Debugging and Monitoring

Applications that use LLMs have some challenges that are well known and understood: LLMs are **slow**, **unreliable** and **expensive**.

These applications also have some challenges that most developers have encountered much less often: LLMs are **fickle** and **non-deterministic**. Subtle changes in a prompt can completely change a model's performance, and there's no `EXPLAIN` query you can run to understand why.

Warning

From a software engineers point of view, you can think of LLMs as the worst database you've ever heard of, but worse.

If LLMs weren't so bloody useful, we'd never touch them.

To build successful applications with LLMs, we need new tools to understand both model performance, and the behavior of applications that rely on them.

LLM Observability tools that just let you understand how your model is performing are useless: making API calls to an LLM is easy, it's building that into an application that's hard.

Pydantic Logfire

Pydantic Logfire is an observability platform developed by the team who created and maintain Pydantic and PydanticAI. Logfire aims to let you understand your entire application: Gen AI, classic predictive AI, HTTP traffic, database queries and everything else a modern application needs.

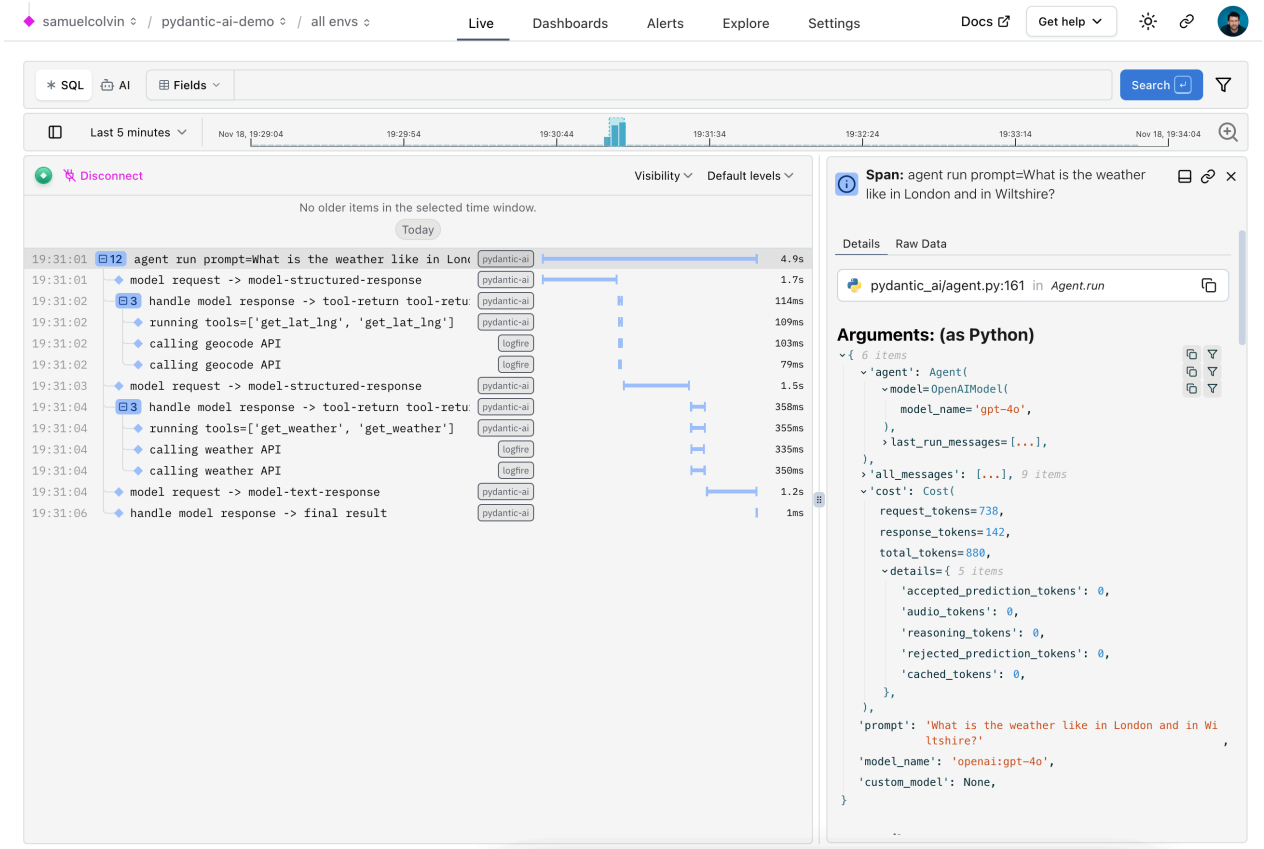
Pydantic Logfire is a commercial product

Logfire is a commercially supported, hosted platform with an extremely generous and perpetual free tier. You can sign up and start using Logfire in a couple of minutes.

PydanticAI has built-in (but optional) support for Logfire via the `logfire-api` no-op package.

That means if the `logfire` package is installed and configured, detailed information about agent runs is sent to Logfire. But if the `logfire` package is not installed, there's virtually no overhead and nothing is sent.

Here's an example showing details of running the Weather Agent in Logfire:



Using Logfire

To use logfire, you'll need a logfire account, and logfire installed:

pip

```
pip install 'pydantic-ai[logfire]'
```

uv

```
uv add 'pydantic-ai[logfire]'
```

Then authenticate your local environment with logfire:

pip

pip

```
logfire auth
```

uv

```
uv run logfire auth
```

And configure a project to send data to:

pip

```
logfire projects new
```

uv

```
uv run logfire projects new
```

(Or use an existing project with `logfire projects use`)

The last step is to add logfire to your code:

adding_logfire.py

```
import logfire

logfire.configure()
```

The logfire documentation has more details on how to use logfire, including how to instrument other libraries like Pydantic, HTTPX and FastAPI.

Since Logfire is build on OpenTelemetry, you can use the Logfire Python SDK to send data to any OpenTelemetry collector.

Once you have logfire set up, there are two primary ways it can help you understand your application:

- **Debugging** — Using the live view to see what's happening in your application in real-time.
- **Monitoring** — Using SQL and dashboards to observe the behavior of your application, Logfire is effectively a SQL database that stores information about how your application is running.

Debugging

To demonstrate how Logfire can let you visualise the flow of a PydanticAI run, here's the view you get from Logfire while running the chat app examples:

Monitoring Performance

We can also query data with SQL in Logfire to monitor the performance of an application. Here's a real world example of using Logfire to monitor PydanticAI runs inside Logfire itself:

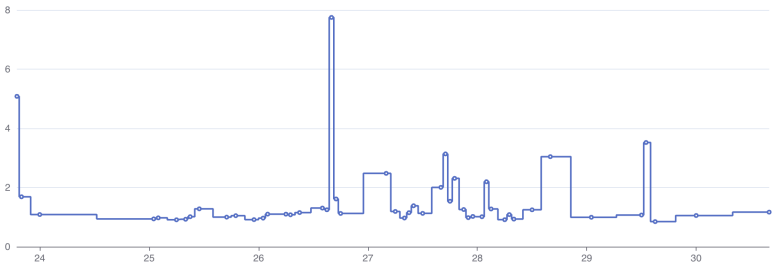
Tab +

```
select
    date_trunc('hour', start_timestamp) as time,
    avg(duration) as avg_duration
from records
    where otel_scope_name='pydantic-ai' and span_name='agent run {prompt=}'
group by time
order by time desc
```

Returned 49 rows in 6.7 seconds.

Limit: 100 Time window: Last 7 days

Table Details Chart



Visualization Type

Line Step

X Axis

time Time

Metric

avg_duration

Group Data By:

Select column...