A Real 'git bisect' Session

Learn about git bisect by implementing an example.

We'll cover the following

Now that you understand a bisect session, you're going to embed this idea by running through an actual `git bisect` session. If the previous lesson felt a bit abstract, then this might help make the lesson more vivid.

Requirements

What you're going to do is create a Git repository with one file (`projectfile`). In this file, you are going to add a line for each commit. The first line will be `1`, the second `2`, and so on until the hundredth commit, which adds the line `100`.

In this scenario, the "bug" is the line `63`, but you don't know that yet. All you know is that you can tell if the bug is in the code with this shell command:

```
if grep -w 63 projectfile
> then
>   echo BAD
> else
>   echo GOOD
```

It outputs `BAD` if the “bug” is found and `GOOD` if it is absent. Obviously, this is an arbitrary example. Your reproduction code for a bug in a real situation might be far more complicated and/or time-consuming.

Implementation

Type this in:

```
mkdir -p lgthw_bisect
cd lgthw_bisect
git init
touch projectfile
git add projectfile
cat > test.sh << END
> if grep 63 projectfile
> then
>   echo BAD
> else
>   echo GOOD
> fi
> END
chmod +x test.sh
git add test.sh
```

Terminal 1

Terminal

So far everything is normal. You’ve created and added a file to a new Git repository and added a test script to determine if the bug is present.

Adding 100 commits

The next bit of code creates 100 lines with an incrementing number in each line, each line committed in turn:

```
for ((i=1;i<=100;i++))
> do
> echo $i >> projectfile
> git commit -am "A$i"
> done
```

Terminal 1

Terminal

Now, check if the history of the repository is as you expect:

```
git log
```

Finding the bug

You're going to start your bisect session to find the bug. Follow the output carefully:

```
git bisect start
./test.sh
git bisect bad
git status
```

Terminal 1

Terminal

By marking the bisect session as bad you have told it where to end its search for the bug. Can you see where it tells you that in the `git status` output?

Now type this in and try and figure out what's going on while you do so.

You will want to throw in some `git status` and `git log` commands as you go through to orient yourself:

```
git checkout HEAD~99
git status
./test.sh
git bisect good
git log
```

Terminal 1

Terminal

The first thing to point out is the checkout command. You should know what `HEAD` is by now. Can you figure out what the `~99` does?

That's right! It refers to the 99 commits previous to this one. You will return to this later.

Where are you now in the history?

```
git status
./test.sh
git bisect good
git log
./test.sh
git bisect bad
```

Terminal 1

Terminal

At this point, you should be getting the idea.

Continue bisecting commits

I'm going to tell you now to keep going: mark commits as good or bad until you get to something that looks like this (Can you spot the difference before you continue?):

Input

```
37  git bisect bad
```

Sample output

```
37cfc7e407bb20392f067f07899c1c0bf8b94560 is the first bad commit
commit 37cfc7e407bb20392f067f07899c1c0bf8b94560
Author: Educative Learner <learner@educative.io>
Date:   Thu Jan 21 08:17:37 2021 +0000

    A51

:100644 100644 96cc558853a03c5d901661af837fceb7a81f58f6 1c5a36f5d2e2f5
293d62440acef04fbbd683e447 Mprojectfile
```

If all was as expected, then the bisect is complete and has reported that `37cfc7e407bb20392f067f07899c1c0bf8b94560` was the first bad commit (the commit identifier may differ for you).

Difference between reported commit and its parent

You can get the diff between this reported commit ID and its parent by using the `^` operator with diff:

```
37  git diff 37cfc7e407bb20392f067f07899c1c0bf8b94560^ \37cfc7e407bb20
392f067f07899c1c0bf8b94560
```

```
diff --git a/projectfile b/projectfile
index dccaef9..86c8a76 100644
--- a/projectfile
+++ b/projectfile
@@ -63,3 +63,4 @@
 63
 64
 65
+66
```

The `^` operator is a handy shortcut that allows you to quickly refer to a commit's parent.

← Back
Bisecting: The Session at a High Level

Next →
The Difference Between ~ and ^

✓ Mark as Completed

Report an Issue

Ask a Question
(https://discuss.educative.io/tag/a-real-git-bisect-session__git-bisect__learn-git-the-hard-way)