

Whitepaper

ADMS.NXT.NOW

Observability Best Practices for Cloud Native Architectures



Abstract

Observability is a popular subject specifically in the context of cloud native architectures driven by the principles of microservices, distributed architectures, and just-in-time environment provisioning. All these elements add to the need of having extremely robust and efficient observability solutions to enable high reliability of IT systems.

In this paper, we will go over some of the important observability best practices that are recommended while implementing the observability strategy and solution for modern cloud native architectures.

We will briefly touch upon Tech Mahindra's NewAgeDelivery (NAD) platform that incorporates these best practices for observability and enabling the IT engineering teams to implement the best-in-class strategy and tooling with ease.

Key Takeaways

- Observability best practices for distributed, microservices based cloud native architectures
- Important aspects of data collection, aggregation visualizations and traceability aspects for observability solutions
- Utilization of machine learning and artificial intelligence for observability
- Observability as code and observability-as-a-service
- Tech Mahindra's NAD platform to enable best-in-class observability strategy and solution

04

DevOps Adoption

09

Observability-as-a-Service

10

Phases of Engagement – Setting up
Observability-as-a-Service Platform



Introduction

Businesses are changing at a rapid speed today – agility to respond to the changing business demands and helping businesses remain ahead in the curve as compared to the competition, is the key to IT modernization. IT teams across industries are adopting cloud native engineering, powered by microservices, containers, digital integrations and so on, to deliver at the right quality with right velocity.

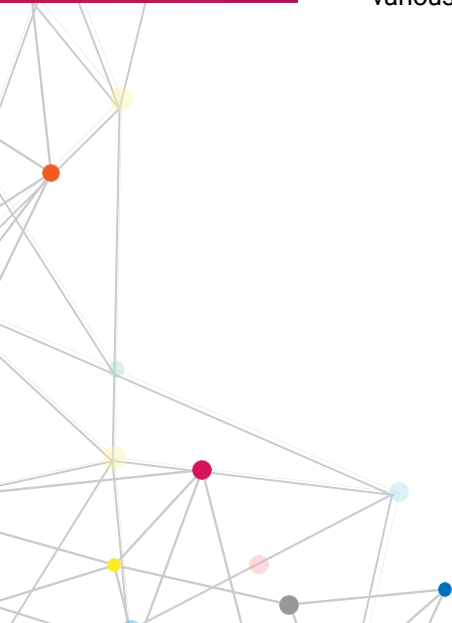
The modern cloud native IT systems provide unparalleled benefits by leveraging just-in-time infra provisioning and highly scalable architectures that can use decoupled microservices to enable changes at rapid speed, without creating any system instability. However, these architectures come at a cost of being distributed and complex too. When it comes to traditional monitoring for a monolith system, it is much simpler with contained infra, networks, and apps to be monitored. The challenge in cloud native and microservices architectures is speed, scale and amount of data that is collected by the monitoring systems, network, and services traces, and logs.

Before we deep dive on this subject of monitoring and observability – it is important to establish the similarities and difference between the two. There can be different ways of understanding this and each SME in this area will have their own way of articulating it as well. In my opinion, **monitoring is the act of collecting data through logs, traces and taking certain actions as needed for the aspects that are known, to prevent issues, while observability is to use that data for a deeper analysis to discover the unknown**, to analyze why something is happening in distributed IT system, what can be the impact and what should be the steps taken. Monitoring is the subset of the overall observability solution – **a system that is not monitored well is not observable!**

Observability is also defined as the ability of the engineering team to be able to infer or answer any IT system behavior and any question around the state of application services, infra, networks by looking at the data from logs, traces and any other form of monitoring and telemetry data.

Highly observable systems provide the insights for the DevOps and SRE teams to be able to take actions (preventive or reactive) quickly by analyzing all the data – some evident and obvious data and some buried data from complex microservices traces.

In this paper, we will discuss some basic principles of designing the highly observable IT systems. One disclaimer is that it is not necessarily a comprehensive list of all the best practices and principles – but a list to cover all the experiences that I have had through various engagements and discussions with various IT teams.





DevOps Adoption

For an observability framework and platform to support microservices driven multi-cloud and distributed systems, and be successful, it is important that the IT teams adopt DevOps practices.

Why it is important is that **true impact of observability comes with ability to trace the events / incidents through the lifecycle to the origination**. That is possible when there is true collaboration between all IT teams not just in terms of people but also tools / frameworks used. Tools across the Dev and Ops landscape need to talk to each other or at least allow data traceability for right level to root cause analysis in case of any event.

It is also important that there is quickest possible response to any event that the observability platform identifies. For this to happen teams need to collaborate, avoid finger pointing and truly work as one team utilizing integrated processes and tools, believe in the process of continuous monitoring and feedback loops. All these are basic principles of DevOps. Hence, **it is important to align with the DevOps culture for the best ROI and favorable outcomes from the investment on observability framework and platform**.

Metrics and Baseline

Before embarking on the journey to set-up best in class observability framework and platform few things that are to be kept in mind:

- **What metrics we are trying to improve?** For example: mean time to resolution, automation index and auto remediations, first time resolution, incident reduction, and so on.
- **Establish the current baseline for the metrics** that we are trying to improve
- Agree on the success / **failure criteria**
- Agree on the **measurement frequency** for the identified metrics post implementation of the observability platform

No Data is Useless

One key best practice is that “Data is King” – **no data across the SDLC pipeline is useless data unless we collect, analyze, and then decide to discard it**. The ability to capture data as the transactions and workflows move across various system layers is one of the first design principle for highly observable systems.

One important point to remember is that while observing the highly distributed microservices based cloud native systems – we will be hit with issues that are categorized as “unknown unknowns” – and the ability to find the trace of the issue origination in the complex maze of

services’ interdependencies, distributed infra and error propagation across cross service interactions is the key to good observability solution. It is important that no data is left untraced and untracked from the observability platform’s ability standpoint. At some point, we can decide to filter out certain data sets and mark as not required to be collected to ensure that we are not just capturing elements that are of no value.

So, **it is ok to collect all the data and then sample or filter – instead of starting with a sampled / filtered view** that does not have the details that we need.



Frequency of Data Collection

Let's again look at a monolith stable application in an abundantly provisioned infrastructure – maybe looking at log data at half-a-minute interval is good enough.

Now **consider a modern cloud native microservices based IT system where servers and containers are created or destroyed every minute due to auto scaling set-up or where 1000s of serverless functions run for less than 15 seconds; the granularity of data collection must be very different** – mostly every second.

Again, when setting up the initial observability framework it is much better to choose higher level granularity and then change to lower as needed at later point.

Data is Important but Traceability is more Important

While we discussed the importance of data having the right traceability, it is equally important to get meaningful insights. Good observability solution enables data tracing in a way that anyone looking at the data-trail should be able to point where the issue originated – for e, g., a performance degradation in the service response time could have been a result of front-end UI issue. So, it is important that log and trace data can provide anyone the ability (irrespective of their role as SRE, operations, Dev, or QA) to identify the root cause.

This is one of the core principles of good observability – data tracing and correlation allow teams to move away from operational silos, where anyone debugging an issue irrespective of their role, identifies the same root cause. The observability solution should be able to collate all data and create meaningful correlations.

Data Aggregation and Centralization

While we talked about traceability and correlation one of the key aspects for **that to be successful is our ability to get all the important data from all the important tools and sources into a central data lake**. Unless there is a centralized data repository, it is not possible to do real trend analysis, anomaly detections, and so on.

Data Visualization

A good observability solution provides the ability for users to be able to clearly visualize the data – starting with the issue and navigating to the root cause of the issue with minimal clicks / screen navigations. Choosing the right metrics to display and identify the most intuitive dashboards is what is expected out of a good observability solution.

It is also important to have multiple personas built into the dashboards based on the pattern-analysis of the historical usage, the user-roles and the incident types.



Utilization of Machine Learning and Artificial Intelligence

The amount of data from trace and logs that come along with highly distributed microservices based cloud native systems can be overwhelming to be analyzed manually for making predictions, trend analysis, looking for auto remediations or identifying anomalies to take some preventive steps. That is why machine learning and AI models are used as part of the observability solutions for risk predictions or making decisions on auto actions or remediations through bots / human interventions.

Another important point to note is that while observability is key to a reliable IT system, many a times, just having very basic checks on certain thresholds provided by auto-alert mechanisms can create lots of unnecessary and frequent noise. It is important to have the right algorithms that can reduce the noise from all the trace and log data and provide meaningful and actionable insights. That is what a good model with the right training from historical data and statistical analysis can do.

Shift Left and Shift Right

Gone are the days when monitoring and observability was part of the operations function – observability solution is not good if it is not giving the observations from development to deployment in production. A good observability solution must be able to correlate what is happening in the application that leads to certain behavior in the production. That is where role of quality engineering and performance engineering teams becomes crucial in the observability solution design.





Embedding the right functional and non-functional testing processes earlier in the development pipeline (shift-left) and extending the pre-production stage monitoring and some elements of testing (A/B, canary releases) into production (shift-right) are also important guiding principles when designing a robust observability solution.



Observability and Security

For any cloud-native highly distributed (or even otherwise) IT system, security is a very crucial aspect. **Having the right threat monitoring and alerting mechanisms, ability to guide for prevention of security issues and quick remediation suggestions or auto actions in case of a security issue**, are some of the core expectations from a good observability solution.

Good observability solution helps in reducing the overall security operations cost.

-  Quick access to all event data for analysis
-  Faster turnaround to isolate threats and critical events
-  Faster remediation turnaround in case of an incident
-  Better ability to do thorough root cause analysis (RCA) for preventive measures

Another critical aspect of security is also around regulatory and compliance requirements for sensitive data. **Robust observability solution would provide enough traces, logs and metrics to ensure any incidents that might lead to a breach of regulatory and compliance requirements**, it can be arrested proactively and prevented.

Observability-as-Code

There is a movement around everything as code – observability is no exception. Observability-as-Code means to configure and launch the alerts, dashboards, and graphs programmatically, using code and APIs.

Having the **ability to dynamically create the visualizations from the logs, traces, and events alert data helps** in real time views, eliminating wait-time for the periodic report generations. Observability-as-Code also means **programmatically executing scripts of some auto remediations, auto healing scripts based on certain observations / alerts or events.**




Observability-as-Code is one of the important best practices to keep in mind while designing the observability solution.



Alerting Best Practices

We can configure many alerts based on the described thresholds, to alert the teams proactively in case of an event or even proactively alerting or in case of possibility that an event might occur.

There are certain best practices to be followed before setting up these alerts:

-  Identify the **best mode of alerting** depending upon the nature and criticality of the event observed
-  **Proactive vs. reactive alerting** – again decision to be based on type of event
-  **Threshold setting** – are they too conservative? Are we triggering unnecessary alert or are we missing on alerting proactively?

All these aspects need to be analyzed well before deciding on alerting strategy and tools.

Knowledge Management and SOPs

Eventual goal of a robust observability solution is to improve the system reliability, availability, and mean time to recovery or restore (MTTR). While designing an observability solution, one area that at times is missed out or overlooked, is how to ensure the right level of knowledge and remediating steps every time an incident is observed and resolved.

Creating SOPs and runbooks to be able to quickly identify the experts and resolve the issues or prevent a critical incident is also core to a good observability solution. Making remediations and issue prevention as process-dependent, instead of people-dependent, helps big time.

Utilizing a structured repository for maintaining the assets, templates, SOPs, and runbooks is a recommended approach to incorporate as part of the observability framework.

Scalability

Cloud native environments can be complex and as mentioned earlier, the volume of logs, traces and metrics generated through the hundreds of containers, host environments, data bases and network logs, can be overwhelming and humongous. **The observability solution needs to be able to scale to terabytes (or beyond) of data and millions of traces.**

These are some of the important aspects to consider and plan for, while designing an observability framework and solution.



What kind of data stores to be used?



How much history needs to be maintained?



Archival strategy



Observability-as-a-Service

Tech Mahindra's ADMS.NXT.NOW approach for platform engineering and delivery is built to support cloud native architectures by providing end-to-end integrated and automated platform for DevSecOps, IT value stream management, monitoring, and observability, SRE, knowledge, learning and asset management.

As the IT teams working in complex cloud-native distributed and highly scalable environments start working on observability, many a times, the teams end up adopting multiple tools, dashboard solutions, alerting solutions and auto remediation / self-healing approaches that quickly become complicated and overwhelming to manage. That is why it is important to create a centralized framework supported by right set of tools, dashboards, central code repositories that can be utilized by all the teams to have a unified approach to observability.

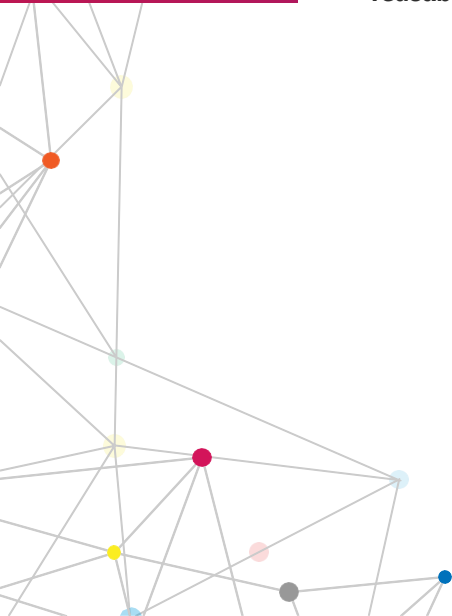
The framework and **platform need to support integration of logs, traces, events from multiple monitoring, logging and service traceability tools and provide actionable insights, intelligence, root cause analysis, preventive steps, auto remediation triggers, bot invocations, SOP / reusable asset creation as needed.**

This is where Tech Mahindra's **NewAgeDelivery (NAD) platform comes in as highly usable platform for providing a centralized observability framework and a platform for cloud-native engineering teams. The platform supports over 300 tool integrations (e.g., planning, build, deployment, release, infra, network, security, monitoring, logging, service traceability with readily usable plug-ins).**

The solution comes with 90+ pre-configured critical dashboards with possibility of configuring event-based alerts. Creation of additional custom dashboards is also feasible.

NAD platform also integrates with over 600 pre-built bots and Tech Mahindra's event driven observability solution for various auto remediation actions. Also comes with an engineering workbench to orchestrate as many auto remediation workflows as needed based on the event or threshold triggers from monitoring tools.

Generally, the large and complex enterprises have cloud infrastructure that is multi-cloud based and **that's another reason to utilize NAD that supports AWS, Azure, GCP and can cut across on-premises and multi-cloud environments. It can integrate with the native monitoring and cloud watch solutions and tools from all three hyper-scalers (AWS, Azure, GCP), and beyond.**





Phases of Engagement – Setting up Observability-as-a-Service Platform

Assessment



**2-3
WEEKS**

This is the most important step in the process when we start getting engaged for observability as a service solution. Assessment of the existing technology, applications, infra and networks landscape including the current set of monitoring and telemetry tools and processes is carried out using the NAD design and POD's assessment tool.

We also utilize tools in conjunction with the assessment questionnaire kit. Tools like Tech Mahindra LCaaS and CAST helps us in creating code and architecture level traceability, dependencies to understand the end-to-end IT landscape including the distributed architecture better to create a holistic observability strategy and solution.

Framework and Platform Implementation



**8-12
WEEKS**

After the assessment is done with the thorough understanding of IT landscape and is documented including tools, technology, processes, and gaps identified; the next step is to define the framework strategy, detailed observability platform architecture that utilizes best tools, solutions from customer's landscape and integrating them with the NAD platform, to create the best-in-class observability-as-a-service platform that can be leveraged and proliferated across the IT and engineering teams.

The idea is to maximize the ROI on the investments already made by the customer and utilize the existing capabilities to the fullest, in conjunction with the capabilities provided by Tech Mahindra NAD platform.

Identifying the right dashboards from the pre-existing 90+ dashboards, ensuring right set of graphs and dashboards are populated and access set-up for the same is also a key activity of this phase.

The process typically takes anywhere between 2-3 months to design and implement the fully functional and highly scalable observability platform that can cover varied and distributed technology landscape.

We also provide support based on the needs of the organization and its change management to ensure seamless adoption.



Run and Continuous Improvement



After the foundation framework, processes and observability processes are set-up – in the ongoing run and continuous improvement phase, the Tech Mahindra team of experts focus on the following, based on the client priorities:

- Increase proliferation and support in wider adoption, by driving various organizational changes and management strategies.
- Continuously monitor usage of data and filter unnecessary data to remove noise
- Identify any missing data captures and improve the data capture strategies
- Identify need for custom dashboards based on feedbacks
- Design, develop, and deploy custom dashboards
- Proactively work on identifying more machine learning and predictive analytics use-cases based on the data
- Real-time data analysis, trend, and anomalies detection
- Enrich the auto-action and auto-remediation bots, create more as needed

The aim of this phase is to maximize the ROI on the capital invested on monitoring tools and observability platform; and ensure that IT teams are exceeding on availability and reliability objectives.

Conclusion

It is evident that observability is a critical solution that requires good strategy, planning and implementation efforts. Also, there are several best practices to be utilized while implementing a robust observability solution.

Once implemented right, the observability solution can add tremendous value and improve the reliability and resilience of complex IT landscapes drastically.

Typical benefits of utilizing a robust observability framework and platform as a service are as following:

Capex Saving

~30-40%

Opex Saving

~50-60%

MTTR Improved by

~45-60%

MTTF Improved by

~45-50%

Author



Anjali Chhabra Nandwani

*Global Head - GTM & Pre-Sales, ADMSNXT
& Digital Assurance Services*

Anjali is a technology leader with over 20 years of IT engineering in technology services industry, with specialization in optimizing software development lifecycle (SDLC), ADMS, DevOps, and quality engineering services. She has worked with large enterprise customers ranging from banking and financial services, telecommunication insurance, RCG, technology, and various other industry verticals. She has a proven track record on DevSecOps transformation initiatives, consulting for DevOps and agile transformation initiatives, automated delivery pipeline (CI/CD/CT) framework design and set-up for large enterprises, community of practice (COP) set-up, DevSecOps program management and governance. In her role, she has worked with diverse IT teams globally to optimize SDLC with usage of predictive analytics, machine learning, and RPA. She comes with in-depth understanding of integrated SDLC automation experience along with framework design / tool selection and ROI analysis.

