Navigating through the git history like a boss

[If you're not into reading why I find git history useful, you may jump right into the command-line coolness!]

I love reading git logs! First, because it reveals the people and processes behind software. Software is not just a bunch of code that works (or more commonly, doesn't work...) - it's also a bunch of people crafting something together for a long time. Not only adding features and fixing bugs, but they also refactor, do dirty tasks and ugly workarounds, explaining their motives in the commit messages.

A second, more concrete reason - quick access to git history helps me make better software. It helps me find out when and how exactly a bug was introduced, learn whom to talk to about a certain change or code, or simply find a desired code snippet in a huge repository - even if it was already deleted. Am I stating the obvious here? I guess so! but I do believe that sometimes this tool is underestimated and underused.

Of course, the better we know git's secrets, the more effective use of git history we could make. I'll try to share here some useful tricks I've collected in the recent years... I bet you know some; I believe you may still find some of them useful.

Alright, enough human communication - let the command-line part begin:

1. Find commits by their commit message - --grep

Impress your boss at how quickly you find this commit!

```
git log -p --grep "nasty bug"

# Feeling lucky?
git show :/'nasty bug'
```

2. Search by commit content - -S

This one will show you only commits that contain the provided string in the commit content itself. Super useful!

```
git log -p -S 'Date.new'
```

3. Search by author

You know who's responsible for this change you're looking for, so why not use this to ease the search?

```
git log -p --author='Linus Torvalds'

# Can even use parts of the name or email:
git log -p --author=orvalds
git log -p --author=gmail.com
```

4. Find by date

```
git log -p --since=2013-01-01 --until=2012-02-01
```

5. By path!

```
git log -p -- path/to/file1 path/to/directory ...

# Add --follow to follow changes through file renames:
git log -p --follow -- path/to/file1
```

6. Search ALL branches

Lost something?

```
git log -S 'populate_database' --all
```

7. Putting (some of) it all together

I actually find such commands useful in practice.

```
git log -p --author=linus --since=2012-06-01 --until=2013-01-01 --grep fix -S 'Date.new'
```

8. A few useful revision-range tricks

a. What's new in my branch compared to remote?

```
git log -p origin/master..master

# But I like the following syntax better; it's exactly the same but self-explanatory:
# all commits of master branch that are not contained in origin/master branch.
git log -p master ^origin/master

# Also possible to put multiple arguments
git log -p master ^origin/master ^my_github/master
```

(Note: I've noticed that zsh needs the '^' character to be escaped!)

b. 3-dots operator: symmetric difference

The less-widely-known-but-very-useful 3-dots operator, shows any commit that is in any of the branches but not in both:

```
git log -p mybranch...master

# --cherry-pick is often useful to avoid seeing "duplicate commits".
# common when cherry-picking, where commits identical content get
# different hashes.
git log -p --cherry-pick mybranch...master
```

Read more!

```
git help log
git help revisions
git help blame
```

This entry was posted in development and tagged FOSS, git on March 28, 2013 [http://www.held.org.il/blog/2013/03/navigating-through-the-git-history-like-a-boss/] by Oren Held.

---

5 thoughts on “Navigating through the git history like a boss”

fboule
March 29, 2013 at 02:33

Awesome. I have been. Working with git for quite a while now (in conjunction with svn with the git-svn module for empirical reasons) and I am still learning it. Like Alice in Wonderland. Thanks for the tips!

Stephen Ball
March 29, 2013 at 18:14

You could also mention the incredibly amazing and useful `git bisect` for hunting down when something broke, or changed, or started working, or started slowing down, whatever.

saurabh kumar

March 29, 2013 at 18:27

Made my day. Definitely something here to take back home. Thanx for this.

Jaco Pretorius

March 29, 2013 at 20:29

Very helpful, thank you!

Matthew Astley

April 3, 2013 at 13:16

I've also found that a knowledge of history is useful in maintaining code - if you can get hold of it efficiently. Some programmers I have worked with (and respected) ignore the history, treating version control purely as a ratchet. Corollary: if you want good history, especially where code moves between repositories, you may have to do it yourself / demonstrate why it's useful.

The commands are useful, thanks. I also recommend getting `gitk --all` to anyone learning about Git near me. Shame it doesn't (at 1.7.9.5) have an option like `git log --cherry-mark`.
