Observability with Dynatrace

FastAPI + Dynatrace Observability Demo

Mehmet Oğuz Arslan
Ezgi Sena Karabacak
Davida Lamagna
Pol Verdura

12.05.2025

What is dynatrace ?

A single platform to monitor, optimize, and secure your apps, infrastructure, and users — all in real time.

- **Full-Stack Monitoring:** From backend to frontend, cloud to user
- **Smart AI (Davis):** Finds root causes and alerts on issues automatically
- **Observability:** Metrics, Logs, Traces, Real User Monitoring (RUM)
- **Cloud-Ready:** Works with Kubernetes, AWS, Azure, GCP
- **Built-in Security:** Detects threats and vulnerabilities at runtime

Observability Concepts

- **Trace:** Tracks the full path of a request through the system, showing timing and errors.
- **Metric:** Numeric data collected over time (e.g., response time, CPU usage).
- **Log:** Text-based records of events or actions for debugging and analysis.
- **RUM:** Captures real user interactions and performance from the browser.

OpenTelemetry

OpenTelemetry is an open-source toolkit to collect traces, metrics, and logs from applications, in a standard, vendor-neutral way.

Why it matters:

- Auto-instruments apps like FastAPI
- Works with tools like Dynatrace
- Sends data in a unified format (OTLP)

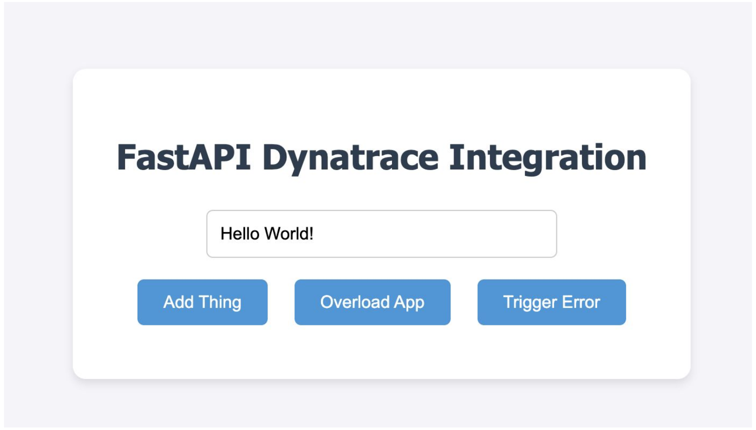
FastAPI + Dynatrace Observability Demo

In this lab, you will:

- Run a FastAPI app
- Configure .env to connect to Dynatrace
- Trigger backend endpoints
- Open the frontend and interact
- View the data live in Dynatrace

Available Endpoints

- Add Thing → Sends a custom log to Dynatrace
- Overload App → Runs CPU load, exports a metric
- Error → Returns HTTP 500 to test tracing + alerts
- RUM (Frontend) → Tracks user behavior via JS tag



Dynatrace Setup

Generate API Token

- Go to Dynatrace → Settings → Integration → API
- Enable scopes: logs.ingest, metrics.ingest, traces, etc.

Create .env File

- Set your environment ID, token, and OTLP endpoint:
  - DYNATRACE_ENV_ID=...
  - DYNATRACE_API_TOKEN=...
  - DYNATRACE_OTLP_ENDPOINT=https://<env>.live.dynatrace.com/api/v2/otlp

Start the App:

- uvicorn main:app --reload

How It Works – Backend Observability

- **Tracing (Automatic)**
    - Each FastAPI route is auto-instrumented to generate distributed traces
    - Captures method, route, status code, and latency
    - Exported to Dynatrace via OTLP exporter
- **Metrics (Custom)**
    - CPU load duration tracked via a custom metric (overload_duration_seconds)
    - Metrics are exported when the overload endpoint is triggered
    - Logs (Manual)
- **Logs sent via Dynatrace Logs API using a custom function**
    - Includes details like log level, route, and error messageS
    - View everything in Dynatrace under Traces, Metrics, and Logs

Test Backend Endpoints

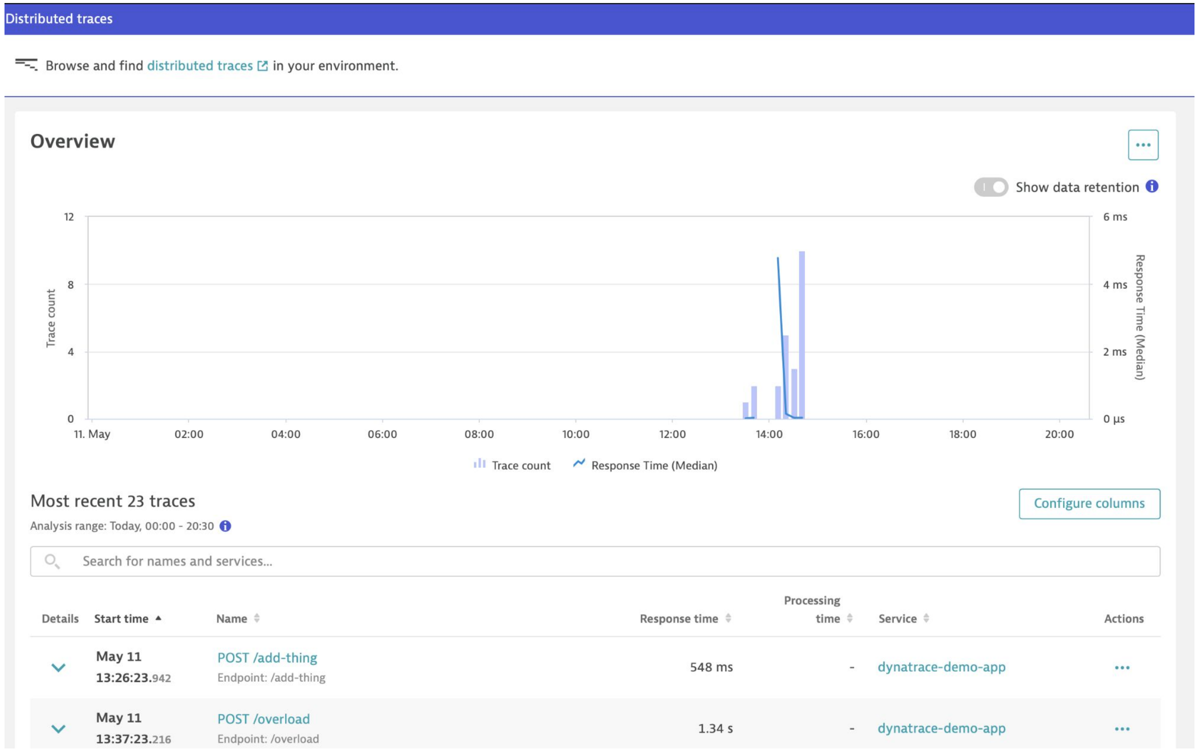
Using the Terminal:

- Send requests to the backend endpoints using tools like **curl**, **httpie**, or **Postman**.
- This allows you to simulate adding logs, triggering CPU load, and causing errors

Using the Frontend Interface:

- Open the index.html file in your browser.
- Use the buttons on the page to:
    - Add a log entry
    - Simulate CPU overload
    - Trigger an intentional error

View Backend Observability

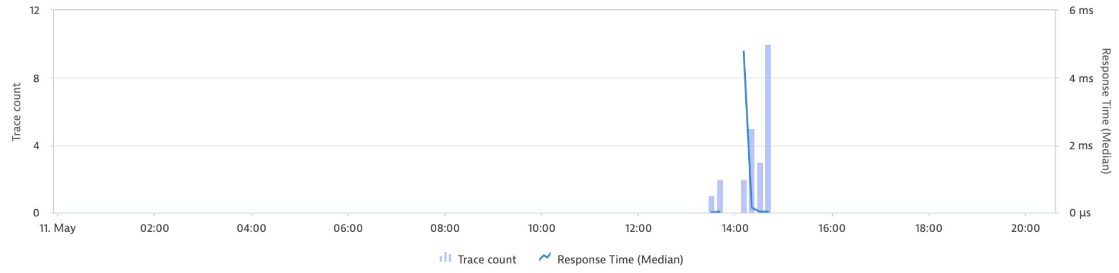


Distributed traces

Browse and find distributed traces in your environment.

Overview

Show data retention



Most recent 23 traces

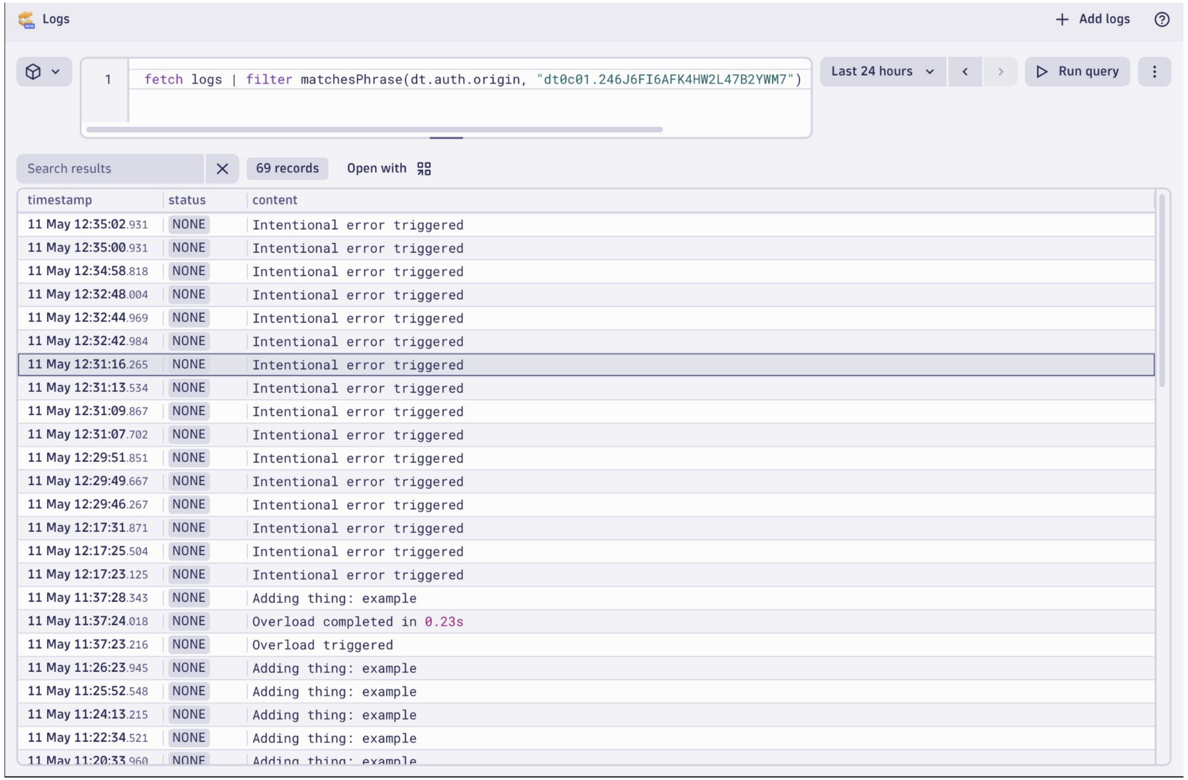
Analysis range: Today, 00:00 - 20:30

Configure columns

Search for names and services...

Details	Start time	Name	Response time	Processing time	Service	Actions
	May 11 13:26:23.942	POST /add-thing Endpoint: /add-thing	548 ms	-	dynatrace-demo-app	•••
	May 11 13:37:23.216	POST /overload Endpoint: /overload	1.34 s	-	dynatrace-demo-app	•••

View Backend Observability



Logs

Add logs

```
fetch logs | filter matchesPhrase(dt.auth.origin, "dt0c01.246J6FI6AFK4HW2L47B2YWM7")
```

Last 24 hours

Run query

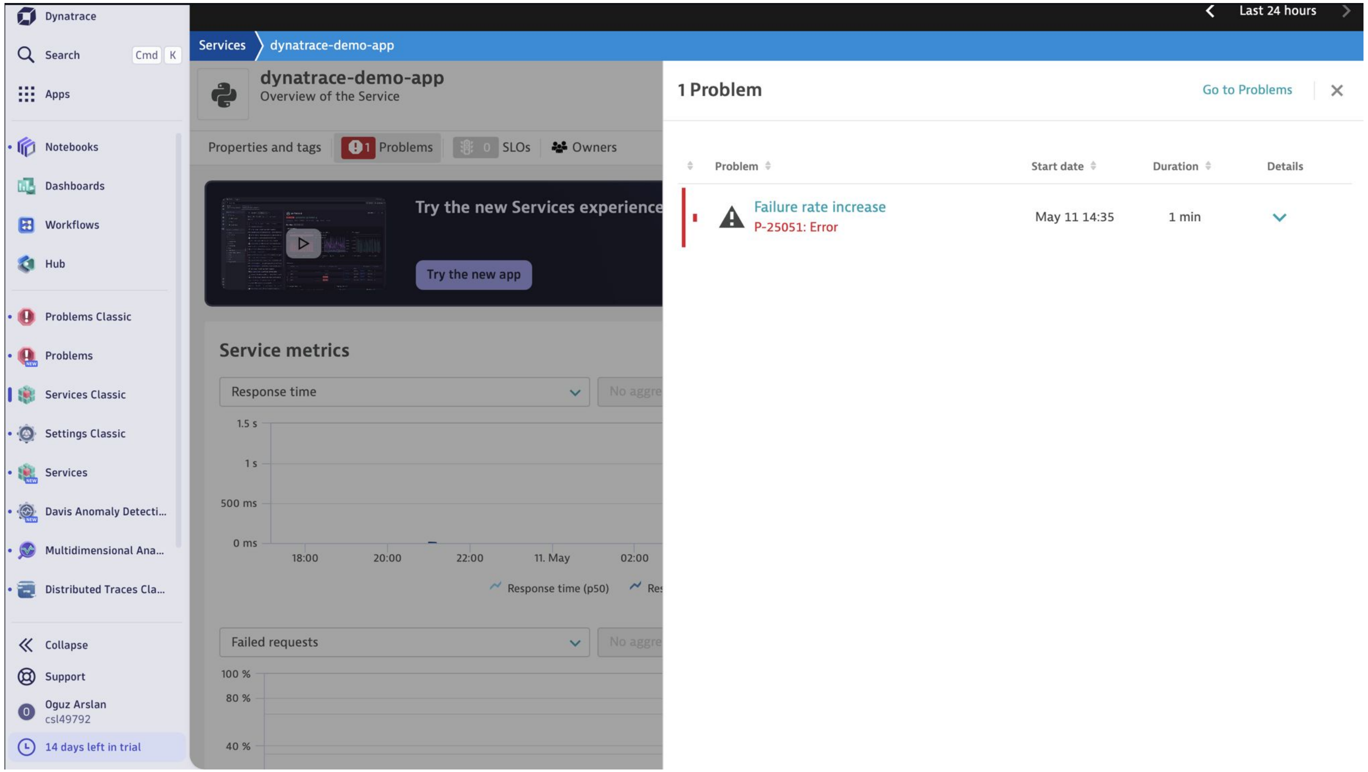
Search results

69 records

Open with

timestamp	status	content
11 May 12:35:02.931	NONE	Intentional error triggered
11 May 12:35:00.931	NONE	Intentional error triggered
11 May 12:34:58.818	NONE	Intentional error triggered
11 May 12:32:48.004	NONE	Intentional error triggered
11 May 12:32:44.969	NONE	Intentional error triggered
11 May 12:32:42.984	NONE	Intentional error triggered
11 May 12:31:16.265	NONE	Intentional error triggered
11 May 12:31:13.534	NONE	Intentional error triggered
11 May 12:31:09.867	NONE	Intentional error triggered
11 May 12:31:07.702	NONE	Intentional error triggered
11 May 12:29:51.851	NONE	Intentional error triggered
11 May 12:29:49.667	NONE	Intentional error triggered
11 May 12:29:46.267	NONE	Intentional error triggered
11 May 12:17:31.871	NONE	Intentional error triggered
11 May 12:17:25.504	NONE	Intentional error triggered
11 May 12:17:23.125	NONE	Intentional error triggered
11 May 11:37:28.343	NONE	Adding thing: example
11 May 11:37:24.018	NONE	Overload completed in 0.23s
11 May 11:37:23.216	NONE	Overload triggered
11 May 11:26:23.945	NONE	Adding thing: example
11 May 11:25:52.548	NONE	Adding thing: example
11 May 11:24:13.215	NONE	Adding thing: example
11 May 11:22:34.521	NONE	Adding thing: example
11 May 11:20:33.960	NONE	Adding thing: example

View Backend Observability



Problem	Start date	Duration	Details
Failure rate increase P-25051: Error	May 11 14:35	1 min	

Frontend + RUM Integration

Create a Web App in Dynatrace

- Go to: Applications → Web → Set up new app

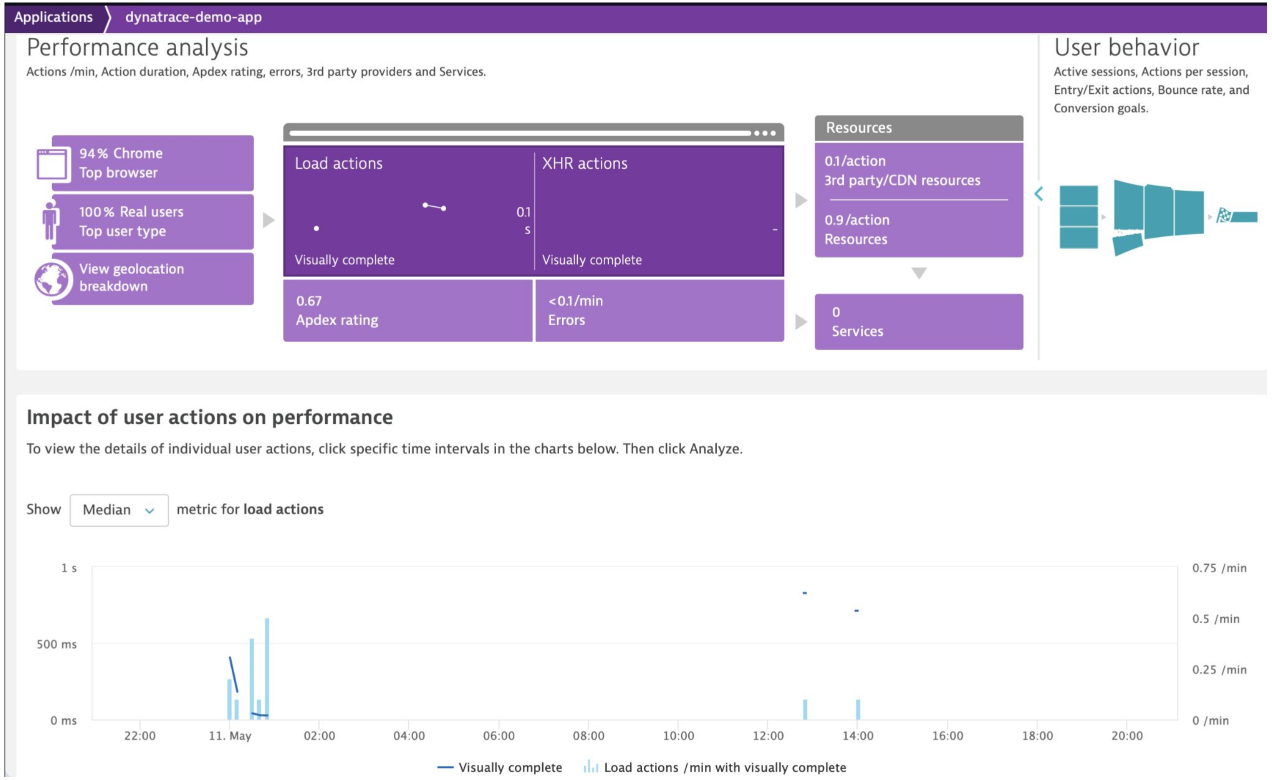
Add Dynatrace JS Tag

- Copy the script and place it in <head> of index.html:
  - <script src="https://<env>.live.dynatrace.com/jstag/<app-id>" async></script>

What RUM Captures:

- Page loads
- User clicks
- JS errors
- Frontend performance

View Frontend Observability


Applications 〉 dynatrace-demo-app

Performance analysis

Actions /min, Action duration, Apdex rating, errors, 3rd party providers and Services.

94% Chrome
Top browser

100% Real users
Top user type

View geolocation
breakdown

Load actions
0.1 s
Visually complete

XHR actions
Visually complete

0.67
Apdex rating

<0.1/min
Errors

Resources

0.1/action
3rd party/CDN resources

0.9/action
Resources

0
Services

User behavior

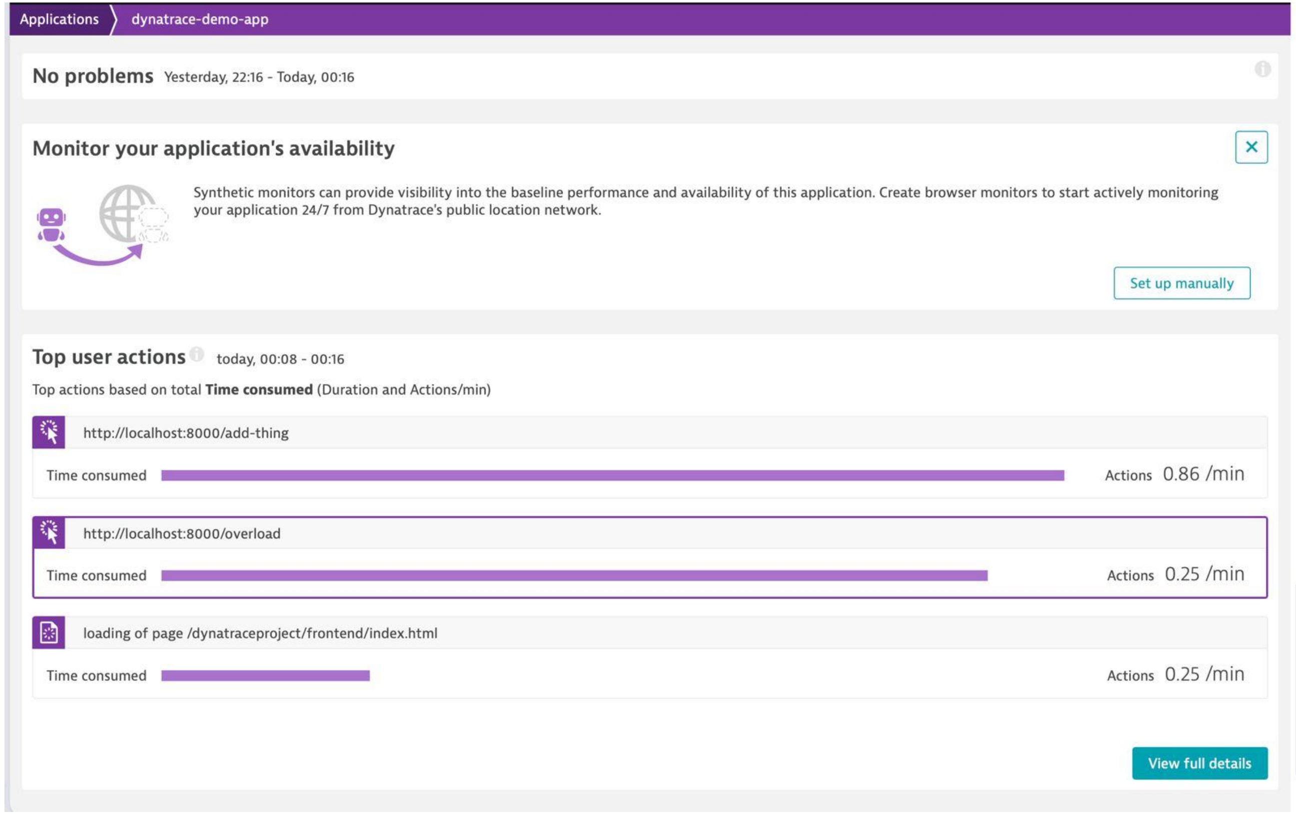
Active sessions, Actions per session, Entry/Exit actions, Bounce rate, and Conversion goals.

Impact of user actions on performance

To view the details of individual user actions, click specific time intervals in the charts below. Then click Analyze.

Show Median metric for **load actions**

View Frontend Observability



Applications 〉 dynatrace-demo-app

No problems Yesterday, 22:16 - Today, 00:16

Monitor your application's availability

Synthetic monitors can provide visibility into the baseline performance and availability of this application. Create browser monitors to start actively monitoring your application 24/7 from Dynatrace's public location network.

Set up manually

Top user actions today, 00:08 - 00:16

Top actions based on total **Time consumed** (Duration and Actions/min)

http://localhost:8000/add-thing
Time consumed — Actions 0.86 /min

http://localhost:8000/overload
Time consumed — Actions 0.25 /min

loading of page /dynatraceproject/frontend/index.html
Time consumed — Actions 0.25 /min

View full details

Summary

A FastAPI application was made **fully observable**

- Both backend and frontend were integrated with Dynatrace
- Observability was demonstrated through T**races**, **Metrics, Logs, and RUM**
- Instrumentation was achieved using **OpenTelemetry** and Dynatrace APIs

Thank you for listening :)

Questions?

Observability with Dynatrace

FastAPI + Dynatrace Observability Demo

Mehmet Oğuz Arslan
Ezgi Sena Karabacak
Davida Lamagna
Pol Verdura

12.05.2025