Committing and reviewing changes to your project

GitHub Desktop tracks all changes to all files as you edit them. You can decide how to group the changes to create meaningful commits.

About commits

Similar to saving a file that's been edited, a commit records changes to one or more files in your branch. Git assigns each commit a unique ID, called a SHA or hash, that identifies:

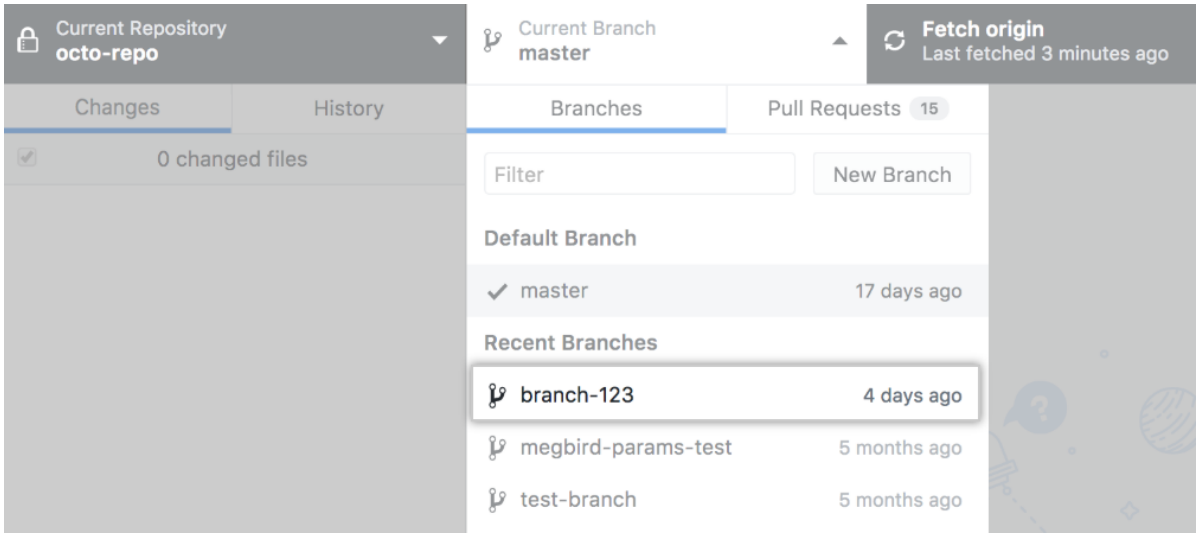
- The specific changes
- When the changes were made
- Who created the changes

When you make a commit, you must include a commit message that briefly describes the changes. You can also add a co-author on any commits you collaborate on.

If the commits you make in GitHub Desktop are associated with the wrong account on GitHub, update the email address in your Git configuration using GitHub Desktop. For more information, see "Configuring Git for GitHub Desktop."

Choosing a branch and making changes

1. Create a new branch, or select an existing branch by clicking **Current Branch** on the toolbar and selecting the branch from the list.



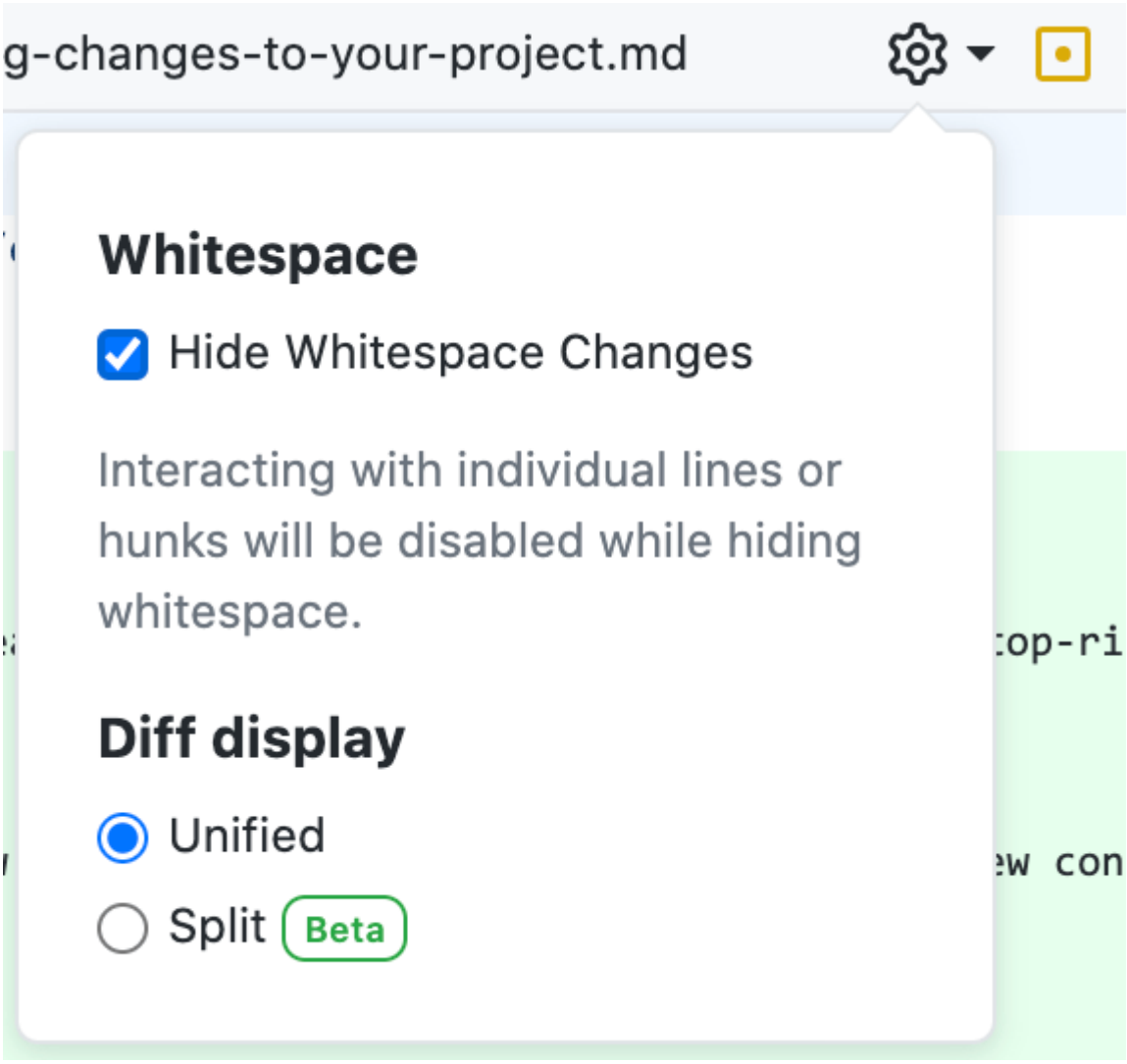
2. Using your favorite text editor, such as Atom, make the necessary changes to files in your project.

Choosing how to display diffs

You can change the way diffs are displayed in GitHub Desktop to suit your reviewing needs.

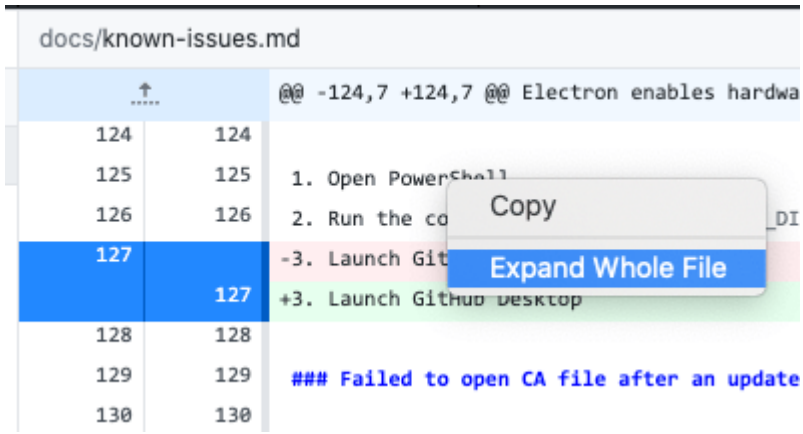
To change how you view diffs, in the top-right corner of the diff view, click .

- To change how the entire diff is displayed, under "Diff display", select **Unified** or **Split**. The Unified view shows changes linearly, while the Split view shows old content on the left side and new content on the right side.
- To hide whitespace changes so you can focus on more substantive changes, select **Hide Whitespace Changes**.



If you need to see more of the file than GitHub Desktop shows by default, you can expand the diff.

- To see the next few lines above or below the highlighted changes, click the arrow above or below the line numbers.
- To see the entire file, right-click in the diff view and click **Expand Whole File**.

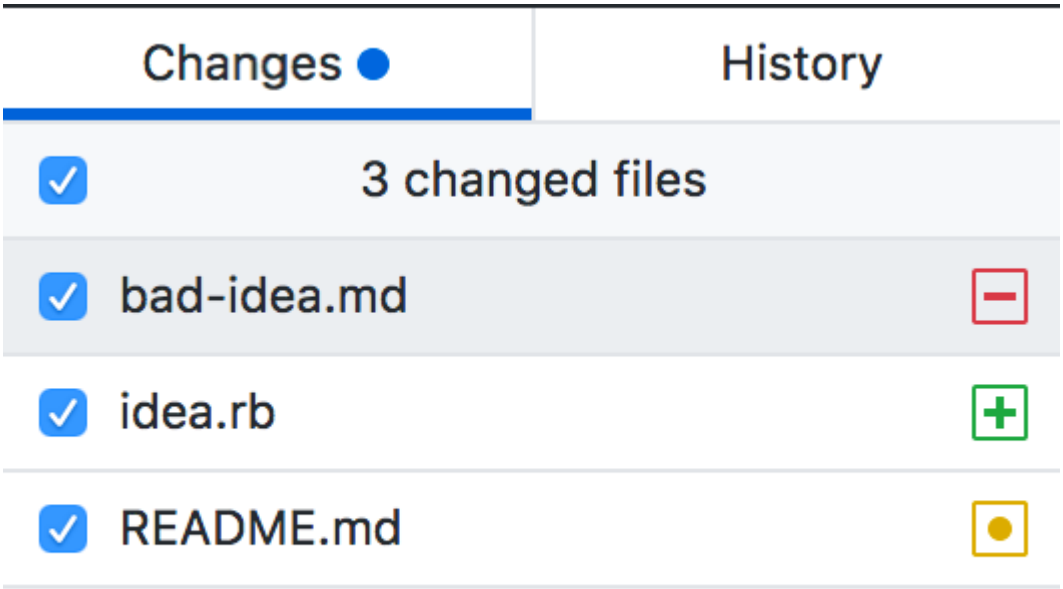


Selecting changes to include in a commit

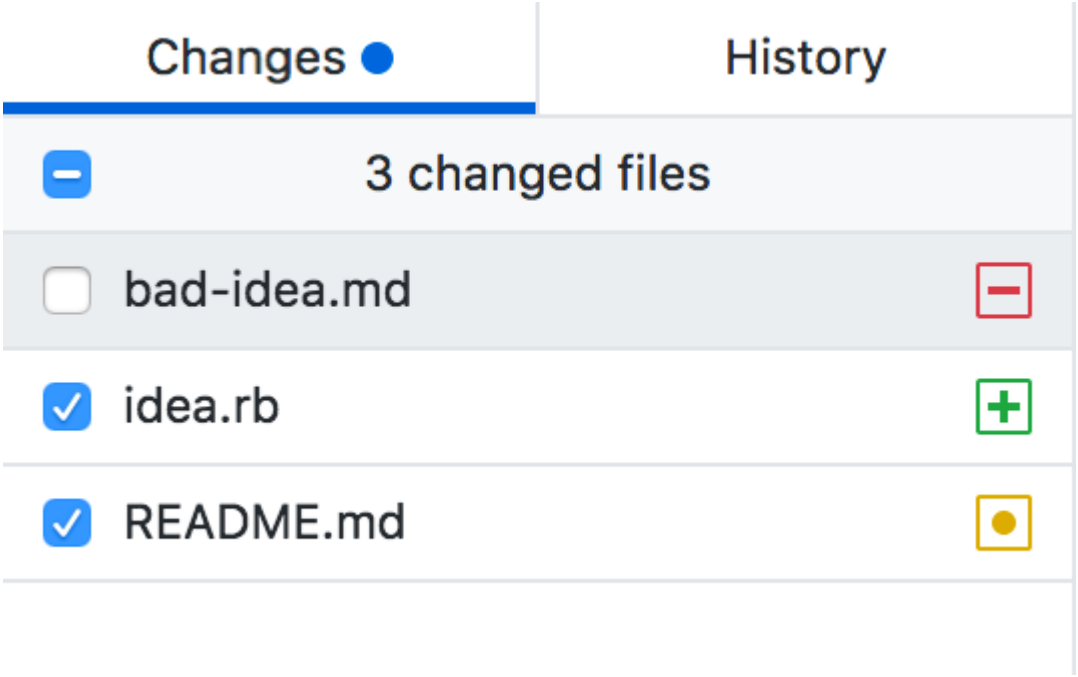
As you make changes to files in your text editor and save them locally, you will also see the changes in GitHub Desktop.

- The red ⊟ icon indicates removed files.
- The yellow ⊡ icon indicates modified files.
- The green ⊞ icon indicates added files.
- To access stashed changes, click **Stashed Changes**.

- To add **all changes in all files** to a single commit, keep the checkbox at the top of the list selected.



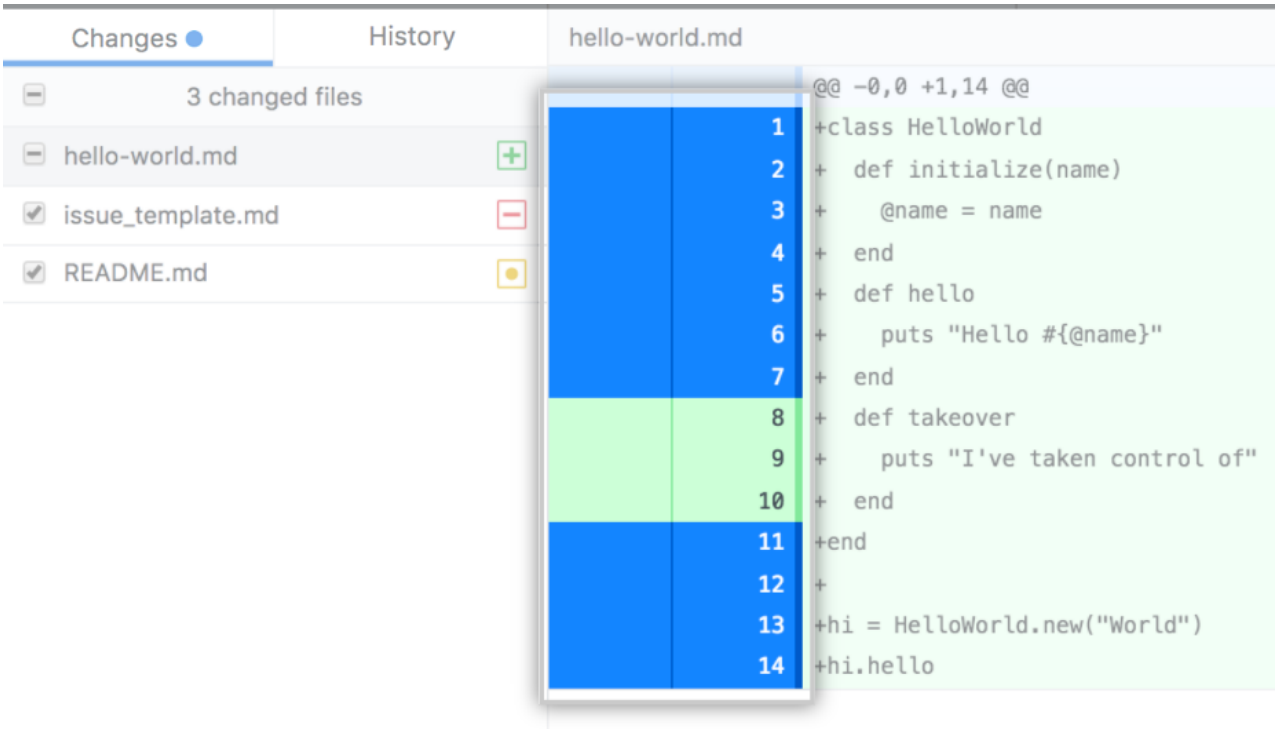
- To add **all changes in one or more files** to a single commit, unselect the checkboxes next to the files you don't want included, leaving only the files you want in the commit. You can toggle the checkbox with the `Spacebar` or `Enter` keys after selecting a file.



Creating a partial commit

If one file contains multiple changes, but you only want some of those changes to be included in a commit, you can create a partial commit. The rest of your changes will remain intact, so that you can make additional modifications and commits. This allows you to make separate, meaningful commits, such as keeping line break changes in a commit separate from code or prose changes.

To exclude changed lines from your commit, click one or more changed lines so the blue disappears. The lines that are still highlighted in blue will be included in the commit.



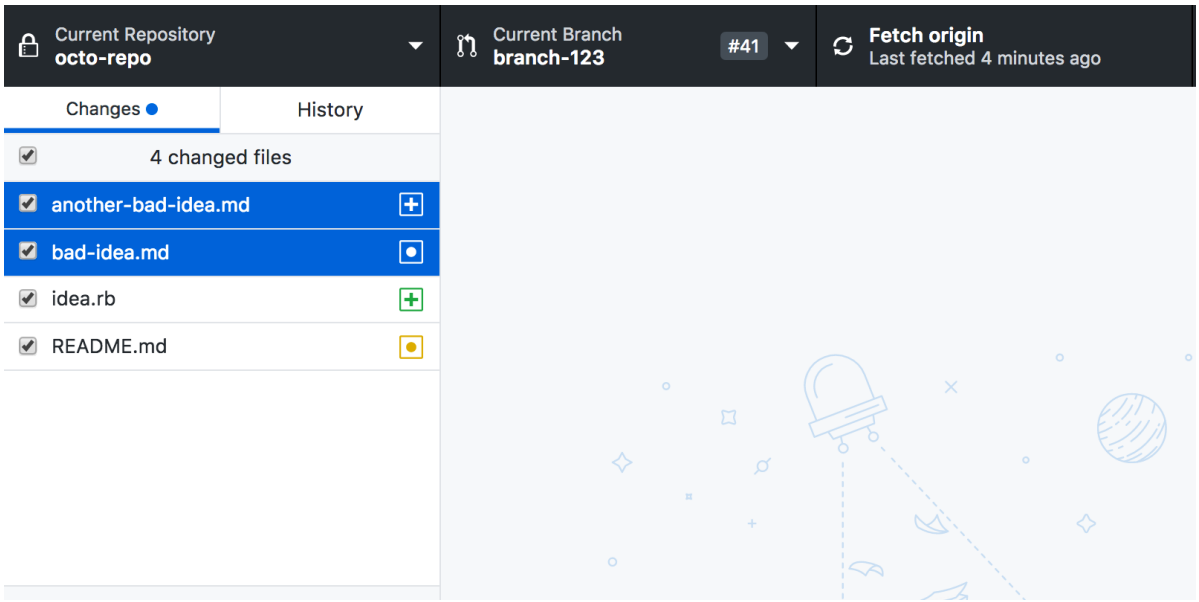
Discarding changes

If you have uncommitted changes that you don't want to keep, you can discard the changes. This will remove the changes from the files on your computer. You can discard all uncommitted changes in one or more files, or you can discard specific lines you added.

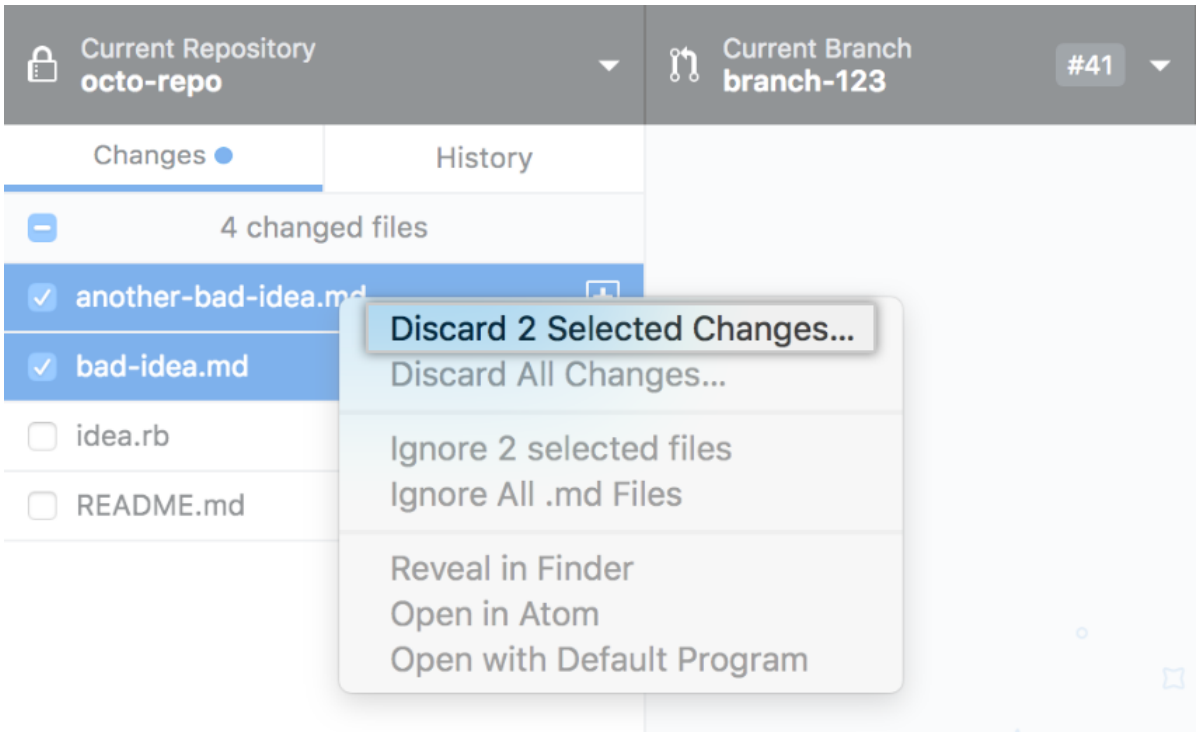
Discarded changes are saved in a dated file in the Trash. You can recover discarded changes until the Trash is emptied.

Discarding changes in one or more files

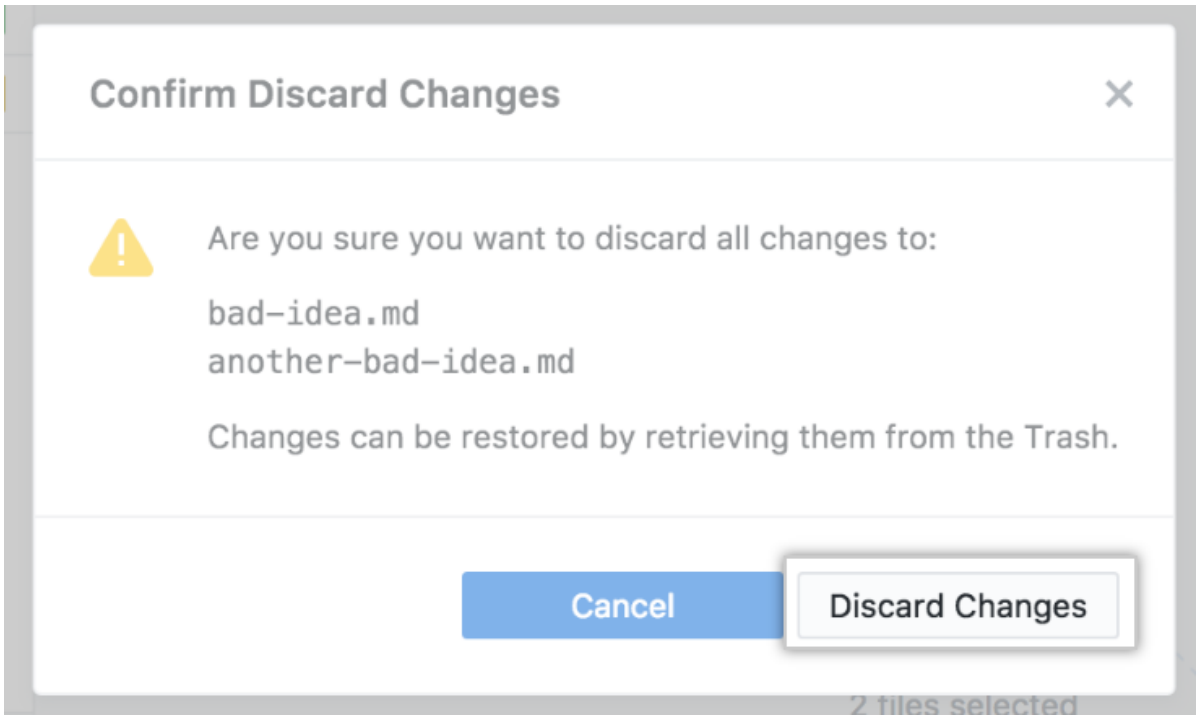
1. In the list of changed files, select the files where you want to discard the changes since the last commit. To select multiple files, click `shift` and click on the range of files you want to discard changes from.



2. Click **Discard Changes** or **Discard Selected Changes** to discard changes to one or more files, or **Discard All Changes** to discard changes to all files since the last commit.



3. To confirm the changes, review the files affected and click **Discard Changes**.

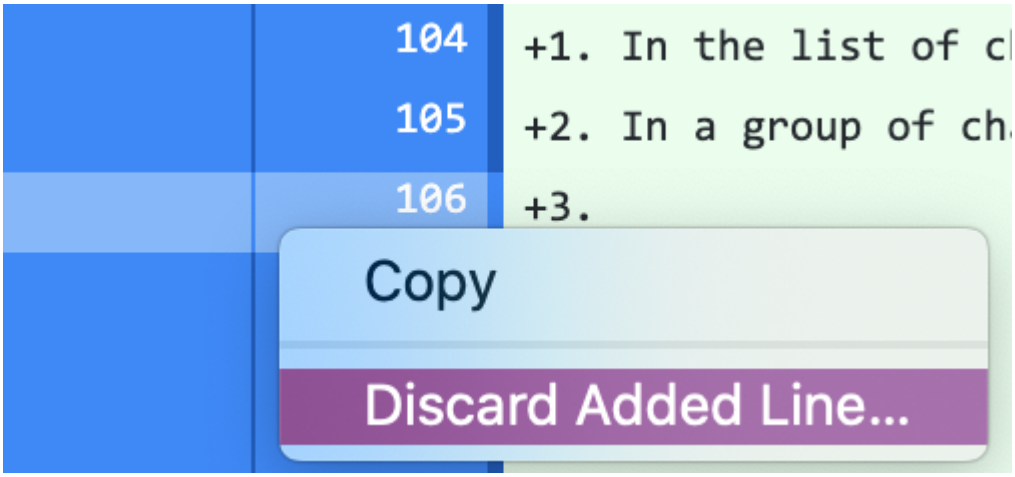


Discarding changes in one or more lines

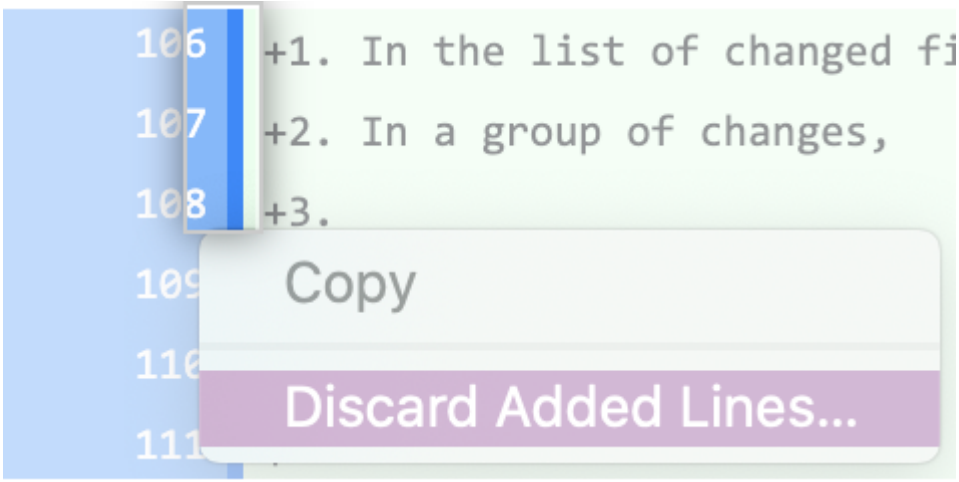
You can discard one or more changed lines that are uncommitted.

> **Note:** Discarding single lines is disabled in a group of changes that adds and removes lines.

To discard one added line, in the list of changed lines, right click on the line you want to discard and select **Discard added line**.



To discard a group of changed lines, right click the vertical bar to the right of the line numbers for the lines you want to discard, then select **Discard added lines**.



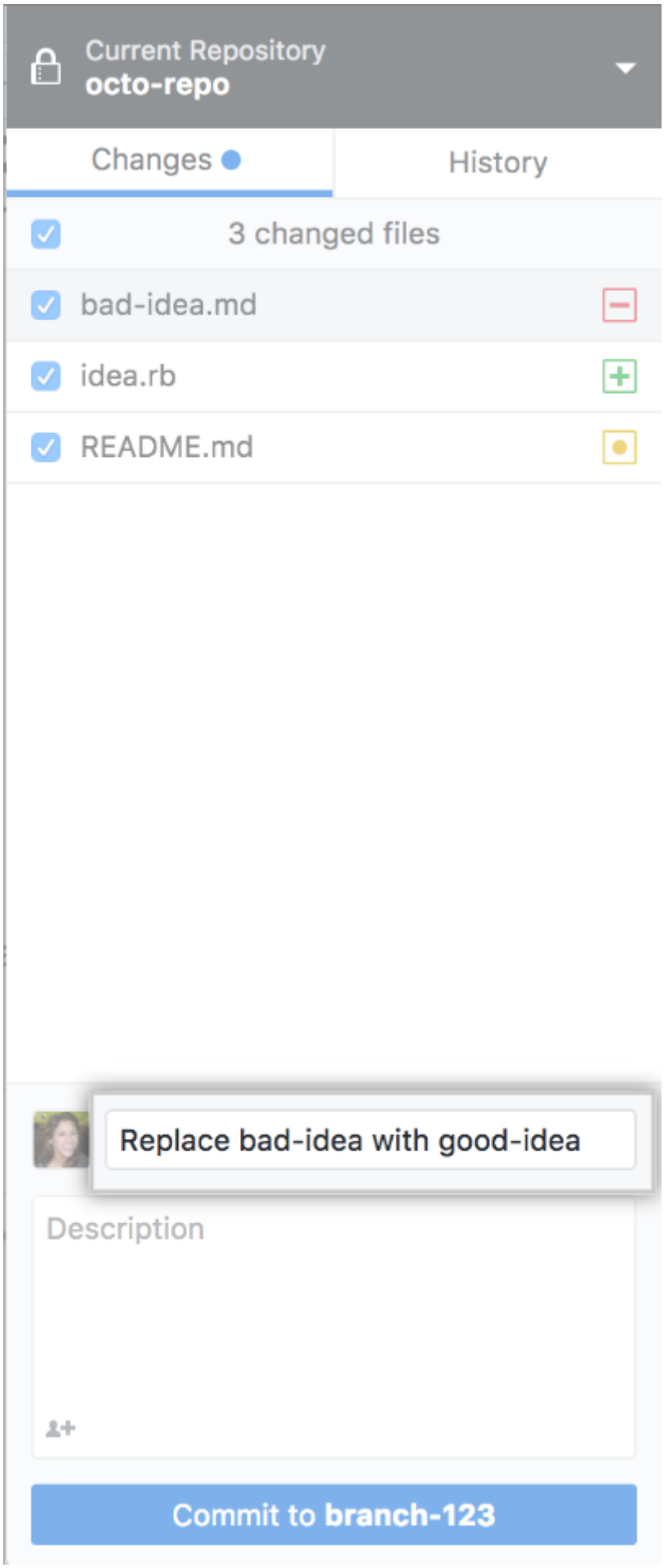
Write a commit message and push your changes

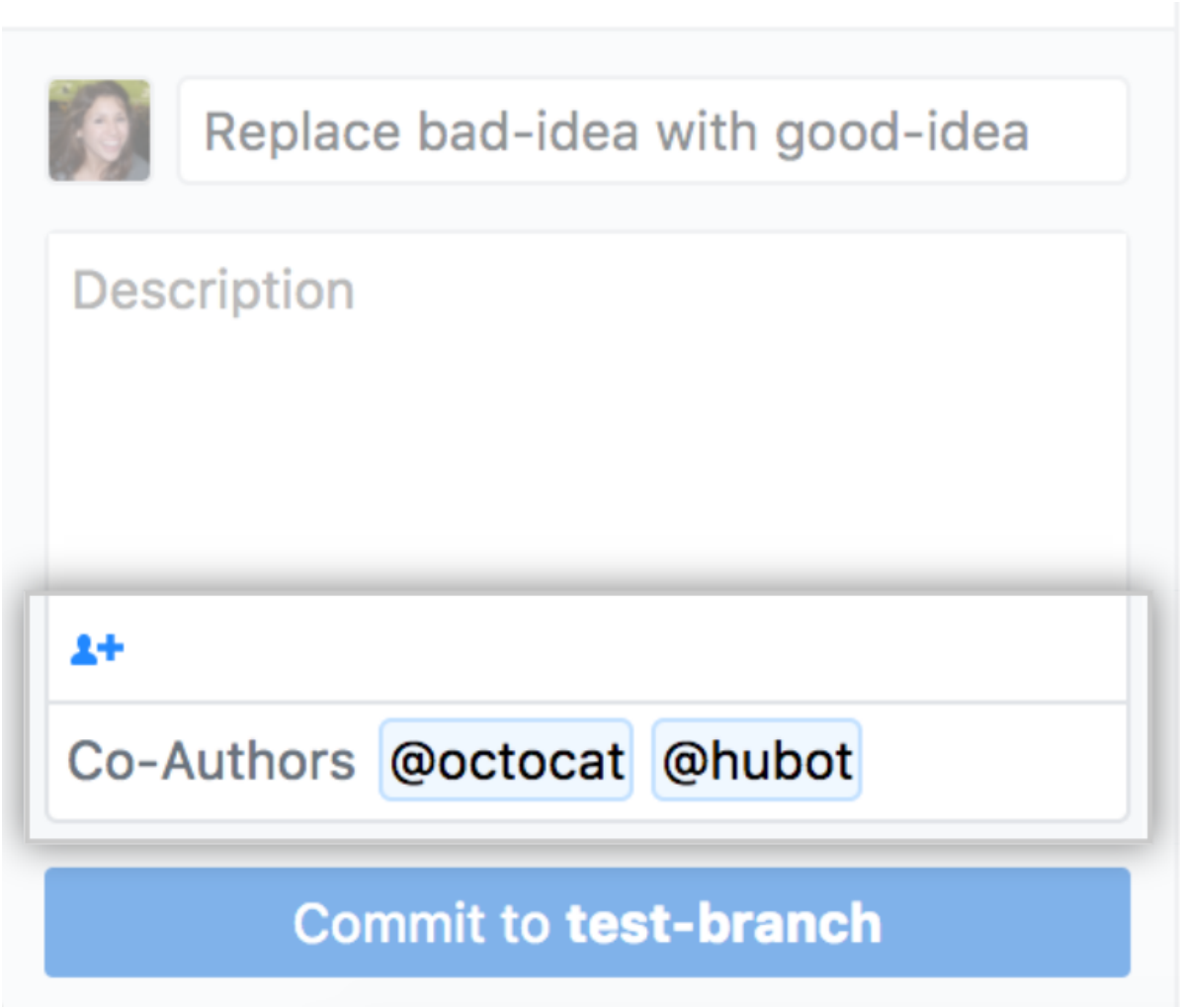
Once you're satisfied with the changes you've chosen to include in your commit, write your commit message and push your changes. If you've collaborated on a commit, you can also attribute a commit to more than one author.

> **Note**: By default, GitHub Desktop will push the tag that you create to your repository with the associated commit. For more information, see "Managing tags."

1. At the bottom of the list of changes, in the Summary field, type a short, meaningful commit message. Optionally, you can add more information about the change in the Description field.

2. Optionally, to attribute a commit to another author, click the add co-authors icon and type the username(s) you want to include.

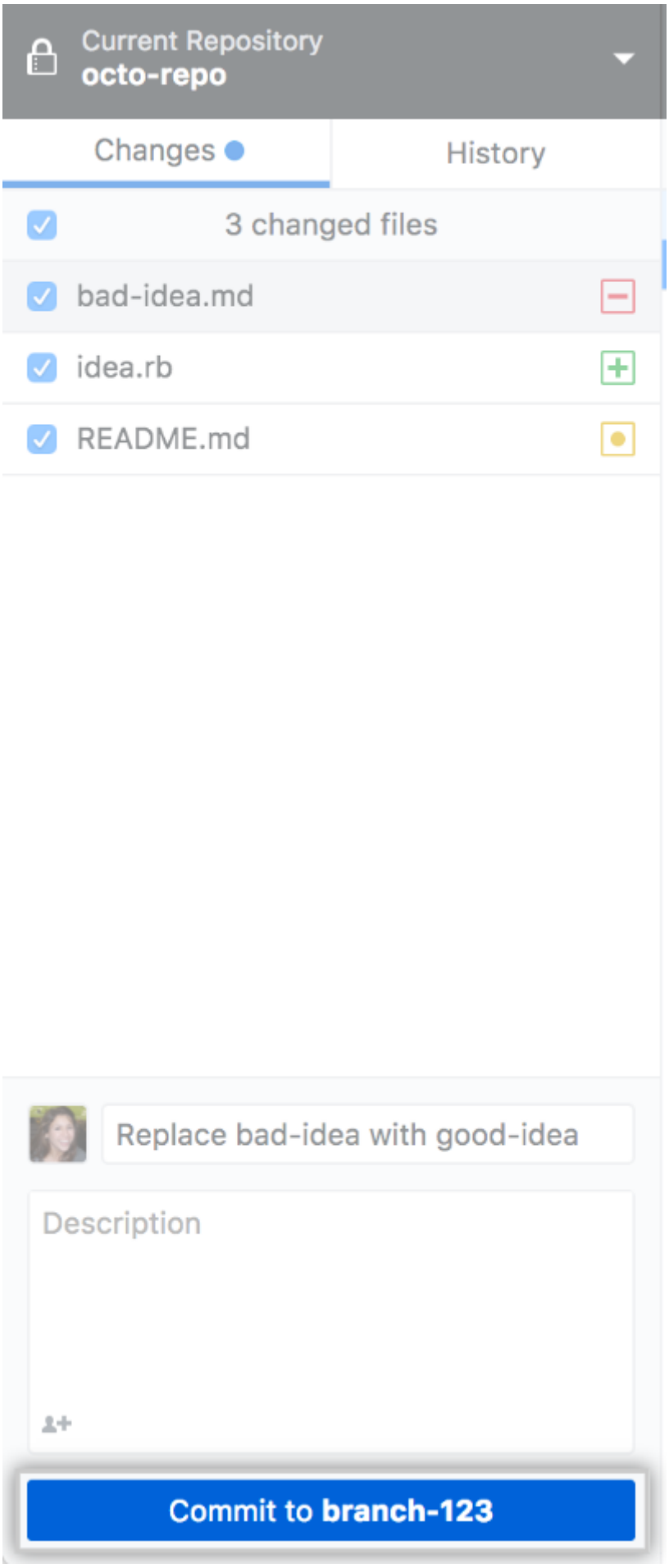


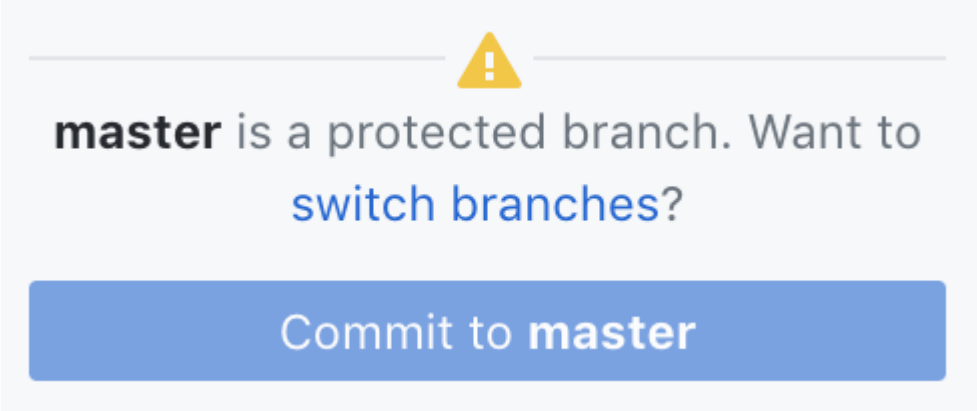


3. Under the Description field, click **Commit to *BRANCH***.

4. If the branch you're trying to commit to is protected, Desktop will warn you.
    - To move your changes, click **switch branches**.
    - To commit your changes to the protected branch, click **Commit to *BRANCH***.

    For more information about protected branches, see "About protected branches".





5. Click **Push origin** to push your local changes to the remote repository.