International Journal of Multidisciplinary Research and Growth Evaluation
ISSN: 2582-7138
Received: 25-02-2021; Accepted: 24-03-2021
www.allmultidisciplinaryjournal.com
Volume 2; Issue 2; March-April 2021; Page No. 298-301

Introduction to Observability and its Three Pillars

Lakshmi Narasimha Rohith Samudrala
Independent Researcher, USA

Corresponding Author: **Lakshmi Narasimha Rohith Samudrala**

DOI: https://doi.org/10.54660/.IJMRGE.2021.2.2.298-301

Abstract
As modern IT ecosystems evolve and become more complex, observability has emerged as a critical capability for ensuring that the IT systems are functioning and performing as expected. Unlike traditional monitoring, which relies on a small set of predefined metrics and custom threshold-based alerts, observability provides a deeper, data-driven understanding of system.

This paper introduces the three pillars of observability Logs, Metrics, and Traces which together provide full visibility into system health. Logs provide a detailed record of events, allowing teams to analyze series of events and debug. Metrics provide quantifiable data; this helps teams track real-time system behavior. Traces map the flow of requests across distributed services, offering invaluable insights into latency, dependencies, and bottlenecks. Each pillar has its own importance, but true observability is only achieved when these data sources are correlated and analyzed together.

This paper also sheds light on the challenges of implementing observability such as data overload, high storage costs, integration complexity, and the need for AI-driven analytics to extract meaningful insights. This paper discusses the best practices for overcoming these challenges.

Keywords: Observability, Logs, Metrics, Distributed Tracing, Open Telemetry, AI-driven Observability, Anomaly Detection, Predictive Analytics, Automated Root-Cause Detection

1. Introduction

The IT environments are growing more complex by the day. Traditional monitoring solutions worked perfectly for simpler architectures, but with cloud-native application, container, hybrid environments the principles on which traditional monitoring work will not be enough. To maintain system reliability, performance, and security a more robust monitoring strategy is required. This leads to the emergence of Observability. Observability offers a more data-driven, proactive approach to system health and performance monitoring.

In the context of IT operations, observability enables engineers to understand, troubleshoot, and optimize applications by collecting and analyzing telemetry data. Unlike traditional monitoring, observability does not require teams to predefine every possible failure mode. Instead, it empowers engineers to explore unknown failures by providing deep insights into system behavior.

A. The need for observability

Organizations are rapidly transitioning from monolithic applications to cloud-native architectures. In doing so, they face new challenges in monitoring and troubleshooting. Microservices, containers, cloud-native, and hybrid environments introduce complexities, making it difficult to track system behavior using conventional monitoring methods.

Conventional/ Traditional monitoring depends on the IT resource's knowledge on failure scenarios, it cannot adapt to unpredictable failure scenarios. Traditional monitoring leads to fragmented visibility across infrastructure, applications, and network layers creating data pools that are not correlated. Such data pools are only useful for reactive monitoring, this results in delayed incident resolution and increased Mean Time to Resolution (MTTR).

Observability addresses these concerns by providing a comprehensive, contextual view of system health. This allows for early detection and faster root-cause analysis by using the correlated telemetry data, improved incident response through real-time alerts, AI-driven root cause analysis, optimized performance, and user experience with actionable insights.

B. Difference between observability and monitoring

The difference between observability and monitoring is often misunderstood. Both observability and monitoring serve distinct

purposes [6]. Monitoring will alert IT teams on a trigger (example: Memory saturation), but it won't explain why. Observability, on the other hand, allows engineers to deep dive into the source of the problem. Below table showcases some key differences between observability and monitoring [6].

Table 1: Difference between observability and monitoring

Monitoring	Observability
Predefined collection of metrics and alerts	Ability to explore and diagnose unknown issues
Reactive (alerts when a known issue occurs)	Proactive (helps discover unknown issues)
Focuses on Uptime, availability, predefined KPIs	Focuses on System behavior, dependencies, and RCA
Cannot troubleshoot unknown failure patterns	Provides full-stack visibility and correlation

2. Three pillars of observbility

Observability is built on three fundamental pillars: logs, metrics, and traces [5]. Each pillar provides a different perspective on system, and when used together, they provide deep visibility into application performance, reliability, and health. While traditional monitoring focuses on predefined failures and alerts, observability enables IT teams to explore and diagnose unknown issues, making it essential for modern, complex distributed systems [5].

A. Logs

Logs tell the story of the system's health and operation [1]. It is a record of events that happen within a system, providing detailed information about what happened and at what time. Logs capture errors, warnings, and other important activities in applications, servers, or network devices [1]. Developers and operation teams use logs to troubleshoot issues, understand user behavior, and detect security threats. However, managing logs can be challenging as they generate massive amounts of data, requiring efficient storage and analysis tools like Elasticsearch, Splunk, or Dynatrace.

Fig 1: shows An Example Log File [1]

B. Metrics

Metrics are numerical measurements that provide insights into a system's health and performance. They help track resource usage, application performance, and several other key KPI over time (timeseries metrics) [2]. Unlike logs, which store detailed event data, metrics are lightweight and aggregated, making them ideal for real-time monitoring and alerting [3]. Tools like Prometheus, Grafana, and Dynatrace help organizations visualize and analyze metrics to detect trends, prevent outages, and optimize system performance. Below image showcases an example of timeseries metric.

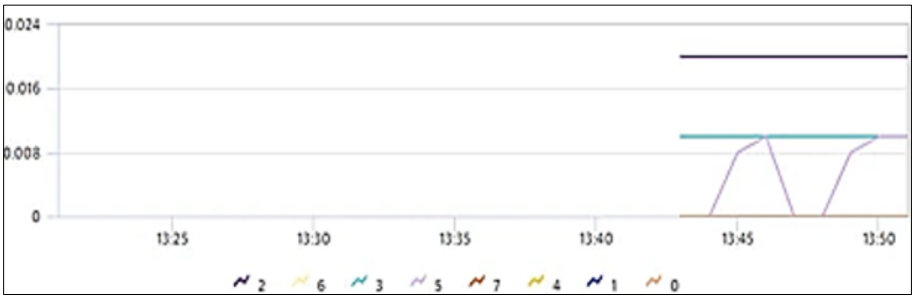

Fig 2: Showcases an Example of Timeseries Metric [3]

C. Traces

Traces track how a request travels through different services in an IT landscape, especially in large complex, distributed architectures like microservices [4]. Trace consists of multiple spans, each span represents a step in the request's journey, such as request to middleware, database queries or API calls [4]. By analyzing traces, teams can quickly find the faulty domain and identify the root-cause of slowdowns, bottlenecks, and failures [4]. Tools like Jaeger, Zipkin, and Dynatrace help capture and visualize traces, making them easier to implement and use. Figure 3 showcases an example of trace.

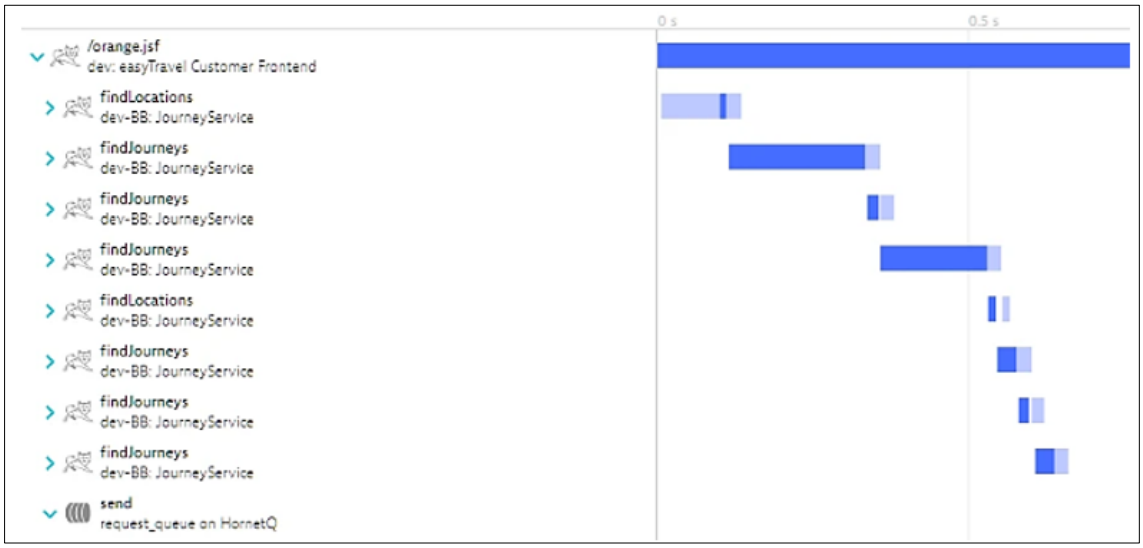

Fig 3: Showcases an Example of Trace (Dynatrace PurePath) [4]

3. Evolution of the three pillars of observability

Observability has evolved to what we know today over the past two decades, driven by advancements in software architecture, cloud computing, and automation. Traditional monitoring relies on static metrics and predefined alerts, modern observability focuses on real-time telemetry, proactive insights, and AI-driven analytics.

The three pillars of observability, which are Logs, Metrics, and Traces were developed independently over time but have proven to be very valuable when converged together into a unified observability framework.

A. Traditional IT monitoring

Traditional monitoring is built primarily with logs and metrics. Traditional monitoring lacked trace data; with simple monolithic applications this wasn't an issue.

In traditional monitoring logs were not structured. IT teams had to rely on manual log files stored in text format [1]. The analysis of logs was primitive, where IT teams would need to manually scan through large log files. In traditional monitoring, Syslog and early event logging systems such as (Windows Event Viewer, Unix logs) provided basic logging capabilities.

In this stage of monitoring the metrics were collected through SNMP (Simple Network Management Protocol) and custom scripts [3]. The focus was primarily on infrastructure metrics rather than application performance. Alerting for these metrics were static threshold based. This led to false positives and alert fatigue.

The primary drawbacks of traditional monitoring were limited visibility beyond infrastructure, lack of real-time analytics, and lack of correlation between logs, metrics, and traces.

B. Application Performance Monitoring (APM)

With the rise of web applications, microservices, and cloud computing, traditional monitoring tools failed to keep up with dynamic, distributed systems. This led to the shift towards Application Performance Monitoring (APM).

In APM, log aggregation tools emerged, enabling centralized log storage and fast querying. APM also introduced AI-powered log analytics, this improved pattern detection and anomaly identification.

The usage of Application Performance Monitoring (APM) tools like New Relic, AppDynamics, and Dynatrace provided the ability to capture real-time application metrics [3]. In addition to this, time-series databases such as Prometheus and InfluxDB revolutionized high-frequency metric collection. The application-based metrics gathered by APM introduced the usage of Service Level Objectives (SLOs) and Service Level Indicators (SLIs) as a business-driven approach to monitoring.

With the rise of microservices architectures tracking requests across multiple services became important and this became impossible to do so with traditional monitoring. This led to the introduction of distributed tracing (first introduced by Google in 2010), which enabled follow requests through complex systems.

C. Observability

Organizations understood the power of the three pillars of observability, more so as a unified full-stack visibility framework. The biggest driver for the emergence of observability as a holistic discipline was the rise of observability platforms such as Dynatrace, New Relic, Datadog, and OpenTelemetry. These platforms integrated log management, real-time metrics collection, and distributed tracing into a single interface. This allowed IT teams to diagnose performance issues, optimize system reliability, and improve incident response times [4].

Another major advancement with observability was the integration of artificial intelligence and automation. AI-driven anomaly detection and root cause analysis (RCA) enabled teams to quickly identify patterns and predict potential failures. Modern observability platforms leverage machine learning algorithms to analyze vast amounts of telemetry data in real-time, reducing mean time to resolution (MTTR) and alert fatigue.

4.Challenges and best practices in implementing observability

Observability provides deep and valuable insights into a system's health, however, implementing observability comes with myriad of challenges. Organizations need to manage high volumes of data, ensure visibility across distributed environments, and optimize performance without overwhelming engineers with excessive information.

One of the primary challenges in implementing observability

is handling the enormous volume of logs, metrics, and traces generated by modern applications. This mountain of data is difficult to understand and use without proper data processing. IT teams can find it extremely difficult to find useful and actionable insights amidst the noise. To overcome this, organizations can leverage AI-driven observability platforms. These platforms like Dynatrace, Datadog, etc. can help filter out noise and detect patterns that indicate potential issues.

Another issue that organizations need to address is the lack of standardization, inconsistent logging formats, varying metric definitions, and lack of trace correlation can make observability data hard to analyze. To overcome this challenge, organizations can use unified observability platform reducing complexity by aggregating data from various sources into a single dashboard.

Also, logs, metrics, and traces may contain sensitive data, such as personally identifiable information or API keys. Organizations need to ensure that observability solutions comply with GDPR, HIPAA, NERC CIP, and other regulatory frameworks while maintaining system visibility. Encrypting logs, redacting sensitive data, and implementing access controls help maintain security while ensuring compliance with industry regulations.

5. Conclusion

Observability is turning into a key practice for maintaining the reliability and performance of modern IT landscapes. By leveraging the three pillars of observability logs, metrics, and traces, organizations can gain deep insights into a system's health, identify issues proactively, and reduce the time needed to resolve issues.

Having said that, implementing observability comes with many challenges such as data overload, standardization issues, and security concerns. Adopting best practices like centralized monitoring, AI-driven insights, and optimized data retention helps organizations overcome these hurdles effectively.

6. References

1. Gavin B. What is a log file (and how do I open one)? How-To Geek. 2018 Jul 27 [cited 2025 Mar 27]. Available from: https://www.howtogeek.com/359463/what-is-a-log-file/
2. 17 Important data visualization techniques | HBS Online. Business Insights Blog. 2019 Sep 17 [cited 2025 Mar 27]. Available from: https://online.hbs.edu/blog/post/data-visualization-techniques
3. Kopp M. Simplify observability for all your custom metrics (Part 3: Scripting languages). Dynatrace News. 2020 Dec 28 [cited 2025 Mar 27]. Available from: https://www.dynatrace.com/news/blog/simplify-observability-for-all-your-custom-metrics-part-3-scripting-languages/
4. Kopp M. PurePath visualization: Analyze each web request from end-to-end. Dynatrace News. 2017 Jun 21 [cited 2025 Mar 27]. Available from: https://www.dynatrace.com/news/blog/purepath-visualization-analyze-web-request-end-end/
5. Williams A, Gain BC. The 3 pillars of observability. The New Stack. 2020 Apr 2 [cited 2025 Mar 27]. Available from: https://thenewstack.io/the-3-pillars-of-observability/
6. Egilmez I. Monitoring vs. observability: What's the difference? The New Stack. 2020 Nov 18 [cited 2025 Mar 27]. Available from: https://thenewstack.io/monitoring-vs-observability-whats-the-difference/.