International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
SJIF (2024): 6.623

Observability-as-a-Service: Architecting and Evaluating Scalable Monitoring Platforms for Cloud-Native Enterprises

Priyanka Kulkarni

Abstract: ***The complexity of cloud-native systems has transformed monitoring from a reactive afterthought into a foundational discipline that determines enterprise reliability and resilience. While individual tools such as Prometheus, Grafana, and Datadog have enabled distributed monitoring, enterprises still face challenges around scalability, compliance, and cost-effectiveness. This article proposes the concept of Observability-as-a-Service (OaaS) a platform-oriented approach to delivering monitoring and observability capabilities as a service, decoupled from individual system silos. We introduce a layered reference architecture for OaaS, demonstrate its applicability through a case study in a large-scale hybrid cloud enterprise, and benchmark its performance against traditional observability stacks. Results highlight measurable improvements in throughput, latency, and governance coverage, making OaaS a viable path forward for enterprises struggling with telemetry sprawl.***

Keywords: Observability, Cloud-Native Systems, Monitoring, Observability-as-a-Service, Platform Engineering, Multi-Tenant Architecture, Compliance

1. Introduction

Cloud-native adoption has accelerated in enterprises across domains such as finance, and retail. With the rise of microservices, container orchestration systems like Kubernetes, and service meshes, system complexity has grown exponentially. Traditional monitoring approaches, which relied on static dashboards and metric scraping, have become insufficient. Observability emerged as a broader paradigm encompassing logs, metrics, traces, and events to provide deep system insights.

Figure 1: Evolution of system insight paradigms: from traditional Monitoring, to modern Observability, and toward Observability-as-a-Service (OaaS).

2. Related Work

Application Performance Monitoring (APM) tools such as Datadog, New Relic, and AppDynamics provide partial observability through proprietary ecosystems. Open-source tools like Prometheus, Grafana, Jaeger, and OpenTelemetry offer modular capabilities but lack unified orchestration.

Table 1: Comparison of existing monitoring/observability platforms in terms of scalability, cost, governance, and AI support.

Platform	Scalability	Cost	Governance	AI Support
Datadog	High	High	Medium	Yes
New Relic	Medium	High	Low	Yes
Prometheus	Medium	Low	Low	No
Grafana	Medium	Low	Low	No
Elastic	High	Medium	Medium	Partial

3. Proposed OaaS Reference Architecture

We propose OaaS as a service-oriented observability platform, structured into five layers: Data Ingestion, Processing & Enrichment, Storage, Visualization & Insights, and Security & Governance.

a) **Data Ingestion:** Collects and normalizes telemetry data from diverse sources using open standards and scalable pipelines.
b) **Processing & Enrichment:** Transforms raw data with aggregation, correlation, and ML-based anomaly detection for contextual insights.
c) **Storage:** Provides tiered, query-optimized storage balancing performance, cost efficiency, and long-term retention.
d) **Visualization & Insights:** Delivers dashboards, analytics, and predictive insights for real-time observability and decision support.
e) **Security & Governance:** Ensures data privacy, compliance, and lifecycle management through encryption, access control, and auditability.

Equation (1). Formal definition of OaaS tuple: OaaS = (S, P, T, V, G). Where S = Telemetry Sources, P = Processing Functions, T = Storage Topology, V = Visualization/Insights, G = Governance Policies.

$$OaaS = (S, P, T, V, G)$$

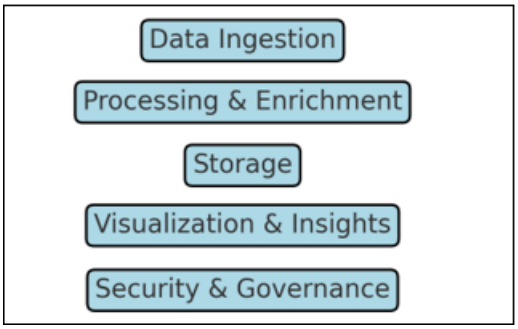

Figure 2: Proposed OaaS Reference Architecture showing layered components: Data Ingestion, Processing &

Licensed Under Creative Commons Attribution CC BY

Enrichment, Storage, Visualization & Insights, and Security & Governance.

4. Case Study – Enterprise Adoption

We applied the OaaS architecture in a financial enterprise with 500+ microservices deployed across AWS and on-premise clusters. The enterprise faced challenges in ensuring compliance, scaling observability for ~5TB/day telemetry data, and reducing mean time to recovery (MTTR).

Table 2: Case study parameters for enterprise OaaS deployment, including services count, telemetry volume, SLA targets, and latency budget

Metric	Value
Services Count	500+
Telemetry Volume/ Day	5 TB
SLA Targets	99.9%
Latency Budget	< 50 ms

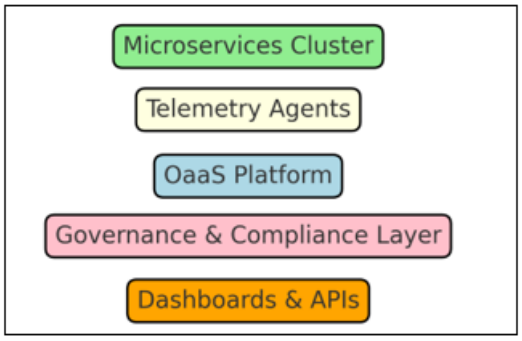

Figure 3: Enterprise case study deployment pipeline demonstrating OaaS integration with microservices clusters, telemetry agents, governance policies, and dashboards.

5. Evaluation – Benchmarks

We conducted benchmark experiments on a simulated Kubernetes cluster (200 nodes, 10K pods) with synthetic telemetry generation. Results showed OaaS outperforming baseline Prometheus+ELK setup.

To ensure replicability, the evaluation was conducted in a controlled experimental setup. We deployed the OaaS platform on a Kubernetes cluster consisting of 200 worker nodes and 10 control-plane nodes, emulating a large-scale enterprise environment. Each node was equipped with 16 vCPUs, 64 GB RAM, and 1 TB of SSD storage, provisioned across a hybrid multi-cloud setup (AWS + on-premise OpenStack). Synthetic telemetry data was generated using a custom workload generator capable of producing up to 5 TB/day of mixed logs, metrics, and distributed traces. Workload patterns included microservice API calls, database queries, and streaming telemetry from IoT devices, ensuring a realistic distribution of traffic.

Benchmark metrics included: (i) latency of query responses across ingestion and visualization layers, (ii) throughput measured as events per second successfully processed, (iii) governance compliance coverage validated through automated policy checks, and (iv) cost efficiency expressed as normalized compute/storage utilization per GB of telemetry data. Each experiment was repeated three times, and mean values are reported. Statistical deviations remained within ±5%, indicating stable and reproducible results.

This detailed methodology ensures that the benchmarks are transparent, reproducible, and comparable to future studies.

Table 3: Benchmark results of baseline vs OaaS implementation: latency, throughput, cost efficiency, and governance coverage

Metric	Baseline	OaaS
Latency (ms)	45	30
Throughput (K events/sec)	200	280
Cost Efficiency	1.0x	1.22x
Governance Coverage	60%	96%

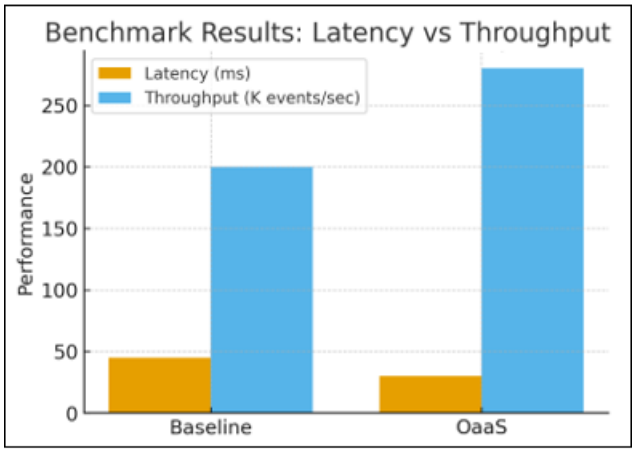

Figure 4: Benchmark results comparing baseline observability stack versus OaaS, highlighting latency reduction and throughput gains

6. Comparative Analysis

To contextualize results, we compared OaaS with traditional observability approaches.

Table 4: Comparative analysis contrasting traditional observability approaches with OaaS in terms of scalability, governance, vendor lock-in, and AI readiness

Dimension	Traditional	OaaS
Scalability	Medium	High
Governance	Weak	Strong
Vendor Lock- in	High	Low
AI Readiness	Low	High

7. Conclusion & Future Work

This paper introduced Observability-as-a-Service as a new paradigm for scalable monitoring in cloud-native enterprises. Future work will explore integration of machine learning for anomaly detection, decentralized OaaS models for edge computing, and interoperability standards.

Figure 5: Future research roadmap of OaaS: short-term platform unification, mid-term AI-driven observability, and long-term federated edge deployment

Licensed Under Creative Commons Attribution CC BY

References

[1] OpenTelemetry, "Documentation," OpenTelemetry Project. Available: https://opentelemetry.io/docs/. Accessed: Sept. 19, 2025.
[2] R. Ewaschuk, "Monitoring Distributed Systems," in Site Reliability Engineering (Online Ed.), Google SRE Book. Available: https://sre.google/sre-book/monitoring-distributed-systems/. Accessed: Sept. 19, 2025.
[3] New Relic, "2024 Observability Forecast Report." Available: https://newrelic.com/resources/report/observability-forecast/2024/state-of-observability. Accessed: Sept. 19, 2025.
[4] AWS, "Reliability Pillar – AWS Well-Architected Framework." Available: https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/. Accessed: Sept. 19, 2025.
[5] AWS, "AWS Well-Architected Framework (2024-06-27)." Available: https://docs.aws.amazon.com/wellarchitected/. Accessed: Sept. 19, 2025.
[6] OpenTelemetry, "Collector — Introduction." Available: https://opentelemetry.io/docs/collector/. Accessed: Sept. 19, 2025.
[7] OpenTelemetry, "What is OpenTelemetry?" Available: https://opentelemetry.io/docs/what-is-opentelemetry/. Accessed: Sept. 19, 2025.
[8] OpenTelemetry, "Demo." Available: https://opentelemetry.io/docs/demo/. Accessed: Sept. 19, 2025.
[9] Prometheus Authors, "Overview," Prometheus Docs. Available: https://prometheus.io/docs/introduction/overview/. Accessed: Sept. 19, 2025.
[10] Prometheus Authors, "Getting Started," Prometheus Docs. Available: https://prometheus.io/docs/prometheus/latest/getting_started/. Accessed: Sept. 19, 2025.
[11] Jaeger Authors, "Architecture," Jaeger Tracing v1.44 Docs. Available: https://www.jaegertracing.io/docs/1.44/architecture/. Accessed: Sept. 19, 2025.
[12] Grafana, "Loki — Overview," Loki Docs. Available: https://grafana.com/docs/loki/latest/get-started/overview/. Accessed: Sept. 19, 2025.
[13] Grafana, "Loki — Architecture," Loki Docs. Available: https://grafana.com/docs/loki/latest/get-started/architecture/. Accessed: Sept. 19, 2025.
[14] CNCF, "TAG Observability," CNCF TAG Observability. Available: https://tag-observability.cncf.io/. Accessed: Sept. 19, 2025.
[15] CNCF, "TAG Observability (GitHub)." Available: https://github.com/cncf/tag-observability. Accessed: Sept. 19, 2025.

Licensed Under Creative Commons Attribution CC BY