Managing a branch protection rule

In this article

You can create a branch protection rule to enforce certain workflows for one or more branches, such as requiring an approving review or passing status checks for all pull requests merged into the protected branch.

> **Who can use this feature**
> People with admin permissions or a custom role with the "edit repository rules" permission to a repository can manage branch protection rules.

> Protected branches are available in public repositories with GitHub Free and GitHub Free for organizations, and in public and private repositories with GitHub Pro, GitHub Team, GitHub Enterprise Cloud, and GitHub Enterprise Server.

About branch protection rules 🔗

You can create a branch protection rule in a repository for a specific branch, all branches, or any branch that matches a name pattern you specify with `fnmatch` syntax. For example, to protect any branches containing the word `release`, you can create a branch rule for `*release*`.

You can create a rule for all current and future branches in your repository with the wildcard syntax `*`. Because GitHub uses the `File::FNM_PATHNAME` flag for the `File.fnmatch` syntax, the `*` wildcard does not match directory separators (`/`). For example, `qa/*` will match all branches beginning with `qa/` and containing a single slash, but will not match `qa/foo/bar`. You can include any number of slashes after `qa` with `qa/**/*`, which would match, for example, `qa/foo/bar/foobar/hello-world`. You can also extend the `qa` string with `qa**/**/*` to make the rule more inclusive.

For more information about syntax options, see the [fnmatch documentation](https://ruby-doc.org/core-2.5.1/File.html).

> **Note:** Although GitHub supports `File::FNM_PATHNAME` in `fnmatch` syntax, `File::FNM_EXTGLOB` is not supported.

If a repository has multiple protected branch rules that affect the same branches, the rules that include a specific branch name have the highest priority. If there is more than one protected branch rule that references the same specific branch name, then the branch rule created first will have higher priority.

Protected branch rules that mention a special character, such as `*`, `?`, or `]`, are applied in the order they were created, so older rules with these characters have a

higher priority.

To create an exception to an existing branch rule, you can create a new branch protection rule that is higher priority, such as a branch rule for a specific branch name.

For more information about each of the available branch protection settings, see "About protected branches."

Creating a branch protection rule 🔗

When you create a branch rule, the branch you specify doesn't have to exist yet in the repository.

> **Note:** Actors may only be added to bypass lists when the repository belongs to an organization.

1. On your GitHub Enterprise Server instance, navigate to the main page of the repository.

2. Under your repository name, click ⚙ **Settings**. If you cannot see the "Settings" tab, select the ••• dropdown menu, then click **Settings**.
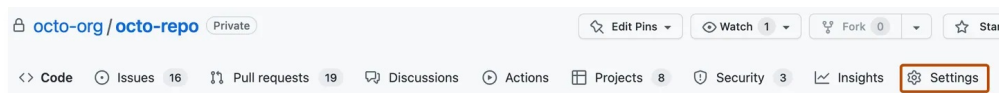
3. In the "Code and automation" section of the sidebar, click ⑂ **Branches**.

4. Next to "Branch protection rules", click **Add rule.**

5. Under "Branch name pattern", type the branch name or pattern you want to protect.

6. Optionally, enable required pull requests.

> **Note:** If you select **Dismiss stale pull request approvals when new commits are pushed** and/or **Require approval of the most recent reviewable push**, manually creating the merge commit for a pull request and pushing it directly to a protected branch will fail, unless the contents of the merge exactly match the merge generated by GitHub for the pull request.
>
> In addition, with these settings, approving reviews will be dismissed as stale if the merge base introduces new changes after the review was submitted. The merge base is the commit that is the last common ancestor between the topic branch and the base branch. If the merge base changes, the pull request cannot be merged until someone approves the work again.

   - Under "Protect matching branches", select **Require a pull request before merging**.
   - Optionally, to require approvals before a pull request can be merged, select **Require approvals**.

     Select the **Required number of approvals before merging** dropdown menu, then click the number of approving reviews you would like to require on the branch.
   - Optionally, to dismiss a pull request approval review when a code-modifying commit is pushed to the branch, select **Dismiss stale pull request approvals when new commits are pushed**.
   - Optionally, to require review from a code owner when the pull request affects

code that has a designated owner, select **Require review from Code Owners**. For more information, see "[About code owners](#)."

- Optionally, to allow specific actors to push code to the branch without creating pull requests when they're required, select **Allow specified actors to bypass required pull requests**. Then, search for and select the actors who should be allowed to skip creating a pull request.
- Optionally, if the repository is part of an organization, select **Restrict who can dismiss pull request reviews**. Then, in the search field, search for and select the actors who are allowed to dismiss pull request reviews. For more information, see "[Dismissing a pull request review](#)."
- Optionally, to require someone other than the last person to push to a branch to approve a pull request prior to merging, select **Require approval of the most recent reviewable push**. For more information, see "[About protected branches](#)."

7 Optionally, enable required status checks. For more information, see "[About status checks](#)."

- Select **Require status checks to pass before merging**.
- Optionally, to ensure that pull requests are tested with the latest code on the protected branch, select **Require branches to be up to date before merging**.
- In the search field, search for status checks, selecting the checks you want to require.

8 Optionally, select **Require conversation resolution before merging**.

9 Optionally, select **Require signed commits**.

10 Optionally, select **Require linear history**.

11 Optionally, to choose which environments the changes must be successfully deployed to before merging, select **Require deployments to succeed before merging**, then select the environments.

12 Optionally, make the branch read-only.

- Select **Lock branch**.
- Optionally, to allow fork syncing, select **Allow fork syncing**.

13 Optionally, select **Do not allow bypassing the above settings**.

14 Optionally, enable branch restrictions.

- Select **Restrict who can push to matching branches**.
- Optionally, to also restrict the creation of matching branches, select **Restrict pushes that create matching branches**.
- In the search field, search for and select the people, teams, or apps who will have permission to push to the protected branch or create a matching branch.

15 Optionally, under "Rules applied to everyone including administrators", select **Allow force pushes**.

Then, choose who can force push to the branch.

- Select **Everyone** to allow everyone with at least write permissions to the repository to force push to the branch, including those with admin permissions.
- Select **Specify who can force push** to allow only specific actors to force push

to the branch. Then, search for and select those actors.

For more information about force pushes, see "[About protected branches](#)."

16. Optionally, select **Allow deletions**.

17. Click **Create**.

Editing a branch protection rule 🔗

1. On your GitHub Enterprise Server instance, navigate to the main page of the repository.

2. Under your repository name, click ⚙ **Settings**. If you cannot see the "Settings" tab, select the ⋯ dropdown menu, then click **Settings**.
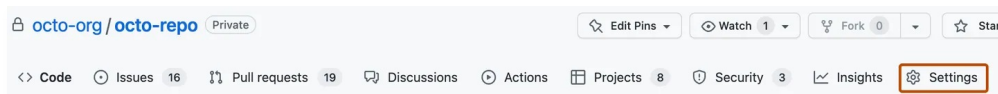
3. In the "Code and automation" section of the sidebar, click ⑂ **Branches**.

4. To the right of the branch protection rule you want to edit, click **Edit**.

5. Make your desired changes to the branch protection rule.

6. Click **Save changes**.

Deleting a branch protection rule 🔗

1. On your GitHub Enterprise Server instance, navigate to the main page of the repository.

2. Under your repository name, click ⚙ **Settings**. If you cannot see the "Settings" tab, select the ⋯ dropdown menu, then click **Settings**.
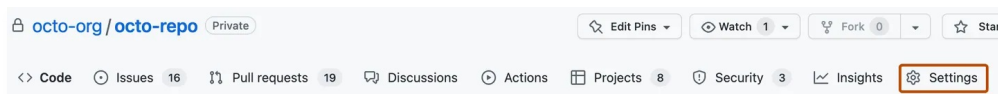
3. In the "Code and automation" section of the sidebar, click ⑂ **Branches**.

4. To the right of the branch protection rule you want to delete, click **Delete**.