An Open-Source Unified Observability Plane for Microservices

Rahul Kamble
Dept. of Information Technology
Pune Institute of Computer Technology
Pune, India
rohitkamble3018@gmail.com

Sahil Katkamwar
Dept. of Information Technology
Pune Institute of Computer Technology
Pune, India
sahilkatkamwar24@gmail.com

Soham Holkar
Dept. of Information Technology
Pune Institute of Computer Technology
Pune, India
sohamholkar800@gmail.com

Ayush Deshmukh
Dept. of Information Technology
Pune Institute of Computer Technology
Pune, India
ayushdeshmukh774@gmail.com

Sachin Pande
Dept. of Information Technology
Pune Institute of Computer Technology
Pune, India
sspande@pict.edu

***Abstract*—Microservices architectures enable scalable and modular system design; however, their distributed topology significantly complicates real-time observability and root cause analysis [22]. Each service generates isolated telemetry signals—logs, metrics, and traces—resulting in fragmented visibility and elevated Mean Time to Resolution (MTTR) [15]. This paper proposes a unified, open-source observability framework leveraging OpenTelemetry, Fluent Bit, Prometheus, Jaeger, OpenSearch, and Grafana to perform intelligent aggregation, correlation, and Machine Learning (ML)-based anomaly detection across all telemetry streams. The multi-tier architecture transitions traditional reactive monitoring into proactive fault prediction, thereby reducing Mean Time to Detect (MTTD) and MTTR while enhancing system reliability in cloud-native and hybrid environments. The primary contribution is the intelligent fusion of logs, metrics, and traces to construct a comprehensive cause-to-effect diagnostic narrative for preemptive failure mitigation.**

***Keywords*—Microservices, observability, logs, metrics, traces, OpenTelemetry, anomaly detection, distributed tracing, cloud-native, Prometheus, OpenSearch.**

I. INTRODUCTION

Microservices have emerged as the dominant paradigm for constructing scalable and maintainable distributed applications [22]. Each service operates autonomously, supports independent deployment, and contributes to overall system resilience. However, the proliferation of services introduces significant operational complexity. The volume and heterogeneity of telemetry data—performance metrics, event logs, and execution traces—scale exponentially with system size, overwhelming traditional monitoring approaches [18].

Observability refers to the ability to infer internal system states from externally emitted telemetry signals [15]. These signals are categorized into three primary types:

- Metrics: Time-series aggregates representing resource utilization, latency, error rates, and throughput—suitable for threshold-based alerting [16].
- Logs: Structured or unstructured event records providing granular execution context for forensic analysis [17].
- Traces: Distributed execution paths that reconstruct request propagation across service boundaries, exposing latency bottlenecks and failure propagation [3].

Conventional monitoring systems treat these signals in isolation, rely on static thresholds, and lack adaptability to dynamic cloud environments [10]. This paper presents a unified observability framework that integrates all three telemetry types into a coherent diagnostic pipeline, enabling proactive anomaly detection and automated root cause analysis.

II. OBJECTIVES AND UNIQUE CONTRIBUTION

A. Project Objectives

1) Aggregate and correlate logs, metrics, and traces from all microservices into centralized storage and analysis platforms to eliminate data silos and enable unified telemetry processing [15].
2) Automatically infer and visualize inter-service dependencies to enable rapid root cause analysis (RCA) of cascading failures through comprehensive dependency mapping [3].
3) Standardize telemetry instrumentation using OpenTelemetry to ensure vendor-agnostic data correlation, context propagation, and interoperability across heterogeneous environments [21].
4) Deploy an adaptive, unsupervised ML-based anomaly detection system for early identification of performance degradation, novel failure modes, and emergent system pathologies [1].
5) Deliver a unified visualization interface (Grafana/OpenSearch Dashboards) that integrates all telemetry streams into a single operational view, supporting streamlined developer and operations workflows [18].

B. What's Unique

While existing solutions typically address individual observability pillars, this framework introduces a Unified Observability Plane through end-to-end integration of logs, metrics, and traces. The core innovation lies in the ML-driven anomaly detection module, which learns baseline behavioral patterns and identifies deviations without requiring labeled training data [4]. This enables correlation of metric anomalies (e.g., CPU saturation) with corresponding traces (e.g., service-level bottlenecks) and log events (e.g., configuration errors), constructing a complete causal chain for each incident [2]. The system transitions from reactive incident response to predictive system health management, significantly reducing MTTD and MTTR.

III. UNIFIED OBSERVABILITY FRAMEWORK

Our framework leverages a 100% open-source technical stack, engineered to overcome the data fragmentation inherent in microservices. Refer to Fig. 1 on the following page.

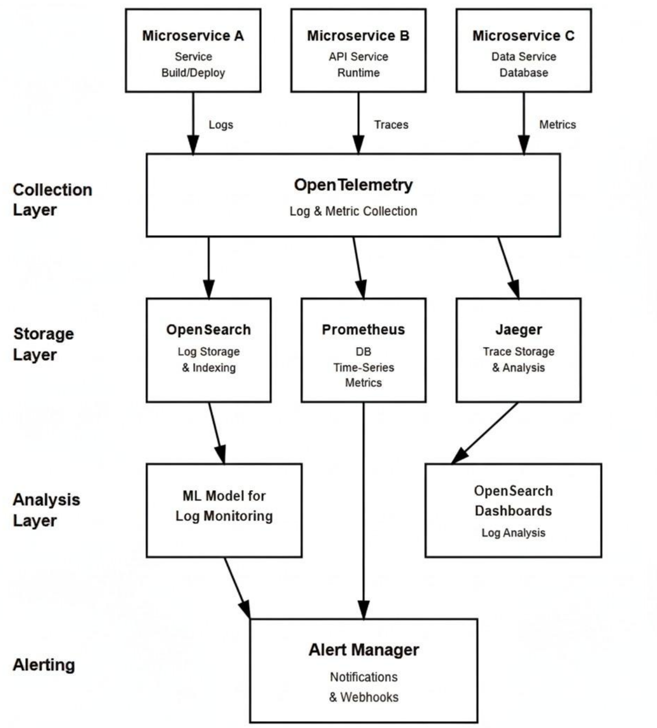

Fig. 1. Architecture: Layered unified observability for microservices.

A. Data Generation and Collection Layer

This layer is responsible for instrumenting microservices and reliably collecting raw telemetry with minimal performance overhead.

- Instrumentation (OpenTelemetry): Microservices are instrumented using OpenTelemetry (OTel) SDKs, which provide a unified set of APIs for generating all three telemetry signals (logs, metrics, and traces) [21]. OTel is critical for injecting correlation IDs (Trace IDs) into log and metric data, enabling the later correlation across the entire stack.
- Metrics (Prometheus): Microservice endpoints expose metrics in a standard format, which the Prometheus server actively scrapes at regular intervals. Prometheus is chosen for its native efficiency with time-series data and powerful PromQL query language [15].
- Logs (Fluent Bit): The highly lightweight and resource-efficient Fluent Bit log forwarder collects logs from service containers/hosts and reliably transmits them to the storage layer [17].
- Traces (OpenTelemetry Collector): OTel agents capture the distributed trace data and export it via the OTLP protocol to the OTel Collector, which then forwards the traces to Jaeger [3].

B. Storage Layer

Specialized backends are utilized to handle the unique indexing and querying requirements of each data type.

- Logs (OpenSearch): Logs are indexed in OpenSearch, a scalable, open-source search and analytics suite, enabling full-text search, aggregation, and rapid querying of high-volume log data [17]. Each log record is stored in a structured JSON format to allow efficient parsing and correlation across services. A typical log entry may look like:

```
{
  "timestamp": "2025-10-31T13:42:15Z",
  "service": "payment-service",
  "level": "ERROR",
  "request_id":
    "d8f9a2b1-239f-4df6-a83c-23fd09bde77f",
  "message": "Transaction timeout while
    connecting to user-service",
  "trace_id": "a1b2c3d4e5f6"
}
```

These logs enable the system to correlate events across distributed services using identifiers such as `trace_id` or `request_id`. OpenSearch indexes each field for efficient querying, making it possible to execute advanced searches (e.g., find all ERROR logs for a specific service within a given time range). For anomaly detection, logs are analyzed for sudden surges in specific error messages or rare event patterns. Machine learning-based log template analysis or sequence modeling can highlight anomalies such as recurring timeout errors or missing log sequences.

- Metrics (Prometheus DB): The built-in TSDB provides long-term retention and fast querying capabilities specifically optimized for metric data, supporting high-cardinality label sets [15]. Each metric is stored as a time-series of timestamped numerical values with identifying labels. A typical metric might look like:

```
http_request_duration_seconds{
  service="checkout",
  instance="node-3",
  method="POST",
  status="200"
```

```
} 0.354 @2025-10-31T13:42:15Z
```

Metrics capture both system-level (CPU, memory, I/O) and application-level (latency, error rate, throughput) indicators. For anomaly detection, PromQL queries combined with algorithms such as z-score analysis, moving averages, ARIMA, or LSTM-based forecasting can identify deviations in expected trends. For instance, a sudden rise in `http_request_error_rate` beyond a defined threshold may signal abnormal service behavior.

- Traces (Jaeger): Jaeger stores and indexes distributed traces, using OpenSearch as a backend for high-volume and efficient searching of spans based on service name, duration, and tags [3]. Each trace consists of multiple spans that represent operations across services. A span typically looks like:


```
{
  "trace_id": "a1b2c3d4e5f6",
  "span_id": "s1234567",
  "parent_span_id": "root",
  "service": "order-service",
  "operation": "validateOrder",
  "start_time": "2025-10-31T13:42:15Z",
  "duration_ms": 132,
  "tags": {
    "http.method": "POST",
    "http.status_code": 500,
    "error": true
  }
}
```


Traces provide an end-to-end view of request execution paths. For anomaly detection, latency outliers, missing spans, or abnormal dependency patterns can reveal issues such as bottlenecks or cascading failures. Graph-based analysis or temporal comparisons across spans can detect deviations from typical call hierarchies or timing distributions.

By correlating logs, metrics, and traces, a multi-modal anomaly detection pipeline can be established. For example, simultaneous spikes in `error-rate` metrics, unusual `ERROR` logs, and increased trace latency across dependent services strongly indicate systemic failures or configuration regressions.

IV. ANALYSIS, CORRELATION, AND ANOMALY DETECTION

A. Unified Correlation and Visualization

The Analysis Layer ensures that all telemetry is consumable through a single pane of glass, facilitating seamless navigation from high-level symptoms to low-level causes.

- Grafana (Metrics + Logs + Traces): Grafana acts as the unified dashboard layer, connecting directly to Prometheus (metrics) and OpenSearch (logs). Crucially, Grafana can integrate with Jaeger (traces) to display distributed trace links directly alongside metric graphs, allowing engineers to "drill down" from a latency spike on a metric chart into the specific traces that caused it [18].
- OpenSearch Dashboards (Log Analysis): Provides specialized tools for querying, visualizing, and performing statistical analysis (e.g., log clustering) directly on the raw log data [17].

B. Proactive ML Anomaly Detection

The ML Anomaly Detection module is a separate service that ingests data from the Storage Layer (primarily OpenSearch and Prometheus) and applies intelligence to spot subtle deviations that evade static thresholds.

1) ML Methodology: The problem of detecting novel failures in a dynamic system is best addressed by Unsupervised Learning techniques, which require no pre-labeled incident data [4].

- Time-Series Anomaly (Metrics): For critical metrics (e.g., P99 Latency, Error Rate), models like LSTM Autoencoders are trained on historical data to learn the normal sequence and temporal correlations. Deviations from the expected reconstructed values flag anomalies [2].
- Pattern Anomaly (Logs): Log data is processed using Natural Language Processing (NLP) techniques, such as clustering (e.g., Isolation Forest applied to log vectors) to identify a sudden shift in the frequency or emergence of rare log messages, indicating a new failure mode [16].
- Trace/Dependency Anomaly: The module analyzes the statistical properties of traces (e.g., standard deviation of span duration, abnormal service call patterns) to detect slow memory leaks or connection pool saturation before they become a critical failure [1].

2) Alerting and Remediation: Once an anomaly is detected, the module generates a rich, context-aware alert that is routed to the Alert Manager. This centralized component handles deduplication, grouping, and escalation policy (e.g., paging the SRE team). The alert contains not only the symptom (e.g., "Latent Anomaly Detected in Service A") but also correlated metadata (e.g., the specific `trace_id` of the first anomalous request) to accelerate RCA [5].

V. DISCUSSION AND IMPLEMENTATION DETAILS

A. Implementation Stack Summary

The entire framework is constructed using highly mature open-source projects under the Cloud Native Computing Foundation (CNCF) umbrella, ensuring high performance and community support. Table III details the components.

B. Challenges and Limitations

While powerful, implementing intelligent observability presents challenges:

- Data Volume and Cardinality: The sheer volume of telemetry data, particularly logs and high-cardinality traces, demands aggressive sampling strategies and scalable storage backends to manage costs and performance [11].
- Model Drift: ML models must be continually retrained to adapt to the dynamic nature of microservices (e.g., new

TABLE I
SUMMARY OF RELATED WORK ON OBSERVABILITY AND ANOMALY DETECTION IN MICROSERVICES

Sr.	Title	Author(s)	Year	Methodology	Features / Key Findings
1	From Point-wise to Group-wise: A Fast and Accurate Microservice Trace Anomaly Detection Approach	Zhe Xie, Changhua Pei, Wanxue Li, Huai Jiang, Liangfei Su, Jianhui Li, Gaogang Xie, Dan Pei	2023	Group-wise trace anomaly detection for microservices, leveraging deep learning-based log analysis and graph methods for improved accuracy and speed	Improves detection accuracy/efficiency, adapts to microservice scalability, enhances reliability insights
2	Root-Cause Metric Location for Microservice Systems via Log Anomaly Detection	Lingzhi Wang, Nengwen Zhao, Junjie Chen, Pinnong Li, Wenchi Zhang, Kaixin Sui	2020	Combines DeepLog-based log anomaly detection and mutual information-based robust correlation analysis; includes data augmentation	Achieves top-15 accuracy for root-cause identification, outperforms prior methods in accuracy and speed, validated on TrainTicket benchmark
3	Graph-Based Trace Analysis for Microservice Architecture Understanding and Problem Diagnosis	Xiaofeng Guo, Xin Peng, Hanzhang Wang, Wanxue Li, Huai Jiang, Dan Ding, Tao Xie, Liangfei Su	2020	GMTA system uses graph-based representation for efficient trace analysis, storage, access, and anomaly detection in microservices	Enables real-time architecture insight and root-cause diagnosis; supports visualization and analytics of dependencies and error propagations
4	Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks (TraceAnomaly)	Ping Liu, Haowen Xu, Qianyu Ouyang, Rui Jiao, Zhekang Chen, Xiaoying Bai, Shenglin Zhang, Jiahai Yang, Linlin Mo, Jice Zeng, Wenman Xue, Dan Pei	2020	TraceAnomaly uses deep Bayesian networks and handcrafted service-level trace vectors for unsupervised anomaly detection/localization	Precision/recall above 0.97, outperforms rule-based/ML baselines, deployed successfully for industrial trace anomaly localization
5	Root Cause Analysis of Failures in Microservices through Causal Discovery (RCD)	Azam Ikram, Sarthak Chakraborty, Subrata Mitra, Shiv Kumar Saini, Saurabh Bagchi, Murat Kocaoglu	2022	RCD: scalable algorithm combining observational and interventional data for efficient causal graph-based failure root cause detection	Higher top-k recall and speed over PC/AutoMAP/CIRCA; robust on synthetic, testbed, and real cloud failures
6	Hybrid Root Cause Analysis for Partially Observable Microservices Based on Architecture Profiling	Isidora Erakovic, Claus Pahl	2025	Architecture mining and anomaly detection with rule-based inference from trace logs, latency, call graphs, and clustering	Effective for partial observability; combines temporal/spatial anomaly tracing; validated on ISP trace logs for accurate fault localization
7	Metric-level Root Cause Analysis for Microservices via Multi-Modal Data (MRCA)	Y. Wang, Z. Zhu, Q. Fu, Y. Ma, P. He	2024	Integrates metrics/logs using extended spectral analysis for end-to-end latency root cause localization in microservices	High precision root cause localization, cross-modal ranking, practical and scalable for latency issues in cloud-native setups
8	Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review	Mehmet Sylemez, Bedir Tekinerdogan, Aya Koluksa Tarhan	2022	Systematic Literature Review (SLR) analyzing 85 primary studies from an initial pool of 3,842 papers	Identifies and organizes architectural challenges into 9 categories and 40 sub-categories, providing mapped solution directions.
9	Data Management in Microservices: State of the Practice, Challenges, and Research Directions	Rodrigo Laigner, Yongluan Zhou, Marcos Antonio Vaz Salles, Yijian Liu, Marcos Kalinowski	2021	Exploratory study using SLR, 9 open-source applications analysis, and a survey of 120+ practitioners	Finds current DB systems insufficient for practitioners' needs with distributed state/consistency, proposes a new feature set for microservice-oriented DBs.
10	Containerization in Multi-Cloud Environment: Roles, Strategies, Challenges, and Solutions for Effective Implementation	Muhammad Waseem, Aakash Ahmad, Peng Liang, Muhammad Azeem Akbar, Arif Ali Khan, Iftikhar Ahmad, Manu Setl, Tommi Mikkonen	2025	Systematic Mapping Study of 121 papers (2013-2024)	Identifies Multi-Cloud Container Monitoring and Adaptation as leading theme; catalogs challenges, solutions, patterns, and strategies.
11	Observability, Profiling, and Debugging in Systems Conference 2015-2025	Not specified	2025	Survey of top-tier systems conference papers (OSDI, SOSP, EuroSys) over 10 years	Traces evolution from ad-hoc observability to always-on frameworks. Identifies overhead-insight trade-off and data deluge as open challenges.
12	Building an Observable System Based on OpenTelemetry	Xi-Zhen Wang, Chien-Chao Tseng	2024	Implements observability system with OpenTelemetry using hook-based, minimally intrusive instrumentation for PHP	Demonstrates OpenTelemetry extends traditional monitoring, capturing contextual and event-based paths.

deployments, feature releases), preventing models from becoming obsolete and generating false positives [16].

- The Cost of Monitoring (Instrumentation Overhead): The tools we use to track the system (OpenTelemetry) can actually slow the system down if not configured carefully. This instrumentation overhead adds non-trivial latency and consumes CPU in performance-sensitive services, forcing us to maintain tight control over how much data we capture (via sampling rates) [21].

VI. CONCLUSION

Recent studies have highlighted the growing importance of unified observability and anomaly detection in complex microservice systems. Xie et al. [1] and Liu et al. [4] demonstrate that traditional point-wise methods often fail to capture inter-service dependencies, advocating for group-wise and service-level analysis to improve accuracy and scalability. Similarly, Wang et al. [2] and Ikram et al. [5] emphasize the effectiveness of multi-modal telemetry and causal discovery in improving root cause localization and reducing false positives. These advancements collectively underline that a cohesive,

TABLE II
SUMMARY OF RELATED WORK ON OBSERVABILITY AND ANOMALY DETECTION IN MICROSERVICES (CONTINUED)

Sr.	Title	Author(s)	Year	Methodology	Features / Key Findings
13	Microservices Architectures Using Spring Boot: Embracing Containerization and Observability	Venugopal Koneni	2024	Review of best practices and real-world cases combining Spring Boot with containerization and observability tools	Concludes that this combination offers a robust, scalable, maintainable foundation for modern applications.
14	Microservices Monitoring with Event Logs	Cinque et al.	2019	Black box tracing with logs for monitoring without source code	Improves visibility, aids troubleshooting in dynamic environments
15	Metrics, Logs, and Traces: Unified Observability	Bhosale	2022	Integrates Prometheus, ElasticSearch, Jaeger; uses tagging, dashboards	Faster RCA, reduced debugging time; promotes OpenTelemetry standardization
16	AI for Microservice Monitoring	Podduturi	2025	AI/ML with deep learning, clustering, reinforcement learning	Enhances real-time monitoring, reduces downtime; addresses data quality
17	Integrated Approach for Logging and Monitoring	Devarakonda	2020	Distributed tracing, ELK/Loki, Prometheus/Grafana, ML anomaly detection	High ingestion, low latency; optimizes resources, response times [17]
18	Observability in Large-Scale Microservices	Madupati	2023	Uses Prometheus, Grafana, Jaeger for health monitoring	Addresses data overload, instrumentation; ensures stability, scalability
19	Microservices Monitoring with Event Logs and Black Box Execution Tracing	Marcello Cinque, Raffaele Della Corte, Antonio Pecchia	2022	Passive network tracing (sniffing) of RESTful HTTP messages to build execution traces without application instrumentation.	Proposes MetroFunnel; reduces log data size by ˜59-78% with low performance overhead; aids in troubleshooting by correlating service calls.
20	LO2: Microservice API Anomaly Dataset of Logs and Metrics	Alexander Bakhtin, Jesse Nyysso¨la¨, Yuqing Wang, Noman Ahmad, Ke Ping, Matteo Esposito, Mika Ma¨ntyla¨, Davide Taibi	2025	Generated a large-scale dataset by testing a production OAuth2 system (Light-OAuth2) with correct and erroneous API calls using Locust.	Provides ˜657k log files (2B+ lines) and 45M+ metric files from a production OSS system; highlights challenges in tracing and enables multimodal anomaly detection research.

TABLE III
OPEN-SOURCE STACK DETAILS

Pillar	Tool	Function
Logs	Fluent Bit	Lightweight log forwarder with efficient data collection and real-time processing capabilities
	OpenSearch	Scalable log storage and indexing with advanced search, analytics features, and robust performance
Metrics	Prometheus	Time-series scraper and database with powerful querying and alerting functionalities
	Grafana	Visualization and dashboarding with customizable panels and real-time data integration
Traces	OpenTelemetry	Unified instrumentation standard for distributed tracing and monitoring across services
	Jaeger	Distributed trace storage and analysis with detailed performance insights and latency tracking
Intelligent Analysis	ML Service	Anomaly detection (Python/PyTorch) with machine learning models for predictive analytics and root cause analysis

cross-layer observability approach is essential for achieving reliability and resilience in distributed microservices. Building on these insights, the proposed framework integrates OpenTelemetry, Prometheus, OpenSearch, Jaeger, and Grafana into a unified observability stack that correlates metrics, logs, and traces in real time. By incorporating machine learning-based anomaly detection across telemetry signals, the system can proactively identify abnormal behaviors and performance bottlenecks before service degradation occurs. This integration bridges the gap between raw observability data and actionable insight, advancing toward self-healing and autonomously managed microservice environments that enhance performance visibility, fault detection, and operational intelligence.

VII. FUTURE SCOPE

The next phase of this research will focus on dramatically advancing the intelligence layer to move beyond detection and into full autonomy and actionable insights.

- Explaining the ”Why” (Explainable AI - XAI): We plan to integrate technology that doesn’t just flag a problem, but tells the engineer exactly why it happened in plain English. For example: ”The performance drop started in Service C (Trace ID XXX) because we saw a sudden surge of database connection errors in the logs (Log ID YYY).”
- Self-Driving Troubleshooting (Automated RCA): We will build causal inference algorithms that automatically trace a failure back to its single, originating service (the first domino to fall), even in complex cascading failures,

eliminating manual investigation time [5].

- Predictive Self-Healing (Adaptive Scaling): By connecting our anomaly predictions directly to Kubernetes autoscaling, we can enable services to automatically scale up in anticipation of predicted load spikes, proactively preventing failures before they even start [10].

ACKNOWLEDGMENT

We owe a huge thank you to the dedicated global communities behind OpenTelemetry, Prometheus, Jaeger, OpenSearch, and Grafana. Their continuous, massive efforts provide the powerful, open-source building blocks that made our intelligent monitoring framework possible.

REFERENCES

[1] Z. Xie et al., "From Point-wise to Group-wise: A Fast and Accurate Microservice Trace Anomaly Detection Approach," in *Proc. ACM SIGSOFT Int. Symp. Softw. Testing Anal. (ISSTA)*, 2023, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3611643.3613861

[2] L. Wang et al., "Root-Cause Metric Location for Microservice Systems via Log Anomaly Detection," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2020, pp. 1–8. [Online]. Available: https://tjusail.github.io/people/chenjunjie/files/ICWS20.pdf

[3] X. Guo et al., "Graph-Based Trace Analysis for Microservice Architecture Understanding and Problem Diagnosis," in *Proc. ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, 2020, pp. 1–12. [Online]. Available: https://taoxie.cs.illinois.edu/publications/esecfse20in-trace.pdf

[4] P. Liu et al., "Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks (TraceAnomaly)," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. (ISSRE)*, 2020, pp. 1–12. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9251058

[5] A. Ikram et al., "Root Cause Analysis of Failures in Microservices through Causal Discovery (RCD)," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 1–14. [Online]. Available: https://surl.lt/wboope

[6] I. Erakovic and C. Pahl, "Hybrid Root Cause Analysis for Partially Observable Microservices Based on Architecture Profiling," in *Proc. Int. Conf. Softw. Archit. Companions (ICSA-C)*, 2025, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/3531056.3542765

[7] Y. Wang et al., "Metric-level Root Cause Analysis for Microservices via Multi-Modal Data (MRCA)," *IEEE Trans. Softw. Eng.*, vol. 50, no. 4, pp. 1–15, Apr. 2024. [Online]. Available: https://www.scitepress.org/Papers/2025/134536/134536.pdf

[8] M. Sylemez et al., "Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review," *Empir. Softw. Eng.*, vol. 27, no. 5, pp. 1–45, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9778893

[9] R. Laigner et al., "Data Management in Microservices: State of the Practice, Challenges, and Research Directions," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, 2021, pp. 1–12. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8826375

[10] M. Waseem et al., "Containerization in Multi-Cloud Environment: Roles, Strategies, Challenges, and Solutions for Effective Implementation," *IEEE Access*, vol. 13, pp. 1–25, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10967524

[11] "Observability, Profiling, and Debugging in Systems Conference 2015–2025," *SSRN Electronic Journal*, 2025. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5234701

[12] X.-Z. Wang and C.-C. Tseng, "Building an Observable System Based on OpenTelemetry," in *Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. (ICSESS)*, 2024, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10967524

[13] V. Koneni, "Microservices Architectures Using Spring Boot: Embracing Containerization and Observability," *IJETCSIT*, vol. 1, no. 1, pp. 1–10, 2024. [Online]. Available: https://www.ijetcsit.org/index.php/ijetcsit/article/view/199

[14] M. Cinque et al., "Microservices Monitoring with Event Logs," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9293310

[15] P. Bhosale, "Metrics, Logs, and Traces: A Unified Approach to Observability in Microservices," *Journal of Artificial Intelligence, Machine Learning, and Data Science*, 2022. [Online]. Available: https://doi.org/10.51219/JAIMLD/Pradeep-bhosale/458

[16] S. Podduturi, "AI for Microservice Monitoring and Anomaly Detection," *SSRN Electronic Journal*, 2025. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract$_id = 5076624$

[17] R. R. Devarakonda, "An Integrated Approach for Logging and Monitoring in a Containerized Microservices Architecture," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, 2020, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9778893

[18] B. Madupati, "Observability in Microservices Architectures: Leveraging Logging, Metrics, and Distributed Tracing in Large-Scale Systems," *SSRN Electronic Journal*, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5076624

[19] M. Cinque et al., "Microservices Monitoring with Event Logs and Black Box Execution Tracing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1–14, 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3456859.3456863

[20] A. Bakhtin et al., "LO2: Microservice API Anomaly Dataset of Logs and Metrics," in *Proc. Int. Conf. Softw. Eng. (ICSE) Datasets*, 2025, pp. 1–10. [Online]. Available: https://www.espjeta.org/jeta-v1i1p122

[21] X.-Z. Wang and C.-C. Tseng, "Building an Observable System Based on OpenTelemetry," in *Proc. 2024 Int. Computer S... Conf.*, Publisher: IEEE, 2024, pp. (Page range, if known). [Online]. Available: https://ieeexplore.ieee.org/document/10959884

[22] M. Sylemez, B. Tekinerdogan, and A. Kolukisa Tarhan, "Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review," *Applied Sciences*, vol. 12, no. 11, p. 5507, May 2022. [Online]. Available: https://shorturl.at/hrgP0

[23] A. Bakhtin et al., "LO2: Microservice API Anomaly Dataset of Logs and Metrics," *arXiv preprint arXiv:2504.12067*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2504.12067

[24] O. Meyer, E. Johnson, and J. Brown, "A Unified Framework for Anomaly Detection and Root Cause Analysis in Microservice Systems," *Computer-Life Journal* (or Conference Name), vol. X, no. Y, pp. Z–W, 2024. [Online]. Available: https://doi.org/10.54097/1gw77589

[25] J. Nobre, E. J. S. Pires, and A. Reis, "Anomaly Detection in Microservice-Based Systems," *Applied Sciences*, vol. 13, no. 13, p. 7891, Jul. 2023. [Online]. Available: https://doi.org/10.3390/app13137891

[26] S. Zhang et al., "Robust Failure Diagnosis of Microservice System through Multimodal Data," Technical Report (or Preprint), Dec. 2023. [Online]. Available: https://nkcs.iops.ai/wp-content/uploads/2023/12/TSC23-DiagFusion.pdf