Dagger

Dagger Cloud: Going 100% Faster, Spending 75% Less

Airbyte

Industry
data integration

Website
airbyte.com

> ***"Dagger is the first thing we've seen that looks at the way CI/CD works and actually tries to turn some of the ideas on their head, which I think is pretty exciting. There's potential to dramatically improve the way our entire software-development supply chain works."***

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 Daggernauts Augustin Lafanechère (aka @alafa on Discord, alafanechere on GitHub) and Conor Barber (aka @conorba on Discord, cpdeethree on Github). Both work at Airbyte, an open-source ELT platform managing hundreds of Docker containers. Augustin is part of the Connector Operations team responsible for managing CI for Airbyte connectors, while Conor works in the Platform Infrastructure team. Over the last few months, Augustin and Conor have transformed Airbyte's maze of GitHub Actions, YAML, and shell scripts into a streamlined CI/CD process powered by Dagger and significantly enhanced Airbyte's developer experience.

Legacy CI: A Maze of YAML and Shell Scripts

Airbyte has an extensive ecosystem of 350+ containerized connectors, all of which need to be tested regularly. On average, Airbyte publishes 17 connectors per day and tests 100+ connectors per night, both as part of regularly scheduled testing cycles and in response to incoming pull requests. Airbyte also regularly tests connectors even if the code hasn't recently changed - they ensure that upstream APIs haven't changed and still work as expected.

Every new connector needs a new test suite; as a result, build and test requirements scale linearly with the number of connectors.

Being able to consistently and reliably build, test, and publish connectors while also ensuring visibility to both platform and development teams is, therefore, a considerable CI/CD challenge.

> ***Not being able to debug locally when there are problems is a pretty major pain point. We were looking for a better way."***

Before Dagger, Airbyte's CI consisted of GitHub Actions YAML, shell scripts and Gradle scripts. Although the system worked, the overall team satisfaction was low:

- The GitHub Actions tooling was a maze of YAML and shell scripts that were slow to execute and difficult to maintain.
- The Gradle tooling was powerful, but complex to understand and use. It worked well for Java builds, but less so for Docker- and Python-related tasks (of which there were many).
- If a CI pipeline failed, developers had no way to simply replicate the failure locally to identify the error and had to spend time and effort searching the previously-mentioned maze for the source of the error.
- The CI scripts had been written by different team members at different times and therefore had no clear owner.
- The outsized nature of Airbyte's test requirements meant
- more, and more complex, CI pipelines and consequently high server costs.

> ***We decided on Dagger because of the expressiveness of being able to do things completely in code. Being able to write things in a more simple, imperative style makes it a lot easier to understand and test."***

"Daggerized" CI: Auto-Scaling Runners with Dagger Cloud Distributed Caching

The Airbyte team investigated various other orchestration tools to solve these problems. In late 2022, Dagger's release of a Python SDK made it the most appealing choice, as Airbyte was already using Python extensively. The team started transitioning to Dagger in February 2023 and ran their first Dagger pipeline in production in May 2023.

The new system employs remote Dagger Engines on a Kubernetes cluster leveraging Dagger Cloud to utilize platform-neutral caching strategies, optimize start-up times, and scale flexibly according to load. Here's how it works:

- Airbyte uses a slim GitHub Action which responds to repository events (pull requests, pushes, merges, ...) by triggering a custom command-line wrapper (CLI) around Dagger. This CLI is used for both remote CI and local development - Airbyte finally had a way to replicate the CI experience locally, and it was fast!
- Dagger, via this CLI wrapper, takes care of run orchestration and pipeline invocation.

> ***Before Dagger it was 'Push and Pray.' Now, we have the same tool running locally and in CI, and that just streamlines everything. Now our mantra is test, test, test, test."***

> ***"We have made a valuable shift. Our CI pipelines are now expressed as code, so refactoring and optimization is much easier. Much of our speed gains were achieved because we can now spot poor, unoptimized CI logic in a simplified way."***

- Pipelines are executed by a fleet of Dagger Engines running on Kubernetes.
- Dagger Cloud provides a distributed caching service that all runners have access to. Each Dagger Engine communicates with Dagger Cloud to read from, and write to, the shared cache.
- Karpenter, a Kubernetes-native node auto-scaler, dynamically adds or removes runner nodes depending on workload requirements. Through this process, Airbyte also gained cost controls they lacked before.

Benefits: CI Pipelines as Code, Unified Local and Remote CI

Some of the benefits of the new "Daggerized" system are:

Faster local debugging: Airbyte's development teams now use the same CLI wrapper as the CI engine. This ensures parity and portability between local and remote CI runs. As a result, developers are able to test code locally, identify errors early, and have fewer surprises when pushing to remote repositories.

Better maintainability: By writing pipelines in Python instead of YAML, CI code is easier to understand, maintain, and optimize. Native-language support makes it easy to create complex conditional pipelines (for example, running a pipeline based on the result of another pipeline). Working entirely in a programming language also makes it easier to follow best practices for testing and packaging CI pipelines as code.

Quicker pipeline execution: Dagger's built-in caching support and integration with Dagger Cloud alleviates the problem of cold starts and enables pipelines to run quicker. Dagger's "caching by default" approach is also easier to work with compared to Gradle, which requires caching to be explicitly configured.

Speed

With Dagger Cloud, CI pipelines are now 2-5 times faster on average. This is because Dagger Cloud intelligently stores the cached data and operations from previous pipeline runs in the cloud and sends this data to new Dagger Engines whenever they are provisioned. As a result, new Dagger Engines start up with a "warm cache" and have all the caching benefits of previous runs.

> ***"In the worst-case scenario, it's about twice as fast as the legacy way and in the best-case scenario, it's often four to five times as fast. The shared remote cache is a core component of why we're able to see a lot of these gains, because it's the foundation upon which we built this distributed architecture."***

> ***"The level of trust between the Airbyte team and the Dagger team is high. It's been a very fruitful and helpful collaboration with the Dagger team, who are willing to step in and make sure that we get things resolved."***

Scalability and Cost

Running Dagger inside a single Kubernetes cluster with auto-scaling runners has improved overall efficiency and drastically reduced CI costs for Airbyte. Airbyte is now able to intelligently scale up and down depending on the workload in CI. Runner nodes are provisioned on-demand and automatically de-provisioned when they are not needed. As a result, server costs are significantly lower than before.

"We examined our spend and there were times in the past six to eight months that it was at least four times what it is now. Switching to a single-runner architecture has both reduced our cost and made our usage of server resources more efficient."

Future Plans

Going forward, Augustin and Conor plan to continue building out Airbyte's CI infrastructure with Dagger. They are working to get other internal teams to adopt Dagger and hope to see those teams contributing more tests to the current pipelines as well as adding their own pipelines in the future. With Dagger Cloud, they are looking to increase the observability of their pipeline internals and gain greater control over caching and parallelization configuration.