Meaning of Github Ahead/Behind Metrics

Asked 8 years, 3 months ago Active 6 years, 4 months ago Viewed 17k times

55

12

In plain language (hopefully with a simple example), what do the ahead/behind metrics on a Github repo's branch mean?

And what are the implications for that branch and the attention it's receiving? Is being "behind" a bad sign for a branch?

github repository branch

edited Aug 27 '12 at 14:35
CharlesB
64.8k 20 154 180

asked Jul 10 '11 at 20:23
LikeMaBell
664 2 9 21

4 Answers

69

✓

Ahead is the number of commits on this branch that do not exist on the base branch. Behind is the number of commits on the base branch that do not exist on this branch.

Ahead and behind are almost like a kind of "age" metric. The ahead number tells you roughly how much impact the branch will have on the base branch should it be merged. The behind number tells you how much work has happened on the base branch since this branch was started.

I find the behind number really useful for judging whether a branch is likely to merge cleanly. When a lot of work has happened on the base branch, it's more likely that the two branches have modified the same line(s). When behind is large, it's a sign that you should probably merge the base branch into this branch to sync up. Once you merge the base branch into this branch, behind will be 0.

answered Jul 11 '11 at 0:48
rtomayko
806 6 2

1 So does this means that if `Branch A` is ahead X commits and behind Y commits w.r.t. `Branch B`, then `Branch B` is ahead Y commits and behind X commits w.r.t. `Branch A`? Is this always true? – mljrg Oct 25 '17 at 9:58

65

If you're more of a visual type, take a look here:

```
◈ - ◈ - A - ◈ - B
     \
       ◈ - C
```

A is 2 commits behind and 0 commits ahead of B
B is 0 commits behind and 2 commits ahead of A
C is 1 commit behind and 2 commits ahead of A
C is 3 commits behind and 2 commits ahead of B

So "behind" means the other branch has commits this one doesn't, and "ahead" means this branch has commits the other does not.

answered Jul 11 '11 at 5:05
Tekkub
23.4k 2 24 20

4 Great visual explanation, helps a lot to follow what's happening. Thanks! – Gabriel Dec 6 '13 at 21:11

1 Great explanation, the thing is. on tools like source tree, you just get an AHEAD and BEHIND metric, with no reference to other branch. It just reads AHEAD... not AHEAD OF BRANCH X how do you make sense of that? – FRR Jan 24 '16 at 23:18

1 This is relatively to the current commit that you "checkout"ed on right now – Adiel Sep 28 '16 at 8:51

Way, way better explanation and visualization. Yes... I'm a visual learner, so this helped greatly. Thank you! – Vippy Nov 8 '18 at 19:12

6

The metrics like those you can see for this project describe, **compare to a branch from the repo (like `master`)**:

- the number of new commits that the GitHub repo has done compared to another branch of another repo: those are the **behind** commits: the other repo is behind compared to the current repo (see those commits).
- the number of new commits another branch of another repo has done compared to the current repo: those are the **ahead** commits: the other repo is ahead compared to the current repo (see those commits).

The technical detail is illustrated by the script "determining which repos are ahead/behind origin":
It is about checking:

- what commits are reachable from another branch, but not from the local branch: ahead

  ```
  git rev-list "$localref..$anotherref"
  ```

- what commits are reachable from the local branch, but not from the other branch: behind

  ```
  git rev-list "$anotherref..$localref"
  ```

answered Jul 10 '11 at 21:23

VonC
900k 330 2925 3517

1

On thing to note is that github's "behind" also counts merge commits. You can check the "behind" stuff with: git log mybranch1 ^mybranch2 and it should show you the same number of commits. If you have merge commits you can exclude them with --no-merges in the last command.

answered May 26 '13 at 9:48

Matjaz Muhic
2,387 2 12 32