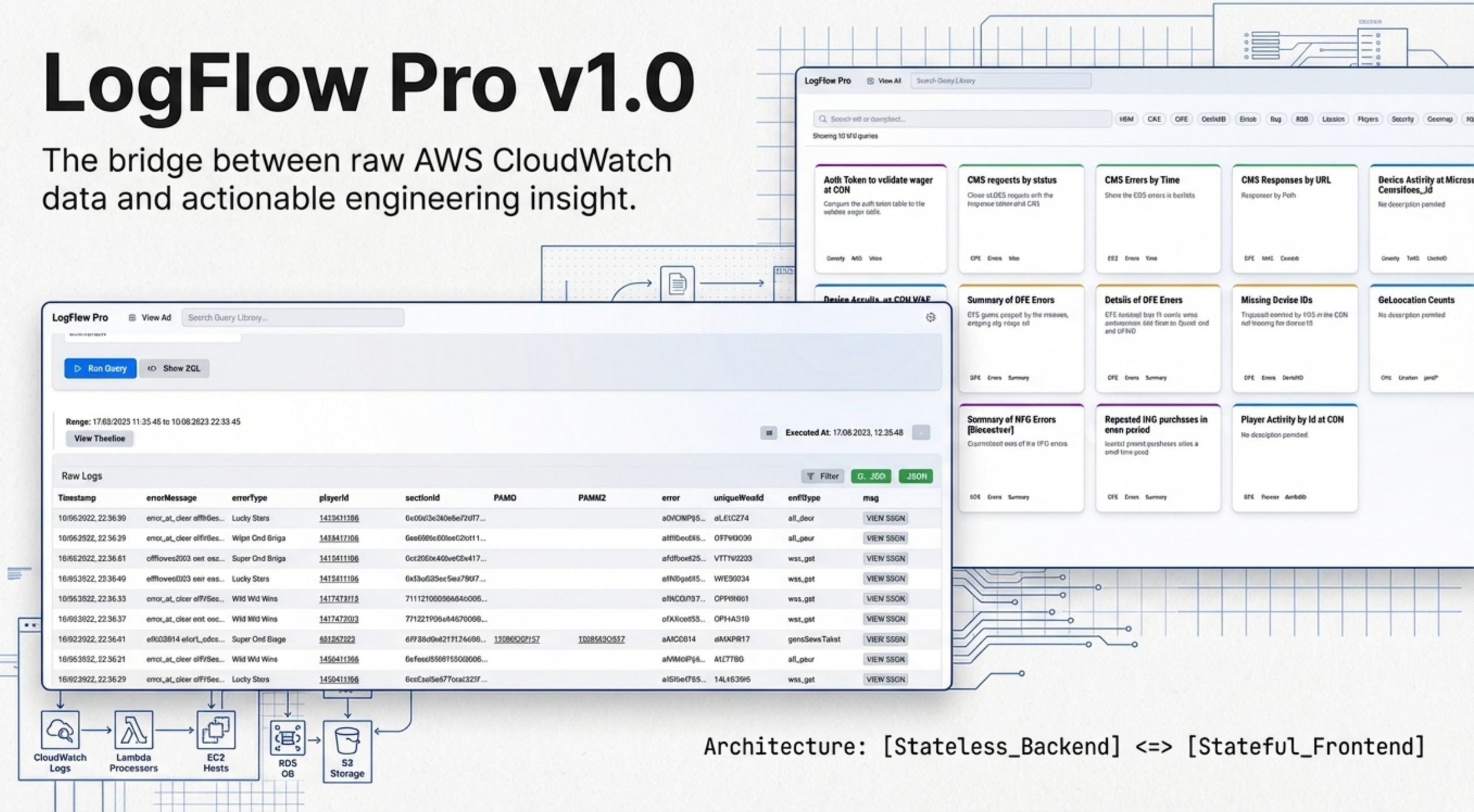
LogFlow Pro v1.0

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

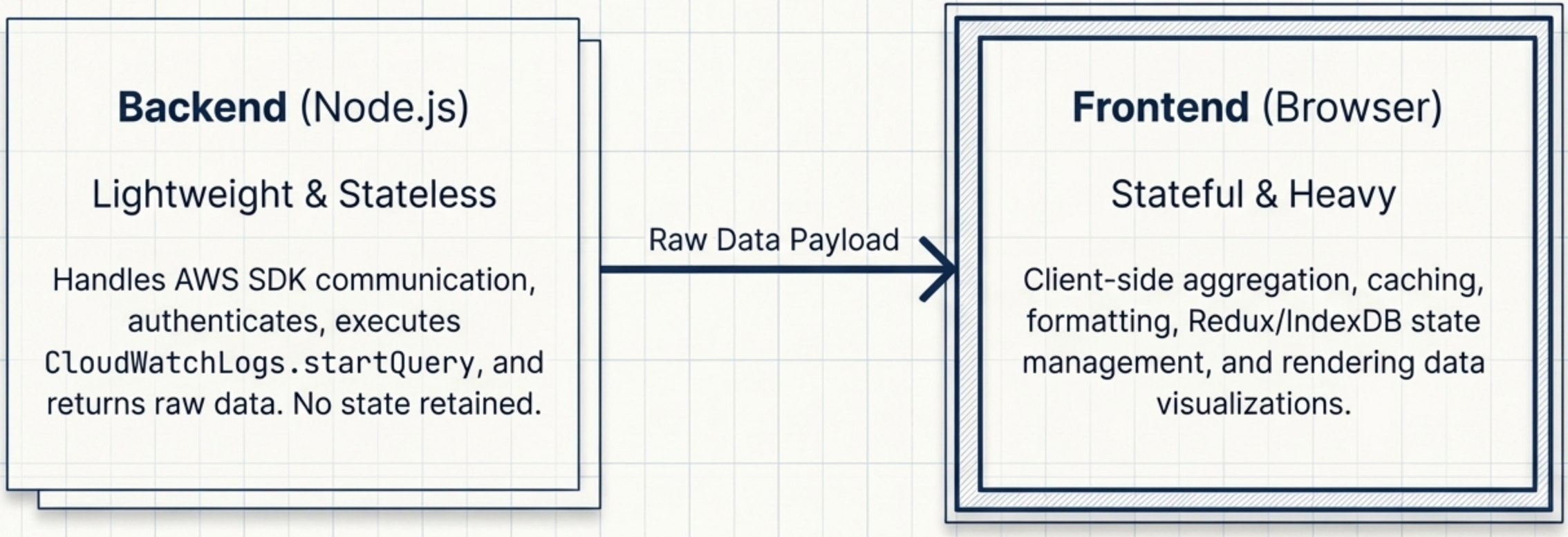


Architecture: [Stateless_Backend] <=> [Stateful_Frontend]

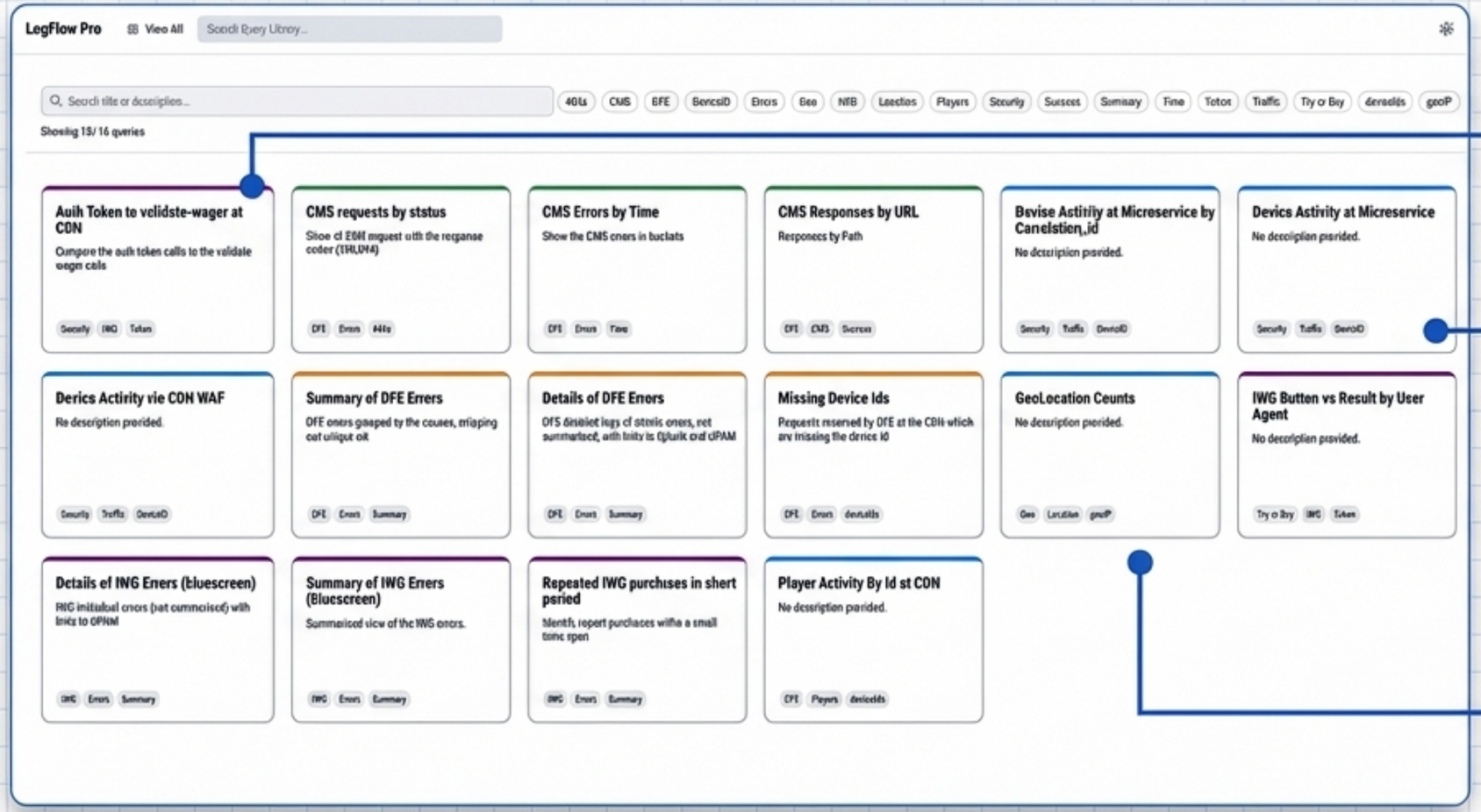
The Diagnostic Matrix: From Ad-Hoc Chaos to Structured Intelligence

	Native AWS CloudWatch	LogFlow Pro
Workflow	Repetitive, manual text entry	Centralized library, dynamic parameters
Query Storage	Unsaved, ephemeral	Persistent `JSON` repository
Time Context	Lost context between runs	Audit trail with precise `executedAt` timestamps
Visualizations	Basic log rows only	Embedded interactive `Chart.js` widgets
State Sharing	Disjointed terminal commands	URL deep links to exact analytical states

Architectural Philosophy: Offloading the State



The Centralized Library: Organizing Institutional Knowledge



Tag-Based Filtering: Organize by functional area (e.g., DFE, Errors, Security).

Visual Cues: Color-coded tile borders defined directly in metadata for rapid visual identification.

Searchability: Instant browsing by title, description, or technical tags.

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

Configuration as Code: Anatomy of a DSL File

```
Ref: SAMPLE-01
Title: Example Error Query
Region: eu-west-2
LogGroups: [/aws/lambda/my-function]

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

Query: fields @timestamp, message |filter level = '{{level}}'
       | sort @timestamp desc | limit 20

Formatters: @timestamp=datetime:YYYY-MM-DD HH:mm:ss

Views: [Log Level Distribution] GroupBy: level, Chart: bar
       [Error Count] GroupBy: @timestamp(1h), Chart: line
```

Metadata & Targeting

Dynamic UI Inputs

Raw AWS CloudWatch SQL

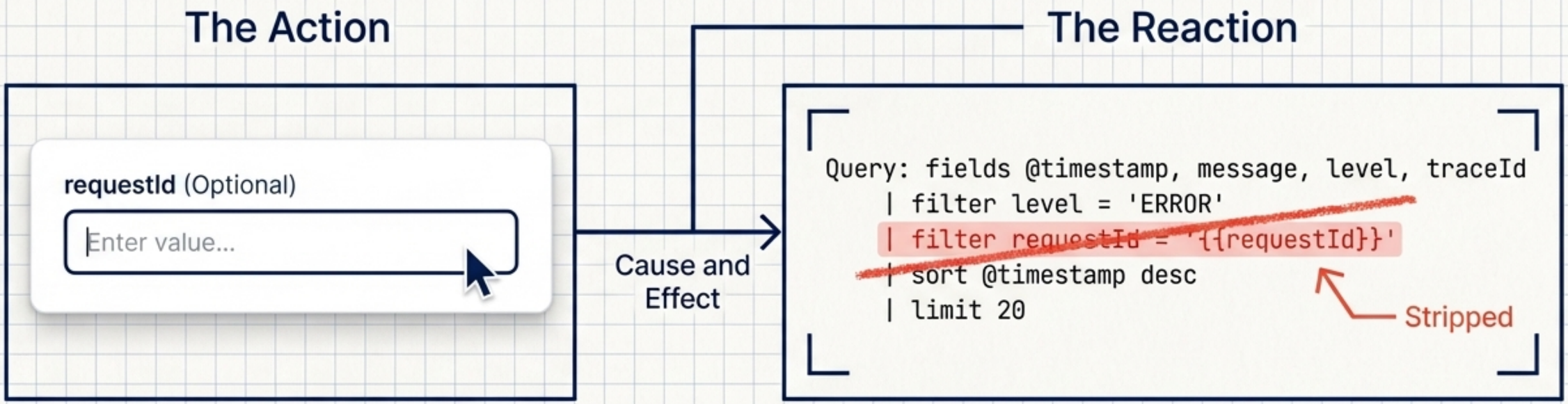
Column Rendering Rules

Client-side Aggregation Definitions

The Parameter Logic Matrix: Handling Dynamic Inputs

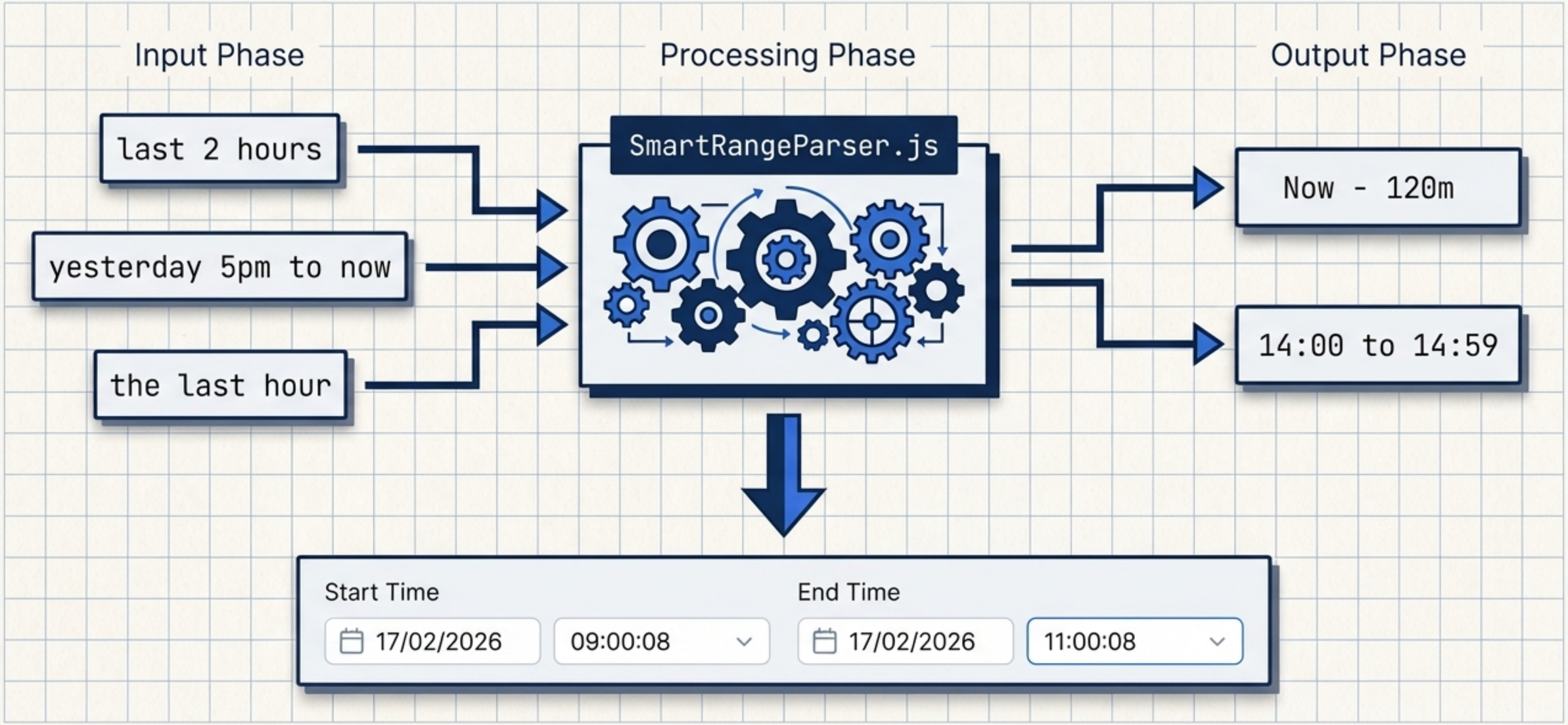
Syntax	User Requirement	Backend Execution Behavior
`{{key}}` ⟶	Mandatory UI input	Red UI warning if empty ⟶ Blocks execution entirely to protect AWS quotas.
`{{key-opt}}` →	Allowed to be left empty	**Triggers the Smart Omission Engine.**

The Smart Omission Engine

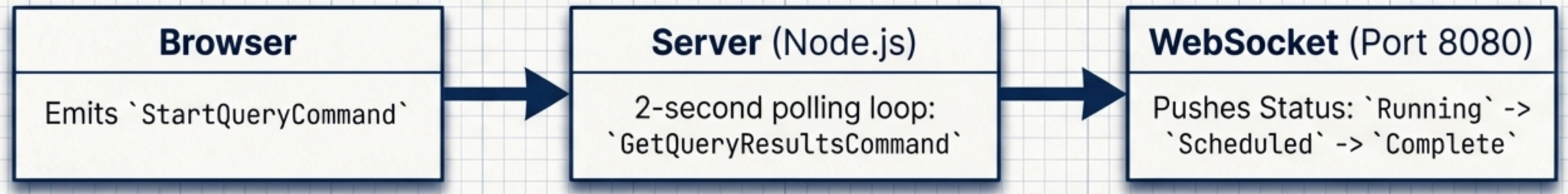


If an optional field is left blank, the backend dynamically strips the entire SQL line containing the placeholder. This prevents AWS syntax errors without requiring complex conditional logic in the query file.

Mastering Time: The Natural Language Funnel



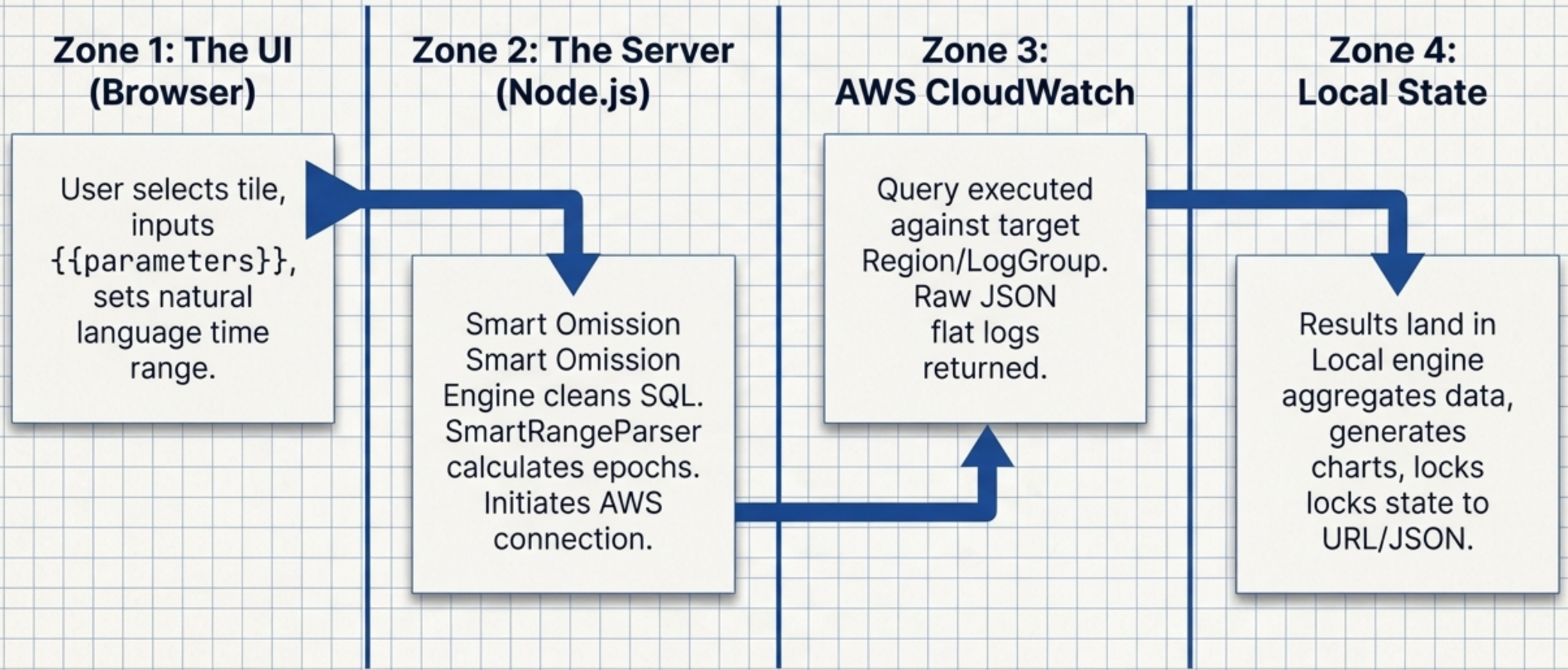
The Architecture of Execution



Self-Healing SSO Credential Guard

Backend continuously monitors for AWS SSO authentication failures. If detected, it executes a hard stop (`process.exit(1)`), instantly signaling the engineer to run `aws sso login` rather than silently failing.

The Lifecycle of a Query: From Input to Insight



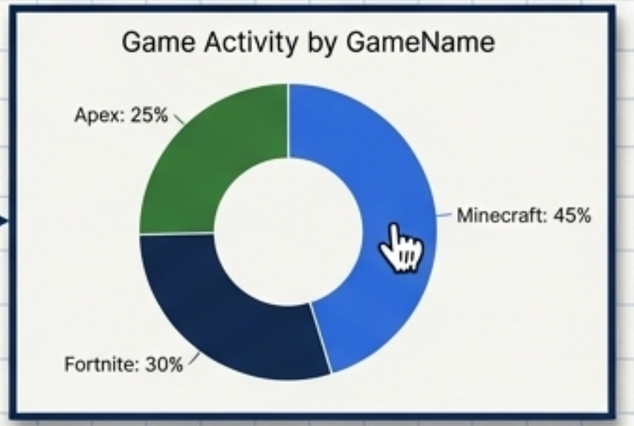
Client-Side Aggregation & Interactive Visualization

Raw Flat JSON Logs

AWS SDK

Client-Side Processing (JavaScript)

GroupBy / Sum

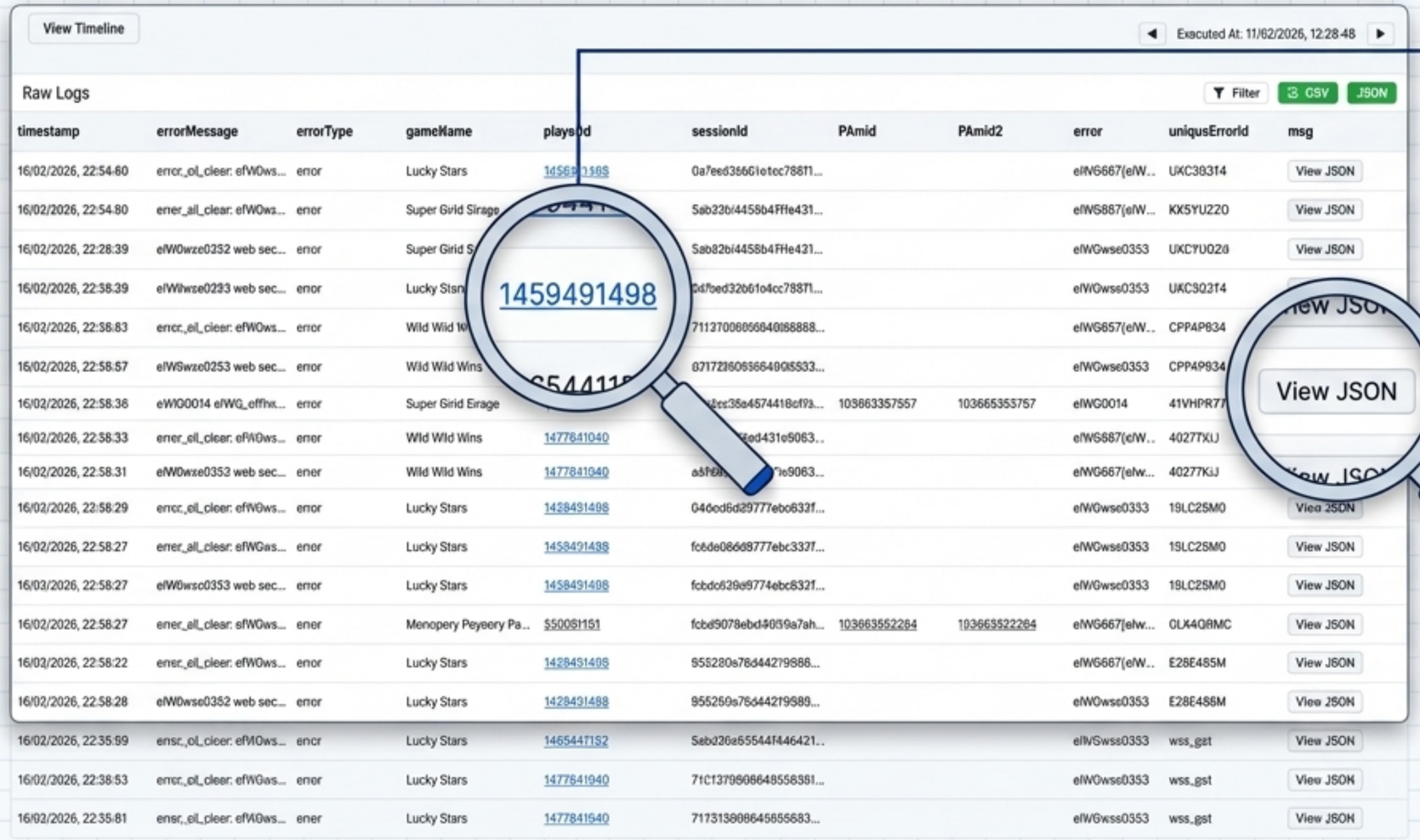


Raw HTML Data Table

GameName	Count(*)	Sum(Duration)
Minecraft	452	3239.54
Fortnite	30	1323.80
Apex	25	205.66
Minecraft	25	897.99

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

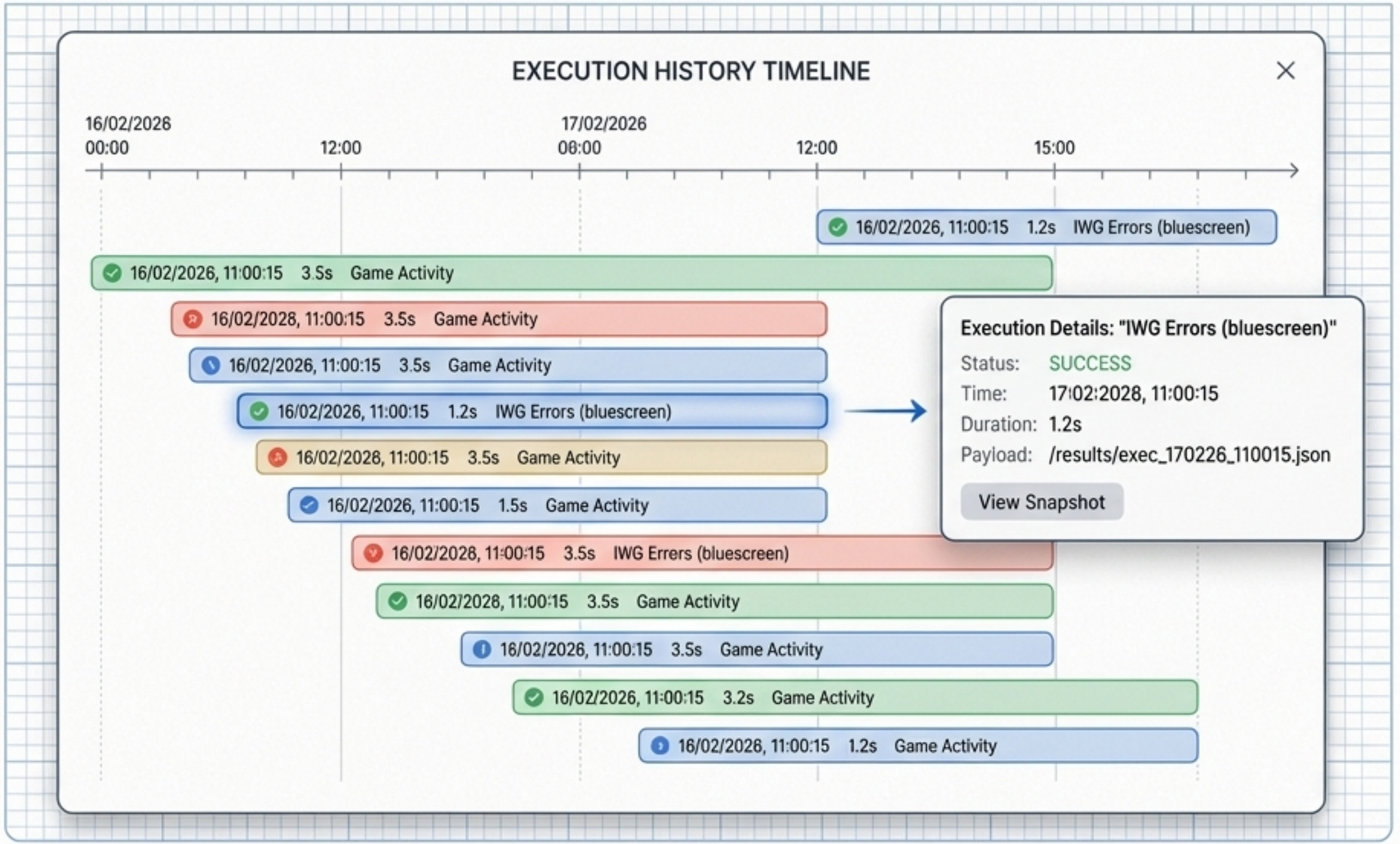
Interactive Data & Contextual Deep Linking



HREF Formatter:
Deep-links to external tools (Splunk, Back Office) by injecting row-context directly into the URL (e.g., /users/{{playerId}}/history).

JSON Formatter:
Opens raw log payloads in an interactive JsonTree.js visual inspector for nested object debugging.

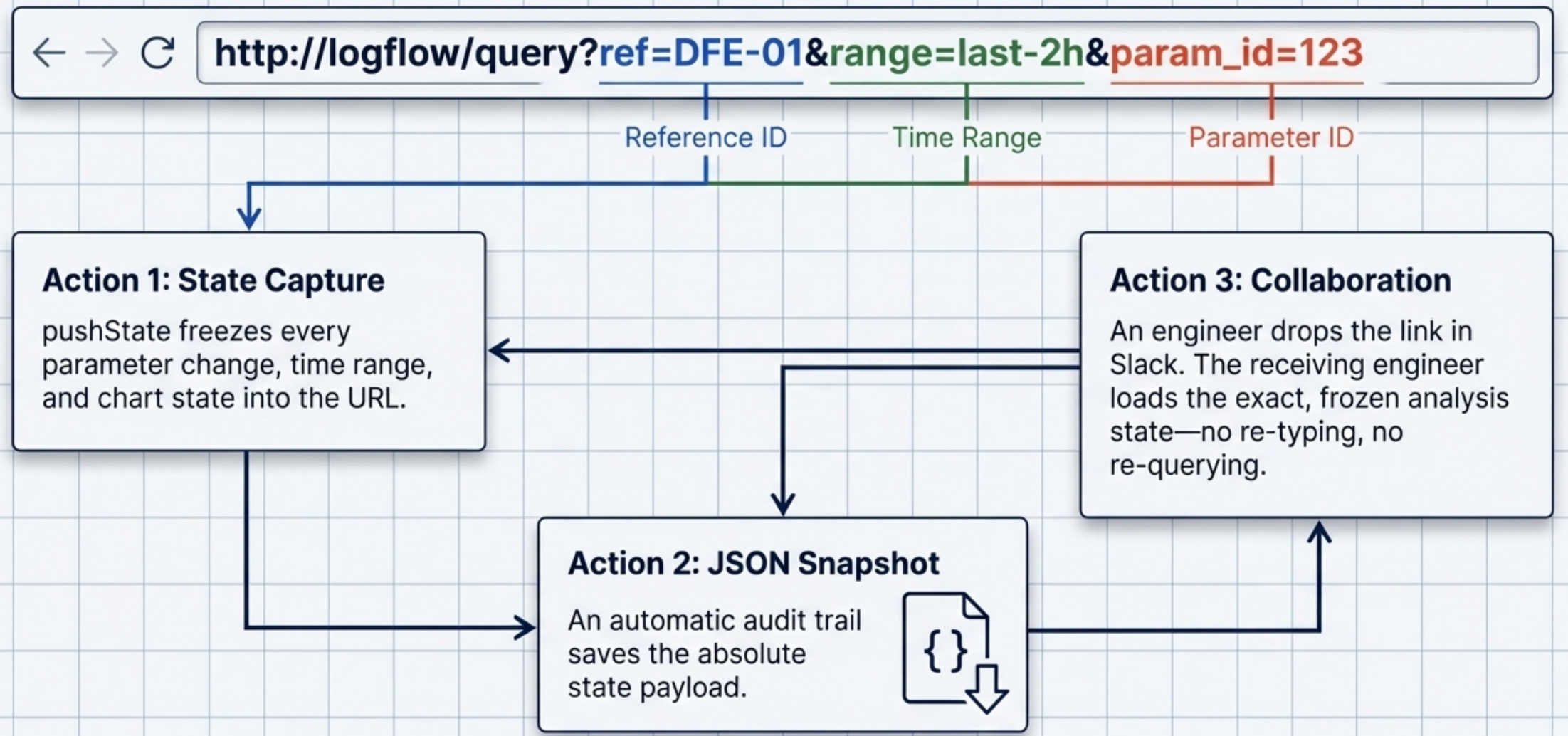
Historical Intelligence: The Timeline God-View



FEATURE HIGHLIGHTS

1. **Visual SVG Modal**: Renders all historical runs as interactive bars on a time-axis.
2. **Persistence:** Every execution is automatically saved as a comprehensive JSON snapshot in the `/results` directory.
3. **Philosophy:** Why re-run an expensive, time-consuming AWS query when the snapshot already exists? Access historical data instantly.

State Persistence & Sharing: Institutional Memory



Technical Stack & Zero-Friction Deployment

The Stack

Core Backend

- **Node.js (v18.x)**: Runtime environment.
- **Express**: Web server framework.
- **WS (WebSockets)**: Library for real-time heartbeats.
- **AWS SDK**: `@aws-sdk/client-cloudwatch-logs`.

Frontend & UX

- **Chart.js**: Data visualization.
- **Phosphor Icons**: Modern, scalable iconography.
- **JsonTree.js**: Payload inspection.

The Deployment Flow

1. Setup

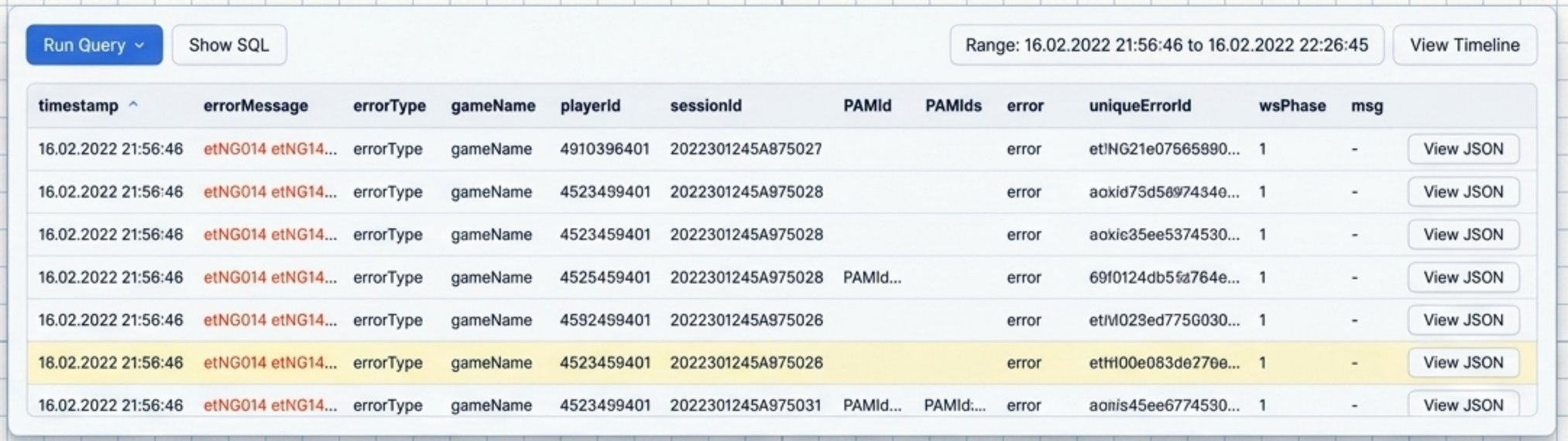
Clone repository.
`-> npm install.`

2. Authenticate

Ensure environment is secured via `aws sso login` **(Critical)**.

3. Execute

Run `node app.js`. (UI Port: 3001, WebSocket: 8080).



STRUCTURED. INTELLIGENT. PERSISTENT.

LogFlow Pro: Turning raw logs into engineering logic.