Home

Pull requests

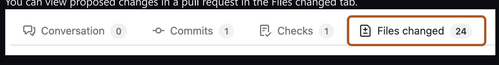
Commit changes to your project
Collaborate with pull requests
Getting started
Working with forks
Code quality features
Propose changes
About branches
Create & delete branches
About pull requests
Compare branches
Creating a pull request
Create a PR from a fork
Using query parameters to create a pull request
Change the state
Request a PR review

About comparing branches in pull requests

Pull requests display diffs to compare the changes you made in your topic branch against the base branch that you want to merge your changes into.

> **Note:** When creating your pull request, you can change the base branch that you're comparing your changes against. For more information, see "Creating a pull request."

You can view proposed changes in a pull request in the Files changed tab.



Rather than viewing the commits themselves, you can view the proposed changes as they'll appear in the files once the pull request is merged. The files appear in alphabetical order within the Files changed tab. Additions to the files appear in green and are prefaced by a `+` sign while content that has been removed appears in red and is prefaced by a `-` sign.

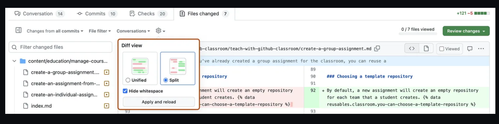
Diff view options

> **Tip:** If you're having a hard time understanding the context of a change, you can click **View** in the Files changed tab to view the whole file with the proposed changes.

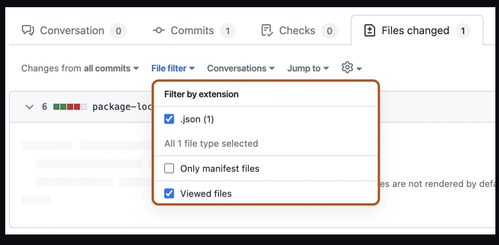
You have several options for viewing a diff:

- The unified view shows updated and existing content together in a linear view.
- The split view shows old content on one side and new content on the other side.
- The rich diff view shows a preview of how the changes will look once the pull request is merged.
- The source view shows the changes in source without the formatting of the rich diff view.

You can also choose to ignore whitespace changes to get a more accurate view of the substantial changes in a pull request.



To simplify reviewing changes in a large pull request, you can filter the diff to only show selected file types, show files you are a CODEOWNER of, hide files you have already viewed, or hide deleted files. For more information, see "Filtering files in a pull request."



Reasons diffs will not display

- You've exceeded the total limit of files or certain file types. For more information, see "About repositories."
- Your file matches a rule in the repository's *.gitattributes* file to block that file from displaying by default. For more information, see "Customizing how changed files appear on GitHub."

Three-dot and two-dot Git diff comparisons

There are two comparison methods for the `git diff` command; two-dot (`git diff A..B`) and three-dot (`git diff A...B`). By default, pull requests on GitHub show a three-dot diff.

Three-dot Git diff comparison

The three-dot comparison shows the difference between the latest common commit of both branches (merge base) and the most recent version of the topic branch.

Two-dot Git diff comparison

The two-dot comparison shows the difference between the latest state of the base branch (for example, `main`) and the most recent version of the topic branch.

To see two committish references in a two-dot diff comparison on GitHub, you can edit the URL of your repository's "Comparing changes" page. For more information, see the Git Glossary for "committish" from the *Pro Git* book site.

For example, this URL uses the shortened seven-character SHA codes to compare commits `f75c570` and `3391dcc`: `https://github.com/github-linguist/linguist/compare/f75c570..3391dcc`.

A two-dot diff compares two Git committish references, such as SHAs or OIDs (Object IDs), directly with each other. On GitHub, the Git committish references in a two-dot diff comparison must be pushed to the same repository or its forks.

If you want to simulate a two-dot diff in a pull request and see a comparison between the most recent versions of each branch, you can merge the base branch into your topic branch, which updates the last common ancestor between your branches.

For more information about Git commands to compare changes, see "Git diff options" from the *Pro Git* book site.

About three-dot comparison on GitHub

Since the three-dot comparison compares with the merge base, it is focusing on "what a pull request introduces".

When you use a two-dot comparison, the diff changes when the base branch is updated, even if you haven't made any changes to the topic branch. Additionally, a two-dot comparison focuses on the base branch. This means that anything you add is displayed as missing from the base branch, as if it was a deletion, and vice versa. As a result, the changes the topic branch introduces become ambiguous.

In contrast, by comparing the branches using the three-dot comparison, changes in the topic branch are always in the diff if the base branch is updated, because the diff shows all of the changes since the branches diverged.

Merging often

To avoid getting confused, merge the base branch (for example, `main`) into your topic branch frequently. By merging the base branch, the diffs shown by two-dot and three-dot comparisons are the same. We recommend merging a pull request as soon as possible. This encourages contributors to make pull requests smaller, which is recommended in general.

Further reading

- "About pull requests"
- "About forks"