Mastering Observability with OS Tools

Luca Raveri

Software Engineer at Talentware

Who Am I

- **Hi**👋 I’m Luca Raveri, Software Engineer and AWS Solutions Architect

💻 Backend Development

☁️ Cloud Architectures

🔍 Observability

📌 Venice, Italy

Talentware

Talentware

- **Mission**: Helping companies build, retain, and grow talent through a skill-based approach.

- **Tech Stack**: Full-stack JavaScript, AWS, and an internal Data Science team.

📌 Milan, Italy

Do you
really know
your system?

Talentware

Problem Description

- We run a complex software system in production.
- With limited testing, some bugs still make it to release.
- Metrics show anomalies but don’t reveal the root cause.
- Logs are massive and hard to analyze.
- Troubleshooting takes days and often requires downtime.

What are Monitoring and Observability?

- **Monitoring**: Continuous measurement of a system's health to detect anomalies, performance issues, or downtime.

- **Observability**: The ability to understand the internal state of a complex system by examining the data it produces (logs, metrics, traces).

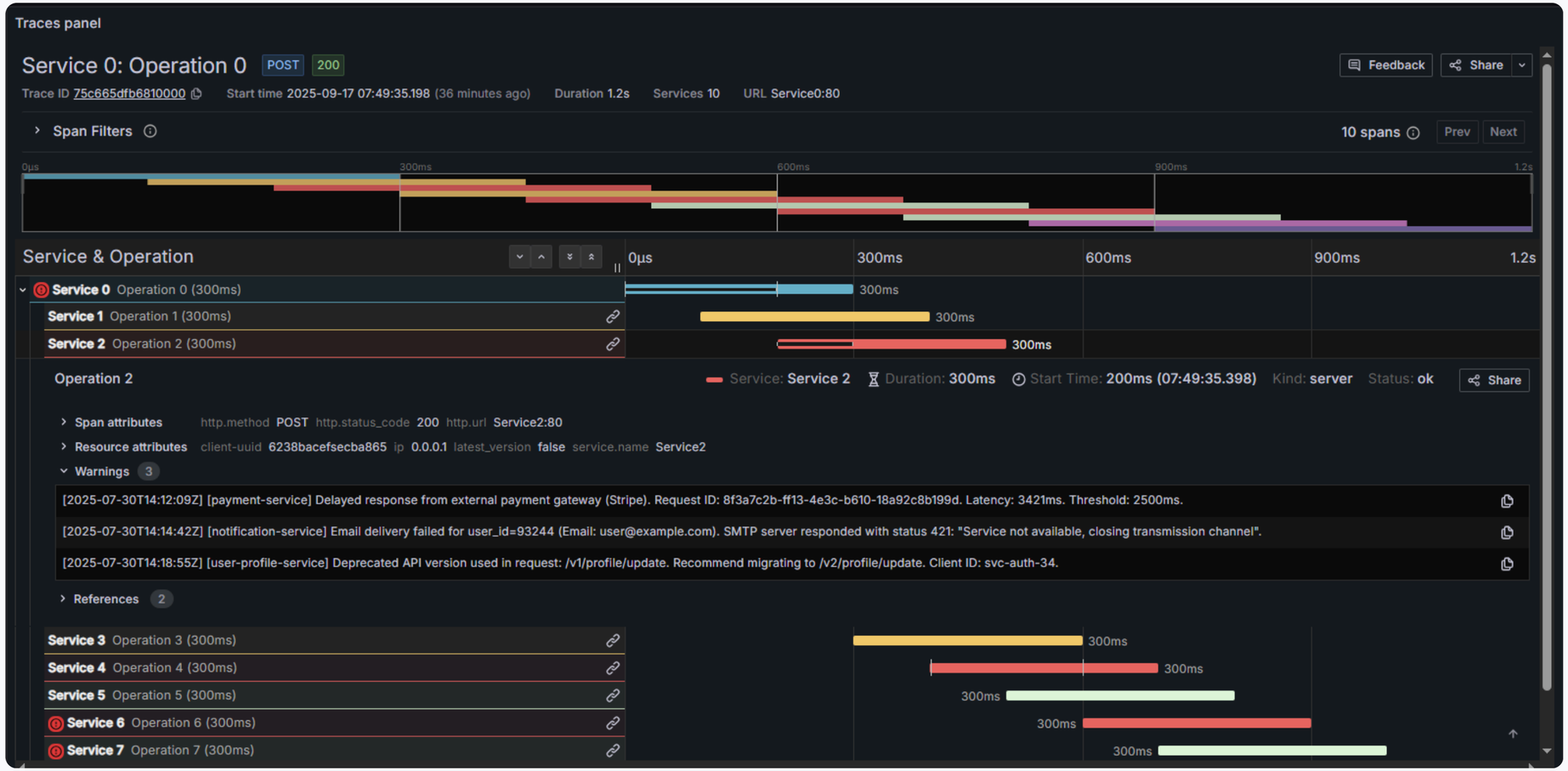
Key difference:

- Monitoring tells you that *something is wrong*.
- Observability helps you understand *why it is wrong*.

The Three Pillars Of Observability

- **Metrics**: Numeric measurements over time (e.g., CPU usage, latency, error rates) → *What is happening*
- **Logs**: Discrete, timestamped records of events (e.g., errors, transactions, system messages) → *Why is happening*
- **Traces:** Detailed records that track the flow of a request through a system, showing how components interact → *Where is happening*

What is a Trace



https://play.grafana.org

Talentware

Key Benefits of Observability

- **Lower MTTD (Mean Time to Detect)**
  Problems are identified quickly, often before users notice.

- **Lower MTTR (Mean Time to Resolve)**
  Root causes are found and fixed faster, reducing downtime and impact.

Other Advantages

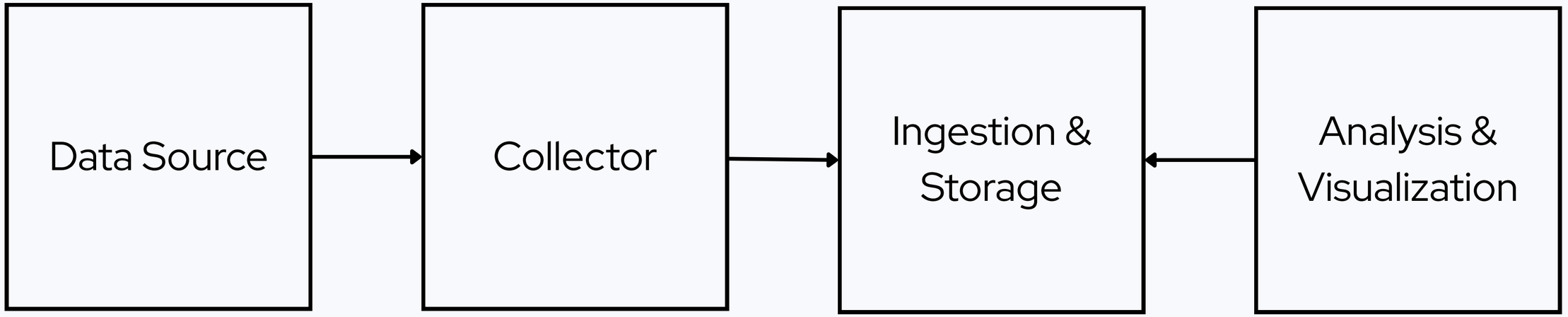
- **Dev Experience**
  Faster debugging, maintainable code

- **Performance**
  Understand current metrics and detect bottlenecks

- **Business**
  Control costs, track features, ensure compliance, make data-driven decisions

Demo Time!

Let’s Build an Observability System



Talentware

LGTM Stack

Loki

Database
for logs

Grafana

Visualization
tool

Tempo

Database
for traces

Mimir

Database
for metrics

Open Source: Pros & Cons

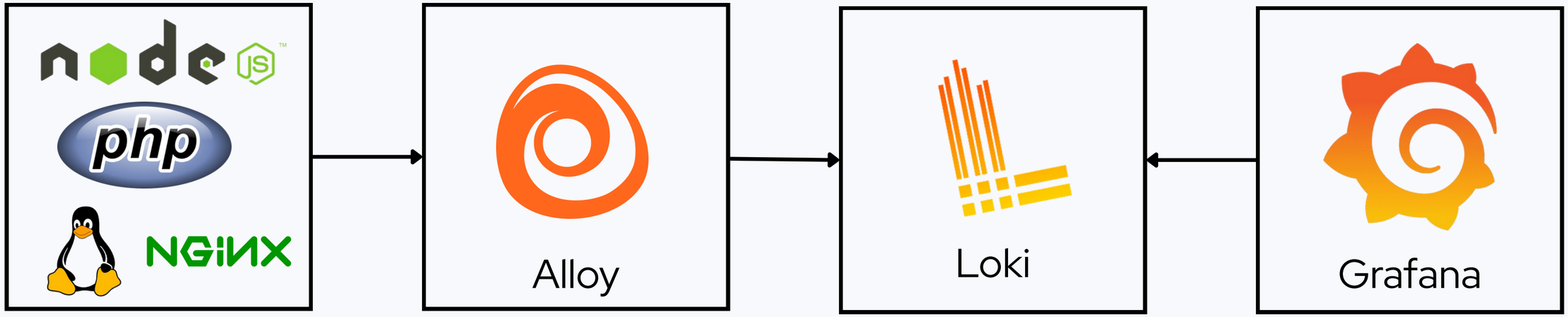
Advantages

- No license costs
- High flexibility and customization
- Strong community support
- Avoid vendor lock-in

Disadvantages

- Higher setup and maintenance effort
- Steeper learning curve
- Limited official support
- Integrations may require extra work

Let’s build an Observability System



Let’s Start Logging

- Use a logger instead of native functions
- Apply log levels and timestamps
- Format logs (JSON)
- Never log secrets and PII
- Set retention policies (e.g., logrotate)

Logging Strategies

- Capture boundaries: log what enters and leaves the system
- Centralize: handle logs in a single, consistent layer if possible
- Correlate: attach a unique ID to every request/transaction
- Balance: more logs mean more insights, but also more overhead (tip: use log levels)

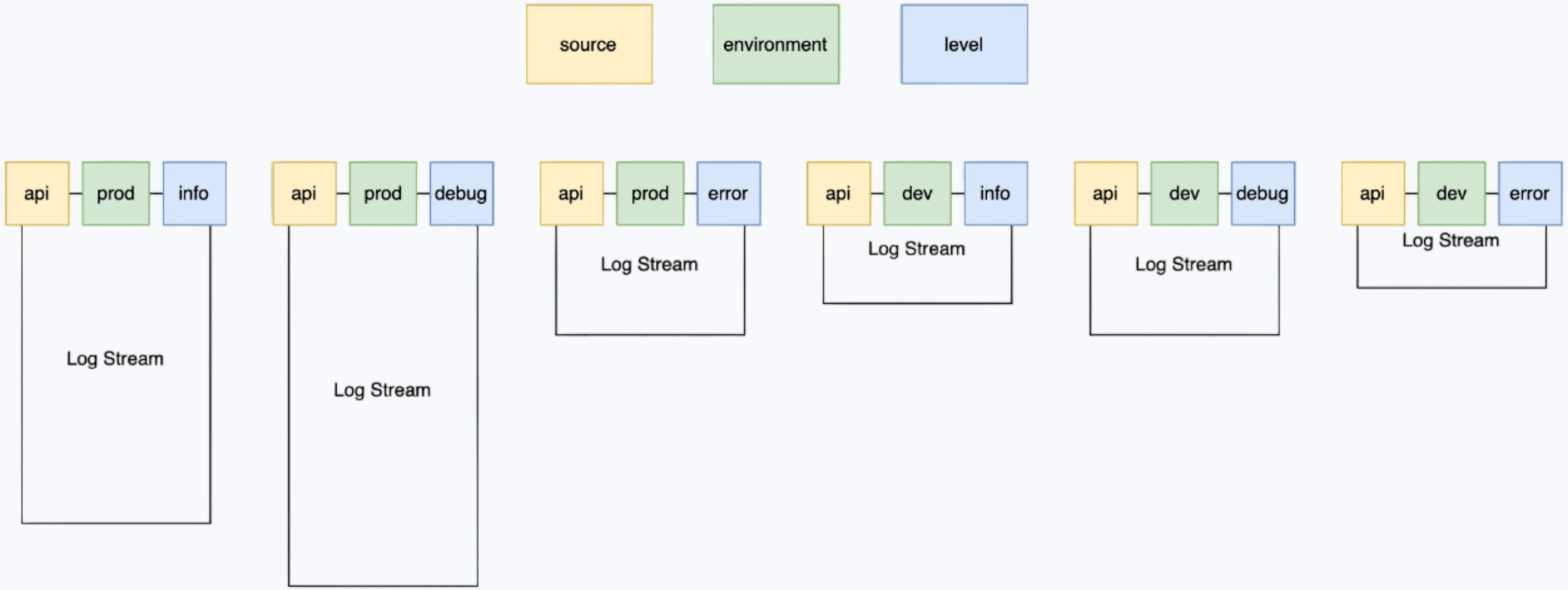
```
console.log(`Customer ${customerName} purchased ${itemsPurchased} items`);
```

```
logger.info('Customer purchase', {
  customerId: customer.id,
  customerName: customer.name,
  itemsPurchased: customer.itemsPurchased,
  totalAmount: customer.totalAmount,
});
```

What is Loki

“Loki is a horizontally scalable, highly available, multi-tenant log aggregation system inspired by Prometheus. It is designed to be very cost effective and easy to operate. It does not index the contents of the logs, but rather a set of labels for each log stream.”

How Loki Works



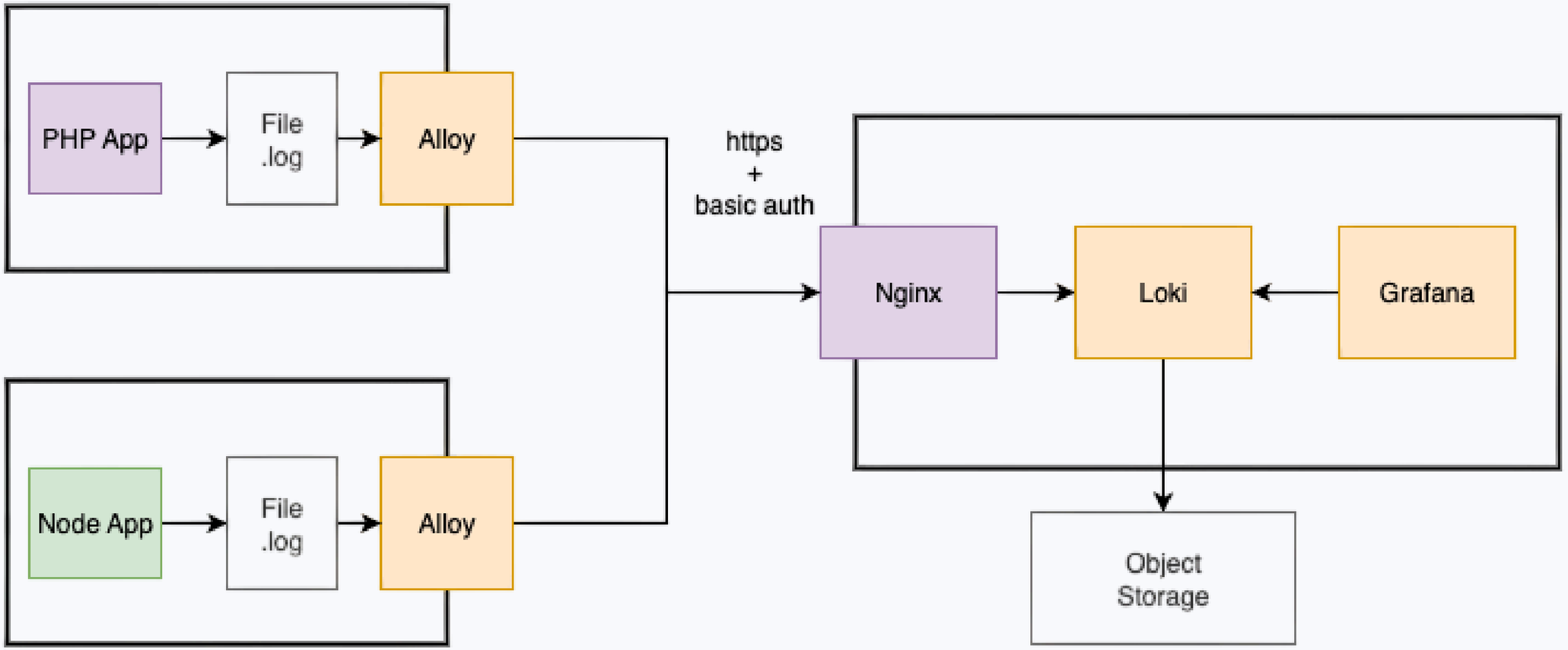
Talentware

How to Send Logs to Loki

Grafana Alloy is a collector that automatically reads your log files and pushes them to Loki. It can:

- Apply **static labels** to all logs from a file
- Extract **dynamic labels** from log content
- Perform **transformations** on log data

Hosting Loki



Talentware

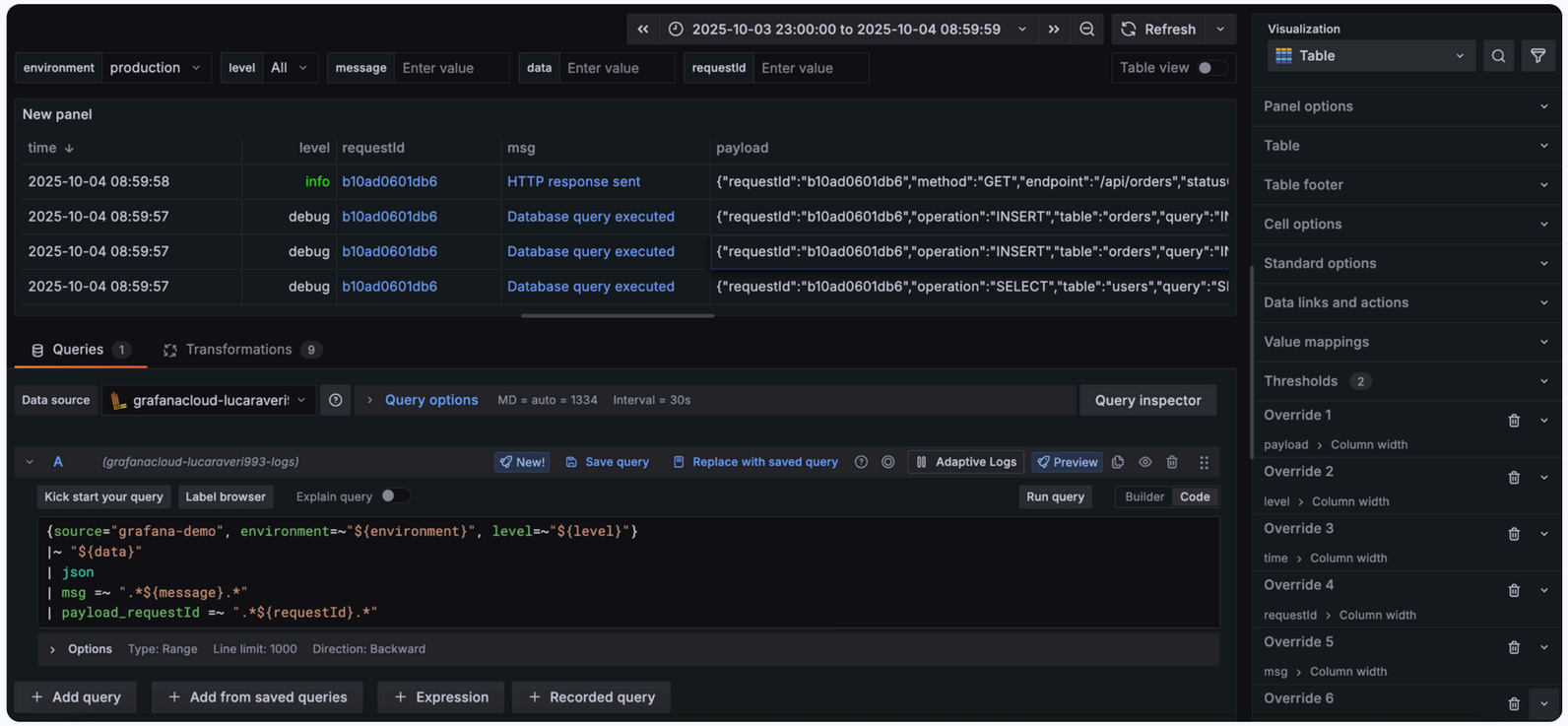
How to Query Logs

```
{source="grafana-demo", environment=~"${environment}", level=~"${level}"}
|~ "${data}"
| json
| msg =~ ".*${message}.*"
| payload_requestId =~ ".*${requestId}.*"
```

```
{job="apache_error_log", environment=~"${environment}", source=~"${source}"}
|~ "${data}"
| regexp "\\[(?P<timestamp>[^\\]]+)\\].*\\[php7:(?P<level>[^\\]]+)\\].*PHP (?P<type>[^:]+): (?P<message>.*?) in (?
P<file>[A-Z]:\\\\[^\"]+?)(?::(?P<line>\\d+)| on line (?P<line_alt>\\d+))"
| line_format "{{ printf `{\"timestamp\":\"%s\", \"level\":\"%s\", \"type\":\"%s\", \"message\":\"%s\",
\"file\":\"%s\", \"line\":\"%s\"}` .timestamp .level .type .message .file (or .line .line_alt) }}"
| message =~ ".*${message}.*"
| file =~ ".*${file}.*"
| message != ""
```

How to Visualize Logs

1. Write the query
2. Apply transformations
3. Create variables
4. Add overrides

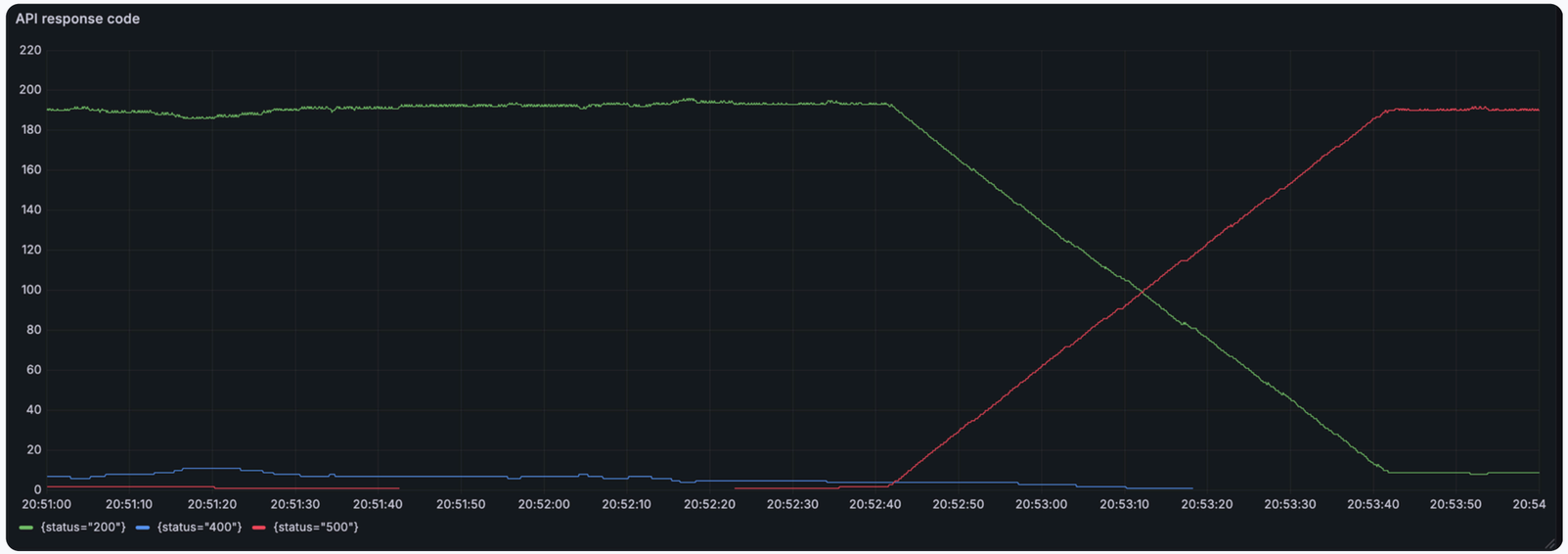


Alerting Strategies

- Single error detection

  Trigger when a critical error appears in a log line.
- Error rate spikes (RED method)

  Alert when the rate of errors suddenly increases, signaling a systemic issue.



Talentware

Results

- Strong team adoption and enthusiasm.
- Significantly improved **MTTD** and reduced **MTTR** from 2 days to 2 hours, enhancing service quality.
- Handled 1 GB/day with 2-week retention (14 GB total), running smoothly on a €6/month instance.

Thank you!

Questions?

Demo Repository