(/learn)

Additional Information on Hooks

Learn if hooks are part of Git content, and learn about “pre-receive” hooks.

We'll cover the following

Are hooks part of Git content?

A question you may be asking yourself at this point is whether the hooks are part of the code or not. You wouldn’t have seen any mention of the hooks in your commits; so does it move with the repository as you commit and push or not?

An easy way to check is to look at the remote bare repository directly.

```
cd ../git_origin
ls hooks
```

Terminal 1

Terminal

Click to Connect...

Examining the output of the above will show that the `pre-commit` script is not present on the bare origin remote. It does not move with the code nor is it considered part of the code committed.

This presents a problem if you are working in a team. If the whole team decides that they do not want any mention of politics in their commits, then they will have to remember to add the hook to their local clone. This isn’t very practical.

But if you (by convention) have a single origin repository, then you can prevent commits from being pushed to it by implementing a `pre-receive` hook. These are a little more complex to implement but arguably more useful, as they can enforce rules per team on a canonical repository.

The `pre-commit` hook you saw before is an example of a “client-side hook” that sits on the local repository. Next, you’ll look at an example of a “server-side hook” that is called when changes are received from another Git repository.

Pre-receive hooks

Type this out, and then I’ll explain what it’s doing. As best you can, try and work out what it’s doing as you go, but don’t worry if you can’t figure it out.

```
cat > hooks/pre-receive << 'EOF'
> #!/bin/bash
> read _oldrev newrev _branch
> git cat-file -p $newrev | grep '[A-Z][A-Z]*-[0-9][0-9]*'
> EOF
```

Terminal 1

Terminal

This time you created a `pre-receive` script, which will be run when anything is pushed to this repository. These `pre-receive` scripts work in a different way to the `pre-commit` hook scripts because the `pre-commit` script allows you to `grep` the content that was being committed; `pre-receive` scripts do not. This is because the commit has been packaged up by Git, and the contents of the commit are delivered up as that packaged content.

The `read` command in the above code is the key one to understand. It reads three variables: `_oldrev`, `newrev`, and `_branch` from standard input (i.e., the data that is coming in to the script). The contents of these variables will match. The previous Git revision reference commit refers to the branch it is committed on. The new Git revision reference commit refers to the branch it is committed on. Git arranges that these references are given to the `pre-receive` script on standard input so that action can be taken accordingly.

Then, you use the (previously unseen in this course) `git cat-file` command to output details of the latest commit value stored in the `newrev` variable. The output of this latest commit is run through a `grep` command that looks for a specific string format in the commit message. If the `grep` finds a match, then it returns no error and all is OK. If it doesn't find a match, then `grep` returns an error as does the script.

Make the script executable:

```
chmod +x hooks/pre-receive
```

Terminal 1

Terminal

Then make a new commit and try to push it:

```
cd ../git_clone
echo 'another change' >> file1
git commit -am 'no mention of ticket id'
git push
```

That should have failed, which is what you wanted. The reason you wanted it to fail is buried in the `grep` you typed in:

```
    grep '[A-Z][A-Z]*-[0-9][0-9]*'
```

This `grep` only returns successfully if it matches a string that matches the format of a JIRA ticket ID (e.g., `PROJ-123`). The end effect is to enforce that the last commit being pushed must have a reference to such a ticket ID for it to be accepted. You might want such a policy to ensure that every set of commits can be traced back to a ticket ID.

Back
A More Sophisticated Example

Next
Assessment #4 - Advanced Git

✓ Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/additional-information-on-hooks__git-hooks__learn-git-the-hard-way)