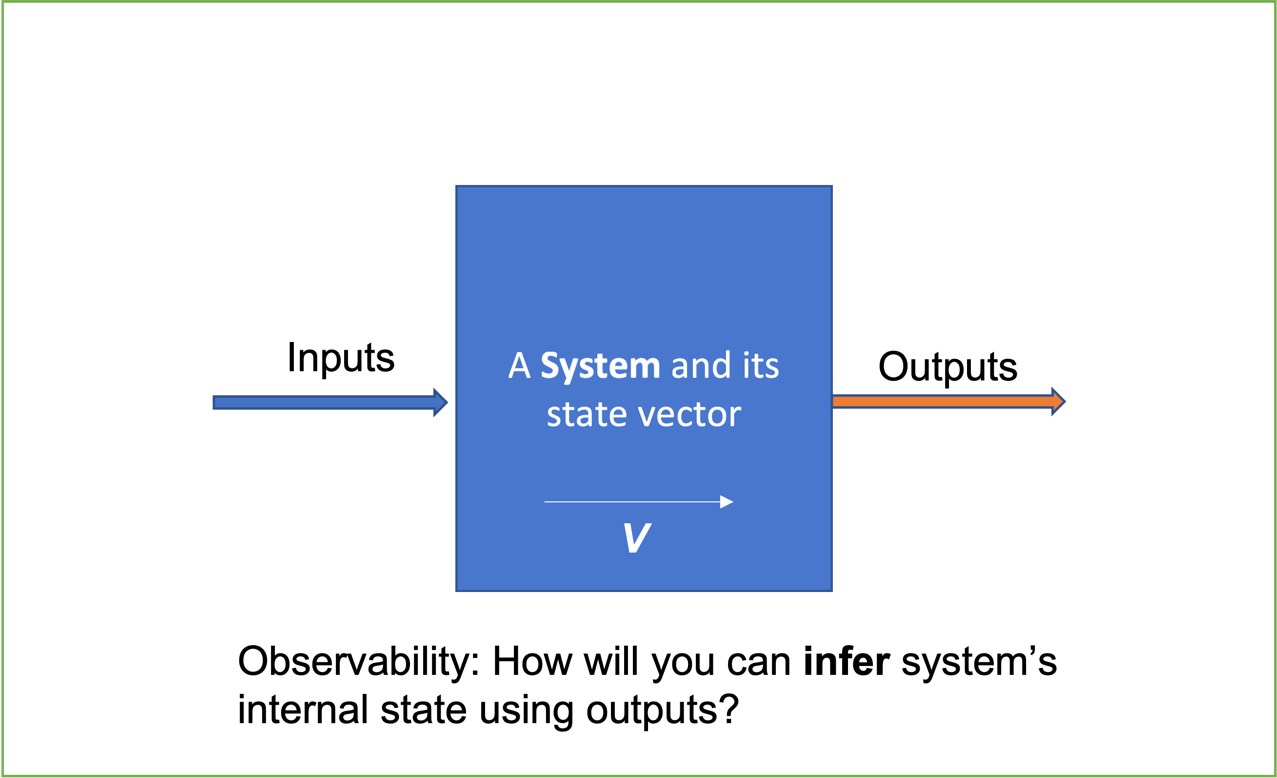
**Observability: Superpower for Engineers**

To get the most out of observability, you need to create an environment where engineers naturally look beyond what went wrong. Observability uses engineering principles like MELT (Metrics, Events, Logs, and Traces) to find the root cause of a problem.

**What is observability?** Observability is not a new concept. It has been applied in the fields of control theory and engineering for many years. The term *Observability* appeared in the field of control theory in the 1960s, thanks to the work of engineer and inventor Rudolf E. Kálmán. Through the lens of control theory, observability is defined as the process of measuring or understanding a system's internal state based on its externally manifested characteristics.



In the mid-2010s, IT workers started integrating observability into their work.

For instance, observability could imply collecting performance data (request frequency, error rates, and request latency) from individual microservices within a distributed application and correlating these data to understand the application's overall health. More significantly, observability uses data analysis and other techniques to forecast how those systems will behave in the future. Overall, observability helps you comprehend the complex state of your system at any given point.

**Observability vs. monitoring, visibility and telemetry**

How Does Observability Compare To Monitoring?

Monitoring and observability are two different concepts that are frequently confused with one another. Both observability and monitoring involve understanding what is happening within your software. However, observability is more of a proactive approach, while monitoring is more of a reactive approach.

Monitoring enables you to recognize when something goes wrong, and observability enables you to identify why things go wrong (the cause of the problem). Observability helps in the prediction of issues before they arise, thereby assisting in the development of a proactive enterprise. By automating observability, it is possible to create self-healing systems capable of fully understanding what is going on within a system and responding to likely outcomes based on systemic predictions.

| Monitoring | Observability |
| --- | --- |
| Answers: Is it broken? | Answers: Why is it broken? |
| Facilitates quick response, AFTER incident occurs. | Prevents and reduces the duration and impact of incident. |
| Is my application (or service) running? | How efficiently is application (or service) running? |
| Siloed data. | Highly correlated data. |
| Passively consume data and metrics about your system. | Actively explore and understand your environment. |

In a similar vein, observability differs from visibility in that it offers the context necessary to comprehend both the *what* and the *why* of software issues. Visibility primarily just alert you to what.

For telemetry, which is the collecting of data from remote systems, observability is different because it offers the context required for comprehensive telemetry data interpretation. Telemetry by itself only provides data, not the intelligence to interpret it.

In a nutshell, telemetry, visibility, and monitoring are some processes that make observability possible. But observability is more thorough and offers a much higher degree of actionability.

**Benefits of observability**

When compared to application performance management techniques that rely solely on processes such as monitoring and telemetry, observability produces even better financial results for the business.

This is partly because observability improves a team's ability to quickly identify and correct root-cause performance issues. This results in less downtime and fewer customer-impacting performance issues, which leads to higher rates of engagement and revenue.

At the same time, observability enables engineering teams to work more efficiently and effectively. Observability tools help teams spend less time tracking down root-cause problems and performing unplanned work by providing the context and correlation that engineers require to resolve complex application issues.

**Observability's criticality in legacy and multi-cloud worlds**

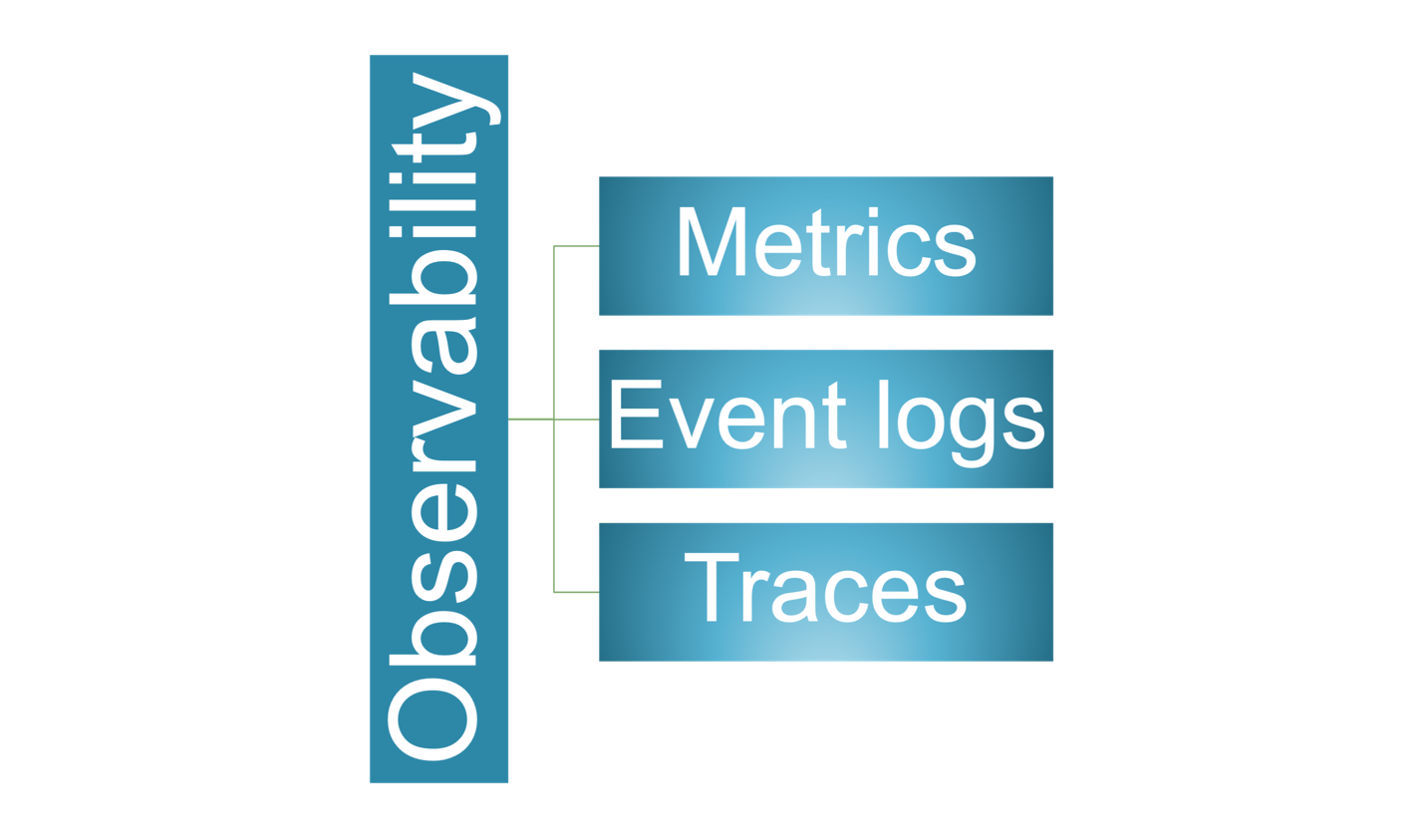
As enterprises adopt modern microservices' architecture, multi-cloud, poly-cloud, and cloud native application architecture, it becomes more difficult to assess and track all application's components.

In such complex contexts, it is quite difficult to monitor and detect any emerging anomalies. This is where observability comes in as superpower. It assists IT professionals in comprehending the "unknown unknowns," providing insight across the entire stack-from networks to hardware to operating systems to even external and internal services. Essentially, it allows IT teams to help identify and resolve specific hindrances and anomalies within dynamic distributed systems without having to switch tools. Modern cloud environments are made up of several interconnected elements such as databases, containers, and virtual machines (VMs) that generate data (logs, metrics, and traces). The teams use this generated data to analyze and understand where the anomalies are.

Legacy applications may not require observability as much as cloud-native applications, but they can still benefit greatly from it. This means that observability can help you regardless of the type of applications your company manages or the technological paradigms it uses.

**The three pillars of observability**

What to observe? Logs, metrics, and traces present their own unique purpose and are complementary. Together, they provide maximum visibility into the behavior of distributed systems

 **Metrics**: Metrics are numerical indications of data that are frequently calculated over time (for example, the number of failed requests). These value counts are derived primarily from a system's performance and can come from various sources such as hosts, services, cloud platforms, and so on. Metrics also contain information about Service Level Indicators (SLIs), which shed light on memory and power usage. Because SLIs are frequently generated, they are efficient and dependable. At ABC, metrics are generally aggregated and reported by AppDynamics or Instana.

**Event Logs**: Logs are computer-generated structured, plain, or binary text. These are time-stamped records that provide a precise understanding of discrete events that occur at a specific time. Event logs help detect unpredictability in distributed system components. At ABC, log aggregation is handled by the Centralized Logging Service(CLS), based on the Elastic stack.

**Traces**: Traces keep track of a request or transaction activity as it moves through an application. These are critical because they provide context for the previously mentioned data. Traces, for example, can help provide insightful information about which metric is more important or which log is more relevant to the specific transaction ID. At ABC, traces generally provided by AppDynamics or Instana.

**How to observe in practice?**

1. Open instrumentation: involves collecting event logs, metrics, and traces from any source, including applications, services, cloud services, mobile applications, etc. It makes all types of diverse environments more observable and makes the entire infrastructure visible.
2. Connected Data: Simply collecting data from all sources is insufficient. It is crucial to link it to other metadata so that the sources generating these data can be recognized and linked. This part provides context for the gathered data so that you can interpret it meaningfully.
3. Automation: This element links all the data to the business goals. Goals like having a smooth customer experience can typically be measured and seen through KPIs in the dashboard.

To deliver a more interactive experience, one should build curated applications on observability platforms in order to fully understand observability in relation to business outcomes.

**Observability is part of our culture**

Tooling is important, but it's not the only factor in the equation. You'll need a platform that can collect, correlate, and analyze data across all aspects of complex systems.

In addition to the right tooling, you need the to create a culture that bleeds and believes in observability.

You must integrate observability into your IT organization's culture in order to get the most benefit from it. As a result, every engineer needs to be aware of the distinctions between observability and monitoring and be ready to develop the necessary instrumentation to support observability.

Building a culture where engineers instinctively ask "*Why* is it wrong?" rather than "*What* is wrong?" is necessary to get the most out of observability. Additionally, it necessitates showing that all stakeholders, especially those in administrative responsibilities, embrace observability.

# Observability Services at

* [AppDynamics](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#appdynamics)
* [Elastic](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#elastic)
* [Instana](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#instana)
* [Moogsoft](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#moogsoft)
* [Catchpoint](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#catchpoint)
* [InfluxDB](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#influxdb)
* [Netscout](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#netscout)
* [Grafana](https://dta.discoverfinancial.com/articles/app-eng_observability_service-catalog#grafana)

## AppDynamics

See [AppDynamics](https://dta-staging.cf-ssb-z3-dev.discoverfinancial.com/articles/app-eng_observability_appdynamics/).

## Instana

See [Instana](https://tupperware.discoverfinancial.com/" \l "/resources/monitoring).

## Elastic (CLS)

* CLS Home
* [Ingestion endpoints](https://github.discoverfinancial.com/et-cls-org/support/wiki/CLS---Target-Endpoints-for-clients)
* See [Logs](https://dta-staging.cf-ssb-z3-dev.discoverfinancial.com/articles/app-eng_observability_logs/).

## Moogsoft

* **Summary**: Aggregates and analyzes telemetry data to provide alerts and reports.
* **Goals**: decrease mean time to detection (MTTD) and resolution (MTTR) of issues, and mean time between failures (MTBF) of services.
* **Product**: Moogsoft AIOps platform (<https://www.moogsoft.com/>)
* **Docs**: <https://docs.moogsoft.com/>
* **Portal**: [https://discover.moogsoft.com](https://discover.moogsoft.com/). Accessible to all.

#### Description

Moogsoft provides "AIOps", which here means analysis-enhanced IT operations. The service analyzes incoming telemetry signals - logs, metrics, traces, etc. - to reduce noise (false positives) and identify likely causes before alerting humans. It also captures human reaction to its analyses to get better over time. ABC is migrating from static alerts with Netcool to dynamic, analysis-driven alerts with Moogsoft.

[Phil Tee](https://www.linkedin.com/in/philtee/), creator of Netcool per his Linkedin page, is the CEO of Moogsoft.

## Catchpoint

* **Summary**: Executes synthesized (fake) activities from external networks to identify problems proactively.
* **Goals**: Ensure services are available, reachable, reliable, and performant.
* **Product**: <https://www.catchpoint.com/>
* **Portal**: [https://portal.catchpoint.com](https://portal.catchpoint.com/). Accessible to all. Enter dfs as namespace in "Company Credentials (SSO)" dialog box.

#### Description

Execute black box tests from many network locations, including far outside ABC's network. Includes tests for APIs, browsers (Selenium), HTML and more.

## InfluxDB

* **Summary**: A managed database system specialized for storing time-series data.
* **Product**: <https://www.influxdata.com/>
* **Project**: <https://github.com/influxdata/influxdb>

#### Description

* Replacing [Broadcom UIM](https://www.broadcom.com/info/aiops/unified-infrastructure-management) and [Ganglia](http://ganglia.info/) at ABC.
* Uses [Telegraf server agents](https://www.influxdata.com/time-series-platform/telegraf/) which support many [plugins](https://docs.influxdata.com/telegraf/v1.14/plugins/plugin-list/).
* Team is focused on VMs and OS's now, containers and K8s clusters later.
* Provides its own dashboards but will provide Grafana integration too.

## Grafana

* **Summary**: Aggregates data sources and provides tools for creating dashboards, reports and alerts.
* **Product**: <https://grafana.com/>
* **Project**: <https://github.com/grafana/grafana>
* **Docs**: <https://grafana.com/docs/grafana/latest/>

## NetScout

* **Summary**: Tap and analyze any and all network traffic.
* **Product**: nGeniusOne, nGeniusPULSE