Difference between author and committer in Git?

Asked 6 years, 2 months ago Active 7 months ago Viewed 66k times

209

45

I am trying to make a commit like

```
git commit --author="John Doe <john@doe.com>" -m "<the usual commit message>"
```

where John Doe is some user in whose name I want to make the commit.

It appears all right in `git log`. However, when I do a `gitk`, **the author name is correct, but the committer name is picked from my global git config settings** (and is thus set to my name/email).

Questions

1. What is the difference between the two (committer vs author)?
2. Should I be setting the committer as well to the other user?
3. If yes, how?

git git-config gitk

edited May 16 '14 at 21:47
user456814

asked Sep 11 '13 at 20:38
mu 無
49.1k 25 109 144

4 jasonnoble.org/2009/04/github-set-authorcommitter.html here is a short description. – René Höhle Sep 11 '13 at 21:01

Possible duplicate of What is the difference between author and committer in Git? – Marcin Armatys Feb 20 '17 at 14:41

Git committer is placed in .gitconfig file. If you --author is same as .gitconfig name, you get only author in commit message. If they are differs, you get both. – poGUIst Nov 14 at 17:42

3 Answers

195

The original poster asks:

> What is the difference between the two (Committer vs author)?

The author is the person who originally wrote the code. The committer, on the other hand, is assumed to be the person who committed the code on behalf of the original author. This is important in Git because Git allows you to rewrite history, or apply patches on behalf of another person. The **FREE** online **Pro Git** book explains it like this:

> You may be wondering what the difference is between *author* and *committer*. The *author* is the person who originally wrote the patch, whereas the *committer* is the person who last applied the patch. So, if you send in a patch to a project and one of the core members applies the patch, both of you get credit — you as the author and the core member as the committer.

The original poster asks:

> Should I be setting the committer as well to the other user?

No, if you want to be honest, you should not be setting the committer to the author, unless the author and the committer are indeed the same person.

edited Sep 13 '13 at 0:53 answered Sep 12 '13 at 3:34
user456814

I'm still confused about this. I had this happen, and in my case, as far as I'm aware, no patch or history rewrite ever occurred (unless some git commands create and apply patches, opaquely, "under the hood"). Are those really the only 2 ways for something like this to happen? – cowlinator Feb 14 at 23:19

Also, calling the author the "person who wrote the code" doesn't make sense. How would git know who wrote it? When you set `git config user` and then `git add` and `git commit` , then git would know who added and who committed, but it still would not know who wrote it. – cowlinator Feb 14 at 23:53

1 @cowlinator It doesn't know who wrote the code. That's why you have to tell it, if it's not you. Keep in mind that the previous distributed version control system before git was invented was sending ~~Linus~~ the project maintainer emails with patches to apply. This functionality is there so ~~Linus~~ the maintainer can apply your patch while still crediting you for it in an 'official' way, rather than just ad-hoc in the commit message. – Nic Hartley Feb 27 at

84

Mailing list + `git format-patch` + `git apply` can generate author != committer

In projects like the Linux kernel where patches are:

- generated by `git format-patch`
- sent by email, either by copy pasting, or more commonly with `git send-email`
- applied by another person with either `git apply` or `git am` : How to use git am to apply patches from email messages?

generating a single new commit with different author and committer:

- the author is who wrote the patch
- the committer is who is a project maintainer, and who merged the patch

See for example this randomly selected patch and the corresponding commit:

- https://lkml.org/lkml/2018/1/25/568
- https://github.com/torvalds/linux/commit/5beda7d54eafece4c974cfa9fbb9f60fb18fd20a

Git web interfaces like GitHub and GitLab may or may not generate author != committer

Since Git(Hub|Lab) hold both the upstream and the fork repositories on a the same machine, they can automatically do anything that you can do locally as well, including either of:

- Create a merge commit.

  Does not generate author != committer.

  Keeps the SHA or the new commit intact, and creates a new commit:

  ```
  * Merge commit (committer == author == project maintainer)
  |\
  | * Feature commit (committer == author == contributor)
  |/
  * Old master (random committer and author)
  ```

  Historically, this was the first available method on GitHub.

This produces two commits per pull request, and keeps a fork in the git history.

- rebase on top of `master`

  GitHub also hacks the commits to set committer == whoever pressed the merge button. This is not mandatory, and not even done by default locally by `git rebase`, but it gives accountability to the project maintainer.

  The git tree now looks like:

  ```
  * Feature commit (committer == maintainer, author == contributor)
  |
  * Old master (random committer and author)
  ```

  which is exactly like that of the `git apply` email patches.

On GitHub currently:

- you choose the method when merging via the dropdown on the merge button
- methods can be enabled or disabled on the repo settings by the owner

https://help.github.com/articles/about-merge-methods-on-github/

How to set the committer of a new commit?

The best I could find was using the environment variables to override the committer:

```
GIT_COMMITTER_NAME='a' GIT_COMMITTER_EMAIL='a' git commit --author 'a <a>'
```

How to get the committer and commit date of a given commit?

Only author data shows by default on `git log`.

To see the committer date you can either:

- format the log specifically for that:

  ```
  git log --pretty='%cn %cd' -n1 HEAD
  ```

- use the `fuller` predefined format:

  ```
  git log --format=fuller
  ```

  See also: How to configure 'git log' to show 'commit date'

- go low level and show the entire commit data:

  ```
  git cat-file -p HEAD
  ```

How to set the committer date of a new commit?

`git commit --date` only sets the author date: for the committer date the best I could find was with the environment variable:

```
GIT_COMMITTER_DATE='2000-01-01T00:00:00+0000' git commit --date='2000-01-
01T00:00:00+0000'
```

See also: What is the difference between author and committer in Git?

How Git stores author vs committer internally?

See: What is the file format of a git commit object?

Basically, the commit is a text file, and it contains two line separated fields:

```
author {author_name} <{author_email}> {author_date_seconds} {author_date_timezone}
committer {committer_name} <{committer_email}> {committer_date_seconds}
{committer_date_timezone}
```

This makes it clear that both are two completely independent data entries in the commit object.

edited Oct 12 '18 at 12:06

answered Apr 16 '14 at 11:29

Ciro Santilli 新疆改造中心法轮功六四事件

180k 39 681 552

1 @adelphus on Git 2.5, it does work if you set both `GIT_{COMMITTER,AUTHOR}_EMAIL` – Ciro Santilli 新疆改造中心法轮功六四事件 Aug 15 '15 at 19:51

1

@Ciro Santilli 新疆改造中心 六四事件 法轮功 proposed to use

> ```
> GIT_COMMITTER_NAME='a' GIT_COMMITTER_EMAIL='a' git commit --author 'a <a>'
> ```

To avoid repeating the name and email, you can reuse them

```
GIT_COMMITTER_NAME='a'; GIT_COMMITTER_EMAIL='a'; git commit --author
"$GIT_COMMITTER_NAME <$GIT_COMMITTER_EMAIL>"
```

which first sets the variables in separate commands, then uses them for the `git commit` call (note the double parentheses).

answered Apr 5 at 12:41
serv-inc
20.7k 6 99 102