Dagger

How Flipt Improved Coverage and Build Times with Dagger

Flipt

Industry
GitOps-enabled feature management platform

Website
flipt.io

> ***Dagger feels like an actual innovation in CI/CD. There's a real attention to getting feedback from everyone and feeding that back into the design and building something that people need."***

> ***Open source is a value that's super important to us. Flipt being 100% open source and the value alignment with Dagger in that respect just made a lot of sense to us.***

We're always interested in hearing how our community is using Dagger – their use cases, their challenges, and their experiences with deploying Dagger in different environments. Our Discord is a great place to find these stories, and to benefit from the knowledge and experience of the Dagger community.

In this blog post, we'll share the story of Flipt, an open source, self-hosted feature flag solution. Flipt uses Dagger as an integral part of its CI/CD pipeline. With Dagger, Flipt has seen its build pipeline with subsequent integration tests go from 10 minutes with an empty cache to 17 seconds on a subsequent run!

The problem: Networking and language limitations for CI testing

Flipt stores its source code in a Go multi-module repository and uses a Go workspace to bring the modules together. Initially, Flipt's CI/CD pipeline used GitHub Actions for both unit and integration tests. The pipeline used a matrix strategy to test different configurations of feature flags and configurations.

There were two main challenges with this approach.

- Test scripts were written in Bash. Flipt, however, is written entirely in Go. This created an inconsistency, wherein application code was written in one language and application tests in a different language.
- Test scripts were executed in separate VMs by Github Actions, which was cumbersome. They had brittle little bash scripts for backgrounding the Flipt process and then running the tests against them.

The Flipt team had previously experimented with Dagger's CUE-based implementation in other projects. Dagger's release of a Go SDK gave them a reason to rethink their CI/CD strategy and consider re-implementing it in Go with Dagger. Dagger being open-source also aligned closely with Flipt's own values.

The solution: Fast, full-featured native-language CI testing with Dagger

The Flipt team currently uses Dagger for both build and test phases of its pipeline, handing off to GoReleaser for the release phase.

- In the build phase, the Dagger pipeline downloads the application source code, adds required dependencies and build tools, and builds the application. The end result of the build phase is two artifacts: a "test" image with the source code plus build and test dependencies, and a "production" image optimized for distribution.
- In the test phase, the Dagger pipeline runs Flipt's complete suite of unit and integration tests using the Go SDK, validating the test image against a common set of Flipt operations. In addition to performing complete end-to-end API testing, the pipeline also performs additional tests related to importing, validating and exporting feature flag configurations. These tests are performed against a variety of databases (including Postgres, MySQL and CockroachDB).

Flipt has seen several benefits after switching to Dagger:

- Native Go support makes it easier for the team to expand the scope of the test suite with additional configurations and permutations -for example, tests against the HTTP or gRPC API, tests with or without authentication enabled, and so on. Since tests are now written in Go, it's much easier for anyone in the team to add new tests or improve existing tests.
- Dagger's built-in support for container networking enables testing to be performed against an isolated, containerized instance of the Flipt API instead of the live API.
- Dagger's built-in support for service containers makes it possible to test against an isolated, containerized instance of the Flipt API instead of the live API.
- Dagger's built-in support for service containers makes it possible to test against multiple database engines, and to quickly add test support for a new database engine.
- The same Dagger pipeline works consistently both locally and remotely, reducing the feedback loop when modifying the pipeline - for example, when adding tests for a new database engine or a new feature flag.

> ***We've got a distributable that can be configured in many ways. We want to test that it still works even when we change the configuration, or when we configure it in different ways. Being able to encode those requirements in Go is a very nice experience."***

Perhaps the biggest benefit, though, is in application build and test times. Flipt's pipeline is fairly large and complex. However, by leveraging Dagger's engine caching and mounted cache volumes, Flipt is able to avoid unnecessary re-downloading or re-computing of assets. When Flipt is compiled with an empty build cache, a fresh run will produce 1 GB of build-cache entries and take approximately 10 minutes. This drops to approximately 17 seconds on subsequent runs, thanks to Dagger's caching...an improvement of ~97%!

The future: Release packaging, profiling and remote caching with Dagger

Flipt currently uses GoReleaser to package and distribute the artifacts generated by the Dagger pipeline. In the near future, the team plans to completely integrate Dagger into the CI pipeline, including for the release phase. Dagger's multi-platform support will be key here, to build and release the Flipt distributable for different architectures.

Flipt is also experimenting with importing and merging Go profiling data from the Dagger pipeline to get a deeper understanding of test coverage and performance for integration tests. And the team is keen to improve the CI pipeline further - for example, by using Dagger to connect to various remote storage engines for caching.