This version of GitHub Enterprise was discontinued on 2023-03-15. No patch releases will be made, even for critical security issues. For better performance, improved security, and new features, upgrade to the latest version of GitHub Enterprise. For help with the upgrade, contact GitHub Enterprise support.

About status checks

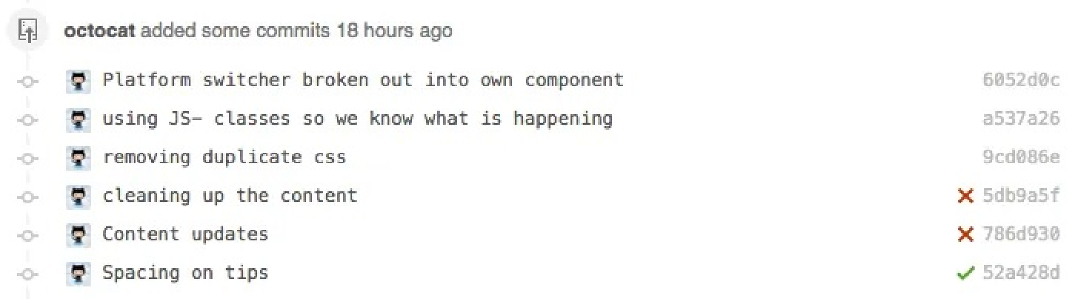
In this article

Types of status checks on GitHub Enterprise Server

Checks

Status checks let you know if your commits meet the conditions set for the repository you're contributing to.

Status checks are based on external processes, such as continuous integration builds, which run for each push you make to a repository. You can see the *pending*, *passing*, or *failing* state of status checks next to individual commits in your pull request.



octocat added some commits 18 hours ago

```
Platform switcher broken out into own component      6052d0c
using JS- classes so we know what is happening       a537a26
removing duplicate css                               9cd086e
cleaning up the content                            ✗ 5db9a5f
Content updates                                    ✗ 786d930
Spacing on tips                                    ✓ 52a428d
```

Anyone with write permissions to a repository can set the state for any status check in the repository.

You can see the overall state of the last commit to a branch on your repository's branches page or in your repository's list of pull requests.

If status checks are required for a repository, the required status checks must pass before you can merge your branch into the protected branch. For more information, see "About protected branches."

Types of status checks on GitHub Enterprise Server

There are two types of status checks on GitHub Enterprise Server:

- Checks
- Statuses

Checks are different from *statuses* in that they provide line annotations, more detailed messaging, and are only available for use with GitHub Apps.

Organization owners and users with push access to a repository can create checks and statuses with GitHub Enterprise Server's API. For more information, see "Checks" and "Commits."

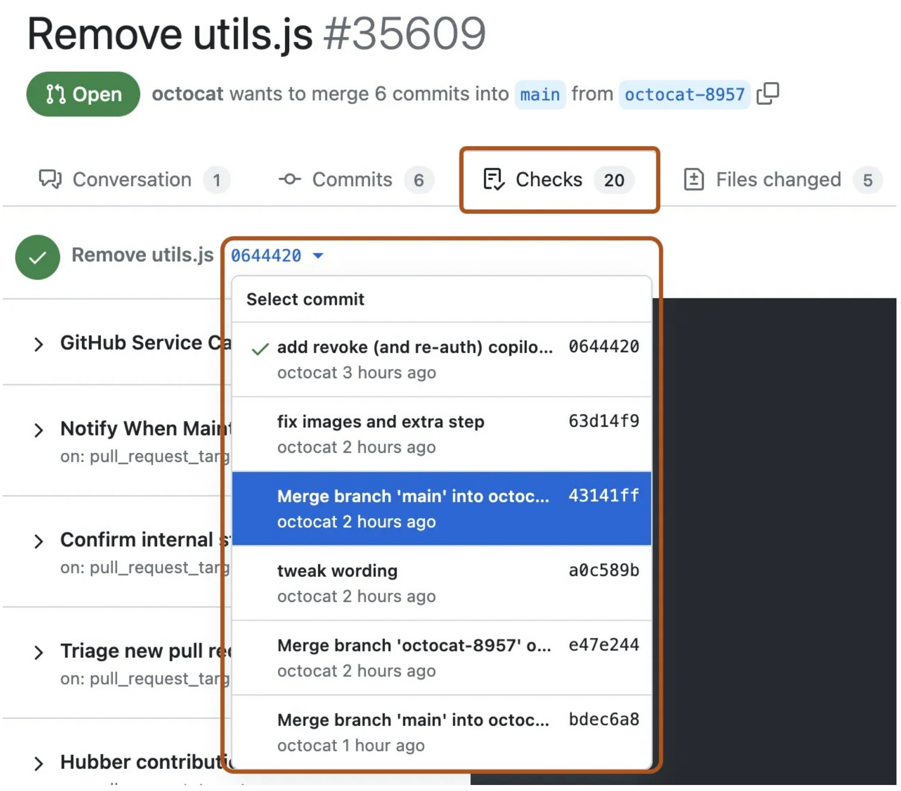
Checks

When *checks* are set up in a repository, pull requests have a **Checks** tab where you can view detailed build output from status checks and rerun failed checks.

Note: The **Checks** tab only gets populated for pull requests if you set up *checks*, not *statuses*, for the repository.

When a specific line in a commit causes a check to fail, you will see details about the failure, warning, or notice next to the relevant code in the **Files** tab of the pull request.

You can navigate between the checks summaries for various commits in a pull request, using the commit drop-down menu under the **Checks** tab.



Skipping and requesting checks for individual commits

When a repository is set to automatically request checks for pushes, you can choose to skip checks for an individual commit you push. When a repository is *not* set to automatically request checks for pushes, you can request checks for an individual commit you push. For more information on these settings, see "Checks."

You can also skip workflow runs triggered by the `push` and `pull_request` events by including a command in your commit message. For more information, see "Skipping workflow runs"

Alternatively, to skip or request *all* checks for your commit, add one of the following trailer lines to the end of your commit message:

- To *skip checks* for a commit, type your commit message and a short, meaningful description of your changes. After your commit description, before the closing quotation, add two empty lines followed by `skip-checks: true`:

```
$ git commit -m "Update README
>
>
skip-checks: true"
```

- To *request* checks for a commit, type your commit message and a short, meaningful description of your changes. After your commit description, before the closing quotation, add two empty lines followed by `request-checks: true`:

  ```
  $ git commit -m "Refactor usability tests
  >
  >
  request-checks: true"
  ```