

How do I get the Git commit count?

Asked 10 years, 8 months ago Active 4 months ago Viewed 256k times



719



225

I'd like to get the number of commits of my Git repository, a bit like SVN revision numbers.

The goal is to use it as a unique, incrementing build number.

I currently do like that, on Unix/Cygwin/msysGit:

```
git log --pretty=format:'' | wc -l
```

But I feel it's a bit of a hack.

Is there a better way to do that? It would be cool if I actually didn't need `wc` or even Git, so it could work on a bare Windows. Just read a file or a directory structure...

git

build-process

revision

edited Nov 12 '17 at 0:57



Peter Mortensen

24.3k 19 89 118

asked Mar 24 '09 at 13:38



Splo

7,846 4 16 14

1 You may find interesting answers here: [what is the git equivalent for revision number?](#) – Sebastien Varrette Jul 26 '12 at 10:34

181 git rev-list HEAD --count [git rev-list](#) – Jake Berger Feb 5 '13 at 18:46 ✎

13 @jberger: I think your comment should be converted to an answer. – utapyngo Mar 8 '13 at 6:29 ✎

@utapyngo: given the 13 other answers, I knew it'd be buried. I've [posted it here](#) then. – Jake Berger Mar 9 '13 at 19:46

@jberger, this answer doesn't work for git1.7.0. – Vorac Mar 31 '16 at 8:14

20 Answers



To get a commit count for a revision (HEAD , master , a commit hash):

1089

```
git rev-list --count <revision>
```



To get the commit count across all branches:



```
git rev-list --all --count
```



I recommend against using this for build identifier, but if you must, it's probably best to use the count for the branch you're building against. That way the same revision will always have the same number. If you use the count for all branches, activity on other branches could change the number.

edited Nov 17 '15 at 19:37

community wiki

4 revs

Benjamin Atkin

26 `git shortlog | grep -E '^[]+\w+' | wc -l` if you want to get total number and `git shortlog | grep -E '^[^]'` if you want to get commits number for every contributor. – [skalee](#) May 24 '11 at 18:58

2 Thanks for pointing out `wc -l`. Minimalism FTW. I incorporated it into my answer. – [Benjamin Atkin](#) May 25 '11 at 21:01

15 This solution is both hacky (similar to the `git log --pretty=format:'' | wc -l` approach given in the original question) and incorrect: you can see this by inverting the match (`git shortlog | grep -Ev '^[]+\w+'`) and seeing that e.g. commits with no message (i.e., "<none>") are not counted. Using `git rev-list HEAD --count` is both more succinct and more accurate. – [ctrueden](#) Mar 1 '13 at 22:55

16 @BenAtkin: My apologies; it was not my intent to be offensive, merely factual. Point taken about the date of the response. At the time, your solution may very well have been the best available one. But I stand by my statement that `git rev-list HEAD --count` is a better solution now. – [ctrueden](#) Mar 23 '13 at 16:36

3 Added an answer as well and works also with old versions: `git log --oneline | wc -l` – [Jimmy Kane](#) Feb 6 '14 at 17:33



`git shortlog` is one way.

150




answered Apr 23 '10 at 17:28

community wiki

Rayne

5 Ty. This worked for me when counting commits in a range; `git shortlog sha1..sha2` – [RJFalconer](#) Apr 13 '11 at 11:07

- 1 Yep, the first line of git shortlog has the number of commits in it. Problem solved. – [Robert Massaioli](#) Jul 31 '11 at 14:23
- 4 The number of commits is grouped by committer, not so good. Can count lines in git shortlog, but this doesn't work over ssh without a terminal for some reason (pager?). The asker's original solution is the best! git log --pretty=format:" | wc -l – [Sam Watkins](#) Feb 4 '12 at 7:15
- 4 However, I would suggest `git rev-list HEAD --count` rather than the original approach given in the OP. In my tests, `git log --pretty=format:" | wc -l` is off by one. – [ctrueden](#) Mar 1 '13 at 22:59 
- 3 @ctrueden `git log --oneline | wc -l` isn't off by one (OS X 10.8.5). – [Andy Stewart](#) Mar 24 '15 at 9:37

git rev-list HEAD --count

106

[git rev-list](#)

`git rev-list <commit>` : List commits that are reachable by following the parent links from the given commit (in this case, *HEAD*).

`--count` : Print a number stating how many commits would have been listed, and suppress all other output.

answered Mar 9 '13 at 19:43



[Jake Berger](#)

4,479 1 24 21

This command returns count of commits grouped by committers:

87

git shortlog -s

Output:

```
14 John lennon
9  Janis Joplin
```

You may want to know that the `-s` argument is the contraction form of `--summary`.

edited Jul 16 at 11:07



[Valerio Bozz](#)


463 4 15

answered Nov 14 '11 at 7:52



[Alex Pliutau](#)

18k 25 92 134

- 9 `git shortlog` by itself does not address the original question of *total* number of commits (not grouped by author). Use `git rev-list HEAD --count` instead. – [ctrueden](#) Mar 1 '13 at 22:58 
- 4 Awesome! You can sort it by `| sort -n` too – [Mohsen](#) Jul 24 '13 at 20:00

If you're looking for a unique and still quite readable identifier for commits, [git describe](#) might be just the thing for you.

53

edited Nov 8 '11 at 9:10

answered Mar 24 '09 at 14:08

[svrist](#)

6,469

7

35

61

[Bombe](#)

68.5k

20

110

119

- 2 That could work and would be more easy to use than a custom-made algo. +1 – [VonC](#) Mar 24 '09 at 14:23
- 2 I didn't know `git describe`. This little number between the tag name and the sha1 is just what I was looking for. Thank you. – [Spl0](#) Mar 25 '09 at 0:33
- 2 Take a look at `GIT-VERSION-GEN` script and how it is used in git repository, and similar script in Linux kernel sources (and how they are used in `Makefile`). – [Jakub Narębski](#) Mar 27 '09 at 3:30
- This gives unique, but not INCREMENTAL id. Doesn't work for me. However Ben Atkin's answer offers commit count, which in practice should be incremental. Aaron Digulla's answer is more sure, but requires also more work. – [JOM](#) Jun 30 '11 at 5:25
- 2 Yes, that's because the *concept* of an *incremental* ID does not make any sense with distributed version control systems. – [Bombe](#) Jun 30 '11 at 6:27

You are not the first one to think about a ["revision number" in Git](#), but `'wc'` is quite dangerous, since commit can be erased or squashed, and the history revisited.

34

The "revision number" was especially important for Subversion since it [was needed in case of merge](#) (SVN1.5 and 1.6 have improved on that front).

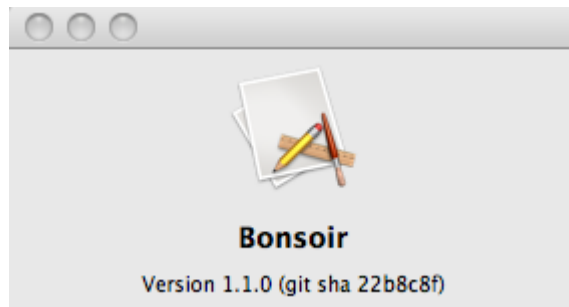
You could end up with a pre-commit hook which would include a revision number in the comment, with an algorithm *not involving* looking up the *all* history of a branch to determine the correct number.

[Bazaar](#) actually came up with [such an algorithm](#), and it may be a good starting point for what you want to do.

(As [Bombe's answer](#) points out, Git has actually an algorithm of its own, based on the latest tag, plus the number of commits, plus a bit of an SHA-1 key). You should see (and upvote) his answer if it works for you.

To illustrate [Aaron's idea](#), you can also [append the Git commit hash into an application's "info" file](#) you are distributing with your application.

That way, the about box would look like:



The applicative number is part of the commit, but the 'application's "info" file' is generated during the packaging process, effectively linking an *applicative* build number to a technical revision *id*.

edited Nov 12 '17 at 0:59



Peter Mortensen

24.3k 19 89 118

answered Mar 24 '09 at 13:54



VonC

914k 336 2982 3585

2 I've updated my script to work with Xcode 3. You can pick up an up to date version from gist.github.com/208825. – Abizern Oct 29 '09 at 22:56

U can just use :

31

```
git shortlog -s -n
```

Result :

```
827 user one
15 user two
2 Gest
```

answered Dec 26 '17 at 9:09

demenvil



A simple way is:

21

```
git log --oneline | wc -l
```

oneline ensures that.

edited Nov 12 '17 at 1:06



Peter Mortensen

24.3k 19 89 118

answered Feb 6 '14 at 15:38



Jimmy Kane

11.9k 7 60 85

1 'wc' is not recognized as an internal or external command, operable program or batch file. – [user815693](#) Jul 23 '15 at 14:35

Well what system are you using? Is it a UNIX one ?/ – [Jimmy Kane](#) Jul 23 '15 at 15:31

1 This seems faster too if you have thousands of commits. All other commands take too much time. – [Danny Coulombe](#) Jun 13 '17 at 20:49

To get it into a variable, the easiest way is:

21

```
export GIT_REV_COUNT=`git rev-list --all --count`
```

edited Aug 10 '16 at 23:11

answered Jul 15 '11 at 21:52



John Gietzen

43.7k 28 132 181

5 Indeed, `git rev-list` is the correct tool to use, not `git log` like the other say. – [Nayuki](#) Nov 12 '11 at 2:40

1 To count the number of commits in the lineage to reach HEAD: `git rev-list --first-parent | wc -l` – [200_success](#) Jan 7 '13 at 22:54

You don't need `wc -l` just use the `--count` switch: `git rev-list --all --count .` – [slm](#) Aug 3 '16 at 2:59

Thanks @slm, I've updated the answer. Although, I suspect the original answer is older than the `--count` switch itself. – [John Gietzen](#) Aug 10 '16 at 23:12

@JohnGietzen - oh yeah I figured that 8-), was just adding this detail to help. – [slm](#) Aug 11 '16 at 1:03

Git shortlog is one way to get the commit details:

15

```
git shortlog -s -n
```

This will give the number of commits followed by the author name. The -s option removes all the commit messages for each commit that the author made. Remove the same option if you would like to see the commit messages also. The -n option is used for sorting the entire list. Hope this helps.

answered Nov 25 '12 at 7:59

[Sri Murthy Upadhyayula](#)

7,920 1 12 20

2 `git shortlog` by itself does not address the original question of *total* number of commits (not grouped by author). Use `git rev-list HEAD --count` instead. – [ctrueden](#) Mar 1 '13 at 23:01

`git rev-parse --short HEAD`

8

answered Nov 24 '09 at 9:24

community wiki
[makuchaku](#)

1 Produces a hash, whereas a number was requested. – [Jesse Glick](#) Sep 17 '14 at 19:57

There's a nice helper script that the Git folks use to help generate a useful version number based on Git describe. I show the script and explain it in my answer to [How would you include the current commit id in a Git project's files?](#).

7

edited Nov 12 '17 at 1:03

[Peter Mortensen](#)

24.3k 19 89 118

answered Mar 24 '09 at 15:22

[Pat Notz](#)

166k 28 84 91

If you're just using one branch, such as master, I think this would work great:

4 `git rev-list --full-history --all | wc -l`

This will only output a number. You can alias it to something like

```
git revno
```

to make things really convenient. To do so, edit your `.git/config` file and add this in:

```
[alias]
  revno = "!git rev-list --full-history --all | wc -l"
```

This will not work on Windows. I do not know the equivalent of "wc" for that OS, but writing a Python script to do the counting for you would be a multi-platform solution.

edited Nov 12 '17 at 1:05



Peter Mortensen

24.3k 19 89 118

answered Oct 26 '12 at 14:50



NuclearPeon

4,012 1 31 42

3 Generate a number during the build and write it to a file. Whenever you make a release, commit that file with the comment "Build 147" (or whatever the build number currently is). Don't commit the file during normal development. This way, you can easily map between build numbers and versions in Git.

answered Mar 24 '09 at 13:59



Aaron Digulla

275k 89 504 731

If two distributed developers did this wouldn't their build numbers collide/intersect periodically? What if they both did a build between the same revs of a shared repo, or perhaps collision would only occur if either had changes not committed to the shared repo. Not sure. – [hobs](#) Jul 17 '11 at 21:32

Sure but the conflict tells you what to do: Just talk to the other guy or always use higher number. Remember: A number can't magically cure a broken build process. It's just a *reminder* or *hint* that you need to check something. – [Aaron Digulla](#) Jul 18 '11 at 12:47

1 Ahh, yes, the magic buildno.txt file is committed along with the rest. Good approach for a small team, or a large team that avoids parallel builds. Only place I can think of that it might not work as well is for a large team using a scripted language (python) that doesn't need a build process (to assign a

single person to do building). – [hobs](#) Jul 19 '11 at 22:08



In our company, we moved from SVN to Git. Lack of revision numbers was a big problem!

3

Do `git svn clone`, and then tag the last SVN commit by its SVN revision number:



```
export hr=`git svn find-rev HEAD`  
git tag "$hr" -f HEAD
```

Then you can get the revision number with help of

```
git describe --tags --long
```

This command gives something like:

```
7603-3-g7f4610d
```

Means: The last tag is 7603 - it's the SVN revision. 3 - is count of commits from it. We need to add them.

So, the revision number can be counted by this script:

```
expr $(git describe --tags --long | cut -d '-' -f 1) + $(git describe --tags --long |  
cut -d '-' -f 2)
```

edited Nov 12 '17 at 1:08



[Peter Mortensen](#)

24.3k 19 89 118

answered Jun 11 '14 at 11:05



[Matvey](#)

31 1



You can try

1

```
git log --oneline | wc -l
```



or to list all the commits done by the people contributing in the repository

```
git shortlog -s
```

answered Nov 15 '16 at 10:24

user7161360

```
git config --global alias.count 'rev-list --all --count'
```

1

If you add this to your config, you can just reference the command;

```
git count
```

answered Sep 11 '17 at 9:46

[Robert Pounder](#)**987** 1 9 27

The one I used to use was:

1

```
git log | grep "^commit" | wc -l
```

Simple but it worked.

answered Jul 31 '11 at 14:24

[Robert Massaioli](#)**12.3k** 6 43 69

4 it takes one commit message line beginning with "commit" to break count. For example: "corrected mistakes and broken tests that I accidentally pushed in last\ncommit" – [Paweł Polewicz](#) Jun 6 '12 at 9:38

Using Bash syntax,

1

```
$(git rev-list --count HEAD)
```

looks fine for purely linear history. If you also want to sometimes have “numbers” from branches (based off `master`), consider:

```
$(git rev-list --count $(git merge-base master HEAD)).$(git rev-list --count ^master HEAD)
```

When run from a checkout of `master` , you get simply `1234.0` or the like. When run from a checkout of a branch you will get something like `1234.13` , if there have been 13 commits made on that branch. Obviously this is useful only insofar as you are basing at most one branch off a given `master` revision.

`--first-parent` could be added to the micro number to suppress some commits arising only from merging other branches, though it is probably unnecessary.

answered Sep 17 '14 at 20:51



[Jesse Glick](#)

19.5k 6 67 93

Use git shortlog just like this

0

```
git shortlog -sn
```

Or create an alias (for ZSH based terminal)

```
# show contributors by commits
alias gcall="git shortlog -sn"
```

answered Nov 5 '15 at 14:01



[Ahmad Awais](#)

15.6k 3 51 45