Back

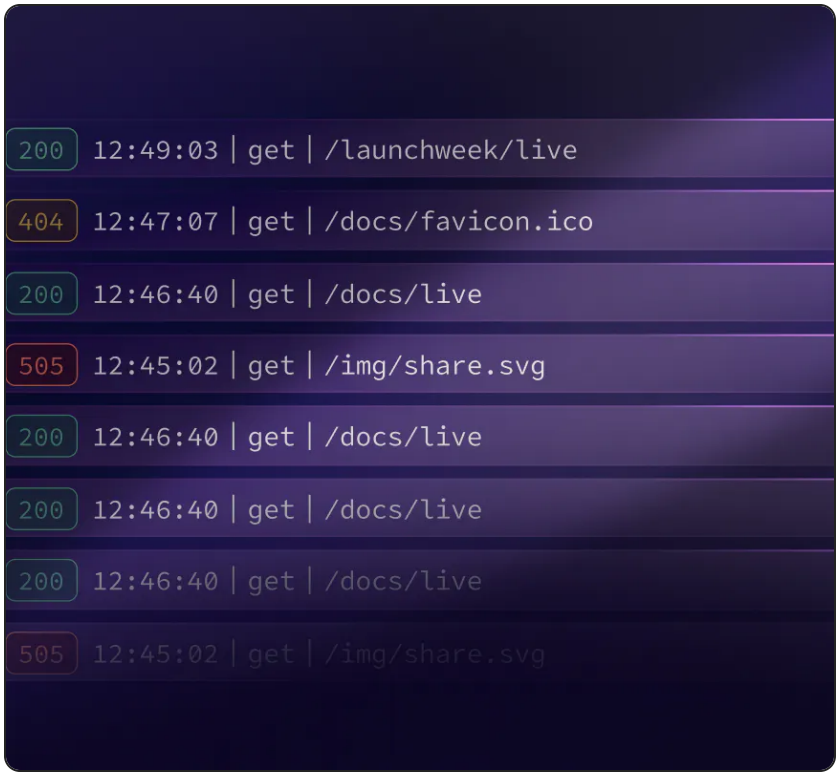
Blog post

Supabase Logs: open source logging server

2023-04-10 • 8 minute read


Chase Granberry
Engineering


Lee TzeYiing
Engineering



Today, we're releasing Supabase Logs for both self-hosted users and CLI development.

Logflare Introduction

Since Logflare joined Supabase over a year ago it's been quietly handling over 1 billion log events every day. These events come from various tools in the Supabase infrastructure - the API gateway, Postgres databases, Storage, Edge Functions, Auth, and Realtime.

Logflare is a multi-node, highly available Elixir cluster, ingesting the log events and storing them into BigQuery for Supabase and Logflare's customers. On average, the cluster has 6 nodes handling every spike our customers throw at it.

To expose log data to customers, we leverage Logflare Endpoints. This provides an HTTP integration into Supabase Studio, powering the log query UIs and most time-series charts. These charts live across the studio, such as the project home page and the new API reports.

Self-hosting Logflare

Logflare was available under a BSL license prior to joining Supabase. We've since changed the license to Apache 2.0, aligning it with our open source philosophy.

In the past few months we've made Logflare more developer-friendly for local development and self-hosting. While you're building a project, you can view and query your logs from any Supabase service, just as you would in our cloud platform.

> 📢 *Check out the new self-hosting docs to get Logflare up and running as your analytics server.*

It currently supports a BigQuery backend, and we are actively working on supporting more.

The Ingestion Pipeline

Logflare receives Supabase log events via multiple methods. Services like Postgres use Vector to clean and

forward log events to the Logflare ingest API. Other services such as Realtime and Storage utilize native Logflare integrations to send the log events directly. These then get processed and streamed into BigQuery.

The Querying Pipeline

The hard part comes after ingesting the logs: searching, aggregating ,and analyzing them at scale. Crunching many terabytes of data on each query is expensive, and exposing the ingested data to Supabase customers in a naive manner would cause our costs to skyrocket.

To solve these issues, we built and refined Logflare Endpoints, the query engine that powers many of Supabase's features, such as the logs views, Logs Explorer, and usage charts.

With Endpoints, you can create HTTP API endpoints from a SQL query, including parameterized queries. Endpoints are like PostgREST views but with some benefits:

Query parameters

You can provide string parameters to the SQL query via the HTTP endpoint.

Read-through caching

Results from the query are cached in memory for fast response times.

A read-through cache provides results if cached results do not exist.

Active cache warming

Query results are proactively warmed at a configurable interval for a combination of fast

response times and as-realtime-as-needed data.

Query sandboxing

If an Endpoint query contains a CTE and the sandbox option is selected, the Endpoint will inject the query string of the `sql` query parameter into the Endpoint SQL replacing the default query (the part of the SQL query after the CTE).

Endpoints parse SQL to allow `select` queries only. No DML or DDL statements are permitted to run through Logflare Endpoints.

With this feature set, Supabase has been able to build any view we've needed on top of billions of daily log events.

Logflare Endpoint Example

Using webhooks, we can send all GitHub events in the Supabase organization to Logflare. The webhook sends structured events, and Logflare transforms the payload into metadata:

```
{
  "event_message": "supabase/supabase | Johanne
  "id": "0d48b71d-91c5-4356-82c7-fdb299b625d0",
  "metadata": {
    "sender": {
      "id": 15695124,
      "login": "JohannesBauer97",
      "node_id": "MDQ6VXNlcjE1Njk1MTI0",
      "site_admin": false,
      "type": "User",
      "url": "https://api.github.com/users/Joha
    },
    "starred_at": "2023-03-30T20:33:55Z"
    //...
  },
  "timestamp": 1680208436849642
}
```

We're interested in the top contributors, which can be extracted with SQL (in BigQuery dialect):

```
select
  count(t.timestamp) as count,
  s.login as gh_user
from
  `github.supabase.webhooks` as t
  cross join unnest(metadata) as m
  cross join unnest(m.sender) as s
where
  timestamp::date > current_date() - @day::int
group by
  gh_user
order by
  count desc
limit
  25
```

With this view in place, we can use Endpoints to provide an API that we can hit from our application:

```
curl "https://logflare.app/endpoints/query/6942
 -H 'Content-Type: application/json; charset=ut
 -G -d "day=30"
```

This returns a JSON response with the top org wide contributors for the last 30 days!

```
{
  "result": [
    { "count": 23404, "gh_user": "vercel[bot]"
    { "count": 10005, "gh_user": "joshenlim" },
    { "count": 7026, "gh_user": "MildTomato" },
    { "count": 6405, "gh_user": "fsansalvadore"
    { "count": 5195, "gh_user": "saltcod" },
    { "count": 3454, "gh_user": "alaister" },
    { "count": 2691, "gh_user": "kevcodez" },
    { "count": 2117, "gh_user": "gregnr" },
    { "count": 1769, "gh_user": "Ziinc" },
    { "count": 1749, "gh_user": "chasers" },
    { "count": 1430, "gh_user": "Isaiah-Hamilto
    //...
```

```
    ]
  }
```

We can configure this Endpoint to cache results for an interval of 10 minutes after the first API request, and proactively update those cached results every 2 minutes - 5 queries across the 10 minute interval. Even if we hit the Endpoint thousands of times, we only sustain the cost of 5 queries.

The initial request is fast because Logflare also performs setup (such as partitioning) on our BigQuery tables appropriately. Subsequent requests are *extremely fast* as they are cached in-memory.

The best part is that all these knobs can be tweaked for your use case. If we have a real-time requirement, we can completely disable caching or reduce the proactive caching to update on a per-second interval.

The Self-hosted Challenge

To change the license, we needed to remove all closed-source dependencies. Previously, Logflare relied on the closed source General SQL Parser under a business licenses. This is incompatible with the Apache License.

We switched to an open source alternative, the rust-based sqlparser-rs library, contributing a few updates for the BigQuery dialect.

Along with the parser, we invested a lot of effort into transforming the multi-tenant architecture into something that was self-hosting friendly and easily configurable. We moved towards environment variable based configuration instead of compile-time configurations, exposing the Endpoints configurations necessary for Supabase Logs.

What's Next?

To further integrate Logflare into the Supabase platform, we are building out 2 main areas: Management API, Multiple Backends.

Management API

The Management API allows users to interact programmatically with Logflare to manage their account and resources. This feature will be available for both Logflare customers and self-hosted users.

You can check out the preview of our OpenAPI spec here: https://logflare.app/swaggerui

Not only that, we intend to expose user account provisioning to select partners. Soon, you'll be able to become a Logflare Partner to provision Logflare accounts through the Partner API. Perfect if you want to resell a log analytics service from your own platform.

Contact us at growth@supabase.com to get in early on that waitlist.

Multiple Backends

Logflare currently supports a BigQuery backend. We plan to add support for other analytics-optimized databases, like Clickhouse. We will also support pushing data to other web services, making Logflare a good fit for any data pipeline.

This will benefit the Supabase CLI: once Postgres support is available, Logflare will be able to integrate seamlessly, without the BigQuery requirement.

Wrapping Up

Logflare has given Supabase the flexibility to quickly deploy features powered by an underlying structured event stream. Materializing metrics from an event stream is a powerful framework for delivering real-time views on analytics streams.

Logflare is the hub of analytics streams for Supabase. We look forward to giving Supabase customers the same superpower.

More Launch Week 7

Designing with AI

Supavisor

Open Source Logging

Share this article

Next post

Supabase Beta March 2023

8 April 2023

Related articles

Supabase Logs: open source logging server

Supabase Beta March 2023

The Supabase AI Hackathon

Designing with AI: Generating unique artwork for every user

Infinite scroll with Next.js, Framer Motion, and Supabase

View all posts

Build in a weekend, scale to millions

Start your project

Product

- Database
- Auth
- Functions
- Realtime
- Storage
- Pricing
- Launch Week 7

Resources

- Support
- System Status
- Integrations
- Experts
- Brand Assets / Logos
- DPA
- SOC2

Developers

- Documentation
- Changelog
- Contributing
- Open Source
- SupaSquad
- DevTo
- RSS

Company

- Blog
- Customer Stories
- Careers
- Company
- Terms of Service
- Privacy Policy
- Acceptable Use Policy
- Service Level Agreement
- Humans.txt
- Lawyers.txt
- Security.txt