We have CI at home. CI at home:

Git Hooks

October 19th, 2023

Presented by

Tales da Aparecida

tdaapare@redhat.com

Tales da Aparecida

tdaapare@redhat.com

🇧🇷

Senior Software Engineer

Continuous Kernel Integration (CKI) Project

We have CI at home. CI at home: **Git Hooks**

1 Git

Version Control System

Source Code Management tool

https://git-scm.com

2 Hooks

Scripts/Executables

Exit codes

Arguments + stdin + worktree files

> man 5 githooks

Hooks are programs triggered at certain points in git's execution.

E.g. `git checkout`, `git commit`, `git rebase`, `git push`, `git am`

There are 28 documented hooks.

Every git repository comes with samples for 13 of them under `.git/hooks`.

I can get by using only 3, at most!

Skip with `--no-verify`

```
man git-hook
Covers the CLI to run git hooks
```

> man 5 githooks

1. `prepare-commit-msg`: Executed by `git commit` after preparing the default commit message but before the editor opens. It allows you to edit the message file.
2. `commit-msg`: Invoked by `git commit`. It enables you to validate and modify commit messages, such as checking for duplicate Signed-off-by trailers.
3. `pre-commit`: Invoked by `git commit`, it runs before committing. You can use it to perform checks like trailing whitespaces.
4. `post-commit`: This hook can be used to send some kind of post-commit notification after running `git commit`.
5. `pre-push`: Triggered by `git push`, it can prevent pushes by inspecting the push details. It's useful for access control and ensuring a clean push.

> man 5 githooks

6. `pre-rebase`: Called by `git rebase`, it can prevent branches from being rebased. It is useful for branch protection.
7. `post-rewrite`: Executed by commands that rewrite commits, such as `git commit --amend` and `git rebase`. It provides information on rewritten commits and notes.
8. `post-checkout`: Invoked by `git checkout` after updating the worktree. It provides information about the change and can be used for repository validity checks or notification.
9. `pre-merge-commit`: Invoked by `git merge` before creating a merge commit. It is useful to run pre-commit checks.
10. `post-merge`: Invoked by `git merge` during a local repository git pull. You can use it to save and restore metadata associated with the working tree.

> man 5 githooks

Email Workflow

11. `applypatch-msg`: This hook is invoked by `git-am(1)`. Similar to `commit-msg`, it allows you to modify or reject commit messages, ensuring they adhere to project standards.
12. `pre-applypatch`: Invoked by `git-am(1)`, it runs after a patch is applied but before a commit is made. Similar to `pre-commit`, you can use it to inspect the working tree and prevent a commit if necessary.
13. `post-applypatch`: Triggered by `git-am(1)`, similar to `post-commit`, this hook serves as a notification after a patch is applied and a commit is made.
14. `sendemail-validate`: Invoked by `git-send-email(1)` for email validation, similar to `pre-push`. You can use it to ensure email content is valid.

man 5 githooks

Random things

15. `pre-auto-gc`: Git occasionally does garbage collection as part of its normal operation, by invoking `git gc --auto`. This hook can be used to abort the collection if now isn't a good time.
16. `reference-transaction`: This hook is invoked by any Git command that performs reference updates. It executes when a transaction is prepared, committed, or aborted, covering various states and transactions.
17. `post-index-change`: Triggered when the Git index is written in `read-cache.c` in the `do_write_locked_index` function. It provides information about whether the working directory and index were updated and if skip-worktree bits changed.
18. `fsmonitor-watchman`: A hook for integration with the Watchman tool when the core.fsmonitor configuration is set to `.git/hooks/fsmonitor-watchman` or `.git/hooks/fsmonitor-watchmanv2`. It communicates changes in the working directory to Git.

> man 5 githooks

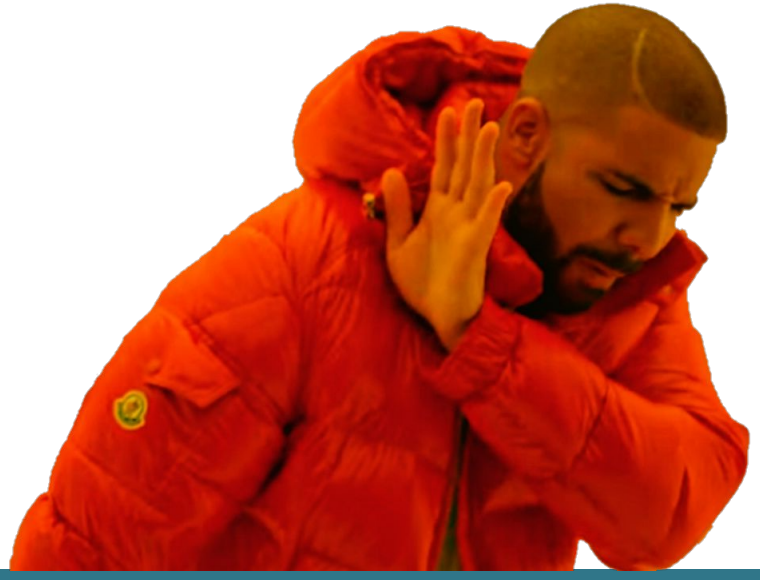
p4 support (other VCS called Perforce)

19. `p4-changelist`: Used by `git-p4 submit` to validate changelist messages. It can edit and normalize changelist text or reject the submit.
20. `p4-prepare-changelist`: Invoked by `git-p4 submit` after preparing the default changelist message but before the editor opens. It allows you to edit the message file before submission.
21. `p4-post-changelist`: This hook is called after a successful submit in Perforce. It's meant for notification and doesn't affect the outcome of the `git-p4 submit` action.
22. `p4-pre-submit`: Invoked by `git-p4 submit`. It performs pre-submit checks and can prevent the `git-p4 submit` action from launching.

> man 5 githooks

Server-side hooks (might cause pushback)

23. `pre-receive`
24. `proc-receive`
25. `post-receive`
26. `update`
27. `post-update`
28. `push-to-checkout`



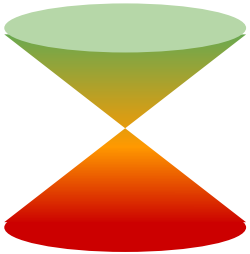
Start with pre-commit

Incremental steps

I. Spell checking
II. Formatting code
III. Linting
IV. Testing

Avoid long running tasks.

Static analysis



Dynamic analysis

Start with pre-commit

```
#!/usr/bin/env bash
# ^ Note the above "shebang" line. This says "This is an executable shell script"
# Name this script "pre-commit" and place it in the ".git/hooks/" directory

# If any command fails, exit immediately with that command's exit status
set -eo pipefail

# Run flake8 against all code in the `source_code` directory
flake8 source code
echo "flake8 passed!"

# Run black against all code in the `source_code` directory
black source code --check
echo "black passed!"
```

Source: https://verdantfox.com/blog/how-to-use-git-pre-commit-hooks-the-hard-way-and-the-easy-way


What if I want to
share

Advance with pre-commit

"A framework for managing and maintaining multi-language pre-commit hooks."

- Community
- Version control
- Customization

pre-commit.com

Advance with **pre-commit**

.pre-commit-hooks.yaml

```yaml
-   id: flake8
    name: flake8
    description: |
        `flake8` is a command-line utility for enforcing
        style consistency across Python projects.
    entry: flake8
    language: python
    types: [python]
    require_serial: true
```

Advance with pre-commit

.pre-commit-config.yaml

```yaml
repos:
-   repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.5.0
    hooks:
    -   id: end-of-file-fixer
    -   id: trailing-whitespace
        exclude: ^tests/fixtures/
-   repo: https://github.com/PyCQA/flake8
    rev: 6.1.0
    hooks:
    -   id: flake8
-   repo: https://github.com/psf/black
    rev: 23.9.1
    hooks:
    -   id: black
        args: [--line-length=79]
```


Is that
safe?

It can be!

Read the code

Pin a version, or even better,

Fork it

THANK YOU

Q&A