Lab: (Command Line) Rewriting History

Estimated time: 10 minutes

> Note: This lab assumes that you are using a command line. If you would prefer to use Sourcetree, there are separate instructions.

In this lab, you will:

1. Amend a commit.
2. Use interactive rebase to remove a commit.
3. Perform a squash merge.

1: Amend a commit.

1. Create a local repository named `projecth`.

2. Create an initial commit that adds a `fileA.txt` file containing the text "mistake". The commit message should be "add fileA.txt with mistake". Note the first to characters of the commit's SHA-1.

3. Amend the commit so that `fileA.txt` is empty and the commit message simply reads "add fileA.txt".

    - Modify `fileA.txt` so that it is empty.
    - Add `fileA.txt` to the staging area.
    - Amend the previous commit by executing `git commit --amend -m "add fileA.txt"`.

4. Notice the SHA-1 of your amended commit. It should be different than the one you noted prior to amending the commit.

    > Congratulations, you have amended a commit.

2: Use interactive rebase to remove a commit.

1. Create a `feature1` branch.

2. Create a commit on the `feature1` branch. Add the string "feature 1 wip" to `fileA.txt`. The commit message should be "add feature 1 wip".

3. Create another commit on the `feature1` branch. The contents of `fileA.txt` should be "feature 1". The commit message should be "add feature 1".

4. Let's say that you think that the "feature 1 wip" commit adds little value to the commit history. Use interactive rebase to remove that commit by squashing it.

    - Checkout the `feature1` branch.
    - View the commit graph of the `feature1` branch.
    - Execute `git rebase -i <SHA-1>`, where is for the initial commit of the repository (which is also the first commit of the `feature1` branch).
    - Explore the interactive rebase editor.
    - Remove the commit with the commit message of "feature 1 wip". Do this by changing the `pick` command to `squash` for the commit with a commit message of "add feature 1". Save the file.
    - An editor then opens for the commit message of the squashed commit. Change it to simply read `add feature 1`.

5. View your commit graph. Verify that the commit with a message of "feature 1 wip" has been removed.

6. Merge the `feature1` branch and delete its branch label.

    > Congratulations, you have performed an interactive rebase.

3: Perform a squash merge.

1. Create a `feature2` branch off of the `master` branch.

2. Create a commit on the `feature2` branch. Add the string "feature 2 wip" to `fileA.txt`. The commit message should be "add feature 2 wip".

3. Create another commit on the `feature2` branch. In `fileA.txt`, change the line "feature 2 wip" to "feature 2". The commit message should be "add feature 2".

4. Perform a squash merge of the `feature2` branch into the `master` branch.

    - Checkout the `master` branch.
    - Execute `git merge --squash feature2`.
    - Execute `git commit`. An editor opens. Specify a commit message of "add feature 2".

5. View your commit graph. Notice that the commit with a message of "feature 2 wip" is not part of the `master` branch.

6. Delete the `feature2` branch label. This will create two "dangling" commits that will be garbage collected.

7. You will not use the `projecth` repository in future labs. You can delete it.

> Congratulations, you have performed a squash merge and have completed this lab.

Copyright © 2018 Atlassian