What is the precise meaning of “ours” and “theirs” in git?

Microsoft
Create scalable apps in the cloud with PHP, Java and Node.js
Try Azure free
AdChoices

This might sound like too basic of a question, but I have searched for answers and I am more confused now than before.

What does "ours" and "theirs" mean in git when merging my branch into my other branch? Both branches are "ours".

In a merge conflict is "ours" always the upper of the two versions displayed?

Does "ours" always refer to the branch that HEAD was pointing to when the merge began? If so then why not use a clear possessive reference like "current branch's" instead of using a possessive pronoun like "ours" that is referentially ambiguous (since both branches are technically ours)?

Or just use the branch name (instead of saying "ours" just say "local master's" or such)?

The most confusing part to me is if I specify in a specific branch's .gitattributes file. Lets say in **test branch** I have the following .gitattributes file:

```
config.xml merge=ours
```

Now I checkout and point HEAD to **master** then merge in **test**. Since **master** is ours, and **test**'s .gitattributes is not checked out, will it even have an effect? If it does have an effect, since **master** is now "ours", then what will happen?

git merge

edited Jul 29 '16 at 9:11
kenorb
38.9k 14 258 254

asked Aug 29 '14 at 21:18
CommaToast
3,568 5 29 42

2 Answers

I suspect you're confused here because it's fundamentally confusing. To make things worse, the whole ours/theirs stuff switches roles (becomes backwards) when you are doing a rebase.

Ultimately, during a `git merge` , the "ours" branch refers to the branch you're merging *into*:

```
git checkout merge-into-ours
```

and the "theirs" branch refers to the (single) branch you're merging:

```
git merge from-theirs
```

and here "ours" and "theirs" makes some sense, as even though "theirs" is probably yours anyway, "theirs" is not the one you were *on* when you ran `git merge` .

While using the actual branch name might be pretty cool, it falls apart in more complex cases. For instance, instead of the above, you might do:

```
git checkout ours
git merge 1234567
```

where you're merging by raw commit-ID. Worse, you can even do this:

```
git checkout 7777777    # detach HEAD
git merge 1234567       # do a test merge
```

in which case there are *no* branch names involved!

I think it's little help here, but in fact, in `gitrevisions` syntax, you can refer to an individual path in the index by number, during a conflicted merge

```
git show :1:README
git show :2:README
git show :3:README
```

Stage #1 is the common ancestor of the files, stage #2 is the target-branch version, and stage #3 is the version you are merging from.

The reason the "ours" and "theirs" notions get swapped around during `rebase` is that rebase works by doing a series of cherry-picks, into an anonymous branch (detached HEAD mode). The target branch is the anonymous branch, and the merge-from branch is your original (pre-rebase) branch: so "--ours" means the anonymous one rebase is building while "--theirs" means "our branch being rebased".

As for the gitattributes entry: it *could* have an effect: "ours" really means "use stage #2" internally. But as you note, it's not actually in place at the time, so it *should not* have an effect here ... well, not unless you copy it into the work tree before you start.

Also, by the way, this applies to all uses of ours and theirs, but some are on a whole file level (`-s ours` for a merge strategy; `git checkout --ours` during a merge conflict) and some are on a piece-by-piece basis (`-X ours` or `-X theirs` during a `-s recursive` merge). Which probably does not help with any of the confusion.

I've never come up with a better name for these, though. And: see VonC's answer to another question, where `git mergetool` introduces yet more names for these, calling them "local" and "remote"!

edited May 23 at 12:18
Community ♦
1 1

answered Aug 29 '14 at 21:43
torek
116k 10 144 196

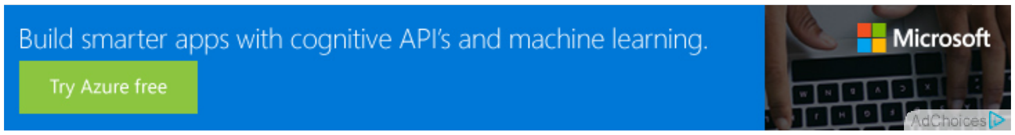
5 +1. About ours and theirs being reversed during rebase, see also: stackoverflow.com/a/2960751/6309 and stackoverflow.com/a/3052118/6309 – VonC Aug 29 '14 at 22:01

1 Two things "merge with" each other. When a merge happens both sides merge "into" each other. I feel it would be incorrect to say that one of the two sides is "not merging into anything". If there are no branch names involved (as you point out) then there are commit names involved (you could say "7777777's" and "1234567's" instead of "ours" and "theirs"). I understand what happens during a rebase and I don't find it to be confusing at all. I think "HEAD's" and "incoming's" would work better than "ours" and "theirs" because there's always a "HEAD" (whether it's detached or not). – CommaToast Aug 29 '14 at 22:18

11 I guess since the head is the seat of the mind, which is the source of identity, which is the source of self, it makes a bit more sense to think of whatever HEAD's pointing to as being "mine" ("ours", since I guess me and the HEAD makes two). If nothing more, that'll just be a good mnemonic device. – CommaToast Aug 29 '14 at 22:20

1 I like that mnemonic, will have to repeat it. Thanks! – torek Aug 29 '14 at 22:35

1 That being said, I still like to make a physical copy of the entire repo before doing anything like rebasing ... :D – CommaToast Nov 17 '15 at 6:11


Build smarter apps with cognitive API's and machine learning.
Try Azure free
Microsoft
AdChoices

The '**ours**' in Git is referring to the original working branch which has authoritative/canonical part of git history.

The '**theirs**' refers to the version that holds the work in order to be rebased (changes to be replayed onto the current branch).

This may appear to be swapped to people who are not aware that doing rebasing (e.g. `git rebase`) is actually taking your work on hold (which is *theirs*) in order to replay onto the canonical/main history which is *ours*, because we're rebasing our changes as third-party work.

The documentation for `git-checkout` was further clarified in Git >=2.5.1 as per `f303016` commit:

> `--ours` `--theirs`
>
> When checking out paths from the index, check out stage #2 ('ours') or #3 ('theirs') for unmerged paths.
>
> Note that during `git rebase` and `git pull --rebase`, 'ours' and 'theirs' may appear swapped; `--ours` gives the version from the branch the changes are rebased onto, while `--theirs` gives the version from the branch that holds your work that is being rebased.
>
> This is because `rebase` is used in a workflow that treats the history at the remote as the shared canonical one, and treats the work done on the branch you are rebasing as the third-party work to be integrated, and you are temporarily assuming the role of the keeper of the canonical history during the rebase. As the keeper of the canonical history, you need to view the history from the remote as `ours` (i.e. "our shared canonical history"), while what you did on your side branch as `theirs` (i.e. "one contributor's work on top of it").

For `git-merge` it's explain in the following way:

> **ours**
>
> This option forces conflicting hunks to be auto-resolved cleanly by favoring our version. Changes from the other tree that do not conflict with our side are reflected to the merge result. For a binary file, the entire contents are taken from our side.
>
> This should not be confused with the ours merge strategy, which does not even look at what the other tree contains at all. It discards everything the other tree did, declaring our history contains all that happened in it.
>
> **theirs**
>
> This is the opposite of ours.

Further more, here is explained how to use them:

> The merge mechanism (`git merge` and `git pull` commands) allows the backend merge strategies to be chosen with `-s` option. Some strategies can also take their own options, which can be passed by giving `-X<option>` arguments to `git merge` and/or `git pull` .

So sometimes it can be confusing, for example:

- `git pull origin master` where `-Xours` is our local, `-Xtheirs` is theirs (remote) branch
- `git pull origin master -r` where `-Xours` is theirs (remote), `-Xtheirs` is ours

So the 2nd example is opposite to the 1st one, because we're rebasing our branch on top of the remote one, so our starting point is remote one, and our changes are treated as external.

Similar for `git merge` strategies (`-X ours` and `-X theirs`).

edited Oct 23 '16 at 13:01

answered Apr 27 '16 at 21:49
kenorb
38.9k 14 258 254

this answer seems out of date => "git merge --ours" is not a valid option – Alexander Mills Oct 23 '16 at 5:21

@AlexanderMills The answer didn't talk about `git merge` , but `git pull` and `git checkout` as example. If you like to use this parameter with `git merge` , you should use `-X ours` . You can still use `--ours` syntax for `git checkout` . I've clarified the answer further more. – kenorb Oct 23 '16 at 12:59