Golden Sun-Rise
International Journal of Multidisciplinary on Science and Management
ISSN: 3048-5037/ Volume 1 Issue 3 Jul-Sep 2024 / Page No: 45-47
Paper Id: IJMSM-V1I3P104 / Doi:10.71141/30485037/V1I3P104

Review Article

Foundations of Observability Engineering

Shekhar Jha[1]

[1]*Chief Architect, Global Architect Leader, AVP, Integration, Data & Cloud COE, USA.*

Received: 14 July 2024 Revised: 02 August 2024 Accepted: 08 August 2024 Published: 30 August 2024

Abstract *- Our global and dynamic life is reliant on behavior in complex systems. Observability engineering provides the means and tools to understand how these systems work, are not working and are healthy. It describes basics of metrics, logs and traces, and how to collect, work with and visualize data streams. This includes real-world design issues for observable systems like selecting appropriate metrics, logging and tracing apps, and good alerting. And segmented by metrics terms like Mean Time to Detect (MTTD), Mean Time to Resolve (MTTR), Change Failure Rate (CFR) etc.*

Keywords *- Observability, Metrics, Logs, Traces, MTTD, MTTR, AI/ML-Powered Observability.*

I. INTRODUCTION

Observability engineering is one of the biggest topics in modern software engineering which allows teams to dive deep into their complex systems. It is all about getting knowledge other than just a dashboard to solve issues on an ongoing basis.

Observationality is an engineering gem, because when we can see how it functions on the other end, we know very precisely what's happening in the system. This is very useful in the present high complexity applications, when the short time to crash can mean the difference between a long crash and crashing. Better, easier to notice, easier to experience that's how engineering teams save lives, maximize margin, and offer better user experience.

II. METHODOLOGY

There are 3 primary data structures that Observability Engineering uses metrics, logs, and trace. Such rules are meant to give an overall picture of how the system works and behaves. Metrics provide data, logs carefully track system health, and traces specify the journey a request takes. All of this allows us to take system dynamics a step further.

- **Data**: Observable features of the behavior of the system in time. CPU time, Memory, Request Latency, Error count, etc.
- **Logs**: Verbal record of system activity. Logs show you minutely what's going on inside the system, its faults and its errors.
- **Traces**: Records about the request as it goes through a distributed system, the order in which it will go and when.

III. PLANNING AND REVIEWS

Techniques to Develop an Observability Plan- Establish baselines and KPIs. Create a feedback cycle to keep on trying. Be sure that the observability platform can scale up with evolving demand. Install real time monitoring and alerting. In short,

- **Beginning with Observability Objectives**: Define observability objectives that are in alignment with the business objectives.
- **Choose the Tools that Fit**: Choose tools that you're comfortable with and integrate with your current system.
- **Create a Data-Driven Culture**: Inspire data-driven culture in your organization.
- **Conquer and Evolve**: You will want to change and revisit your observability workflows as your customers

change and demand.

How Does It Perform? You need to know the incident resolution rates, alert accuracy, and ability to detect problems before they affect users in order to know how well the observability tools are performing. The only issue with these tools is turning raw data into decision-making intelligence. A breakdown of metrics and considerations:

- **Coverage Observability**: How much coverage your observability tools cover all the critical components of your system. Wide coverage is good, you know everything your system does.
- **Business Impact**: How much business impact is there on observability such as revenue, customer engagement, and cost. Demonstrating the business use case for observability can fund tools and infrastructure.
- **Mean Time to Detect (MTTD)**: Mean time it takes to detect an incident or anomaly. A lower MTTD equals earlier problem detection and faster fixing.
- **Mean Time to Resolve (MTTR):** Average amount of time to fix an incident when it is discovered. The smaller the MTTR, the faster the response time, the less the failure of the service and more user satisfaction.
- **Change Failure Rate (CFR):** Outage number, Outage due to code update or infrastructure change. Lower CFR, which means that your observability capabilities are catching and reversing problems with change.
- **Alert Fatigue**: Engineers get tired from a lot of noise and overload they might miss alerts, leading to critical defects being missed and low engineer performance.
- **Loyalty**: Your observability tool suite is supporting your engineers. If users are happy, then your tools help with an awesome engineering experience.

If we observe and analyze these metrics carefully, we can accurately track the performance of our observability tools and continuously evolve observability programs.

IV. FUTURE OF OBSERVABILITY TECHNOLOGY

For predicting and anomaly detection, AI and Machine Learning are going to be the observability technologies of the future. The more sophisticated the infrastructure, the more complex the tools. And much more than that, it'll be all about integrating observability tools into the current DevOps architecture. Observability Technology will be the main take-overs of AI/ML Based Observability, Observability for Cloud-native Modern Architectures, Edge Computing, Serverless Observability, Data-based Observability, Open Standards, Interoperability, Observability Platforms as a Service (OpaaS) and more.

Details on important trends:

A. Observability for Modern Architectures

- **Cloud-Native Insights**: Observability tools will be continually upgraded to accommodate cloud-native challenges such as microservices, serverless functions, and Kubernetes.
- **Edge Computing**: Observability tools will have to be deployed on edge devices and IoT ecosystems to see how distributed systems behave and perform.
- **Serverless Observability**: Problems associated with serverless observability, including its temporary state and distributed execution will be overcome through special tools and methods.

B. AI/ML-Powered Observability

- **Predictive Analytics**: AI/ML algorithms will continue to be introduced in observability platforms to anticipate failures, identify anomalies, and act as early alarms for engineers.
- **Automatic Root Cause Recognition**: AI can automatically extract the root cause from large amounts of telemetry to automatically identify performance problems and incidents, helping to drastically speed up troubleshooting.
- **Intelligent Alerting**: AI can learn from the past and filter out noise to minimize alert fatigue, so engineers can stay on track with critical incidents.

C. Integration and Open Standards

- **Open Standards and Interoperability**: The more attention given to open standards and interoperability, the more easily the observability tools and platforms can be standardized.
- **Observability Platforms as a Service (OpaaS)**: Managed observability platforms will open the door for organizations to tap into higher level features and knowledge without investing in large in-house systems.

V. CONCLUSION

Observability is engineering's biggest idea, and it can force teams to own things in the best possible way. In the age of improving technology, repurposing observability methods will be important to ensure the machine is operating as it should. Engineering teams can stay on the cutting edge by thinking about their systems as reimagined with new tools.

Observation engineering is a core activity of today's software engineering teams to understand how their large-scale systems are working holistically. Through metrics, logs and trace as well as cutting-edge technologies like AI/ML and data-driven solutions, enterprises can identify and resolve issues faster, increase systems' efficiency and availability, and accelerate their digital transformation. It's always learning and evolving, if we're going to get the best out of observability, and make sure that it's worth our while in an ever-changing technological landscape.

VI. REFERENCES

1. Aditya Pawar, The Three Pillars of Observability: Metrics, Logs and Traces, eginnovations, 2024. Online: https://www.eginnovations.com/blog/the-three-pillars-of-observability-metrics-logs-and-traces/
2. Sonja, API Observability Dashboard with OpenTelemetry and Grafana, Tyk 5.7, 2023. Online: https://community.tyk.io/t/api-observability-dashboard-with-opentelemetry-and-grafana/6580
3. AI-Powered Observability, logz.io. Online: https://logz.io/platform/
4. What is MTTD? Why does it matter for ITOps?, BigPanda, 2024. Online: https://www.bigpanda.io/blog/what-is-mean-time-to-detect-mttd/
5. What is Observability Engineering?, Dynatrace. Online: https://www.dynatrace.com/knowledge-base/observability-engineering/