Continuous integration with GitHub Actions

Contents

In this tutorial, we'll show you **a few short steps** to get started using GitHub Actions as your continuous integration (CI) workflow for commits and pull requests.

One benefit of Nix is that **CI can build and cache developer environments for every project** on every branch using binary caches.

An important aspect of CI is the feedback loop of, **how many minutes does the build take to finish?**

There are a several good options, but Cachix (below) and integrating with GitHub's built-in cache (at the end) are the most straightforward.

Caching builds using Cachix

Using Cachix you'll never have to waste time building a derivation twice, and you'll share built derivations with all your developers.

After each job, just-built derivations are pushed to your binary cache.

Before each job, derivations to be built are first substituted (if they exist) from your binary cache.

Skip to main content

1. Creating your first binary cache

It's recommended to have different binary caches per team, depending who will have write/read access to it.

Fill out the form on the create binary cache page.

On your freshly created binary cache, follow the **Push binaries** tab instructions.

2. Setting up secrets

On your GitHub repository or organization (for use across all repositories):

1. Click on `Settings`.
2. Click on `Secrets`.
3. Add your previously generated secrets (`CACHIX_SIGNING_KEY` and/or `CACHIX_AUTH_TOKEN`).

3. Setting up GitHub Actions

Create `.github/workflows/test.yml` with:

```yaml
name: "Test"
on:
  pull_request:
  push:
jobs:
  tests:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - uses: cachix/install-nix-action@v25
      with:
        nix_path: nixpkgs=channel:nixos-unstable
    - uses: cachix/cachix-action@v14
      with:
        name: mycache
        # If you chose signing key for write access
        signingKey: '${{ secrets.CACHIX_SIGNING_KEY }}'
        # If you chose API tokens for write access OR if you have a private cache
        authToken: '${{ secrets.CACHIX_AUTH_TOKEN }}'
    - run: nix-build
    - run: nix-shell --run "echo OK"
```

Skip to main content

Once you commit and push to your GitHub repository, you should see status checks appearing on commits and PRs.

Caching builds using GitHub Actions Cache

A quick and easy way to speed up CI on any GitHub repository is to use the Magic Nix Cache. The Magic Nix Cache doesn't require any configuration, secrets, or credentials. This means the caching benefits automatically work for anyone who forks the repository.

One downside to the Magic Nix Cache is it only works inside GitHub Actions. For more details, check out the readme and the limits of GitHub Actions caching.

Create `.github/workflows/test.yml` with:

```yaml
name: "Test"
on:
  pull_request:
  push:
jobs:
  tests:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - uses: cachix/install-nix-action@v25
      with:
        nix_path: nixpkgs=channel:nixos-unstable
    - uses: DeterminateSystems/magic-nix-cache-action@v2
    - run: nix-build
    - run: nix-shell --run "echo OK"
```

Next steps

- See GitHub Actions workflow syntax
- To quickly setup a Nix project read through Getting started Nix template.