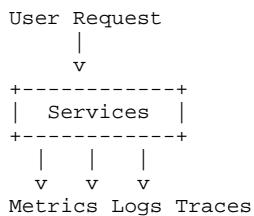


Observability – Complete 2 Hour Demo Guide

1. What is Observability?

Observability is the ability to understand the internal state of a system by examining its externally produced signals. In software systems, these signals are Metrics, Logs, and Traces.



2. Why Observability Exists

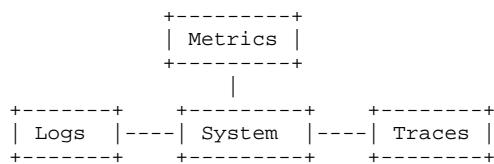
Modern systems are distributed, dynamic, and complex. Traditional monitoring cannot explain unknown failures. Observability enables teams to ask new questions without predefined dashboards.

3. Observability vs Monitoring

Monitoring	Observability
-----	-----
Known problems	Unknown problems
Static alerts	Exploratory analysis
Reactive	Proactive

4. Three Pillars of Observability

Metrics, Logs, and Traces together form complete observability.



5. Metrics – Detailed Explanation

Metrics are numeric measurements collected over time. Examples: CPU usage, latency, error rate, throughput. Used for alerting, dashboards, and trend analysis.

6. Logs – Detailed Explanation

Logs are timestamped records of events. They provide context and details for debugging failures.

7. Traces – Detailed Explanation

Traces show the end-to-end journey of a request across services. Each trace consists of spans.

```
Trace ID: 123
[API Gateway]
  |
[User Service]
  |
[Order Service]
  |
[Payment Service]
```

8. Golden Signals of Observability

Golden signals help measure user experience and system health.

1. Latency -> How long requests take
2. Traffic -> How many requests
3. Errors -> Failed requests
4. Saturation -> Resource exhaustion

9. Latency – Deep Dive

Latency includes tail latency (p95, p99). Even if average latency is low, tail latency impacts users.

10. Traffic – Deep Dive

Traffic measures system load. Examples: Requests per second, messages per second.

11. Errors – Deep Dive

Errors indicate user-visible failures. HTTP 5xx errors are critical; 4xx indicate client issues.

12. Saturation – Deep Dive

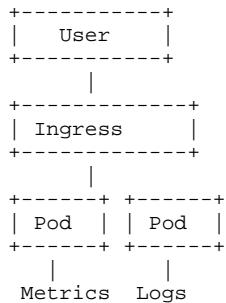
Saturation shows how close the system is to its limits. CPU, memory, disk, queues.

13. Real-Time Use Case: Microservices Debugging

```
User -> API -> Order -> Payment
  |
  Timeout
```

Observability helps identify which service and dependency caused the issue.

14. Kubernetes Observability



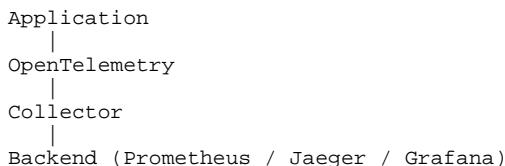
15. Business Impact Correlation

Observability correlates technical failures to revenue and user impact.

16. Why Observability is Critical

- Reduces MTTR - Improves reliability - Enables SRE practices - Supports continuous delivery

17. Observability Architecture



18. OpenTelemetry Overview

OpenTelemetry is the industry standard for collecting metrics, logs, and traces. Vendor-neutral and cloud-native.

19. Demo Flow for 2 Hours

1. Concepts (30 min)
2. Golden Signals (20 min)
3. Real Use Cases (30 min)
4. Architecture & Tools (20 min)
5. Q&A (20 min)

20. Summary

Observability is essential for operating modern distributed systems. Without it, systems become unmanageable at scale.