Detail change after Git pull

Asked 15 years, 3 months ago Modified 6 years, 11 months ago

Viewed 53k times

138

After a Git pull, its output gives a summary on the change amount.

How can I see each or some of the files detailed changes?

Okay, here is my question to Jefromi:

1. How do I know if I was pulling to master? All I did is "git pull".
2. What does master point to and what is the difference between master and HEAD, the two default heads of Git?
3. How do I see the detailed change in a specific file?
4. How do I see the change in the summary output by the last `git pull` again?
5. What's difference between `git diff` and `git whatchanged`?

git **git-pull**

Share

edited Dec 31, 2017 at 19:31

Improve this question

Follow

Peter Mortensen
31.6k ● 22 ● 109 ● 133

asked Sep 1, 2009 at 15:03

Tim
99k ● 145 ● 395 ● 637

4 Okay, this repeated adding of new questions via edits is not exactly the way the system is intended to be used. You can also very easily answer a lot of your questions by looking at man pages or just by trying things. For example, `git diff` clearly outputs a diff, while `git whatchanged` clearly outputs a list of commit information, each containing a list of what files changed. – Cascabel Sep 1, 2009 at 15:36

Probably because of your low rep. – T.E.D. Sep 1, 2009 at 15:47

@T.E.D. It only takes 50 rep to leave comments, and 15 to upvote. – Cascabel Sep 1, 2009 at 15:52

On my laptop with Ubuntu, it sometimes work sometimes don't. I temporarily found another computer with Centos and am making this comment. On both computers I am using Firefox. – Tim Sep 1, 2009 at 15:59

Very odd. You might want to head over to meta and see if it's a known problem/report it. – Cascabel Sep 1, 2009 at 16:02

4 Answers

Sorted by: Highest score (default)

217

Suppose you're pulling to master. You can refer to the previous position of `master` by `master@{1}` (or even `master@{10.minutes.ago}`; see the specifying revisions




section of the git-rev-parse man page), so that you can do things like

- See all of the changes: `git diff master@{1} master`
- See the changes to a given file: `git diff master@{1} master <file>`
- See all the changes within a given directory: `git diff master@{1} master <dir>`
- See the summary of changes again: `git diff --stat master@{1} master`

As for your question of "how do I know if I'm on master"... well, using branches is an important part of the Git workflow. You should always be aware of what branch you're on - if you pulled changes, you want to pull them to the right branch! You can see a list of all branches, with an asterisk by the currently checked-out one, with the command `git branch`. The current branch name is also printed along with the output of `git status`. I highly recommend skimming the man pages of commands to use - it's a great way to slowly pick up some knowledge.

And your last question: `HEAD` is the name for the currently checked out branch. You can indeed use `HEAD` and `HEAD@{1}` in this context as well, but it's a bit more robust to use the branches, since if you go and check out another branch. `HEAD` is now that second branch, and `HEAD@{1}` is now `master` - not what you want!

To save having to ask a lot of little questions like this, you should probably have a look at a Git tutorial. There are a

million on the web, for example:

- The Pro Git book
- Git Magic
- and the 4.5 million hits on Google for "Git tutorial"

Share Improve this answer

Follow

edited Dec 31, 2017 at 19:35

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 1, 2009 at 15:09

Cascabel

496k ● 75 ● 381 ● 319

5 this is better than my solution :) – Christian Oudard Sep 1, 2009 at 15:14

2 I know this is old, but... It should be the other way around: `git diff master@{1} master`, otherwise the change is shown "backwards", i.e. insertions become deletions etc. – ultracrepidarian Mar 11, 2014 at 16:46

2 `git diff master@{1} master` didn't work for me instead `git diff master~1 master` did the job for me. – unrealsoul007 Aug 24, 2015 at 8:49

5 @unrealsoul007 Then your situation was different. master~1 is the parent commit of the one master's currently pointing to; you're going to see the diff for just that commit. master@{1} is the previous commit master pointed to; if for example you just pulled, that'd be the position of master before the pull as described here. If it didn't do that, then you've probably done something else to master since you pulled. Try `git reflog master` to understand what. – Cascabel Aug 24, 2015 at 17:26

@Jefromi `fatal: ambiguous argument 'firstDesign@': unknown revision or path not in the working tree.` I keep getting this error. Although git reflog firstDesign has this output. – unrealsoul007 Aug 24, 2015 at 21:54

54

Say you do a git pull like this:

```
$ git pull
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 4), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
From git@dev.example.com:reponame
   a407564..9f52bed  branchname   ->
origin/branchname
Updating a407564..9f52bed
Fast forward
 .../folder/filename          |  209 ++++++++++-----
 .../folder2/filename2        |  120 ++++++++++--
-------
 2 files changed, 210 insertions(+), 119
deletions(-)
```

You can see the diff of what changed by using the revision numbers:

```
$ git diff a407564..9f52bed
```

Share Improve this answer

Follow

answered Sep 1, 2009 at 15:12

Christian Oudard

49.8k ● 25 ● 70 ● 70

5 And you can get the summary using " `git diff --stat a407564..9f52bed` " or for just a summary " `git diff --summary a407564..9f52bed` " – Jakub Narębski Sep 1, 2009 at 17:13

14 For newer versions of git, git pull no longer outputs the list of files that were changed. To get that, you need to do `git pull --stat' – user10 Feb 14, 2011 at 14:49

6

> **1.** How do I know if I was pulling to master? All I did is "git pull".

The command itself works like this:

```
git pull [options] [<repository> [<refspec>…]]
```

and per default refers to the current branch. You can check your branches by using

```
git branch -a
```

This will list your local and remote branches like for e.g so (Added a `---` as divider between local and remote to make it more clear)

```
*master
foo
bar
baz
---
```

```
origin/HEAD -> origin/master
origin/deploy
origin/foo
origin/master
origin/bar
remote2/foo
remote2/baz
```

When you then take a look at one remote repo, you will see what you are referring to:

```
git remote show origin
```

will list like the following:

```
* remote origin
  Fetch URL:
ssh://git@git.example.com:12345/username/somerepo.gi
  Push  URL:
ssh://git@git.example.com:12345/username/somerepo.gi
  HEAD branch: master
  Remote branches:
    foo    tracked
    master tracked
  Local refs configured for 'git push':
    foo    pushes to foo    (up to date)
    master pushes to master (fast-forwardable)
```

So it's quite easy to be sure where to pull from and push to.

> **3.** how to see the detail change in a specific file?
>
> **4.** how to see the change in summary output by last git pull again?

The easiest and **most elegant way** (imo) is:

```
git diff --stat master@{1}..master --
dirstat=cumulative,files
```

This will give you two blocks of information about the changes in between your last pull an the current state of work. Example output (I added a `---` as divider between `--stat` and `--dirstat` output to make it more clear):

```
 mu-plugins/media_att_count.php
|  0
 mu-plugins/phpinfo.php
|  0
 mu-plugins/template_debug.php
|  0
 themes/dev/archive.php
|  0
 themes/dev/category.php
| 42 ++++++++++++++++++
 .../page_templates/foo_template.php
|  0
 themes/dev/style.css
|  0
 themes/dev/tag.php
| 44 ++++++++++++++++++++
 themes/dev/taxonomy-post_format.php
| 41 +++++++++++++++++
 themes/dev/template_parts/bar_template.php
|  0
 themes/someproject/template_wrappers/loop_foo.php
| 51 +++++++++++++++++++++
---
 11 files changed, 178 insertions(+)
  71.3% themes/dev/
  28.6% themes/someproject/template_wrappers/
 100.0% themes/
  27.2% mu-plugins/
   9.0% themes/dev/page_templates/
```

```
 9.0% themes/dev/template_parts/
63.6% themes/dev/
 9.0% themes/someproject/template_wrappers/
72.7% themes/
```

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Sep 18, 2014 at 23:43

kaiser

22.3k ● 18 ● 91 ● 111

2

This way's kind of hacky, but it'll allow you to use graphical tools like `gitk` or `gitg` or `git-gui`:

```
git pull
git reset HEAD@{1}
gitg (or gitk or whatever tool you like)
```

The answer with the most upvotes gives the best way using the git tool, but I use this method because I can then utilize tools with GUI to see the changes :P

I'd then have the extra step of doing a `git checkout .` and then doing `git pull` again so that I properly pull and merge, but I value the ability to examine differences in a GUI enough to deal with the extra two steps.

Share Improve this answer

Follow

answered Mar 5, 2014 at 22:22

Jack

5,394 ● 7 ● 36 ● 44