How to list branches that contain a given commit?

993

How can I query git to find out which branches contain a given commit? gitk will usually list the branches, unless there are too many, in which case it just says "many (38)" or something like that. I need to know the full list, or at least whether certain branches contain the commit.

git version-control

244

edited Dec 20 '12 at 14:49
ThiefMaster
247k 63 490 570

asked Sep 14 '09 at 4:03

Andrew Arnott
59.1k 23 117 157

4 See also: How to list all tags that contain a commit?. – Andrew Marshall Jan 18 '13 at 17:51

Related question for an equivalent commit per comments: stackoverflow.com/questions/16304574/… – UpAndAdam Feb 25 '15 at 16:38

2 Answers

1346

From the git-branch manual page:

```
git branch --contains <commit>
```

> Only list branches which contain the specified commit (HEAD if not specified). Implies `--list` .

```
git branch -r --contains <commit>
```

See also this git ready article.

> The `--contains` tag will figure out if a certain commit has been brought in yet into your branch. Perhaps you've got a commit SHA from a patch you thought you had applied, or you just want to check if commit for your favorite open source project that reduces memory usage by 75% is in yet.

```
$ git log -1 tests
commit d590f2ac0635ec0053c4a7377bd929943d475297
Author: Nick Quaranto <nick@quaran.to>
Date:   Wed Apr 1 20:38:59 2009 -0400

    Green all around, finally.

$ git branch --contains d590f2
  tests
* master
```

Note: **if the commit is on a remote tracking branch, add the `-a` option**.
(as MichielB comments below)

```
git branch -a --contains <commit>
```

MatrixFrog comments that it only shows which branches contain that *exact* commit.
If you want to know which branches contain an "equivalent" commit (i.e. which branches have cherry-picked that commit) that's `git cherry` :

> Because `git cherry` **compares the changeset rather than the commit id (sha1)**, you can use `git cherry` to find out if a commit you made locally has been applied `<upstream>` under a different commit id.
> For example, this will happen if you're feeding patches `<upstream>` via email rather than pushing or pulling commits directly.

```
          __*__*__*__*__> <upstream>
         /
```

edited Jul 7 '17 at 22:24

Gerold Broser
8,383 4 22 63

answered Sep 14 '09 at 4:08

VonC
878k 318 2841 3407

3 `tests` and `master - master` is the current branch, therefore the asterisk. – blueyed Mar 25 '11 at 13:31

49 This only shows which branches contain *that exact commit*. If you want to know which branches contain an "equivalent" commit (i.e. which branches have cherry-picked that commit) that's `git cherry` : "Because git cherry compares the changeset rather than the commit id (sha1), you can use git cherry to find out if a commit you made locally has been applied <upstream> under a different commit id. For example, this will happen if you're feeding patches <upstream> via email rather than pushing or pulling commits directly." kernel.org/pub/software/scm/git/docs/git-cherry.html – MatrixFrog Apr 14 '11 at 1:04

57 Add a `-a` parameter to also check remote branches. – Raman Jan 25 '12 at 23:45

25 You can also do `git tag --contains <commit>` . See Searching for all tags that contain a commit?. – Andrew Marshall Jan 18 '13 at 17:42

4 For the `git cherry` part @UpAndAdam asked the question here: stackoverflow.com/questions/16304574/…, alas, the question has not (yet) been answered. – adeelx Aug 6 '14 at 19:24

17

You may run:

```
git log <SHA1>..HEAD --ancestry-path --merges
```

From comment of last commit in the output you may find *original branch* name

Example:

```
       c---e---g--- feature
      /         \
-a---b---d---f---h---j--- master

git log e..master --ancestry-path --merges

commit h
```

edited Oct 29 '16 at 13:58

answered Oct 29 '16 at 13:53

Eugen Konkov
7,166 3 43 68

6 Nice! I used `git log <SHA1>..master --ancestry-path --merges --oneline | tail -n1` to get this in one line – James EJ Aug 11 '17 at 19:59

1 If you would like to use pure git command, you could use: `git log <SHA1>..master --ancestry-path --merges --oneline -1` – Bartosz Mar 23 '18 at 12:03