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.

Mac Windows

About commits

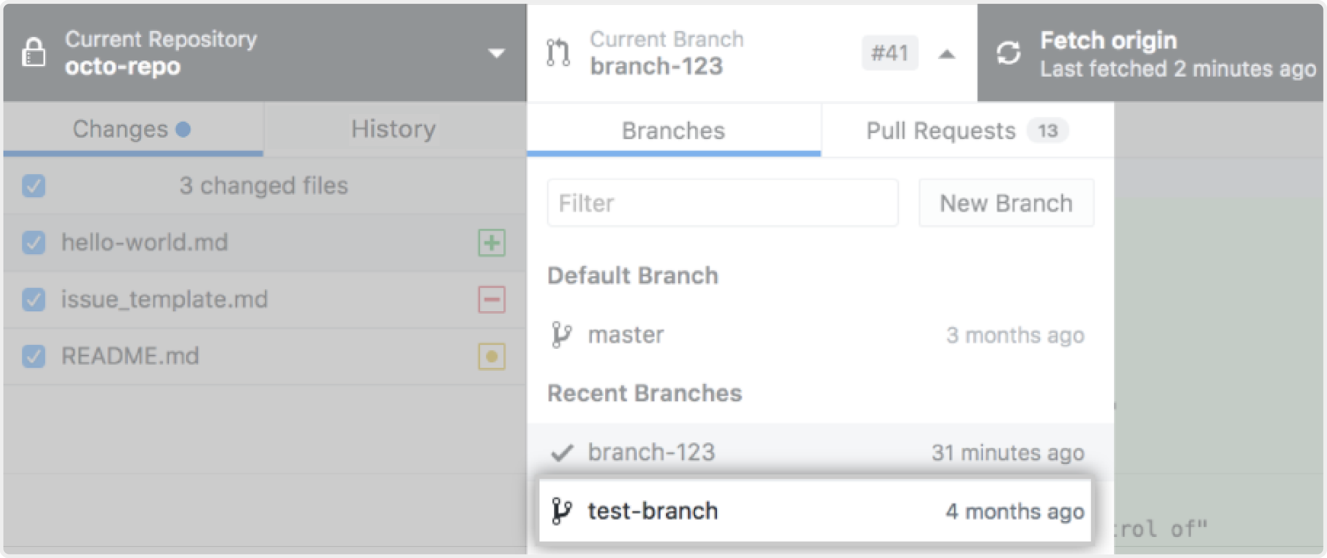
Similar to saving a file, a commit is a change to one or more files in your branch. Git assigns each commit a unique ID, called a SHA or hash, that tracks:

- 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.

1. 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.

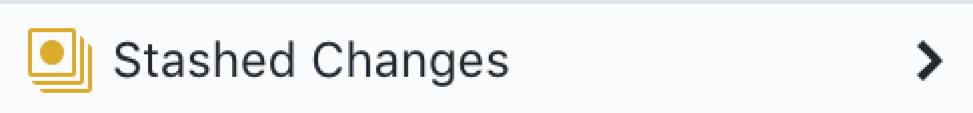


2 Using your favorite text editor, such as Atom, make the necessary changes to files in your project.

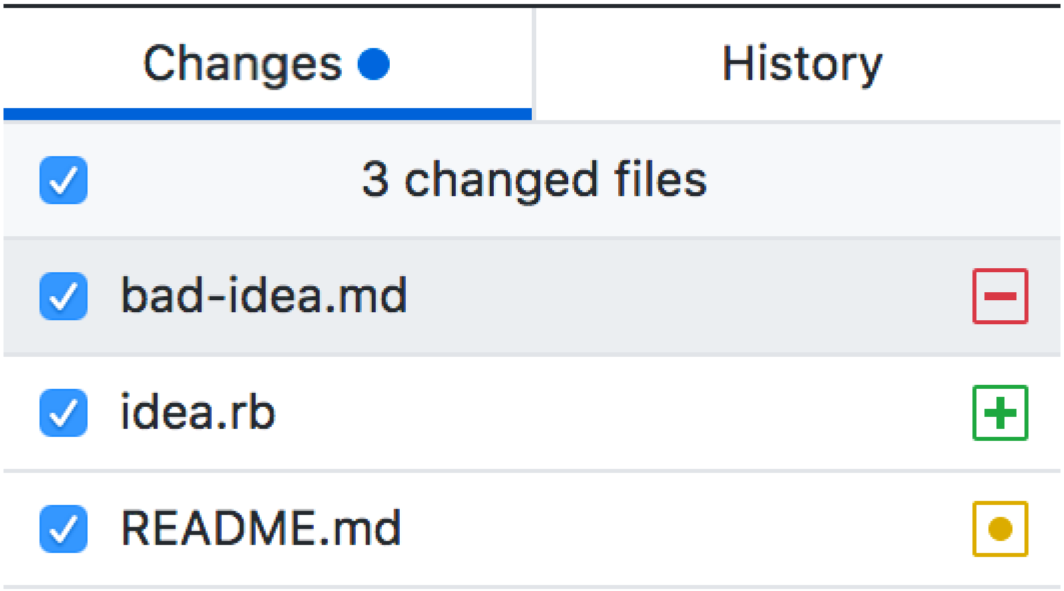
2. Selecting changes to include in a commit

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

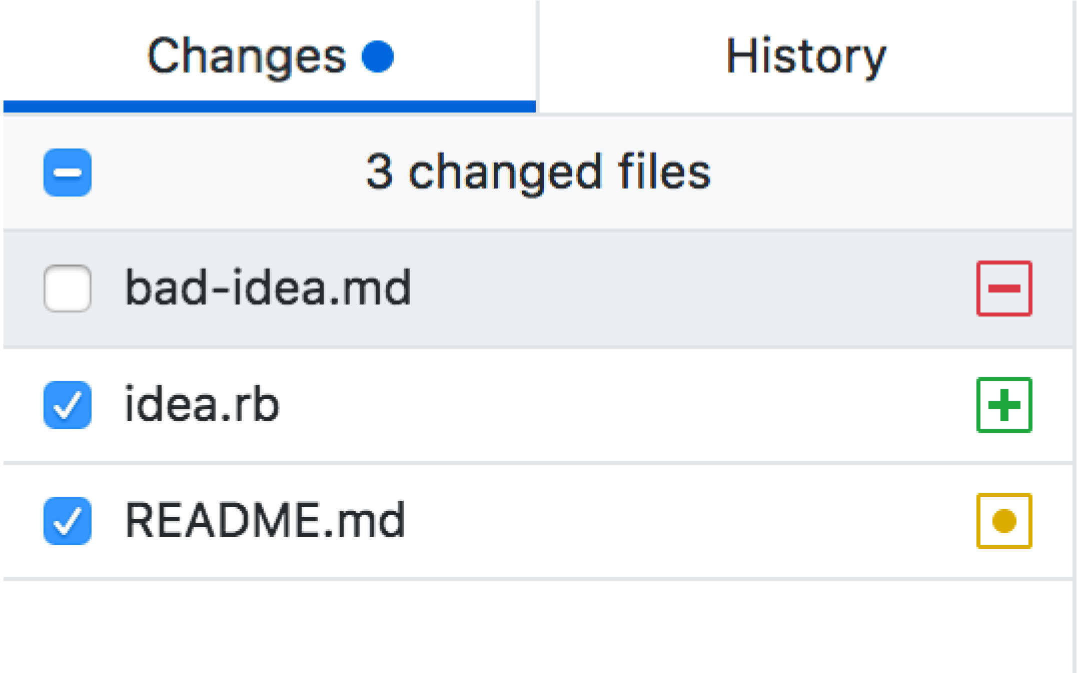
- The red ⊟ icon indicates removed files.
- The yellow ⊡ icon indicates modified files.
- The green ⊞ icon indicates added files.
- To access stashed changes, click **Stashed Changes**.



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



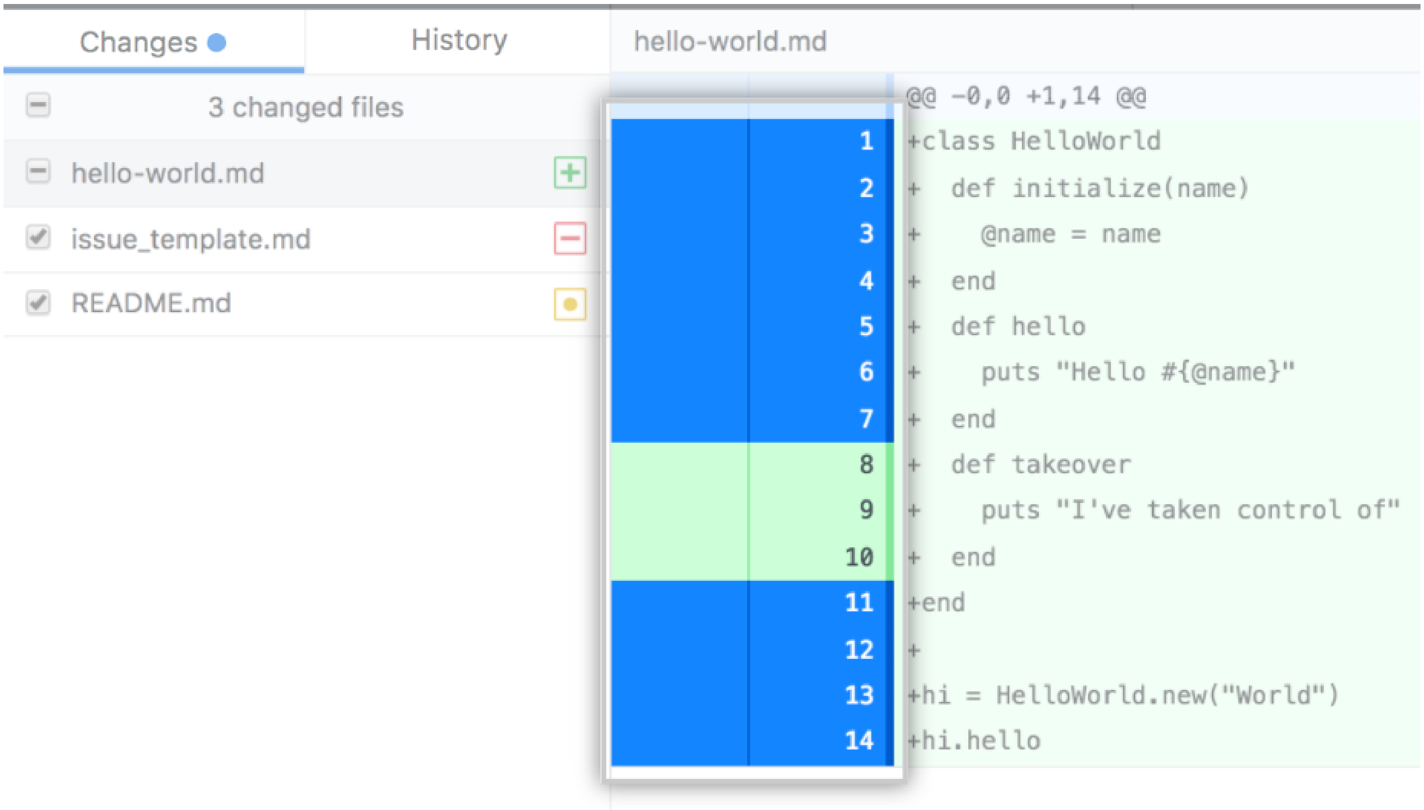
- To add **all changes in one or more files** to a single commit, unselect the checkboxes next to the files you don't want included, leaving only the files you want in the commit. You can toggle the checkbox with the `Spacebar` or `Enter` keys after selecting a file.



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.

When you review the diff of the file, the lines that will be included in the commit are highlighted in blue. To exclude the change, click the changed line so the blue disappears.

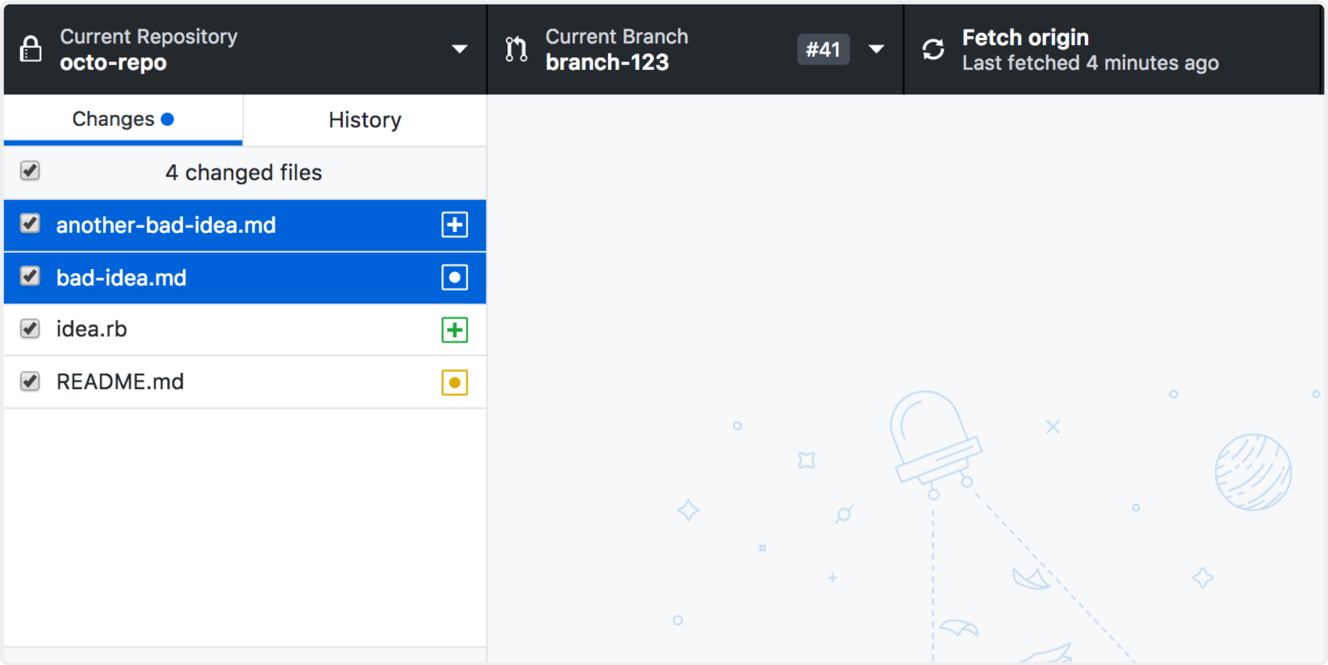


```
@@ -0,0 +1,14 @@
+class HelloWorld
+  def initialize(name)
+    @name = name
+  end
+  def hello
+    puts "Hello #{@name}"
+  end
+  def takeover
+    puts "I've taken control of"
+  end
+end
+
+hi = HelloWorld.new("World")
+hi.hello
```

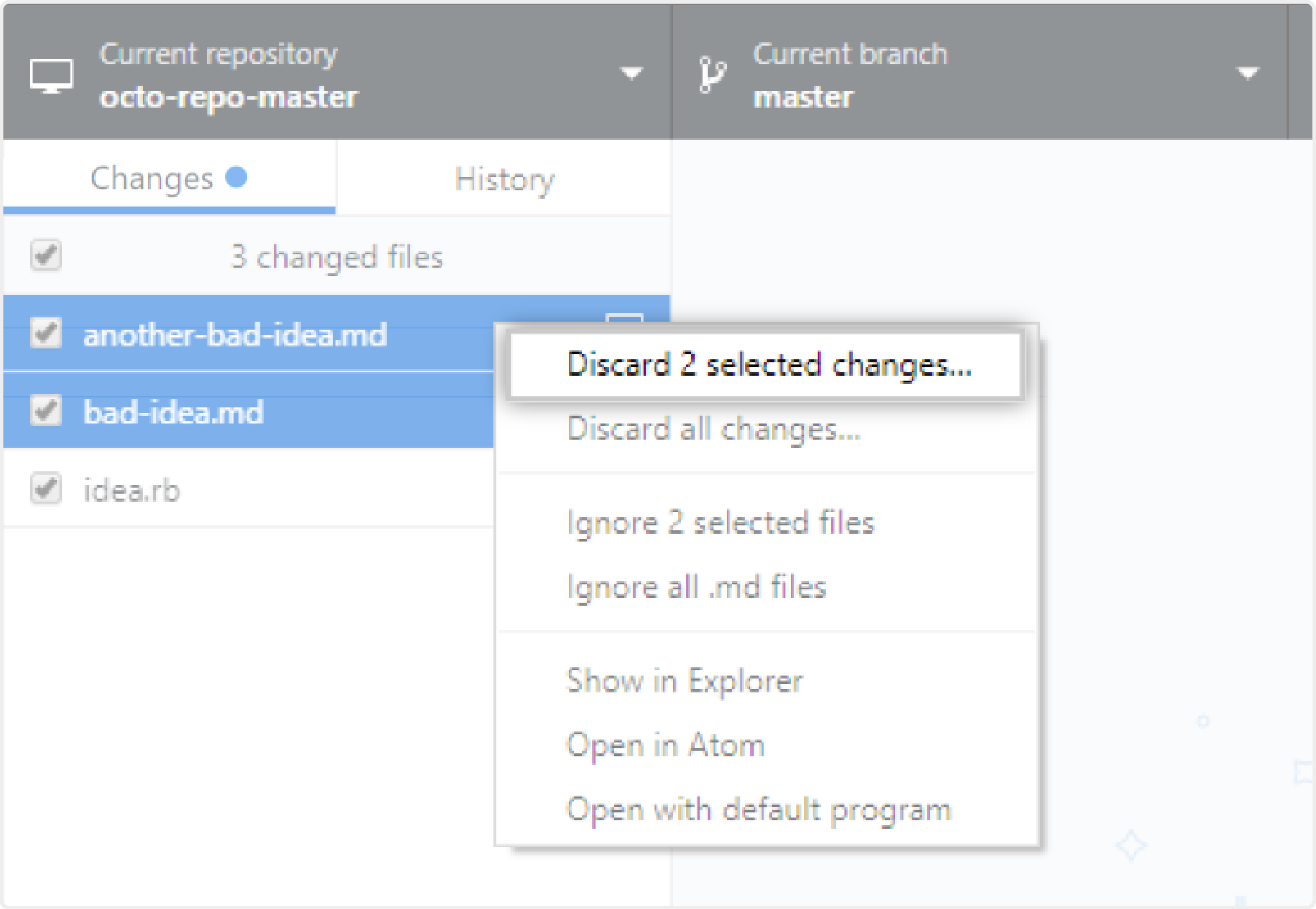
Discarding changes

You can discard all the uncommitted changes in one file, a range of files, or discard all changes in all files since the last commit.

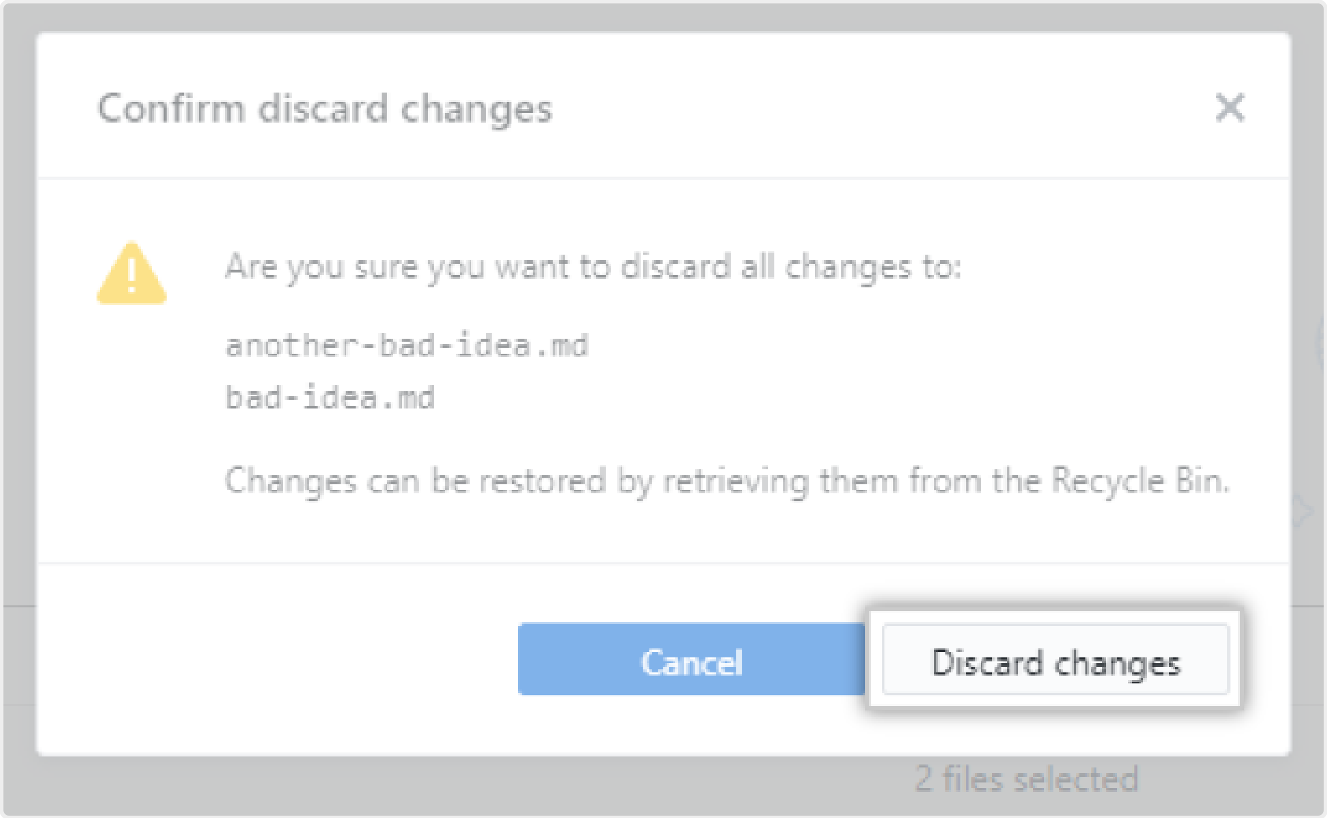
1 In the list of changed files, select the files where you want to discard the changes since the last commit. To select multiple files, click `shift` and click on the range of files you want to discard changes from.



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**.



Confirm discard changes

Are you sure you want to discard all changes to:

another-bad-idea.md
bad-idea.md

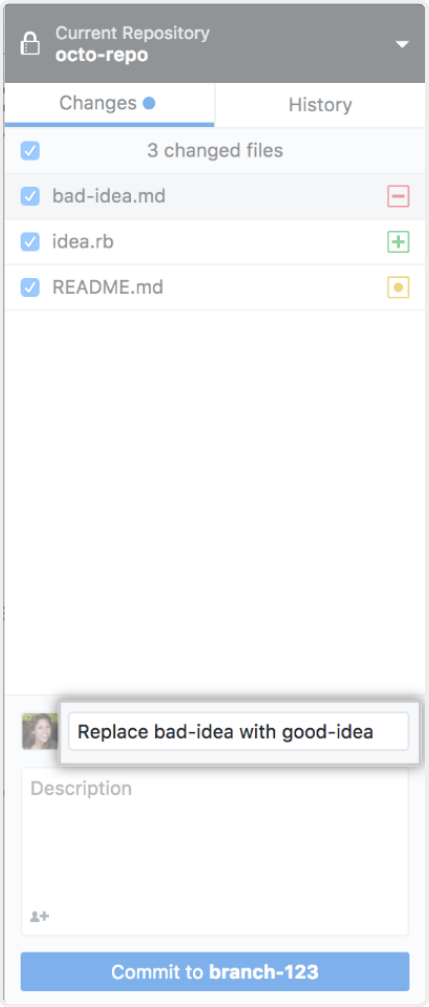
Changes can be restored by retrieving them from the Recycle Bin.

Tip: The changes you discarded are saved in a file in the Recycle Bin and you can recover them until it is emptied.

3. Write a commit message and push your changes

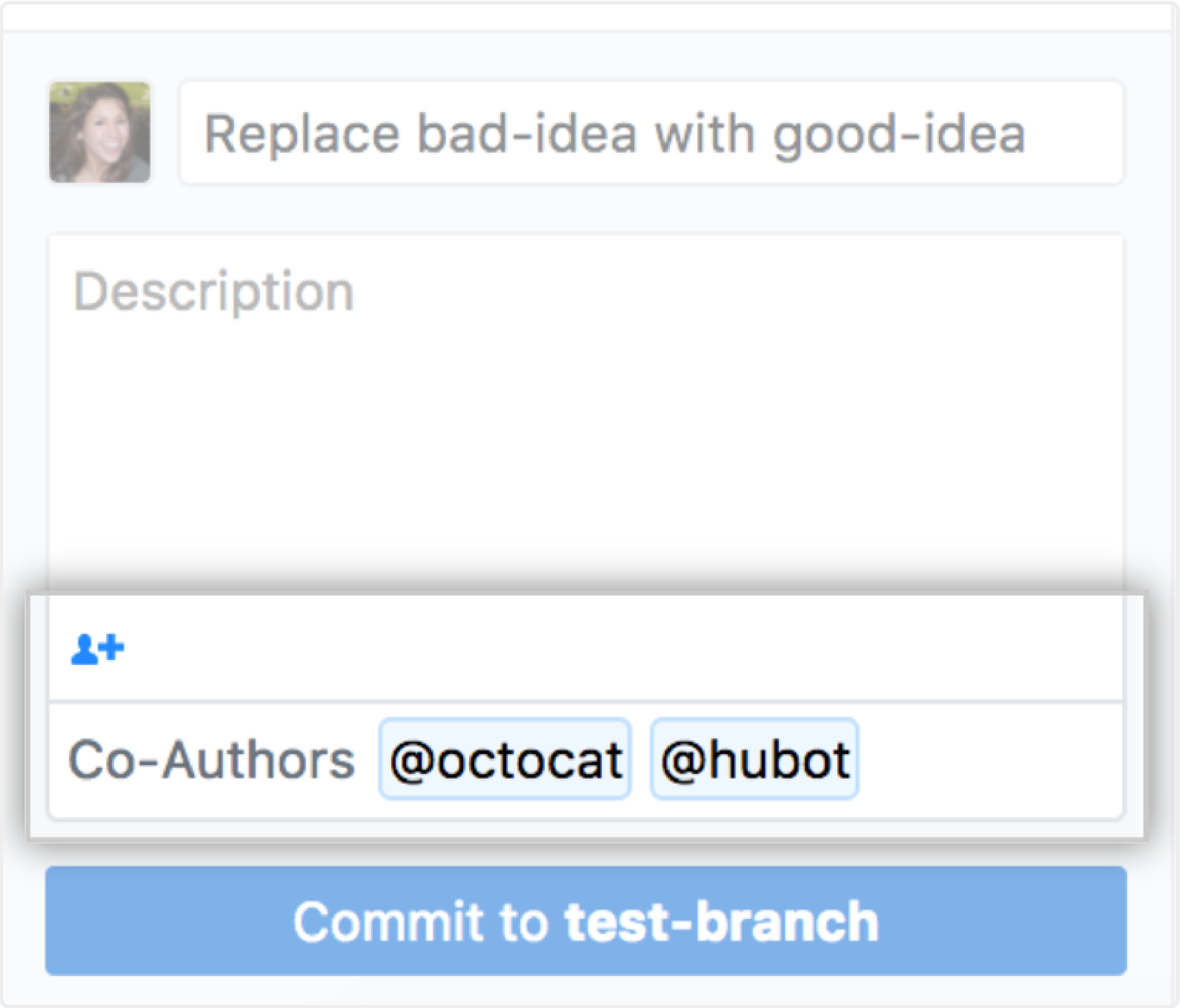
Once you're satisfied with the changes you've chosen to include in your commit, write your commit message and push your changes. If you've collaborated on a commit, you can also attribute a commit to more than one author.

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.



Replace bad-idea with good-idea

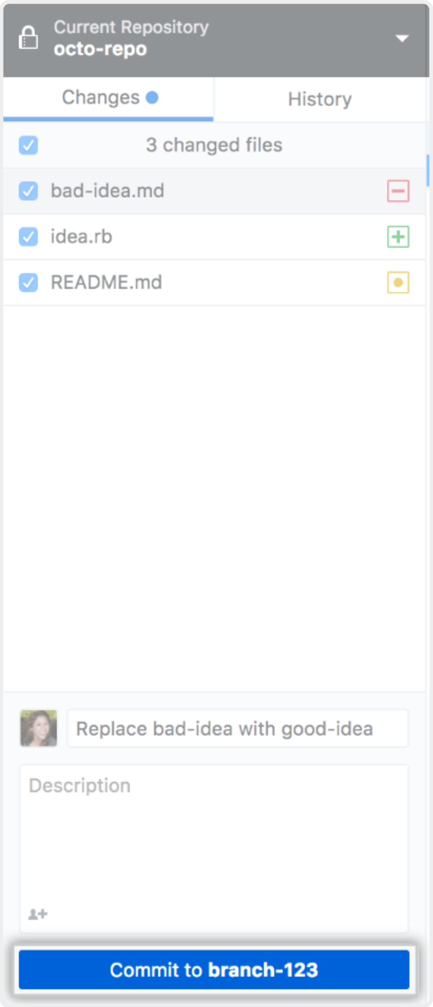
2 Optionally, to attribute a commit to another author, click the add co-authors icon and type the username(s) you want to include.



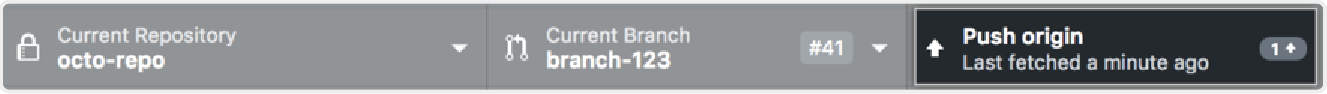
Replace bad-idea with good-idea

Co-Authors @octocat @hubot

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



4 Click 🡅 **Push origin** or **Force push origin** to push your changes to the remote.



Ask a human

Can't find what you're looking for?

Contact us

GitHub

Product

Features

Security

Enterprise

Case Studies

Pricing

Resources

Platform

Developer API

Partners

Atom

Electron

GitHub Desktop

Support

Help

Community Forum

Training

Status

Contact GitHub

Company

About

Blog

Careers

Press

Shop

Terms Privacy