Resolving a merge conflict using the command line

MAC | WINDOWS | LINUX

You can resolve merge conflicts using the command line and a text editor.

Merge conflicts may occur if competing changes are made to the same line of a file or when a file is deleted that another person is attempting to edit. For information on how to resolve these situations, see "Competing line change merge conflicts and "Removed file merge conflicts."

Competing line change merge conflicts

To resolve a merge conflict caused by competing line changes, you must choose which changes to incorporate from the different branches in a new commit.

For example, if you and another person both edited the file *styleguide.md* on the same lines in different branches of the same Git repository, you'll get a merge conflict error when you try to merge these branches. You must resolve this merge conflict with a new commit before you can merge these branches.

1. Open Terminal.
2. Navigate into the local Git repository that has the merge conflict.

```
cd REPOSITORY-NAME
```

3. Generate a list of the files affected by the merge conflict. In this example, the file *styleguide.md* has a merge conflict.

```
$ git status
# On branch branch-b
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add ..." to mark resolution)
#
# both modified:      styleguide.md
#
no changes added to commit (use "git add" and/or "git commit -a")
```

4. Open your favorite text editor, such as Atom, and navigate to the file that has merge conflicts.
5. To see the beginning of the merge conflict in your file, search the file for the conflict marker `<<<<<<<`. When you open the file in your text editor, you'll see the changes from the HEAD or base branch after the line `<<<<<<< HEAD`. Next, you'll see `=======`, which divides your changes from the changes in the other branch, followed by `>>>>>>> BRANCH-NAME`. In this example, one person wrote "open an issue" in the base or HEAD branch and another person wrote "ask your question in IRC" in the compare branch or `branch-a`.

```
If you have questions, please
<<<<<<< HEAD
open an issue
=======
ask your question in IRC.
>>>>>>> branch-a
```

Article versions

GitHub.com
GitHub Enterprise 2.12
GitHub Enterprise 2.11
GitHub Enterprise 2.10
GitHub Enterprise 2.9

6. Decide if you want to keep only your branch's changes, keep only the other branch's changes, or make a brand new change, which may incorporate changes from both branches. Delete the conflict markers `<<<<<<<`, `=======`, `>>>>>>>` and make the changes you want in the final merge. In this example, both changes are incorporated into the final merge:

```
If you have questions, please open an issue or ask in our IRC channel if it's mo
```

7. Add or stage your changes.

```
$ git add .
```

8. Commit your changes with a comment.

```
$ git commit -m "Resolved merge conflict by incorporating both suggestions."
```

You can now merge the branches on the command line or push your changes to your remote repository on GitHub and merge your changes in a pull request.

Removed file merge conflicts

To resolve a merge conflict caused by competing changes to a file, where a person deletes a file in one branch and another person edits the same file, you must choose whether to delete or keep the removed file in a new commit.

For example, if you edited a file, such as *README.md*, and another person removed the same file in another branch in the same Git repository, you'll get a merge conflict error when you try to merge these branches. You must resolve this merge conflict with a new commit before you can merge these branches.

1. Open Terminal.

2. Navigate into the local Git repository that has the merge conflict.

```
cd REPOSITORY-NAME
```

3. Generate a list of the files affected by the merge conflict. In this example, the file *README.md* has a merge conflict.

```
$ git status
# On branch master
# Your branch and 'origin/master' have diverged,
# and have 1 and 2 different commits each, respectively.
#  (use "git pull" to merge the remote branch into yours)
# You have unmerged paths.
#  (fix conflicts and run "git commit")
#
# Unmerged paths:
#  (use "git add/rm ..." as appropriate to mark resolution)
#
#  deleted by us:   README.md
#
# no changes added to commit (use "git add" and/or "git commit -a")
```

4. Open your favorite text editor, such as Atom, and navigate to the file that has merge conflicts.

5. Decide if you want keep the removed file. You may want to view the latest changes made to the removed file in your text editor.

   To add the removed file back to your repository:

```
$ git add README.md
```

To remove this file from your repository:

```
$ git rm README.md
README.md: needs merge
rm 'README.md'
```

6. Commit your changes with a comment.

```
$ git commit -m "Resolved merge conflict by keeping README.md file."
[branch-d 6f89e49] Merge branch 'branch-c' into branch-d
```

You can now merge the branches on the command line or push your changes to your remote repository on GitHub and merge your changes in a pull request.

Further reading

- "About merge conflicts"
- "Checking out pull requests locally"

Contact a human