Nothing was broken.
But the system was behaving differently.

JUNE 2026

SS

Samarth Sharma
Software Engineer @ DataGenie

THE INCIDENT

Checkout requests failing intermittently

We had logs. We had traces. We had metrics.

Everything modern observability tells us to collect.

ON CALL

Everything looked healthy.


Observability dashboard — all metrics green, but users reporting checkout failures

The system isn't obviously broken. It's behaving differently.

THE REALITY

This is what debugging looks like

```
08:14:02.331 INFO [api] incoming request GET /checkout
08:14:02.334 INFO [auth] validating token...
08:14:02.341 INFO [auth] token valid
08:14:02.345 INFO [payments] processing charge $49.99
08:14:02.412 INFO [db] executing query SELECT * FROM orders
08:14:02.510 INFO [db] query completed 98ms
08:14:02.512 INFO [api] response 200 OK
08:14:03.102 INFO [api] incoming request GET /checkout
08:14:03.108 INFO [auth] validating token...
INFO [auth] token valid
```

"Where would you even start?"

CURRENT OBSERVABILITY

Logs

Individual events.
Isolated. Contextless.

Metrics

Aggregated numbers.
No request-level detail.

Traces

Span timelines.
Verbose. Not searchable by pattern.

But where do we search *behavior*?

Systems don't fail as events.
They fail as behaviors.

THE PROBLEM

We are indexing the wrong thing

Today we store thousands of isolated events.

But humans debug by reconstructing execution flows.

So why aren't we indexing flows?

THE IDEA

Execution paths as documents

Each request becomes a single searchable document representing how it actually moved through the system.

BEFORE

Raw Trace Spans

```
GET /checkout — 312ms
  GET /auth — 45ms
    GET /payments — 180ms
      GET /db — 90ms
      GET /db — timeout
+ 12 middleware spans
+ 8 tcp.connect spans
+ 4 router spans
```

AFTER

Behavioral Document

```
{
  "path": ["api", "auth", "payments", "db"],
  "edges": ["api→auth", "auth→payments", ...],
  "has_retry": true,
  "path_string": "api→auth→payments→db",
  "duration": 1200
}
```

NOW POSSIBLE

What becomes searchable

🔄 `has_retry: true`
Find all retry-heavy request paths

🚨 `edges: "payments→db" AND status: "timeout"`
Find timeout paths involving payments → db

👻 `sort by rarity (asc count)`
Find the rarest execution flows — paths never seen before

🔍 `NOT path: "auth"`
Find requests that skipped authentication entirely

KEY INSIGHT

The rarest behavior is the most *interesting*

EXECUTION PATH	OCCURRENCES
api → auth → payments → db	432
api → auth → payments → db → db	18
• api → payments → db	2 ⚠️

"This path skipped auth entirely. That's where I'd start."

LIVE DEMO

Same system.
Different visibility.

We'll run the same traffic through the same services —
first with logs, then with behavior search.

TRADITIONAL VIEW — LOGS

▶ Generate Traffic | Clear

Click "Generate Traffic" to simulate requests...

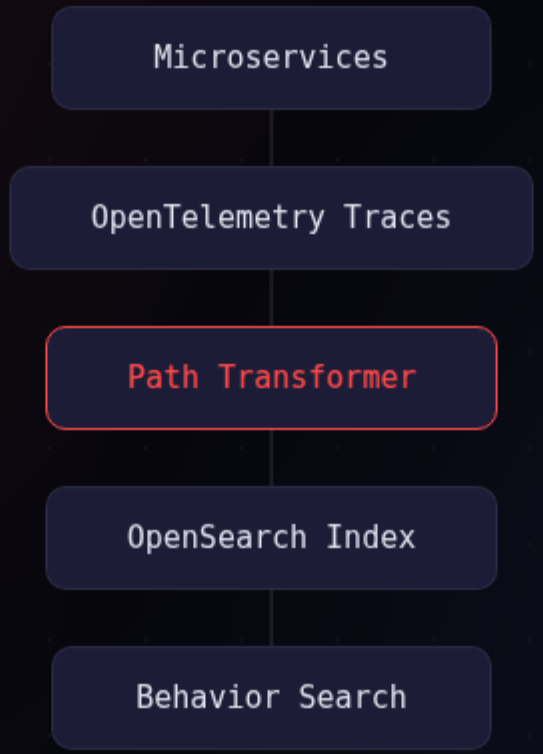
BEHAVIOR VIEW — EXECUTION PATHS

All Paths | Retry Paths | Rare Paths

Select a query to search behavior...

HOW IT WORKS

The pipeline is simple



The important innovation isn't the tooling — it's the representation.

WHY OPENSEARCH

Built for this kind of search

Keyword Arrays

Index execution paths and edges as keyword arrays for exact matching.

Aggregations

Count path frequencies, find rare flows, compute rarity scoring natively.

Scoring

Rank abnormal behavior higher using custom scoring and boosting.

WHY THIS MATTERS

A new layer of observability

- 🎯 Searchable system behavior
- ⚡ Faster anomaly discovery
- 🧠 Better incident reasoning
- 🔬 Behavioral observability as a practice

WHAT'S NEXT

Future possibilities

🤖 ML over Paths

Train models on execution paths to auto-detect anomalies

🔑 Security Detection

Flag unusual auth patterns or bypassed service flows

🧬 Behavior Fingerprints

Unique signatures per deployment or feature flag

📉 Production Drift

Detect when system behavior drifts from baseline

We don't need more
observability data.

We need better representations.

Search behavior. Not just logs.

SS

Samarth Sharma
Software Engineer @ DataGenie

CONNECT ON LINKEDIN