Observability and SLOs with Elastic

Date: January 29, 2026

Observability and SLOs with the Elastic Stack

Building Reliable Systems with Service Level Objectives

Rajesh Kesavalalji | 01/29/2026 | Elastic Seattle User Group

AGENDA

1. Introduction to Observability and SLOs
2. Understanding SLIs, SLOs, and Error Budgets
3. Elastic Stack for Observability
4. Demo: Checkout API with SLO Tracking
5. Key Takeaways

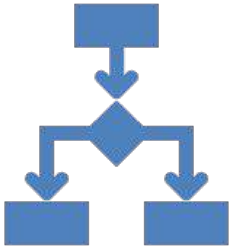
WHAT IS OBSERVABILITY?

The Three Pillars

- 📊 **Metrics**: Quantitative measurements over time
  - Request rate, latency, error rate, resource utilization
- 📝 **Logs**: Discrete events with timestamps
  - Application logs, access logs, error logs
- 🔍 **Traces**: Request flow through distributed systems
  - Distributed tracing, span context, service dependencies

Observability = Understanding system behavior from the outside

WHY OBSERVABILITY MATTERS



From Reactive to Proactive

❌ **Traditional Monitoring**: "Something is broken, let's investigate"

✅ **Observability**: "Why is the system behaving this way?"

Benefits with Observability

⚡ Faster incident resolution

🔮 Proactive problem detection

😊 Better user experience

📈 Data-driven decisions

SERVICE LEVEL OBJECTIVES (SLOs)

What are SLOs?

SLO: A target level of reliability for a service

Example: "99.9% of requests should succeed"

Purpose:

- Set clear reliability expectations
- Enable data-driven decisions
- Balance reliability vs. feature velocity

SLO ≠ SLA

- SLO: Internal target
- SLA: External commitment (with consequences)

SERVICE LEVEL INDICATORS (SLIs)

What We Measure

SLI: A quantitative measure of service reliability

Common SLIs:

- ✅ **Availability**: Percentage of successful requests
- ⏱️ **Latency**: Response time (p50, p95, p99)
- 📊 **Throughput**: Requests per second
- ❌ **Error Rate**: Percentage of failed requests

Example: "99.9% of checkout requests succeed within 200ms"

ERROR BUDGETS

The Safety Margin

Error Budget: The acceptable amount of unreliability

Calculation: 100% - SLO Target

Example:

- SLO: 99.9% availability
- Error Budget: 0.1% (43.2 minutes/month)

Purpose:

- Enable risk-taking
- Guide deployment decisions
- Balance reliability vs. innovation

BURN-RATE ALERTS

Early Warning System

Burn Rate: How fast error budget is consumed

🚨 **Fast-Burn Alert**: Error budget consumed 2x faster than sustainable

- Short window (6 hours)
- Immediate action required

⚠️ **Slow-Burn Alert**: Error budget consumed 1.5x faster than sustainable

- Long window (30 days)
- Proactive intervention

Goal: Catch issues before SLO violation

ELASTIC STACK OVERVIEW

The Observability Platform

🔍 **Elasticsearch**: Search and analytics engine

- Stores metrics, logs, traces
- Real-time indexing and querying

📊 **Kibana**: Visualization and dashboards

- APM UI, SLO tracking, custom dashboards

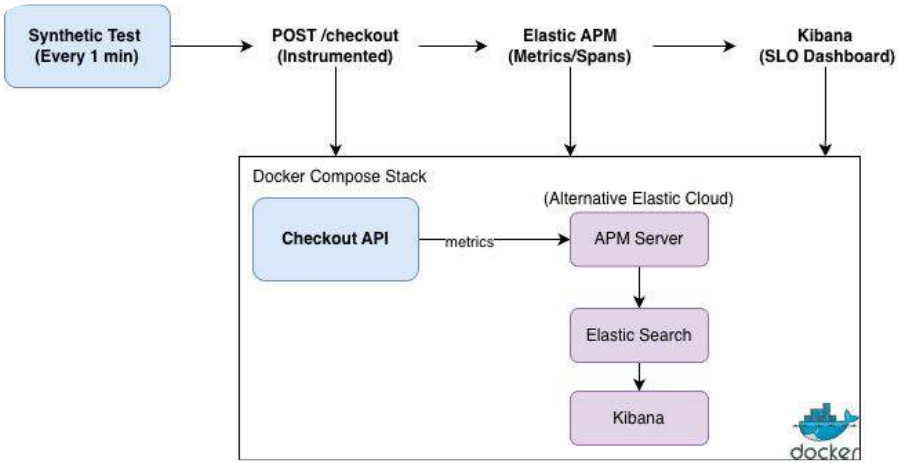
📡 **APM Server**: APM data collection

- Receives data from APM agents
- Processes and indexes to Elasticsearch

🔌 **APM Agents**: Application instrumentation

- Automatic transaction and span tracking

DEMO ARCHITECTURE



Checkout API with Elastic Stack

DEMO: CHECKOUT API

What We're Building

Service: E-commerce checkout API

SLIs:

✅ Availability: 99.5% success rate

⏱️ Latency: p95 < 200ms

Instrumentation: Elastic APM Node.js agent

Scenarios:

- Normal operation
- High latency (database issues)
- High error rate (payment failures)

Github: https://github.com/rajesharma470/elastic-observability-demo

DEMO FLOW - PART 1

Setting Up SLOs

1. **Create Availability SLO**:
   - Service: `checkout-api`
   - Transaction: `POST /checkout`
   - Target: 99.50%
   - Window: 30 days
2. **Create Latency SLO**:
   - Service: `checkout-api`
   - Transaction: `POST /checkout`
   - Target: p95 < 200ms (for demo 99.50%)
   - Window: 30 days
3. **Baseline**: Normal operation with 0% errors

SLOs

Search your SLOs ...

Overview

2	0	0	0	0
Healthy	Violated	No data	Degrading	Stale

Showing **1-2** of 2 **SLOs**

Checkout API Availability

Target 99.499%

99.709%

Checkout API Latency

Target 99.5%

99.778%

DEMO FLOW - PART 2

Simulating Issues

1. **Latency Scenario:**
   - Simulate slow database (5x latency multiplier)
   - Watch p95 latency exceed 200ms
   - Observe latency SLO violation
   - Error budget consumption
2. **Error Scenario**:
   - Simulate payment failures (10% error rate)
   - Watch availability drop below 99.50%
   - Observe availability SLO violation
   - Error budget burn

SLOs

Search your SLOs ...

Overview

0	2	0	0	0
Healthy	Violated	No data	Degrading	Stale

Burn rate

1
Active ale

Showing **1-2** of 2 **SLOs**

So

Checkout API Availability

Target 99.8%

99.767%

Checkout API Latency

⚠ 1

Target 99.62%

99.516%

KEY FEATURES DEMONSTRATED

What We Saw

✅ Automatic Instrumentation: No code changes needed for basic APM

✅ Real-time Monitoring: Live metrics and traces

✅ SLO Tracking: Built-in SLO management in Kibana

✅ Error Budget Tracking: Visual error budget consumption

✅ Burn-Rate Alerts: Proactive alerting

BEST PRACTICES

SLO Implementation Tips

1. **Start Simple**: Begin with 1-2 critical SLIs
2. **User-Focused**: Measure what users experience
3. **Realistic Targets**: Set achievable SLOs
4. **Review Regularly**: Adjust based on data
5. **Document Decisions**: Why this SLO? Why this target?
6. **Error Budgets**: Use them to guide deployments
7. **Automation**: Automate SLO tracking and alerts

Thank you

LinkedIn: https://www.linkedin.com/in/rajesh-sharma-sfbay/

Github Repo: https://github.com/rajesharma470/elastic-observability-demo