(/learn)

Generate and Apply Patch

Learn what to do when a cherry-pick fails.

We'll cover the following

- Generate patch
- Apply patch

Generate patch

So far you've seen that the cherry-pick didn't do what you wanted, and you've seen why. Now you're going to use a different technique to achieve what you want.

First, revert the cherry-pick you started:

```
1   git cherry-pick --abort
```

Terminal 1

Terminal

Click to Connect...

Now, you're going to do two things:

- Create a patch file that has the diff contained in the `abranchtag` commit within it.
- Apply that diff to the `master` branch.

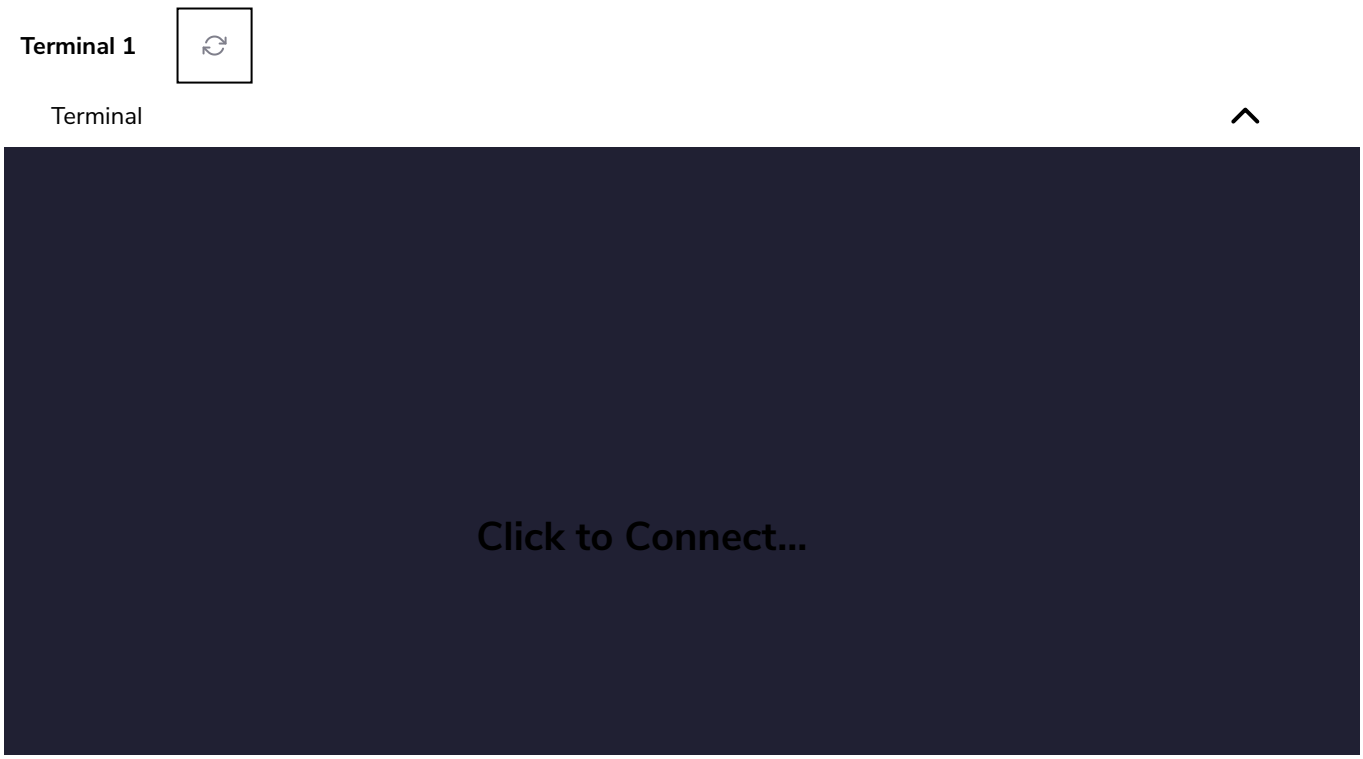
Patch files were originally created to pass changes round by email. The `git format-patch` command does this by default, creating a patch ready for email submission.

There are only a few people that do that these days, and we're not emailing this change, so you're going to use the `git diff-tree` command.

To create a patch file, run this:

```
2   git diff-tree -p abranchtag > abranchtag.patch
```

Take a look at the contents of that file by using `cat abranchtag.patch` command.



As usual, the identifiers will differ. But with a little patience, you will see the changes. Try and figure out what the `+`, `-`, and `@` symbols in the above output mean. To find out in more detail, the unified diff format used above is documented here (https://www.gnu.org/software/diffutils/manual/html_node/Detailed-Unified.html#Detailed-Unified).

Apply patch

Now that you have the patch for that commit (and only that commit), you can apply it to the master branch:

```
3   cat abranchtag.patch | git apply
```

...but that fails!

It fails because there was a conflict. Take a moment to try and work out why.

Git could not apply the diff because the addition of the `Second change, on abranch` line to the `afile` has a line above it (`First change, on abranch`), which is just not present in the same file on the `master` branch. Git doesn't know what to do and gives up trying to apply the patch.

If you want to check what's going on at this point, run some `git status` and `git log` commands to check that the master branch is still in a good state. You will see that no changes were made to the repository.

To apply the diffs that can be applied, run the same command with the `--`

`reject` flag:

```
4   cat abranchtag.patch | git apply --reject
```

Terminal 1

Terminal

This time, changes were applied. The conflict you saw before was put in a `.rej` file. The `.rej` file is created with the name of the file the conflict relates to and the `.rej` appended. In this case, the file created was `afile.rej`.

Take a look at it:

```
5   cat afile.rej
```

At this point, you have a couple of choices about what to do. You can try to adjust the `.rej` file and re-apply it using `git apply`. Or just inspect the conflict described in the diff, and apply the change directly to the file.

The second approach is the one I invariably take, as fiddling with patch files directly is not for the faint-hearted!

You can finish the job yourself; apply the change manually, and then commit the changes, and inspect your repository to understand what you've done.

Back

Attempting to Apply Cherry-Pick

Next

Introduction: Git Hooks

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/generate-and-apply-patch__cherry-picking-and-three-way-merges__learn-git-the-hard-way)