How to figure out which commit causes a bug

Something worked. And now it doesn't. This is called a regression.

This happens. Most commits fix/improve things, but every once in a while, something breaks. Sometimes, by fixing a bug, we introduce another. Tiki has so many features that it's difficult to test everything. Lots of eyeballs and good bug reporting is essential

You'd like to report the bug and inform the right person. But how to figure out who introduced the problem and when?

If you are using Git

The following actions must be done on a working copy (usually local) of your Tiki after all commits and conflicts are solved (check with git status if everything is up-to-date)

You can use `git bisect` to find out what commit is the cause of the regression.
The `git bisect` command requires just two information to start:

1. A revision where things were definitely good.
2. A revision where the bug is present.

Now, `git bisect` will help to track the bug somewhere between the "bad" and "good" commits. It will splits the range of commits between "good" and "bad" in half - and checks out a commit in the middle.

For each iteration you will have to test and confirm if the bug is still there or gone.

In practice (sample)

Go in your terminal, shell, etc. and enter the following where *10df1cb84df250eaae72cef31aca79a32dffef87* is a commit where the bug is present; bad and *165cc018e5dd74485b7e6aa13b53361db1fb3596* is a commit where the bug is not present; good and . Note you can use the number or the commit SHA.

start Git Bisect

```
git bisect start
```

declare the commit that has the bug

```
git bisect bad 10df1cb84df250eaae72cef31aca79a32dffef87 //If it is the last version you can use HEAD
```

declare the commit that doesn't have the bug

```
git bisect good 165cc018e5dd74485b7e6aa13b53361db1fb3596
```

The output from the last command should look like this:

```
Bisecting: 3 revisions left to test after this (roughly 2 steps)
[e0375ad20652b1c5974a634809b269d68d6902a5] [FIX] bs4: Use btn-secondary for tracker item preview buttons, and btn-link for cancel/close like wiki page edit. (missed from 390f4082, thanks @Jyhem
```

Now you need to go back to you browser and check if, using this version, the bug is there or not.
If it is there continue tracking using

continue the search

```
git bisect bad
```

If the bug is not there continue tracing using

continue the search

```
git bisect good
```

It will repeat the process (splitting commits in half and on) until you are able to successfully singled out the bad commit!

Once it is done don't forget to reset your working copy

stop Git Bisect and reset

```
git bisect reset
```

The output should be something like that:

```
Previous HEAD position was e0375ad206 [FIX] bs4: Use btn-secondary for tracker item preview buttons, and btn-link for cancel/close like wiki page edit. (missed from 390f4082, thanks @Jyhem)
Switched to branch 'bernardsfez_23x' Your branch is up to date with 'origin/bernardsfez_23x'.
```

Related

- https://mozilla.github.io/mozregression/
- http://www.selenic.com/mercurial/hg.1.html#bisect
- https://launchpad.net/bzr-bisect
- http://www.monotone.ca/docs/Bisecting.html
- http://www.fossil-scm.org/xfer/help/bisect
- Subversion
- https://quality.mozilla.org/docs/bugzilla/guide-to-triaging-bugs-for-firefox/finding-a-regression-window/

alias

- Bisect
- SVN-Bisect
- SVN Bisect
- Git Bisect
- How to figure out what revision number causes a bug