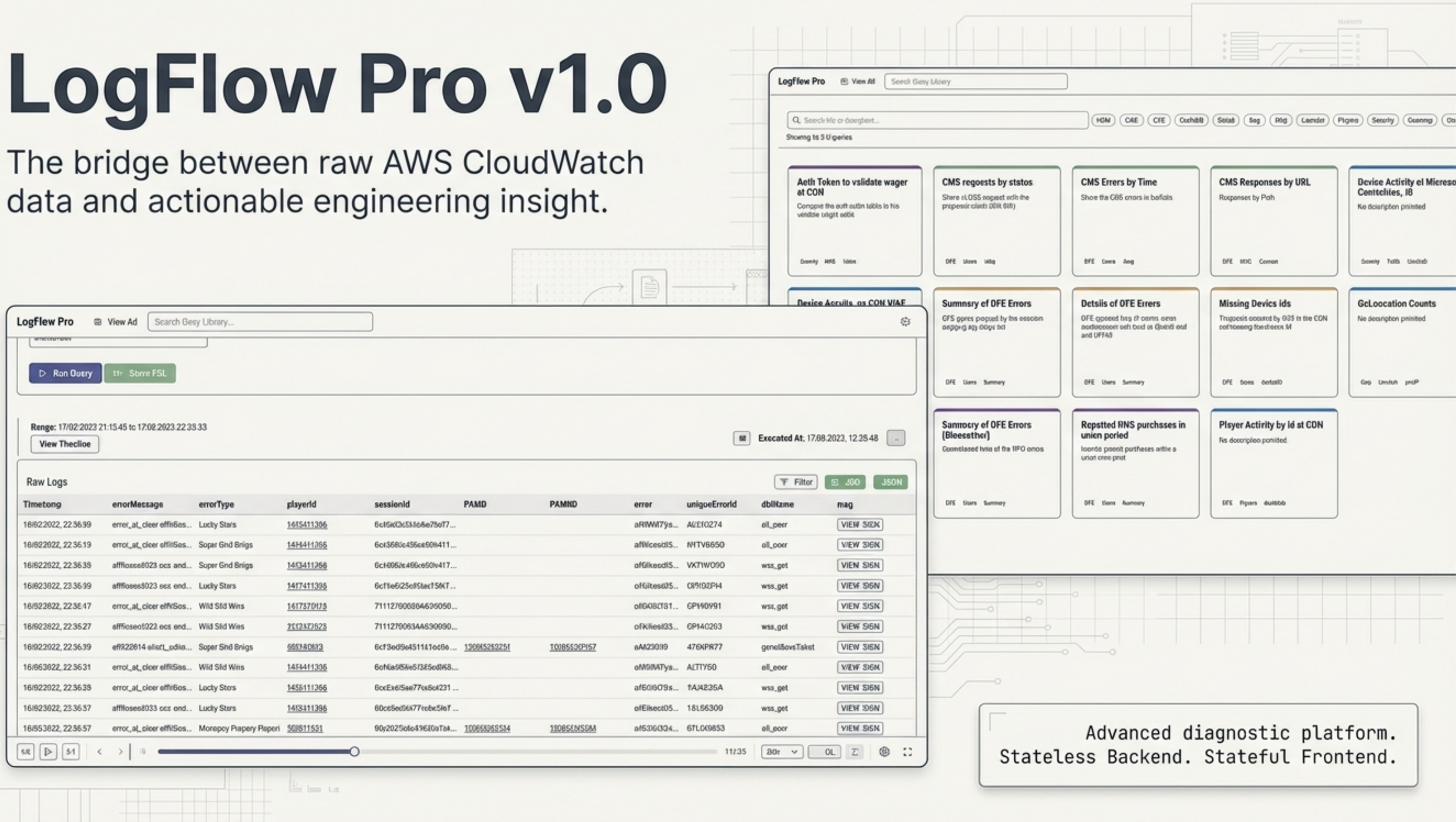
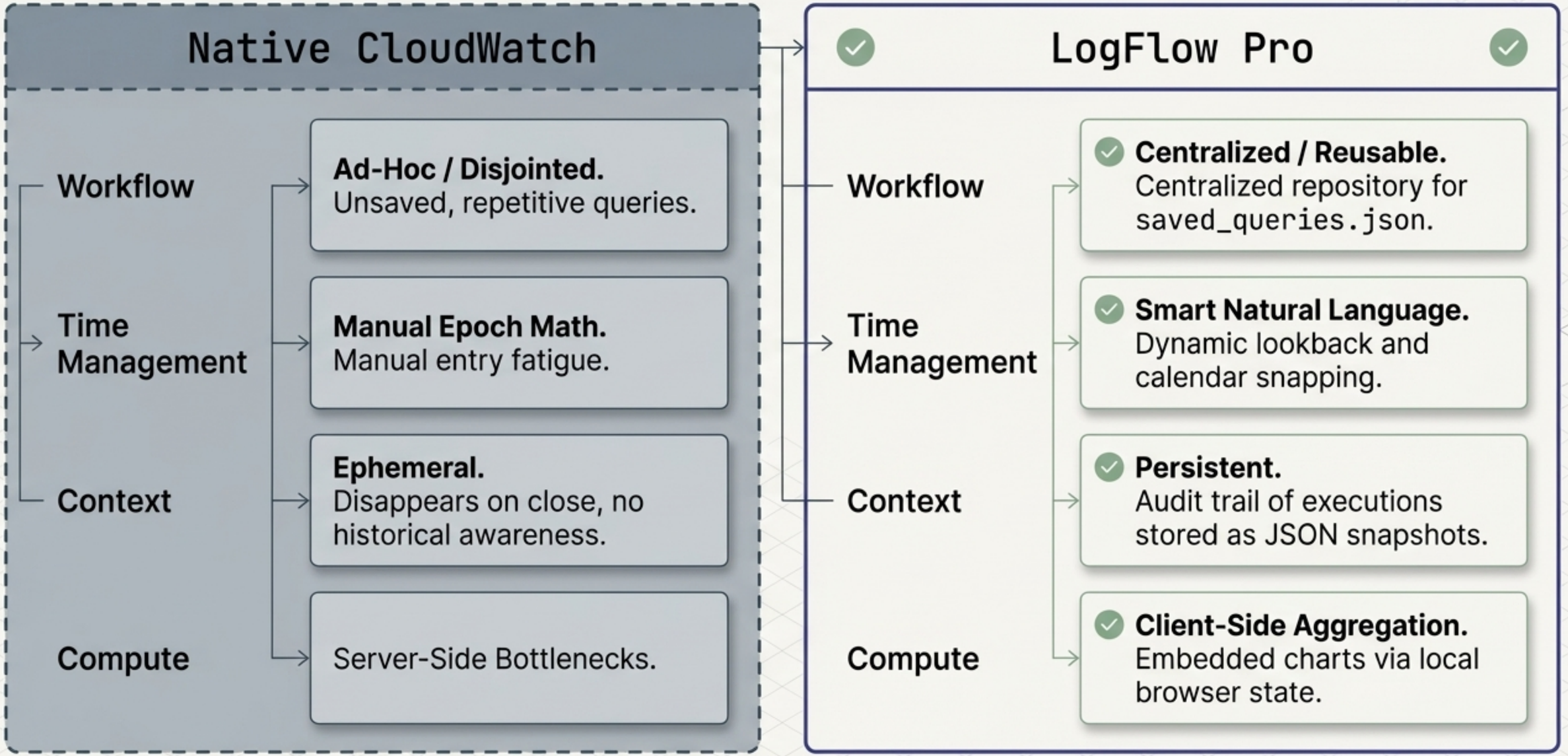
LogFlow Pro v1.0

The bridge between raw AWS CloudWatch data and actionable engineering insight.



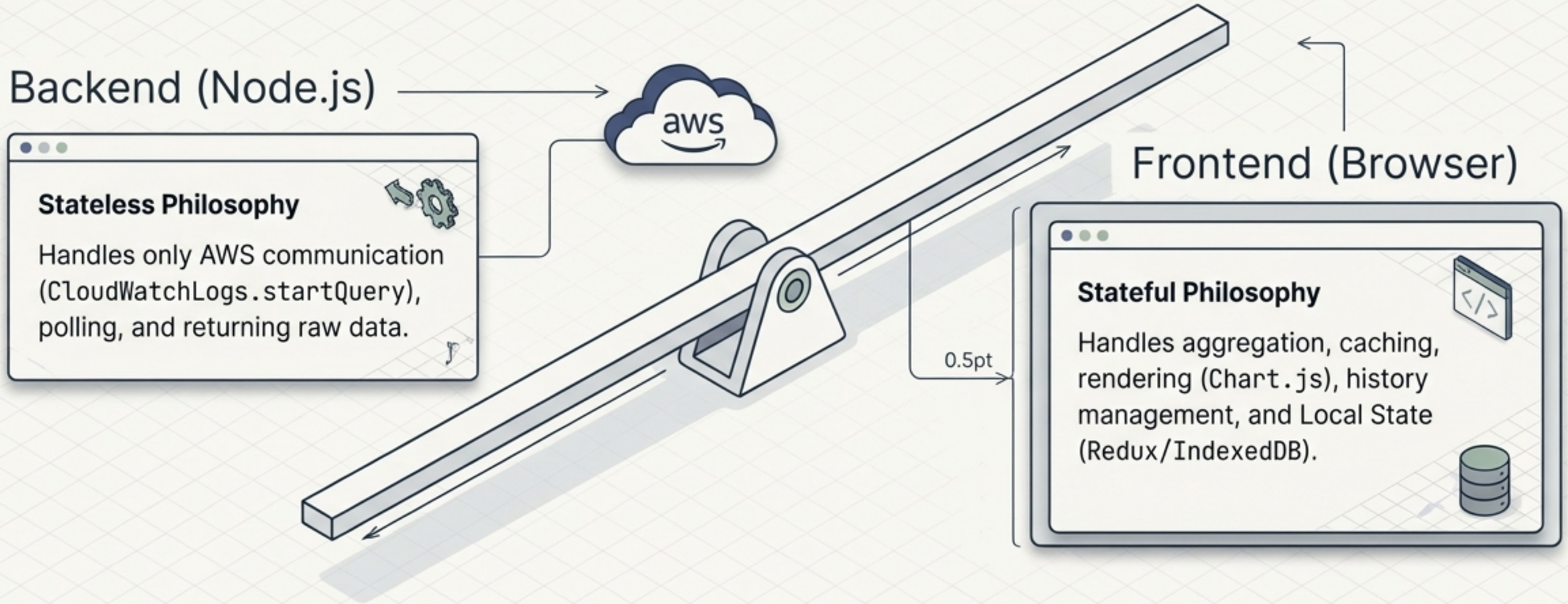
Advanced diagnostic platform.
Stateless Backend. Stateful Frontend.

From Ad-Hoc Chaos to Structured Intelligence



The Editorial Engineering Blueprint

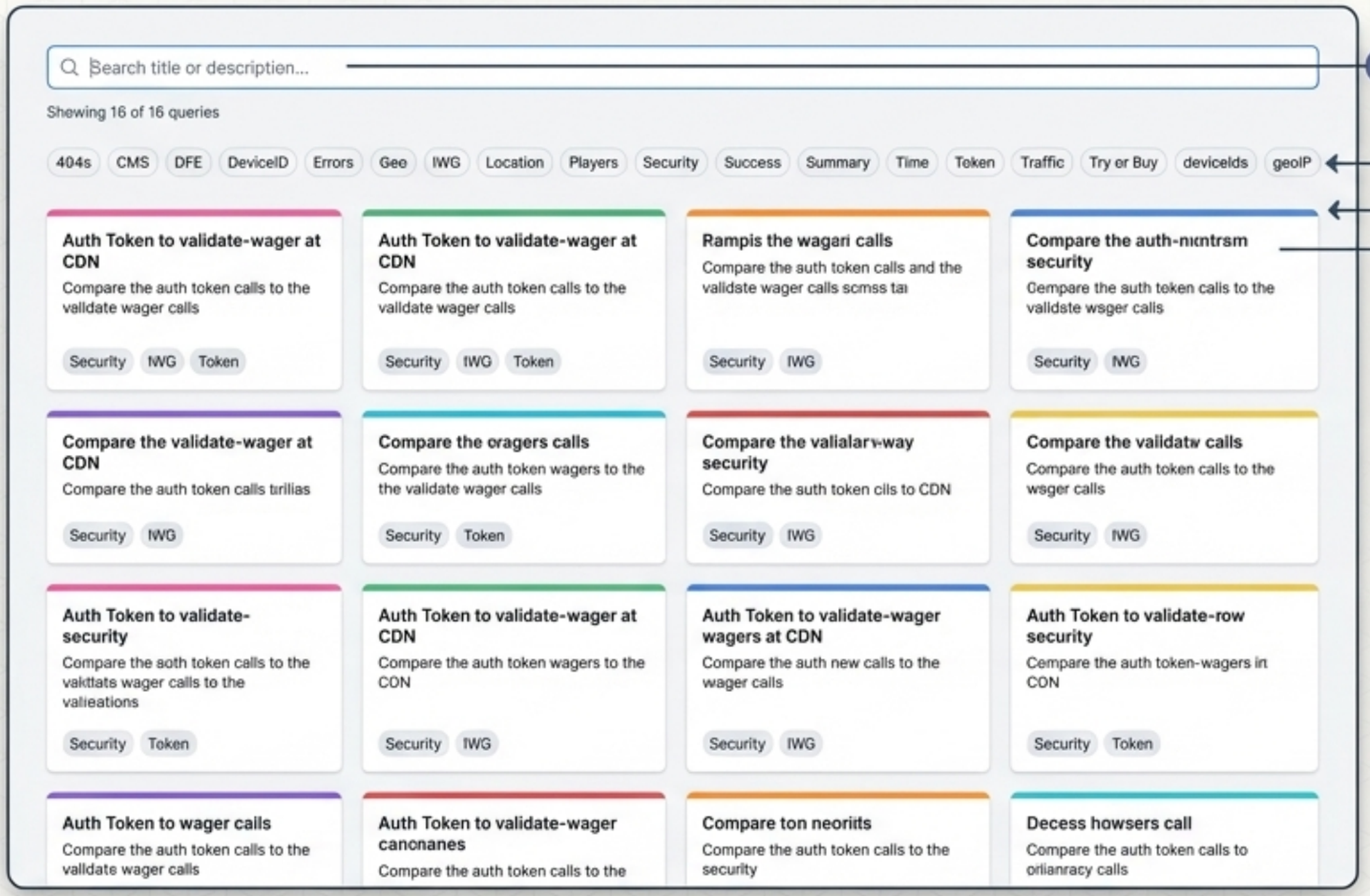
Architectural Philosophy: State Distribution




```
>_ Client-Side Processing > Server-Side Aggregation
```

The Centralized Query Library

Technical Blueprint meets High-Fidelity Software UI



1 Tag-Based Filtering

Organize by functional area (e.g., DFE, Errors, Security).

2 Visual Cues

Color-coded borders defined in metadata for rapid identification.

3 Searchability

Instant browsing by title, description, or tags.

Takeaway

Move beyond ad-hoc searching. Buiid a shared, version-controlled library of engineering knowledge.

Configuration as Code: The DSL Anatomy

query.txt

```
Ref: SAMPLE-01
Title: Example Error Query

Region: eu-west-2

Parameters: level=text:ERROR:Find specific log level

Query: fields @timestamp | filter level = '{{level}}'

.. virts
Views: [Log Level Distribution] GroupBy: level
```

Targeting Infrastructure

Defines the geographic region for AWS resource access and data compliance.

Dynamic Inputs

Allows user-supplied values at runtime to customize query execution.

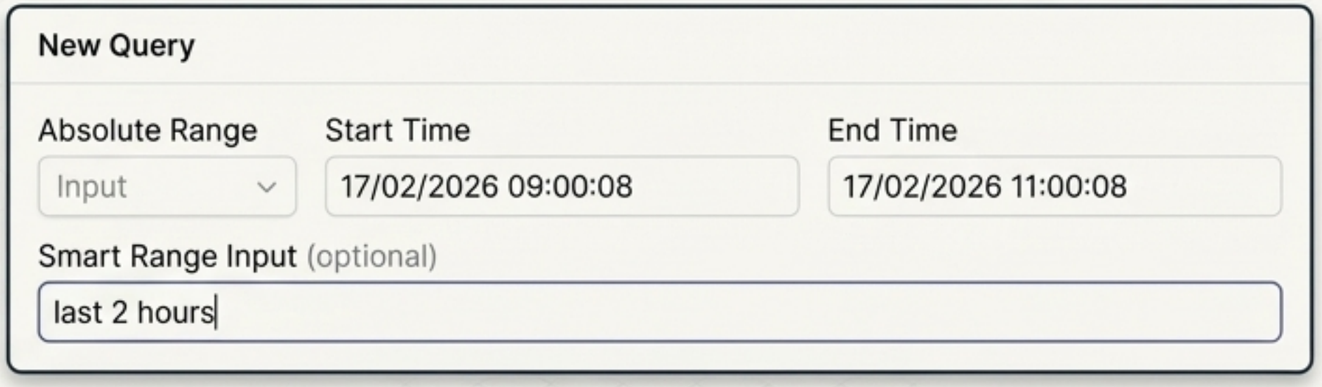
Raw AWS CloudWatch SQL

Directly executes standard CloudWatch Logs Insights query syntax.

Client-side Aggregation Rules

Specifies how to process and visualize results within the frontend application.

Dynamic Parameter Engine & Smart Omission



Condition A: Field Completed

1459491498

↓

```
| filter playerId = '1459491498'
```

Condition B: Field Left Blank

text-opt

↓

~~`| filter playerId = '{{playerId}}'`~~

Smart Omission: Prevents SQL syntax errors without complex conditional logic. If an optional placeholder is empty, its entire line is dynamically excised before execution.

The Parameter Logic Matrix

Parameter Type	Syntax </>	UI Behavior	Backend Logic
Mandatory	`key=text:default:hint`	Renders a red highlight block if empty. Blocks the 'Run' button.	Halts execution immediately if payload is missing.
Optional	`key=text-opt:default:hint`	Standard styling. Allowed to proceed if empty.	Triggers Smart Line Omission. Removes containing SQL line.

SmartRange Translation Engine



Natural Language Features

Rolling Periods

`'last 2 hours'` translates to **dynamic lookback**.

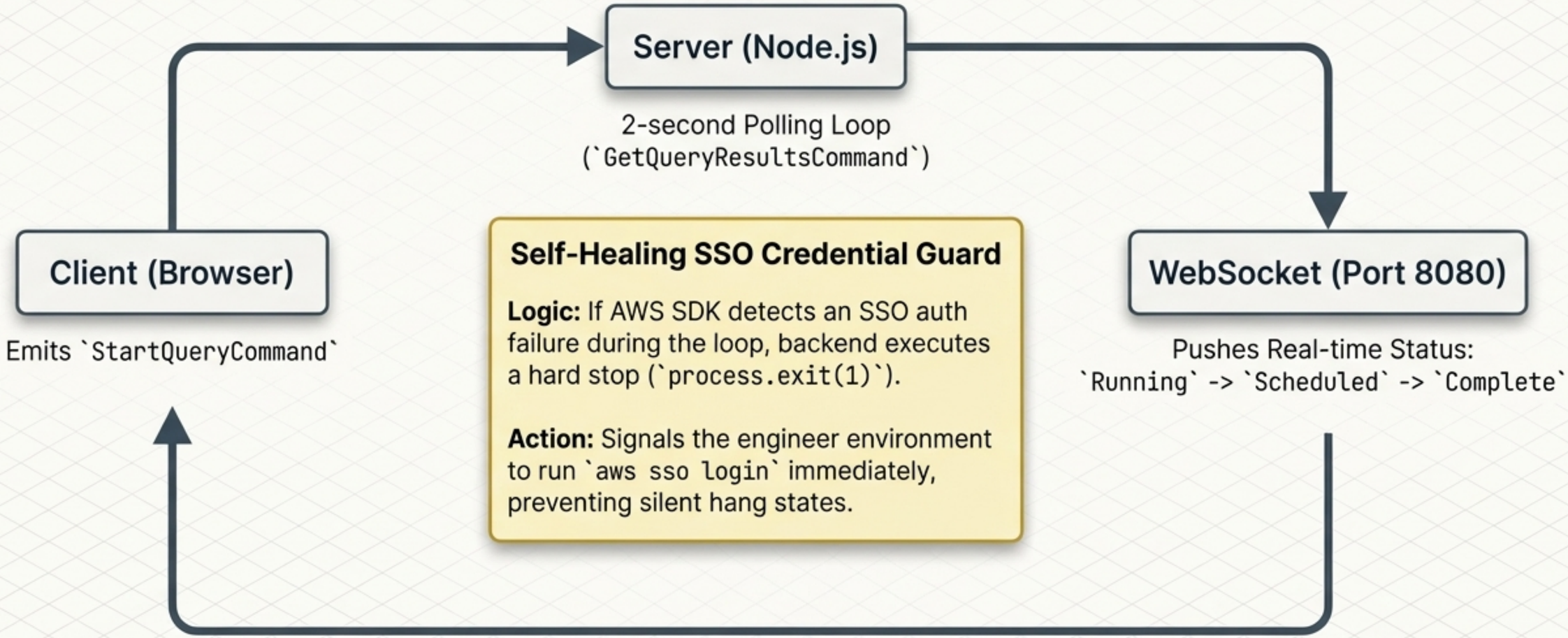
Calendar Snapping

`'the last hour'` snaps precisely from `14:00` to `14:59`.

UI Sync

Resolves complex strings into **absolute time pickers** to lock the range.

Execution Pipeline & Resilience



Client-Side Aggregation Funnel

Raw Log Rows
(From Server)

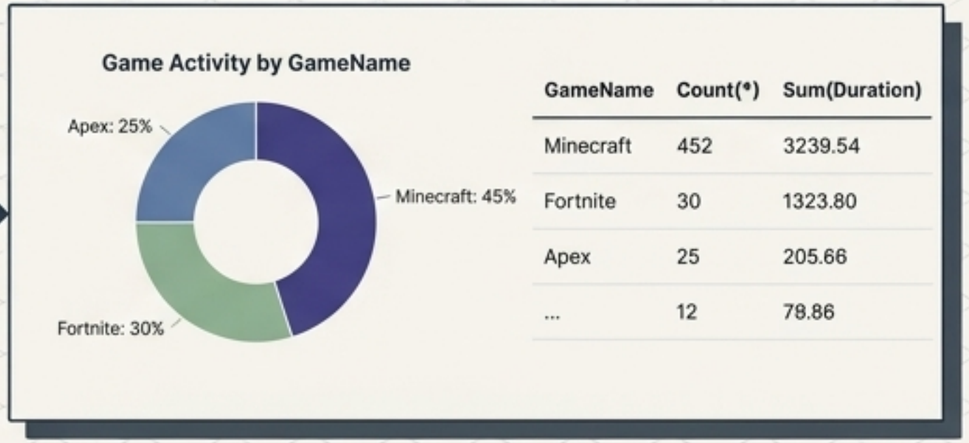
Local Engine (JavaScript)

Local Engine (JavaScript)

Grouping
(e.g., `GroupBy: gameName`)

Metrics
(Calculates 'sum' and 'count'(*)'').

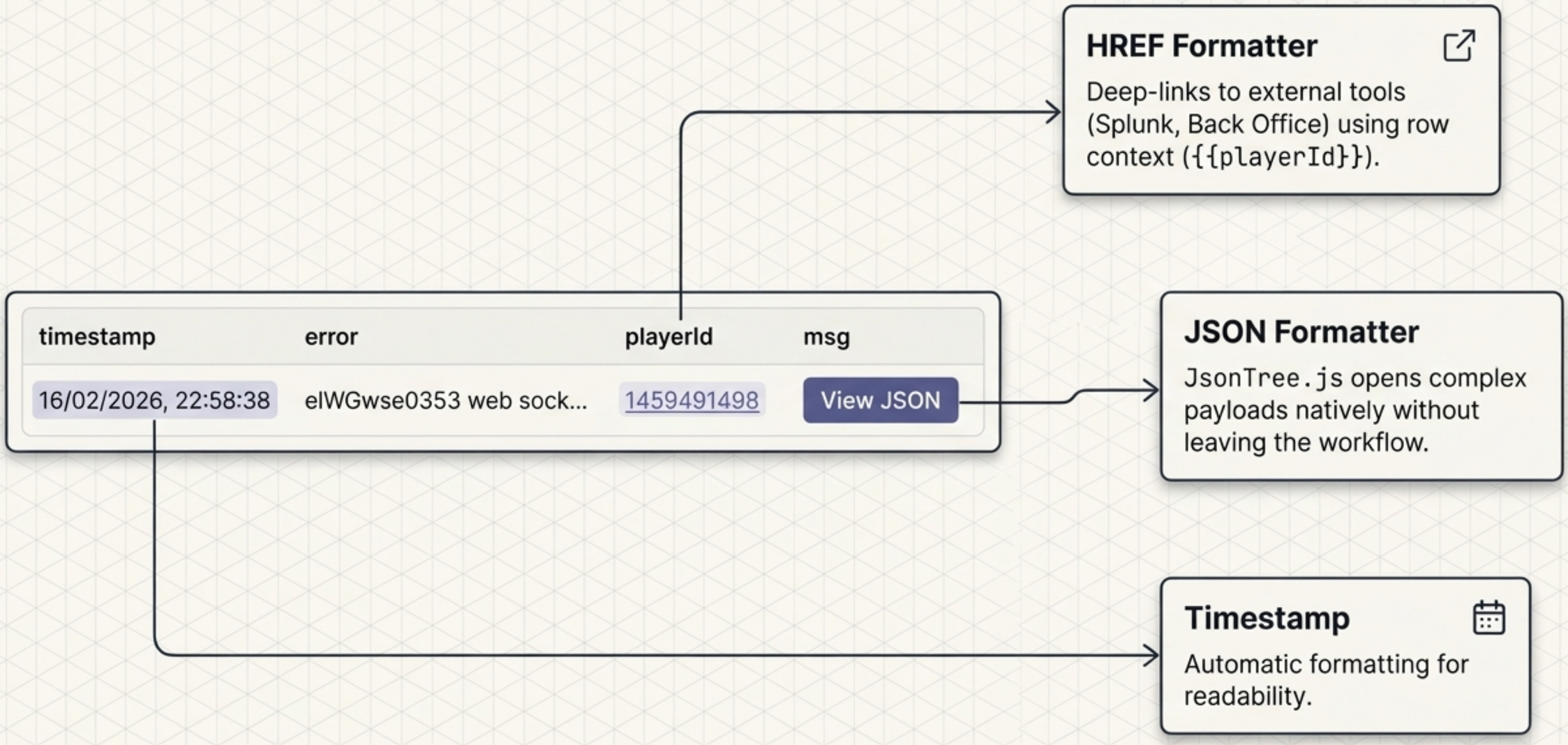
```
JetBrains Mono {
  count(*), sum(Duration);
}
```



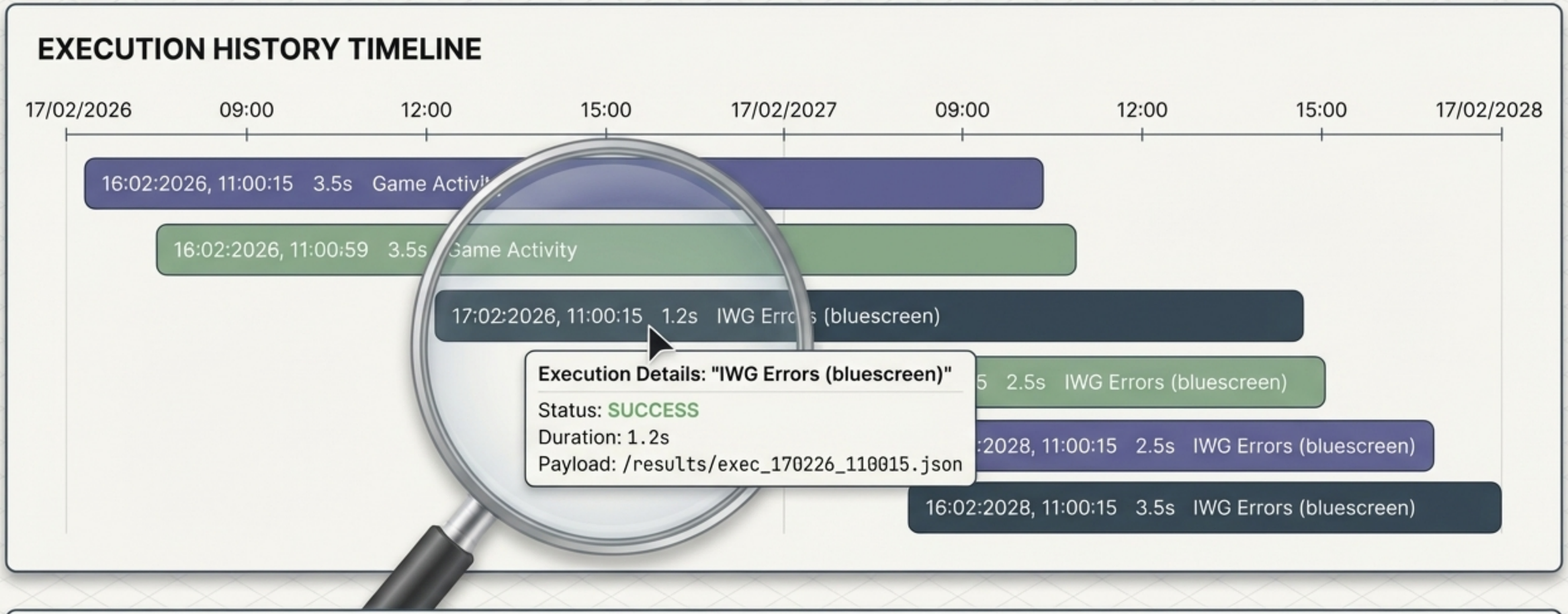
GameName	Count(*)	Sum(Duration)
Minecraft	452	3239.54
Fortnite	30	1323.80
Apex	25	205.66
...	12	78.86

Interactivity: Clicking a chart segment filters the raw data table instantly without re-querying the AWS server.

Deep-Linking Topography



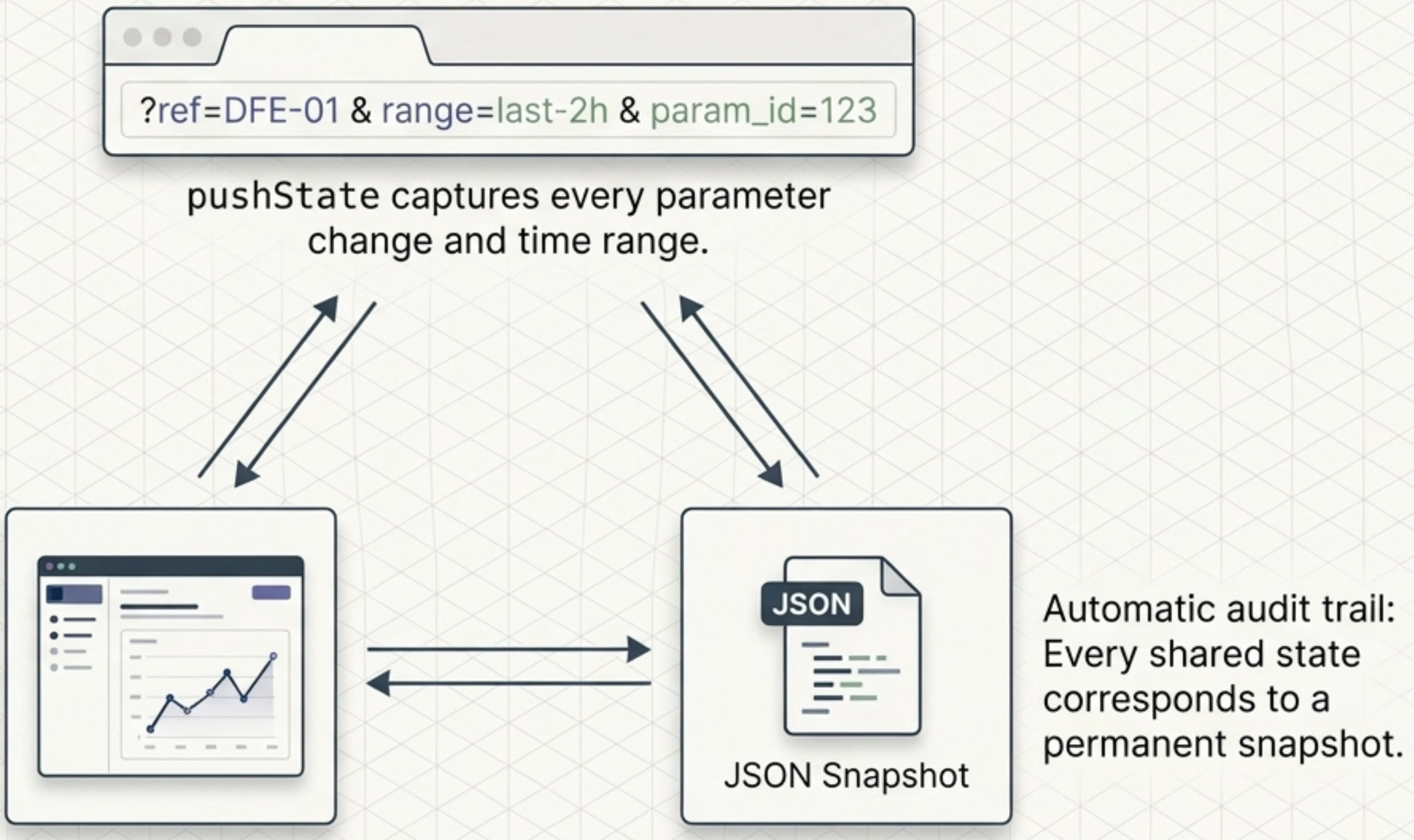
Historical Intelligence: The Timeline



Why re-run a query when the snapshot already exists? Access historical data instantly via locally stored JSON execution payloads.

Editorial Engineering Blueprint

State Persistence & Collaboration



Share the link to load the exact "frozen" analysis state for a colleague.

Automatic audit trail: Every shared state corresponds to a permanent snapshot.

Technical Stack & Deployment

Core Backend

- Node.js (v18.x)
- Express
- WS (WebSockets)
- AWS SDK

Frontend & UX

- Chart.js
- Phosphor Icons
- JsonTree.js

1. Setup

```
npm install
```

2. Authenticate

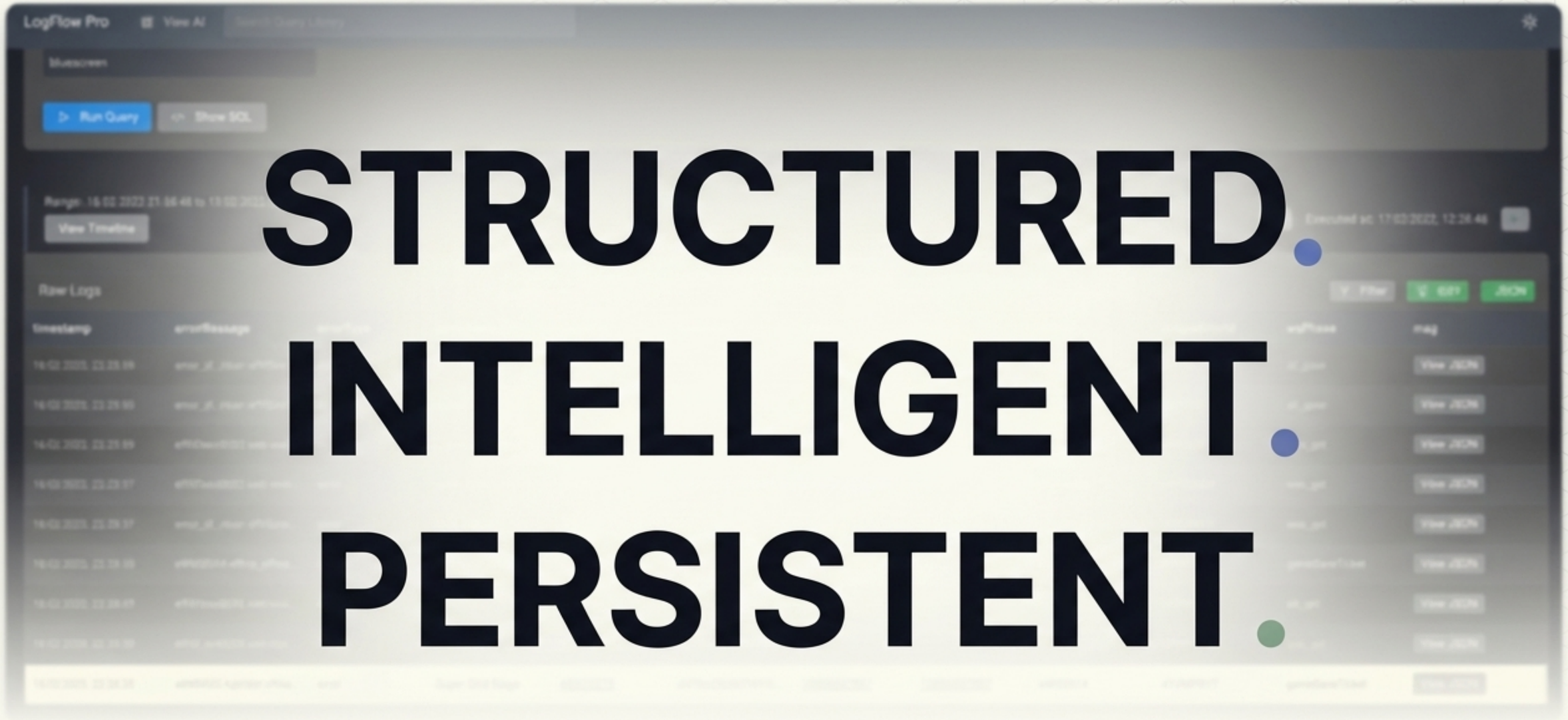
```
aws sso login
```

(Critical)

3. Execute

```
node app.js
```

Port 3001



LogFlow Pro: Turning logs into logic.