About branches

Use a branch to isolate development work without affecting other branches in the repository. Each repository has one default branch, and can have multiple other branches. You can merge a branch into another branch using a pull request.

Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository.

You always create a branch from an existing branch. Typically, you might create a new branch from the default branch of your repository. You can then work on this new branch in isolation from changes that other people are making to the repository. A branch you create to build a feature is commonly referred to as a feature branch or topic branch. For more information, see "Creating and deleting branches within your repository."

You can also use a branch to publish a GitHub Pages site. For more information, see "About GitHub Pages."

You must have write access to a repository to create a branch, open a pull request, or delete and restore branches in a pull request. For more information, see "Access permissions on GitHub."

About the default branch

When you create a repository with content on GitHub, GitHub creates the repository with a single branch. This first branch in the repository is the default branch. The default branch is the branch that GitHub displays when anyone visits your repository. The default branch is also the initial branch that Git checks out locally when someone clones the repository. Unless you specify a different branch, the default branch in a repository is the base branch for new pull requests and code commits.

By default, GitHub names the default branch `main` in any new repository.

You can change the default branch for an existing repository. For more information, see "Changing the default branch."

You can set the name of the default branch for new repositories. For more information, see "Managing the default branch for your repositories," "Managing the default branch name for repositories in your organization," and "Enforcing repository management policies in your enterprise account."

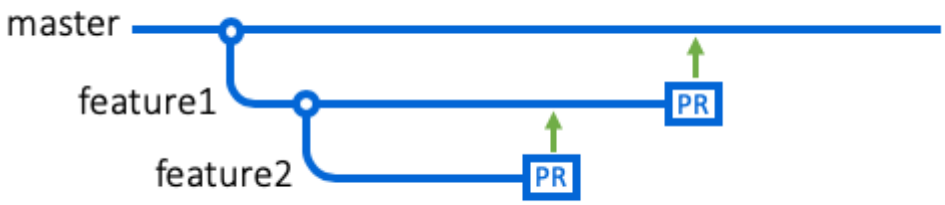
Working with branches

Once you're satisfied with your work, you can open a pull request to merge the changes in the current branch (the *head* branch) into another branch (the *base* branch). For more information, see "About pull requests."

After a pull request has been merged, or closed, you can delete the head branch as this is no longer needed. You must have write access in the repository to delete branches. You can't delete branches that are directly associated with open pull requests. For more information, see "Deleting and restoring branches in a pull request"

If you delete a head branch after its pull request has been merged, GitHub checks for any open pull requests in the same repository that specify the deleted branch as their base branch. GitHub automatically updates any such pull requests, changing their base branch to the merged pull request's base branch. The following diagrams illustrate this.

Here someone has created a branch called `feature1` from the `master` branch, and you've then created a branch called `feature2` from `feature1`. There are open pull requests for both branches. The arrows indicate the current base branch for each pull request. At this point, `feature1` is the base branch for `feature2`. If the pull request for `feature2` is merged now, the `feature2` branch will be merged into `feature1`.



In the next diagram, someone has merged the pull request for `feature1` into the `master` branch, and they have deleted the `feature1` branch. As a result, GitHub has automatically retargeted the pull request for `feature2` so that its base branch is now `master`.

Now when you merge the `feature2` pull request, it'll be merged into the `master` branch.

Working with protected branches

Repository administrators can enable protections on a branch. If you're working on a branch that's protected, you won't be able to delete or force push to the branch. Repository administrators can additionally enable several other protected branch settings to enforce various workflows before a branch can be merged.

> **Note:** If you're a repository administrator, you can merge pull requests on branches with branch protections enabled even if the pull request does not meet the requirements, unless branch protections have been set to "Include administrators."

To see if your pull request can be merged, look in the merge box at the bottom of the pull request's **Conversation** tab. For more information, see "About protected branches."

When a branch is protected:

- You won't be able to delete or force push to the branch.
- If required status checks are enabled on the branch, you won't be able to merge changes into the branch until all of the required CI tests pass. For more information, see "About status checks."
- If required pull request reviews are enabled on the branch, you won't be able to merge changes into the branch until all requirements in the pull request review policy have been met. For more information, see "Merging a pull request."
- If required review from a code owner is enabled on a branch, and a pull request modifies code that has an owner, a code owner must approve the pull request before it can be merged. For more information, see "About code owners."
- If required commit signing is enabled on a branch, you won't be able to push any commits to the branch that are not signed and verified. For more information, see "About commit signature verification" and "About protected branches."
- If you use GitHub's conflict editor to fix conflicts for a pull request that you created from a protected branch, GitHub helps you to create an alternative branch for the pull request, so that your resolution of the conflicts can be merged. For more information, see "Resolving a merge conflict on GitHub."

Further reading