

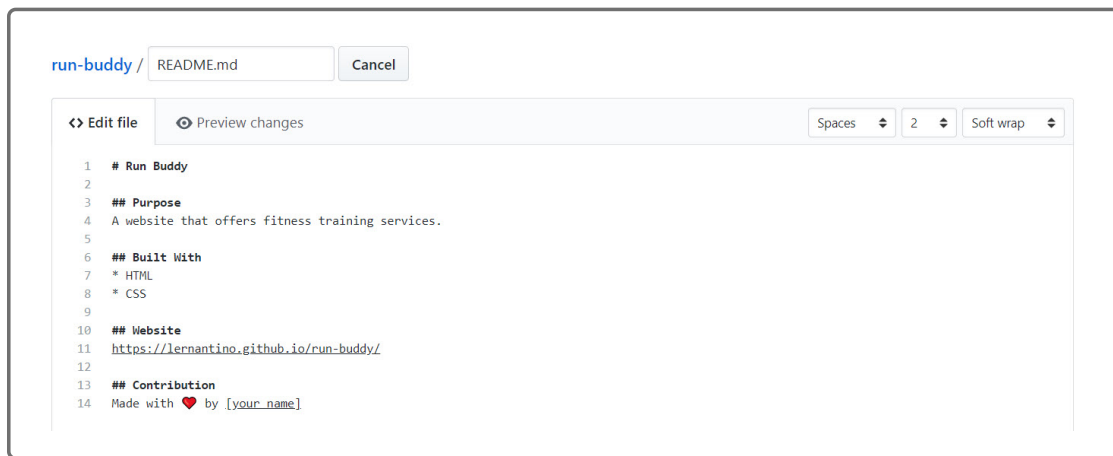
2.1.7 Practice Resolving a Merge Conflict

You can probably imagine that multiple developers working on all of these feature branches could lead to some conflicts when the time comes to merge them all together into the `develop` branch. What happens if two developers edit the same lines of code? How will Git know which one to accept and which one to reject?

These **merge conflict** scenarios are fairly common. When they occur, you can resolve them using the command line or you can use tools such as VS Code. Let's walk through how a merge conflict can be resolved in VS Code.

For this exercise, we'll pretend that Run Buddy's legal team wants us to add Run Buddy's copyright info to the `README.md` file on GitHub.

Go to the remote repo's landing page, switch to the `master` branch if you're not already there, and click the edit button or pencil icon. The "Edit file" pane will open:



Replace the text with the following Markdown:

```
# Run Buddy, Inc

## Purpose
A website that offers fitness training services.

## Built With
* HTML
* CSS

## Website
https://lernantino.github.io/run-buddy/

## Contribution
Made with ❤️ by [your name]

### ©2019 Run Buddy, Inc
```

Now commit those changes directly to the remote `master` branch by clicking the green "Commit changes" button shown in this screenshot:

Commit changes

Update README.md

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

Now that we've made changes to the remote `master` branch, let's make a quick change to our local `README.md` file in the local `master` branch. This will create the merge conflict.

First, use `git branch` and/or `git checkout master` to make sure you are on the correct branch.

Next, open the local `README.md` file and update the first line to look like this:

```
# Run Buddy Inc.
```

Save the `README.md` file, then add and commit these changes locally. Next, run the Git command to update the local branch with the remote `master` branch.

PAUSE

Is the Git command to update the local active branch a `pull` or `push`?

[Show Answer](#)

Altogether, you should run these three commands:

```
git add .  
git commit -m "update readme file"  
git pull origin master
```

The returning message in the command prompt should look something like this:

```
Local-Computer% git pull origin master  
remote:  
remote: Enumerating objects: 6, done.  
remote:  
remote:  
remote:  
remote:  
remote:  
remote:  
remote:  
remote: Counting objects: 100% (6/6), done.  
remote:  
remote:  
remote:  
remote:  
remote:  
remote: Compressing objects: 100% (4/4), done.  
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (4/4), done.  
From https://github.com/markjnkim/FSF0-module-2  
* branch          master      -> FETCH_HEAD  
  0b44c57..032fa61  master      -> origin/master  
Auto-merging README.md  
CONFLICT (content): Merge conflict in README.md  
Automatic merge failed; fix conflicts and then commit the result.
```

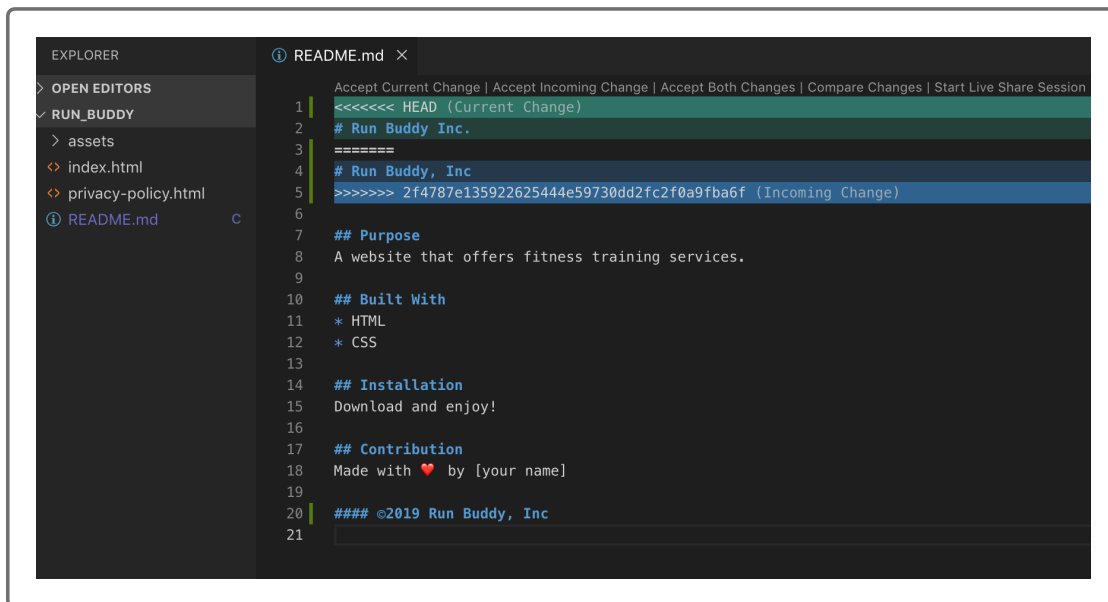
The most important lines in this corresponding message are the last two that say there is a merge conflict and the merge failed.

Let's go back to VS Code and open the `README.md` file. Can you figure out why this conflict occurred?

Git tracks all the changes in the files in both the remote and local repos. When Git sees that the same line has been modified in the same file in both versions, it can't identify which modification is correct so the merge fails. In this case, Git offers the developer an opportunity to decide how to proceed. This is called resolving a **merge conflict**.

To proceed with the merge, Git requires input from the user to determine which modification to keep—either the change on the active branch (our local change) or the incoming change from the target branch (in our case, the remote branch).

The following image shows how VS Code presents this conflict:



```
1 | <<<<<< HEAD (Current Change)
2 | # Run Buddy Inc.
3 | =====
4 | # Run Buddy, Inc
5 | >>>>> 2f4787e135922625444e59730dd2fc2f0a9fba6f (Incoming Change)
6 |
7 | ## Purpose
8 | A website that offers fitness training services.
9 |
10 | ## Built With
11 | * HTML
12 | * CSS
13 |
14 | ## Installation
15 | Download and enjoy!
16 |
17 | ## Contribution
18 | Made with ❤️ by [your name]
19 |
20 | ### ©2019 Run Buddy, Inc
21 |
```

Let's go over the three major areas of interest (note that the highlighting shown here is specific to the formatting in VS Code):

- The highlighted areas represent the two versions of the same code or text. The green highlight is labeled as the "Current Change." This represents the active branch modification. The blue highlight indicates the second version of the same conflicting code or text that is labeled "Incoming Change" noting it is coming from the target branch (in this case, the remote branch). To learn more about the notation for the HEAD and commit reference, please take a look at the [Atlassian docs about resolving merge conflicts](https://confluence.atlassian.com/bitbucket/resolve-merge-conflicts-704414003.html) (<https://confluence.atlassian.com/bitbucket/resolve-merge-conflicts-704414003.html>).
- Above the green highlight, several options to resolve this conflict appear. The first is to "Accept Current Change," which refers to the green highlighted code and represents the active branch modification. The second is to "Accept Incoming Change," which refers to the blue highlighted code or the target branch.
- Note that in the Explorer side panel, the `README.md` font color is purple. In a large project with an extensive folder tree, many files could be impacted by conflicts that must be resolved in order for the merge process to complete. VS Code makes this easier by highlighting the affected files and folders.

ON THE JOB

In the real world, merge conflicts typically occur when multiple team members work on the same file. As our example showed, even if the work is done on separate branches, if the same file has modifications in the same areas, conflicts can occur. You can take measures to prevent this, such as making sure team members strictly work on separate files when possible.

Let's resolve this conflict and decide which version to keep. In our case, we'll keep the "Incoming Change" from GitHub (the title with the comma before "Inc."). So go ahead and click the "Accept Incoming Change" option.

VS Code will now resolve two versions into one, eliminating the local change. Because this altered the file, you will need to save it again.

PAUSE

Why was the last line of code, `### ©2019 Run Buddy, Inc`, added without any conflict?

[Show Answer](#)

Notice how the purple highlight of the `README.md` file remained even though the merge conflict was resolved. Only when we add and commit the file changes will the merge conflict truly be resolved, so let's do that now:

```
git add .  
git commit -m "resolved merge conflict in title of README.md file"
```

Congratulations—you just navigated through your first merge conflict! This is a very typical process in software development, and knowing how to avoid and resolve these is an important skill that employers will seek.

Now that the local `master` branch is updated, our development environment needs an update. Checkout into the `develop` branch by using the command `git checkout develop`, then use `git merge master`.

Now our development environment is ready to go! Please note that it is not typical to make commits directly to the `develop` or `master` branch in a real-world team environment in a workplace. In our example, we created a merge conflict in the `README.md` so as not to alter the code in any way.

Now let's get back into our Git workflow and learn about a service provided by GitHub called Git issues.

© 2020 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.