# GitHub flow

**In this article**

Follow GitHub flow to collaborate on projects.

## Introduction

GitHub flow is a lightweight, branch-based workflow. The GitHub flow is useful for everyone, not just developers. For example, here at GitHub, we use GitHub flow for our site policy, documentation, and roadmap.

## Prerequisites

To follow GitHub flow, you will need a GitHub account and a repository. For information on how to create an account, see "Signing up for GitHub." For information on how to create a repository, see "Create a repo."

## Following GitHub flow

> **Tip:** You can complete all steps of GitHub flow through the GitHub web interface, command line and GitHub CLI, or GitHub Desktop. For more information about the tools you can use to connect to GitHub, see "Connecting to GitHub."

### Create a branch

Create a branch in your repository. A short, descriptive branch name enables your collaborators to see ongoing work at a glance. For example, `increase-test-timeout` or `add-code-of-conduct`. For more information, see "Creating and deleting branches within your repository."

By creating a branch, you create a space to work without affecting the default branch. Additionally, you give collaborators a chance to review your work.

### Make changes

On your branch, make any desired changes to the repository. For more information, see "Creating new files", "Editing files", "Renaming a file", "Moving a file to a new location", or "Deleting files in a repository".

Your branch is a safe place to make changes. If you make a mistake, you can revert your changes or push additional changes to fix the mistake. Your changes will not end up on the default branch until you merge your branch.

Commit and push your changes to your branch. Give each commit a descriptive message to help you and future contributors understand what changes the commit contains. For example, `fix typo` or `increase rate limit`.

Ideally, each commit contains an isolated, complete change. This makes it easy to revert your changes if you decide to take a different approach. For example, if you want to rename a variable and add some tests, put the variable rename in one commit and the tests in another commit. Later, if you want to keep the tests but revert the variable rename, you can revert the specific commit that contained the variable rename. If you put the variable rename and tests in the same commit or spread the variable rename across multiple commits, you would spend more effort reverting your changes.

By committing and pushing your changes, you back up your work to remote storage. This means that you can access your work from any device. It also means that your collaborators can see your work, answer questions, and make suggestions or contributions.

Continue to make, commit, and push changes to your branch until you are ready to ask for feedback.

> **Tip:** Make a separate branch for each set of unrelated changes. This makes it easier for reviewers to give feedback. It also makes it easier for you and future collaborators to understand the changes and to revert or build on them. Additionally, if there is a delay in one set of changes, your other changes aren't also delayed.

## Create a pull request 🔗

Create a pull request to ask collaborators for feedback on your changes. Pull request review is so valuable that some repositories require an approving review before pull requests can be merged. If you want early feedback or advice before you complete your changes, you can mark your pull request as a draft. For more information, see "[Creating a pull request](#)."

When you create a pull request, include a summary of the changes and what problem they solve. You can include images, links, and tables to help convey this information. If your pull request addresses an issue, link the issue so that issue stakeholders are aware of the pull request and vice versa. If you link with a keyword, the issue will close automatically when the pull request merges. For more information, see "[Basic writing and formatting syntax](#)" and "[Linking a pull request to an issue](#)."

In addition to filling out the body of the pull request, you can add comments to specific lines of the pull request to explicitly point something out to the reviewers.

Your repository may be configured to automatically request a review from specific teams or users when a pull request is created. You can also manually @mention or request a review from specific people or teams.

If your repository has checks configured to run on pull requests, you will see any checks that failed on your pull request. This helps you catch errors before merging your branch. For more information, see "[About status checks](#)."

## Address review comments 🔗

Reviewers should leave questions, comments, and suggestions. Reviewers can comment on the whole pull request or add comments to specific lines or files. You and reviewers can insert images or code suggestions to clarify comments. For more information, see "[Reviewing changes in pull requests](#)."

You can continue to commit and push changes in response to the reviews. Your pull request will update automatically.

## Merge your pull request 🔗

Once your pull request is approved, merge your pull request. This will automatically merge your branch so that your changes appear on the default branch. GitHub retains the history of comments and commits in the pull request to help future contributors understand your changes. For more information, see "[Merging a pull request](#)."

GitHub will tell you if your pull request has conflicts that must be resolved before merging. For more information, see "[Addressing merge conflicts](#)."

Branch protection settings may block merging if your pull request does not meet certain requirements. For example, you need a certain number of approving reviews or an approving review from a specific team. For more information, see "[About protected branches](#)."

## Delete your branch 🔗

After you merge your pull request, delete your branch. This indicates that the work on the branch is complete and prevents you or others from accidentally using old branches. For more information, see "[Deleting and restoring branches in a pull request](#)."

Don't worry about losing information. Your pull request and commit history will not be deleted. You can always restore your deleted branch or revert your pull request if needed.