Minimal Git CI using hooks

I have a personal server where I am hosting my git repos. Setting up a new git repo is easy: I ssh into my server, cd to a new directory and run `git init --bare`. That's it. Cloning my newly created repo is simple as `git clone server:/home/me/repo`.

For some projects I've needed a CI, and I looked around for some existing solutions. The CI systems I found are convoluted, yaml-ridden, slow, messy or hard to self-host.

Luckily, I do not need many "modern" CI features like complete isolation for my builds or secret management. I just need something that runs some tests, builds a project or moves some files.

What I have settled on is a simple CI using git hooks. Specifically, I added a remote `post-receive` hook, which is a hook that runs whenever I push to a project. It is just a shell-script that I add to the `hooks` directory in a bare git repository.

The thing to watch out for with `post-receive` hooks is if the script fails, the code you push will be rejected. Also, if the script is slow, pushing will take a while. You generally don't want either. I solved this by using `nq`, a minimal job queue, to queue up my jobs in the background instead. The result is quite sleek: my post-receive hook just executes `nq`, and I can access the logs by running `ssh server nqtail -a`.

The CI runs instantly, and is very easy to set up, so I am happy with the result. If you'd like to set it up for yourself, I created a short tutorial.

There are many ways you could expand the base I provide in the tutorial: you could sandbox the builds (for example with landdown), use podman to isolate builds from the environment, or use sops to manage secrets. For bazaar-style development, receiving git patches via email should be enough. You can also set up cathedral-style development using git-shell or git http-backend.

⇒ Reply
