Lab: Finding the bug in your project

The slowest, most tedious way of finding a bad git commit is something we've all done before. You checkout some old commit, make sure the broken code isn't there, then checkout a slightly newer commit, check again, and repeat over and over until you find the flawed commit.

Using `git bisect` is a much better way. It's like a little wizard that walks you through recent commits, asks you if they are *good* or *bad*, and narrows down the broken commit. In this blog post, I encourage you to create a fresh git repository and walk through each step. Hopefully, you'll gain an intrinsic understanding of `git bisect` by the end of the exercise.

Setting up your lab environment

1. **Create an experimental git repository.**

   Copy and paste the following lines:

   ```
   mkdir git_bisect_tests
   cd git_bisect_tests
   git init
   ```

2. **Create a history of commits to play with.**

   Copy and paste the following lines. This will create a commit history for a file called 'test.txt' that contains the opening lyrics to the childrens song "Row Row Row Your Boat".

   ```
   echo row > test.txt
   git add -A && git commit -m "Adding first row"
   echo row >> test.txt
   git add -A && git commit -m "Adding second row"
   echo row >> test.txt
   git add -A && git commit -m "Adding third row"
   echo your >> test.txt
   git add -A && git commit -m "Adding the word 'your'"
   echo boat >> test.txt
   git add -A && git commit -m "Adding the word 'boat'"
   echo gently >> test.txt
   git add -A && git commit -m "Adding the word 'gently'"
   sed -i -e 's/boat/car/g' test.txt
   git add -A && git commit -m "Changing the word 'boat' to 'car'"
   echo down >> test.txt
   git add -A && git commit -m "Adding the word 'down'"
   echo the >> test.txt
   git add -A && git commit -m "Adding the word 'the'"
   echo stream >> test.txt
   git add -A && git commit -m "Adding the word 'stream'"
   ```

   Output:

```
>> echo row > test.txt
>> git add -A && git commit -m "Adding first row"
[master afc3484] Adding first row
 1 files changed, 0 insertions(+), 8 deletions(-)
>> echo row >> test.txt
>> git add -A && git commit -m "Adding second row"
[master 142e028] Adding second row
 1 files changed, 1 insertions(+), 0 deletions(-)
>> echo row >> test.txt
>> git add -A && git commit -m "Adding third row"
[master bdce17b] Adding third row
 1 files changed, 1 insertions(+), 0 deletions(-)
>> echo your >> test.txt
>> git add -A && git commit -m "Adding the word 'your'"
[master c3a8a2f] Adding the word 'your'
 1 files changed, 1 insertions(+), 0 deletions(-)
>> echo boat >> test.txt
>> git add -A && git commit -m "Adding the word 'boat'"
[master b190f74] Adding the word 'boat'
 1 files changed, 1 insertions(+), 0 deletions(-)
>> echo gently >> test.txt
>> git add -A && git commit -m "Adding the word 'gently'"
[master 72e834a] Adding the word 'gently'
 1 files changed, 1 insertions(+), 0 deletions(-)
>> sed -i -e 's/boat/car/g' test.txt
>> git add -A && git commit -m "Changing the word 'boat' to 'car'"
[master 8d60e7a] Changing the word 'boat' to 'car'
 1 files changed, 1 insertions(+), 1 deletions(-)
>> echo down >> test.txt
>> git add -A && git commit -m "Adding the word 'down'"
[master e9a03bf] Adding the word 'down'
 1 files changed, 1 insertions(+), 0 deletions(-)
>> echo the >> test.txt
>> git add -A && git commit -m "Adding the word 'the'"
[master 7496ee1] Adding the word 'the'
 1 files changed, 1 insertions(+), 0 deletions(-)
>> echo stream >> test.txt
>> git add -A && git commit -m "Adding the word 'stream'"
[master c8ba38a] Adding the word 'stream'
 1 files changed, 1 insertions(+), 0 deletions(-)
```

3. **Find the bug in 'test.txt' file.**

   Look at the contents of test.txt. There is a glaring error: the word 'car' is where the word 'boat' should be. That happened during our commit history -- at some point, the commit of the word 'boat' was overwritten by the commit for the word 'car'.

```
>> cat test.txt
row
row
row
your
car
gently
down
the
stream
```

The Experiment

1. **Use git log to find a *good* commit, and a *bad* commit.**

   First off, let's try to find a commit that still had the word 'boat', and not the word 'car'. Check out your git log:

```
>> git log
commit b428150d2cd1378fc1e7ddddfdd68eb37e77afc4
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding the word 'stream'

commit 498009c1870f5ee1b2399f37f74a29bbf7df668f
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding the word 'the'

commit 8edf7f3e47a78dfe43a77f4dd801271fd380fa02
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding the word 'down'

commit 409e8b6793570543f467be7eec15a4878537e344
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Changing the word 'boat' to 'car'

commit 73d31feceaaf5f72301489617a2a9eb85dafe03f
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding the word 'gently'

commit ddf0fcc5478c90cb20be331d641b4a73e8fee575
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding the word 'boat'

commit 53b0d4a4ea72e31b1fb27b53563ccbe8d6e04f0a
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding the word 'your'

commit 5dee23a7fc41ced3e3eac30a912f8d2ac527b82f
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding third row

commit be2bd3aca4653113524e21e08bdaa077f4c3d7bc
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding second row

commit f317696e909856429a411a2bdbe036a9a44e74a2
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding first row
```

We know that the newest commit in the log is bad, so we will classify this as our *bad* commit. For the sake of simplicity, let's say our *good* commit is the first commit we made for the word 'boat':

```
commit ddf0fcc5478c90cb20be331d641b4a73e8fee575
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Adding the word 'boat'
```

2. **With a *good* commit, and a *bad* commit, we're ready for the process of elimination -- `git bisect`**

   Git bisect is like a wizard: it guides you step by step through a process of elimination until you find the commit that broke your code. Once you type `git bisect start`, you have started the wizard and it won't end until you type `git bisect reset`.

3. **Start up the git bisect wizard**

```
>> git bisect start
```

4. **Let the git bisect wizard know of a *good* commit.**

   In this case, it was the 'Adding the word boat' commit we picked above.

```
>> git bisect good ddf0fcc54
```

5. **Let the git bisect wizard know of a *bad* commit.**

   In this case, let's use the very last commit in the repository, since we know that had the word 'car' in it.

```
>> git bisect bad b428150d2cd1
Bisecting: 2 revisions left to test after this (roughly 1 step)
[409e8b6793570543f467be7eec15a4878537e344] Changing the word 'boat' to 'car'
```

6. **Look at the contents of test.txt**

   When we typed `git bisect bad` in this step, git bisect checked out an old commit for us -- the commit halfway between the latest *bad* commit and the known *good* commit. This is how bisect works -- it cuts the commit history down in halves until it finds the original bad commit.

```
>> cat test.txt
row
row
row
your
car
gently
```

7. **The word 'car' is still there = BAD!**

   This commit is still bad, so we tell bisect about how bad it is: `git bisect bad`

```
>> git bisect bad
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[73d31feceaaf5f72301489617a2a9eb85dafe03f] Adding the word 'gently'
```

8. **Look at the contents of test.txt again.**

   Once again, git bisect checked out a new commit halfway between the latest *bad* commit and the known *good* commit.

```
>> cat test.txt
row
row
row
your
boat
gently
```

9. **The word 'boat' is there = GOOD!**

   Now we see the word 'boat', so this is a *good* commit. Let's let git bisect know how good it is: `git bisect good`

```
>> git bisect good
409e8b6793570543f467be7eec15a4878537e344 is the first bad commit
commit 409e8b6793570543f467be7eec15a4878537e344
Author: Tony Rasmussen <tony@metaltoad.com>
Date:   Tue Apr 17 10:58:29 2012 -0700

    Changing the word 'boat' to 'car'
```

10. **ALL DONE - The bad commit has been found!**

    Plain as day, git reports back to you: "I found the first bad commit"

    You are now free to investigate the problem, notify the author, write patches, and take care of business.

11. **To end your git bisect wizard, simply type git bisect reset**

```
>> git bisect reset
Previous HEAD position was 73d31fe... Adding the word 'gently'
Switched to branch 'master'
```

With `git bisect`, We can narrow down broken code within a few seconds on repositories with a dozen developers adding several dozen commits per hour.