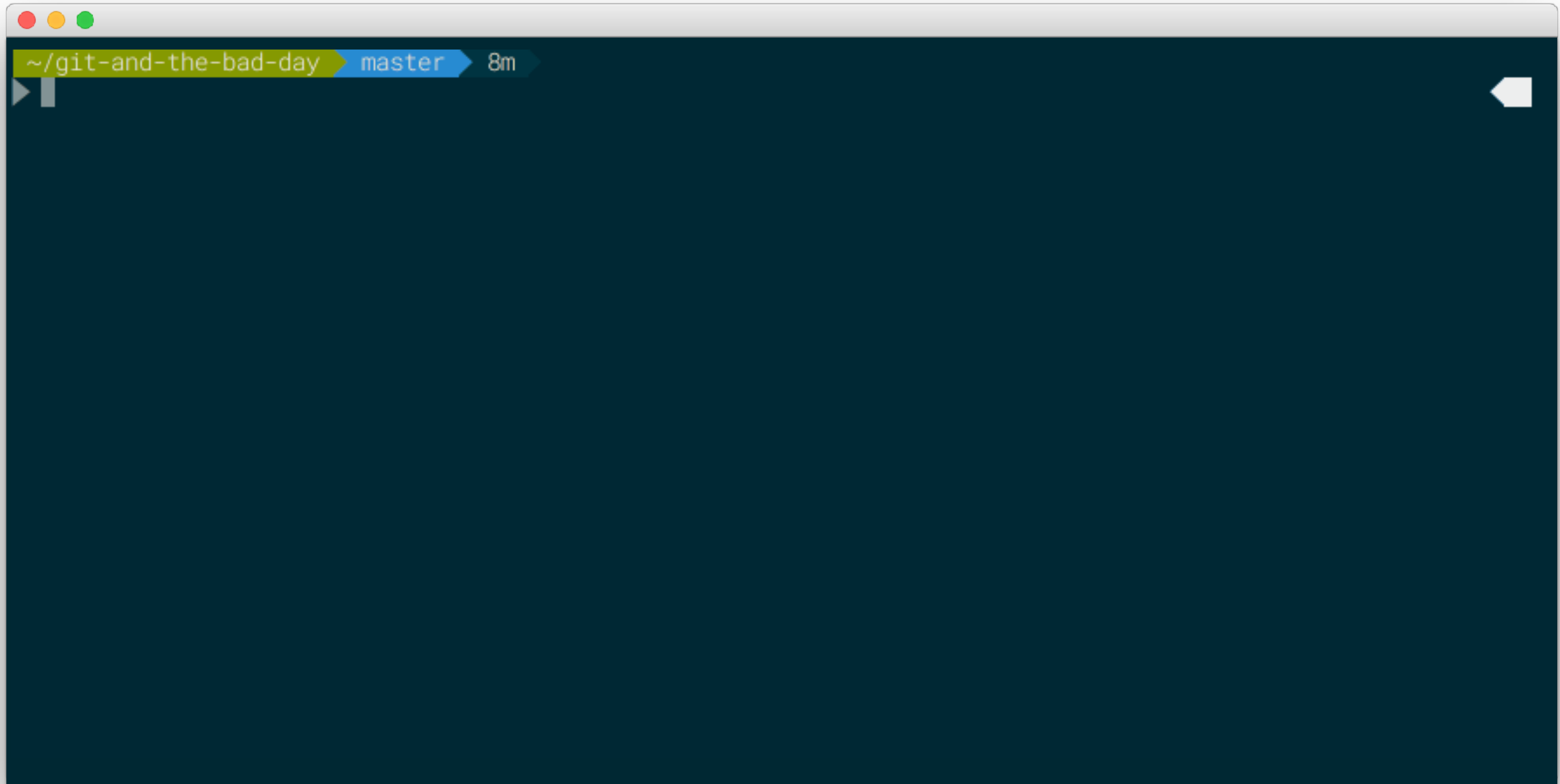


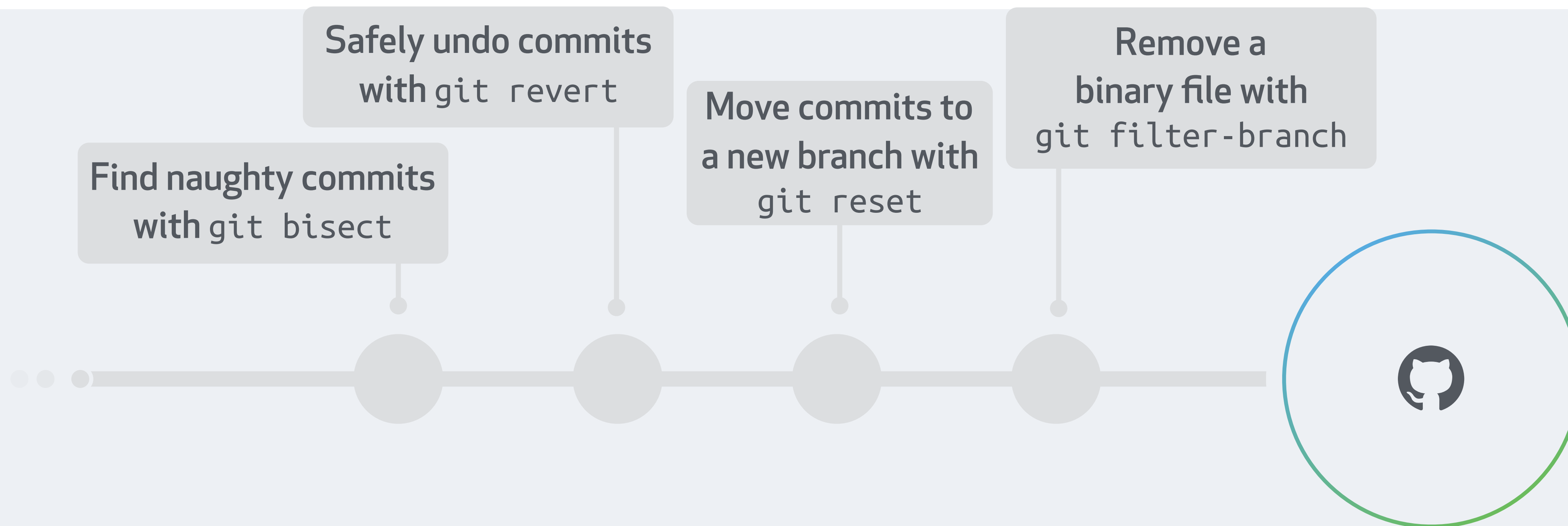

```
new file mode 100644
index 0000000..9569944
Binary files /dev/null and b/git-and-the-bad-day.key differ
diff --git a/git-and-the-bad-day.md b/git-and-the-bad-day.md
index 5fcb71d..c9b6c9f 100644
--- a/git-and-the-bad-day.md
+++ b/git-and-the-bad-day.md
@@ -46,6 +46,8 @@ But before bed I decided to give it another shot. I was successful with `git fil

I pushed my changes up to GitHub, and my push was rejected. It's because I altered commit history so I had to

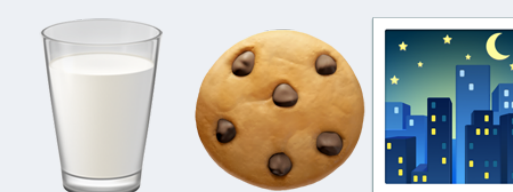
+I could've also used BFG.
+
This was, indeed, a terrible, horrible, no good, very bad day.

My friends say that some days are like that. Even without Git! But Git did indeed stick with me through every
~/git-and-the-bad-day master 3m
▶ git diff e3427...903d ◀
diff --git a/git-and-the-bad-day.key b/git-and-the-bad-day.key
new file mode 100644
index 0000000..9569944
Binary files /dev/null and b/git-and-the-bad-day.key differ
~/git-and-the-bad-day master 3m
▶ ◀
```





Let Hector read you a story



Git and the Terrible, Horrible, No Good, Very Bad Day

Illustrations by Nelson Torres





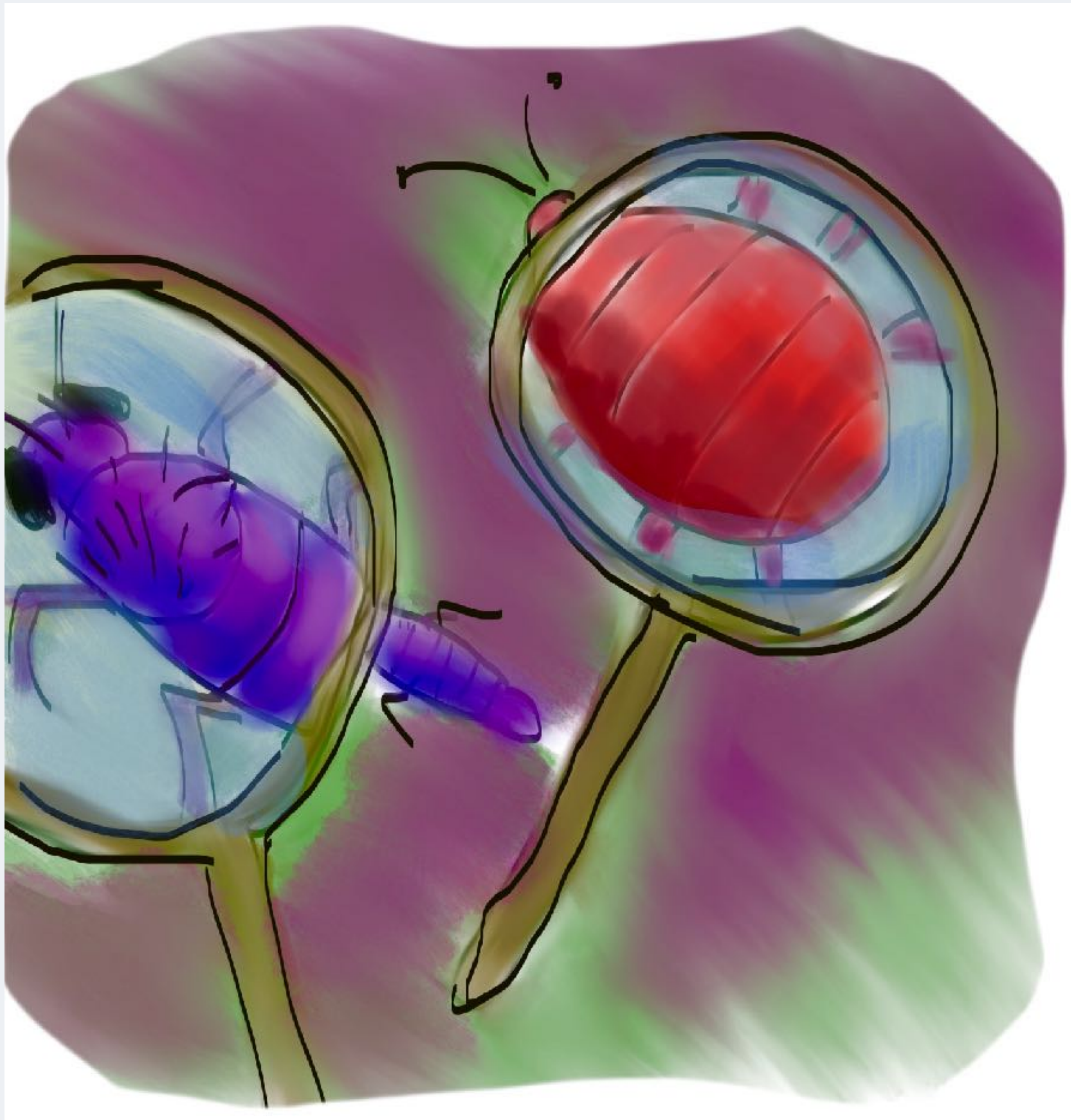
I did some late-night committing.

And I thought everything was fine.

And when I got out of bed, I spilled my morning coffee and by mistake I started a software update on my machine which bricked it for most of the morning.

When I finally logged on I realized I committed part of this story directly to master, which is horrible practice. So I could tell it was going to be a terrible, horrible, no good, very bad day.



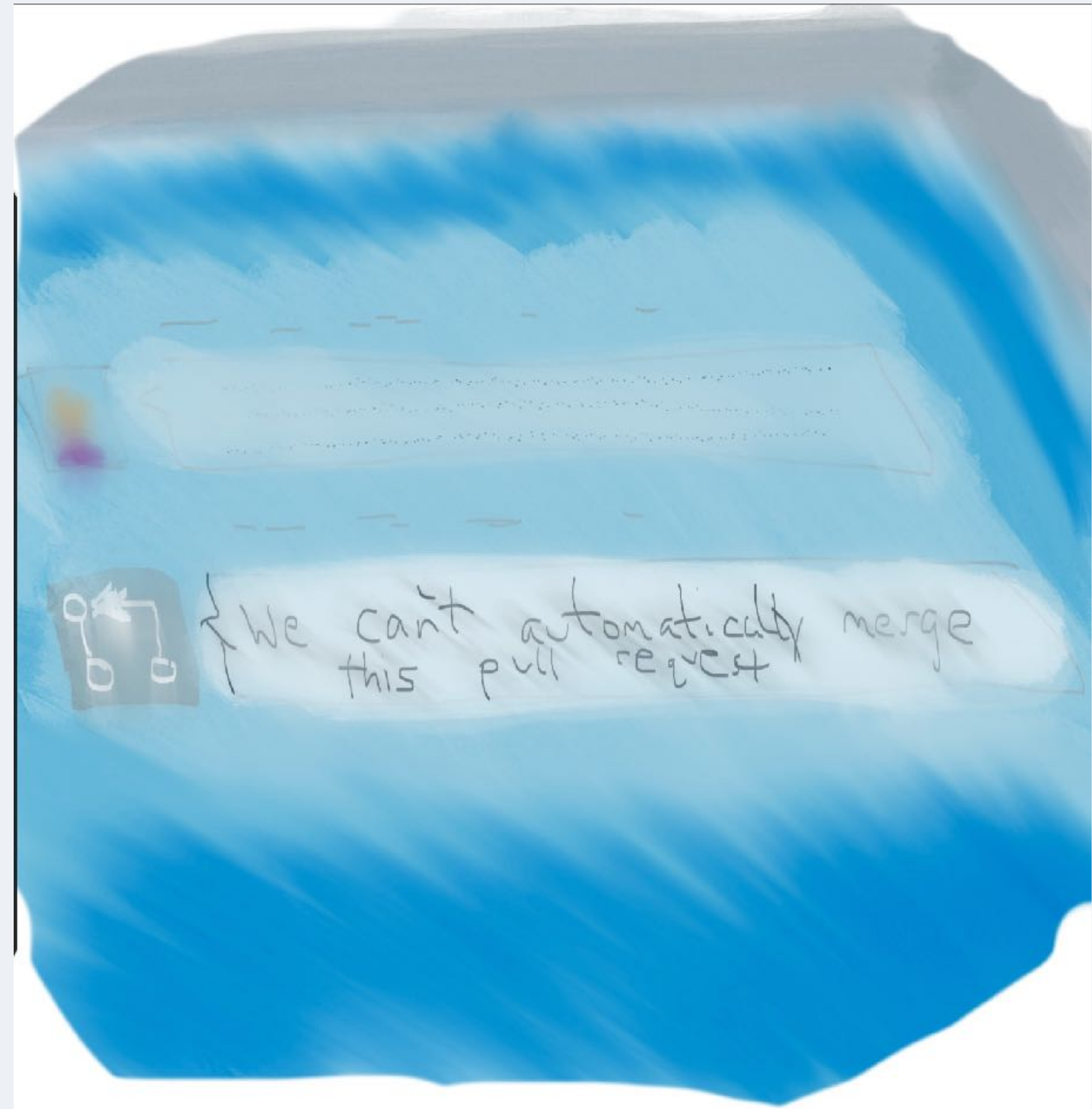


So I decided to do some `git bisect` because on any other day it would probably help me find which commit broke my story. Besides, my friends Matt and Eric always found their bugs with `git bisect`.

My most recent commit was obviously bad.

And I don't even remember the last time I looked at my story, so I figured only the first commit was safe.

It took a couple of tries, but soon enough I found the commit that contained the offending line. But when I reverted the commit, I got a merge conflict!



When Matt used `bisect` he reverted
a missing semicolon,
a misspelled variable name,
and a pretty alarming lack of comments.

When Eric used `bisect` he reverted
some missing environment variables,
some badly indented blocks,
and part of his grocery list that got accidentally pasted into his code.

When I used `bisect`, I just got a gnarly merge conflict.

After breakfast, Briana got to teach her favorite group of students.



Cynthia spent some time watching trains with her son.

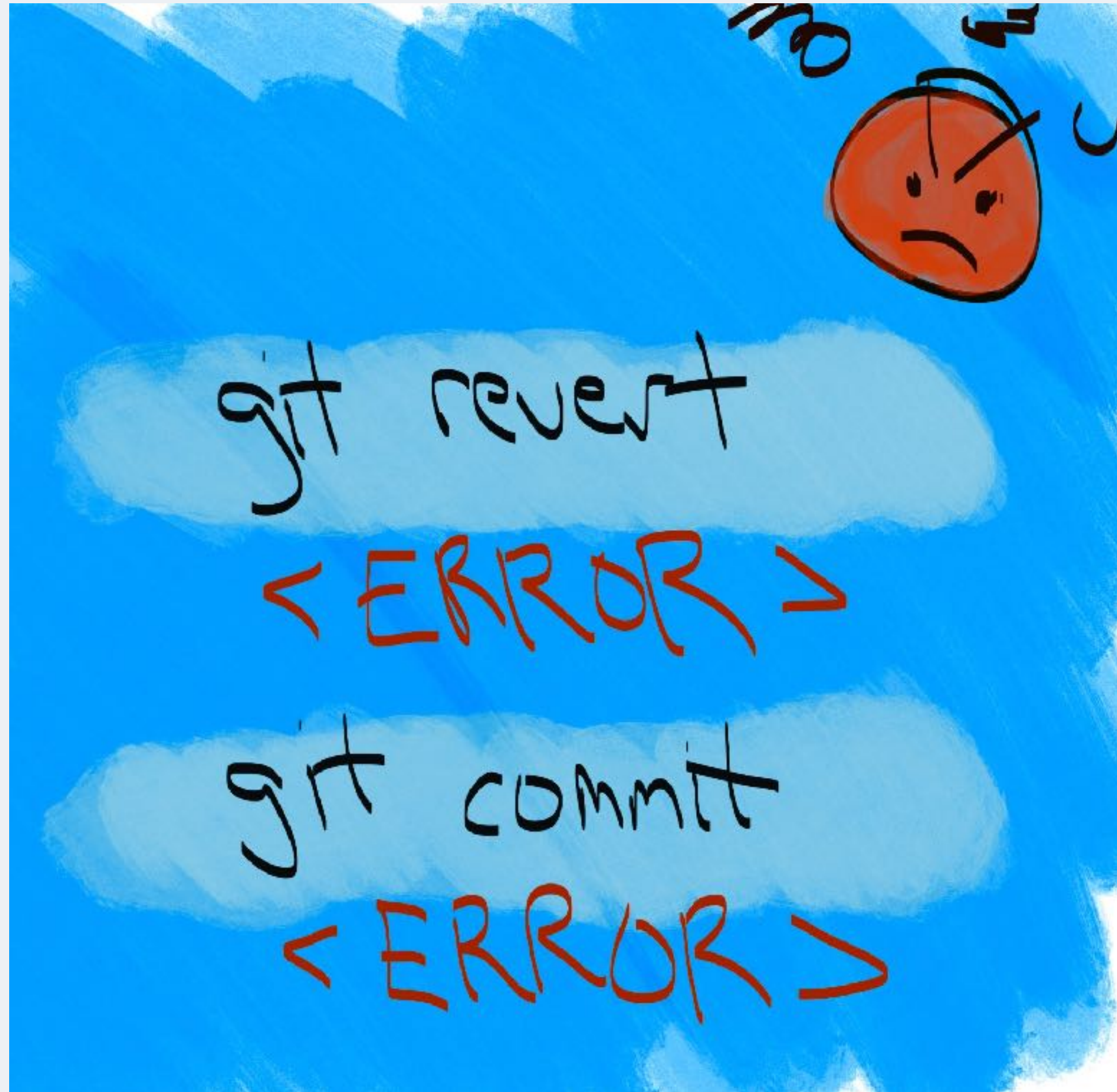




I was stuck trying to figure out how to resolve my merge conflict.

I could tell it was going to be a terrible, horrible, no good, very bad day.

Before lunch, I treated the conflict as any other conflict. I opened the file, and removed the conflict markers. Then I wondered: how do I finish this revert?



``git revert`` didn't work

``git commit`` didn't work

I wish I could revert this day.



Once the revert was finished, I realized I should've done it in its own branch. I wanted to `git reset` this terrible, horrible, no good, very bad day!

That's what it was, because that afternoon I actually tried `git reset`. My commit was made to the wrong branch, so I figured I'd create a new branch, then reset to the the previous commit, and when I switched to the new branch my revert was there!

I then realized a `git cherry-pick` would've done the same thing for me.

I wished I could cherry-pick myself out of this day.



Later that night, I realized I uploaded my Keynote presentation binary to the repo. What a waste of space!

I figured I could just revert the commit, but that just means that any previous commits would still have the file. This was a terrible, horrible, no good, very bad day.

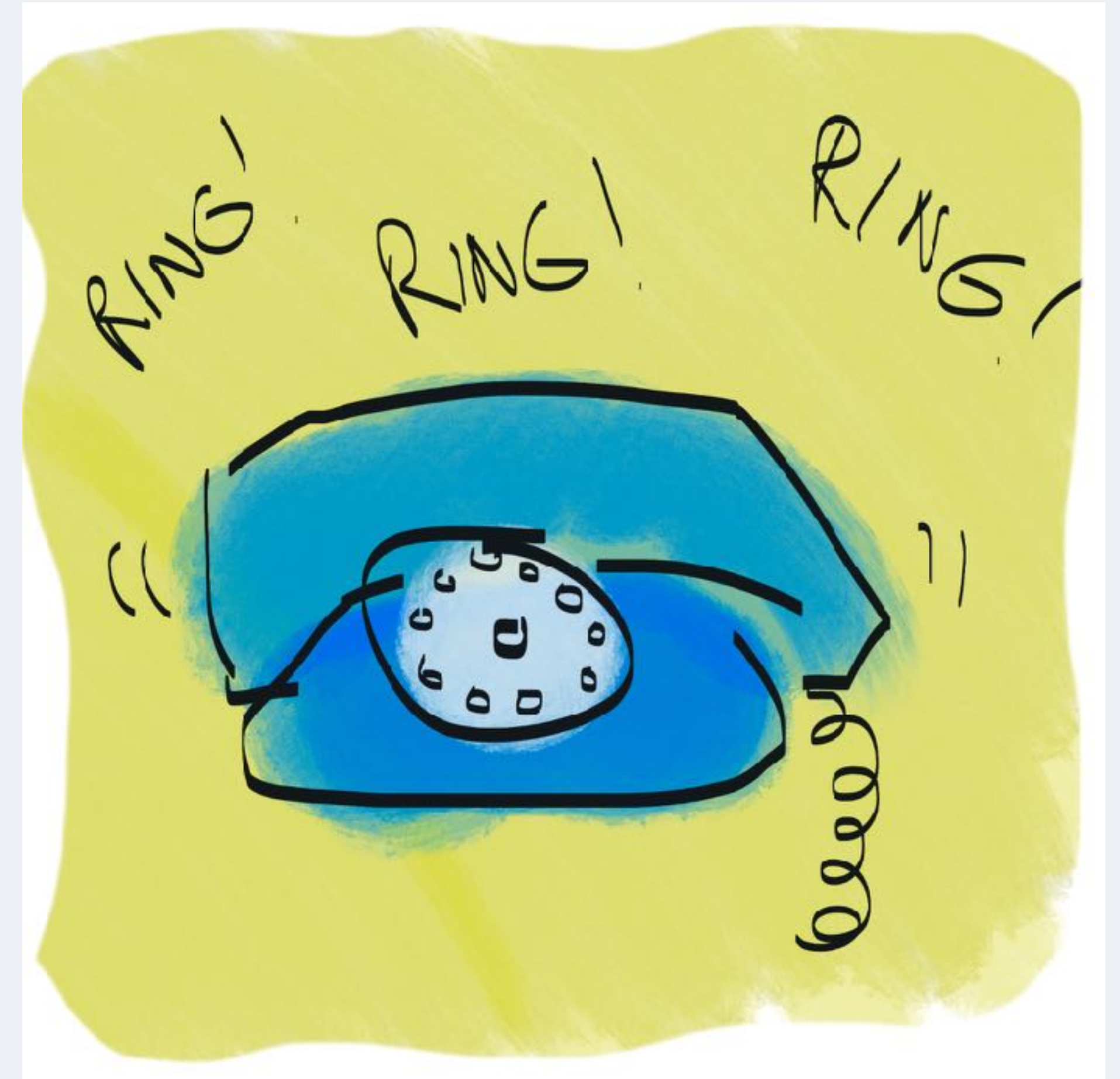
Because each commit is a complete snapshot of the codebase, and I had no idea when I committed the binary, I knew this would be a messy clean-up. But ``git filter-branch`` should help me get out of this mess.



But before bed I was successful with `git filter-branch` I was happy with myself, but then I realized a local backup of what I cleaned up was still in my local git directory. So I had to clean that up, too.

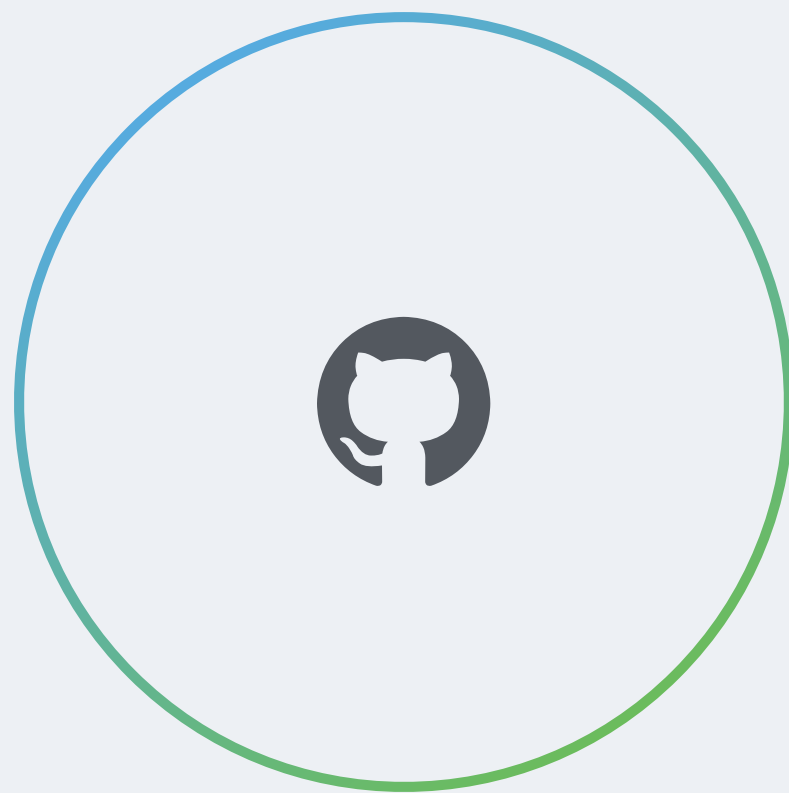
I pushed my changes up to GitHub, and my push was rejected. It's because I altered commit history so I had to force push. Cynthia then told me that GitHub caches commits that may still have my credentials. So I had to contact Support and they removed the cached commits from GitHub.

This was, indeed, a terrible, horrible, no good, very bad day.





My friends say that some days are like that. Even without Git! But Git did indeed stick with me through every sticky situation, and its integrity was never compromised so, hey, Git helped me navigate my terrible, horrible, no good, very bad day.



Shoutouts to:

- Katie Saylor-Miller, ohshitgit.com
- @lorenallensmith, GitHub

@hectorsector
Trainer, GitHub