OpenSearchCon EUROPE

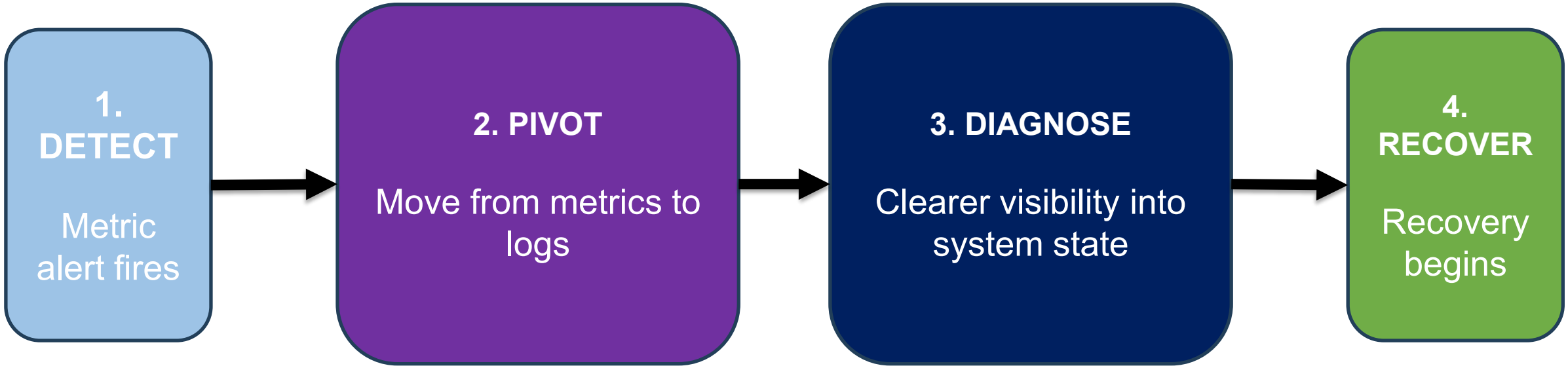
From Logs to Signals: Rethinking System Monitoring with OpenSearch

Shruti Chaturvedi
DevOps Engineer

A Common Workflow for Detecting System State

1. Metrics detect first. They tell us:
   - Errors are spiking.
   - Latency is increasing.
   - We have shifted from our desired state.
2. Engineers search logs for context:
   - Is a downstream dependency failing?
   - Which region impacted?
   - What percentage of users are impacted?
   - Which endpoint is affected?
3. Infer system state manually:
   - Payment dependency is timing out.
   - Impact is isolated to eu-west-1
   - Checkout endpoint is degraded, not fully down.

Fast Detection. Slower Diagnosis. Slower Recovery.



|----------INVESTIGATION DELAY----------|

The Shift: Logs As Signals

	Before: Logs as evidence	After: Logs as signals
Process	Alert fires, then engineers search logs.	Queries evaluate logs directly.
Role of logs	Logs explain what already happened.	Logs help detect and classify what is happening
Diagnosis	Scope and cause is inferred manually.	Cause and scope is returned with the evaluated result.
Output	Broad symptom, limited context.	Explicit state, localized context.
Example	*/checkout: latency high*	DEPENDENCY_FAILURE: payment-service failing

Deriving State from Logs with OpenSearch

Structured logs

JSON fields: dependency, latency, error_type, status.

Indexed in OpenSearch

Mapped for filtering and aggregations.

Query index over a time window

Compute counts, p95, and group by dependency or region.

Explicit System State

Evaluate state, and return clearer operational conclusion.

This turns **logs from supporting evidence into an evaluated signal.**

Structured Logs – Key Requirement!

Machine Parsable	Consistent	Context-rich	Semantic
• Emit JSON • No raw text	• Same keys and types across services.	• Include region, dependency, latency, etc.	• Log meaningful events, not generic messages.

If logs are not machine-parsable, consistent, context-rich, and semantic, state cannot be derived reliably.

Example: Structured Logs in Practice

{} checkout-service-log.json > ...

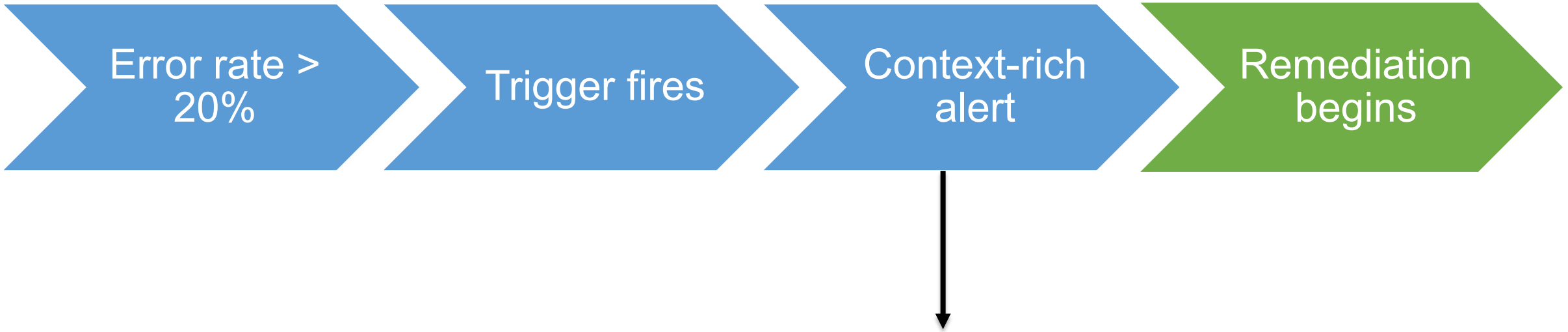
```json
{
    "service": "checkout",
    "endpoint": "/checkout",
    "status": "error",
    "latency_ms": 201,
    "retry_count": 2,
    "dependency": "payment-service",
    "error": "Payment Service Unavailable",
    "error_type": "DEPENDENCY_FAILURE",
    "@timestamp": "2026-04-09T10:00:05Z"
}
```

- ✓JSON: Machine Parsable
- ✓Context-rich
- ✓Consistent fields
- ✓Semantic

Demo Scenario: Dependency Failure

Monitoring flow	Simulated problem
checkout-service emits structured JSON logs.	*payment-service* begins failing.
Logs are mapped into an OpenSearch index.	Checkout requests start producing errors.
OpenSearch monitor queries the index and aggregates failures.	The issue is localized dependency degradation, not full system downtime.
OpenSearch trigger evaluates the query result.	The pattern should map to *DEPENDENCY_FAILURE*
When the error rate reaches 20%, the trigger fires and an alert is sent.	The alert context identifies a dependency failure.

Demo: Alert with Context



Investigation Delay Shortened

Before	After
Symptom detected.	Failure classified.
Infer system state manually.	Context comes with the alert explaining system state.
Diagnosis delayed.	Diagnosis starts immediately.

Structured logs turned symptoms into diagnosis before humans had to investigate.

The Essence: Alert With Context

dependency-alert.log

```
2026/04/12 08:19:20 🚨 ALERT RECEIVED
2026/04/12 08:19:20 Monitor: Dependency Failure Monitor

Total Requests:
50

Endpoint Breakdown:
- /checkout : 50

Status Breakdown:
- success : 29
- error : 21

Dependency Failures:
- payment-service :
    dependency_payment_service_down : 21

Time Window:
2026-04-12T02:48:19.632Z → 2026-04-12T02:49:19.632Z
```

When This Works and When It Does Not

Works well when:

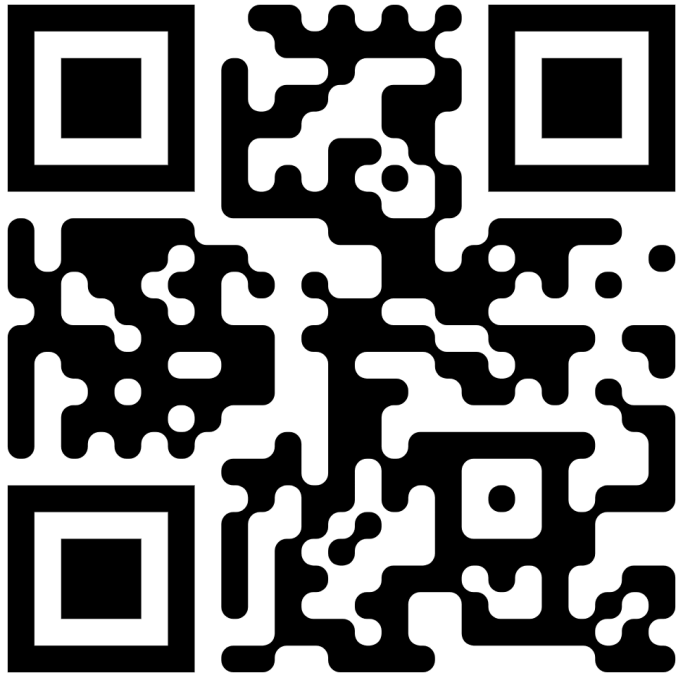
- Logs are structured, consistent, and context-rich.
- Queries can map patterns from logs to state.
- You want diagnosis earlier in the incident flow.

Breaks down when:

- Logs are inconsistent, missing context.
- Volume or noise overwhelms the signal.
- You try to use logs as a full replacement for metrics.

OpenSearch turned structured logs into actionable signals.

Demo Repo



Thank you!

Shruti Chaturvedi
GitHub: ShrutiC-git


OpenSearchCon
EUROPE