Changing a commit message

In this article

If a commit message contains unclear, incorrect, or sensitive information, you can amend it locally and push a new commit with a new message to GitHub Enterprise Server. You can also change a commit message to add missing information.

Rewriting the most recent commit message

You can change the most recent commit message using the `git commit --amend` command.

In Git, the text of the commit message is part of the commit. Changing the commit message will change the commit ID--i.e., the SHA1 checksum that names the commit. Effectively, you are creating a new commit that replaces the old one.

Commit has not been pushed online

If the commit only exists in your local repository and has not been pushed to your GitHub Enterprise Server instance, you can amend the commit message with the `git commit --amend` command.

1. On the command line, navigate to the repository that contains the commit you want to amend.
2. Type `git commit --amend` and press **Enter**.
3. In your text editor, edit the commit message, and save the commit.
    - You can add a co-author by adding a trailer to the commit. For more information, see "Creating a commit with multiple authors."

The new commit and message will appear on your GitHub Enterprise Server instance the next time you push.

> You can change the default text editor for Git by changing the `core.editor` setting. For more information, see "Basic Client Configuration" in the Git manual.

Amending older or multiple commit messages

If you have already pushed the commit to your GitHub Enterprise Server instance, you will have to force push a commit with an amended message.

> We strongly discourage force pushing, since this changes the history of your repository. If you force push, people who have already cloned your repository will have to manually fix their local history. For more information, see "Recovering from upstream rebase" in the Git manual.

Changing the message of the most recently pushed commit

1. Follow the steps above to amend the commit message.
2. Use the `push --force-with-lease` command to force push over the old commit.

   ```
   git push --force-with-lease origin EXAMPLE-BRANCH
   ```

Changing the message of older or multiple commit messages

If you need to amend the message for multiple commits or an older commit, you can use interactive rebase, then force push to change the commit history.

1. On the command line, navigate to the repository that contains the commit you want to amend.
2. Use the `git rebase -i HEAD~n` command to display a list of the last `n` commits in your default text editor.

   ```
   # Displays a list of the last 3 commits on the current branch
   $ git rebase -i HEAD~3
   ```

   The list will look similar to the following:

   ```
   pick e499d89 Delete CNAME
   pick 0c39034 Better README
   pick f7fde4a Change the commit message but push the same commit.

   # Rebase 9fdb3bd..f7fde4a onto 9fdb3bd
   #
   # Commands:
   # p, pick = use commit
   # r, reword = use commit, but edit the commit message
   # e, edit = use commit, but stop for amending
   # s, squash = use commit, but meld into previous commit
   # f, fixup = like "squash", but discard this commit's log message
   # x, exec = run command (the rest of the line) using shell
   #
   # These lines can be re-ordered; they are executed from top to bottom.
   #
   # If you remove a line here THAT COMMIT WILL BE LOST.
   #
   # However, if you remove everything, the rebase will be aborted.
   #
   # Note that empty commits are commented out
   ```

3. Replace `pick` with `reword` before each commit message you want to change.

   ```
   pick e499d89 Delete CNAME
   reword 0c39034 Better README
   reword f7fde4a Change the commit message but push the same commit.
   ```

4. Save and close the commit list file.
5. In each resulting commit file, type the new commit message, save the file, and close it.
6. When you're ready to push your changes to GitHub, use the push --force command to force push over the old commit.

```
git push --force origin EXAMPLE-BRANCH
```

For more information on interactive rebase, see "Interactive mode" in the Git manual.

> As before, amending the commit message will result in a new commit with a new ID. However, in this case, every commit that follows the amended commit will also get a new ID because each commit also contains the id of its parent.

> If you have included sensitive information in a commit message, force pushing a commit with an amended commit may not remove the original commit from GitHub Enterprise Server. The old commit will not be a part of a subsequent clone; however, it may still be cached on GitHub Enterprise Server and accessible via the commit ID. You must contact your site administrator with the old commit ID to have it purged from the remote repository.

Further reading

- "Signing commits"