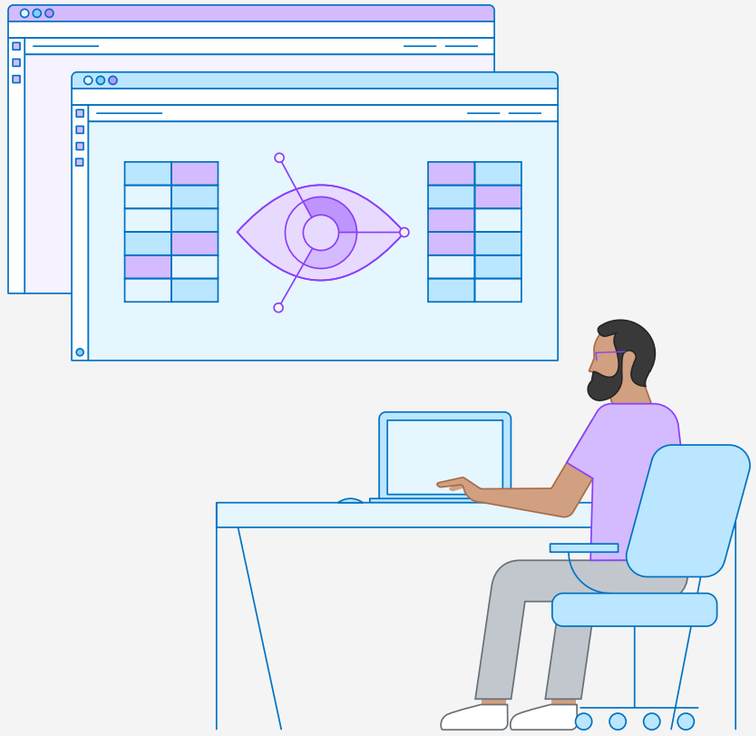
# Observability vs. Monitoring: What's the Difference?

Start here →



# Monitoring and observability are two ways to identify the underlying cause of problems — how are they similar and different?

When something goes wrong with an application, it impacts customers and, ultimately, impacts the business. Teams need a way to find the root cause of problems and quickly resolve them. That's where monitoring and observability come in.

Monitoring and observability are two ways to identify the underlying cause of problems. Monitoring tells you when something is wrong, while observability can tell you what's happening, why it's happening and how to fix it. To better understand the difference between observability and monitoring, let's look at how each works and the roles they play today within software development.

## What is observability?

Observability is the ability to understand a complex system's internal state based on external outputs. When a system is observable, a user can identify the root cause of a performance problem by looking at the data it produces without additional testing or coding.

The term comes from control theory, an engineering concept that refers to the ability to assess internal problems from the outside. For example, car diagnostic systems offer observability for mechanics, giving them the ability to understand why your car won't start without having to take it apart.

In IT, an observability solution analyzes output data, provides an assessment of the system's health and offers actionable insights for addressing the problem. An observable system is one where DevOps teams can see the entire IT environment with context and understanding of interdependencies. The result? It allows teams to detect problems proactively and resolve issues faster.

## What is monitoring?

Monitoring is the task of assessing the health of a system by collecting and analyzing aggregate data from IT systems based on a predefined set of metrics and logs. In DevOps, monitoring measures the health of the application, such as creating a rule that alerts when the app is nearing 100% disk usage, helping prevent downtime. Where monitoring truly shows its value is in analyzing long-term trends and alerting. It shows you not only how the app is functioning, but also how it's being used over time.

Monitoring helps teams watch the system's performance and detect known failures; however, monitoring has its limitations. For monitoring to work, you have to know what metrics and logs to track. If your team hasn't predicted a problem, it can miss key production failures and other issues.

# Observability vs. monitoring: How it works

When it comes to monitoring vs. observability, the difference hinges upon identifying the problems you know will happen and having a way to anticipate the problems that might happen. At its most basic, monitoring is reactive, and observability is proactive. Both use the same type of telemetry data, known as the three pillars of observability.

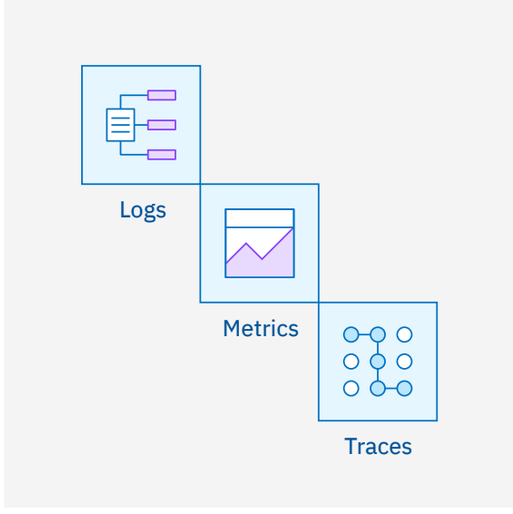
The three pillars of observability are as follows:

- **Logs:** A record of what's happening within your software.
- **Metrics:** A numerical assessment of application performance and resource utilization.
- **Traces:** How operations move throughout a system, from one node to another.

When monitoring, teams use this telemetry data to internally define the metrics and create preconfigured dashboards and notifications. They also identify and document dependencies, which reveal how each application component is dependent on other components, applications and IT resources.

An observability platform takes monitoring a step further. DevOps, site reliability engineers (SREs), operations teams and IT staff can correlate the gathered telemetry in real-time to get a complete view of application performance. This way, they not only understand what's in the system but how different elements relate to each other. The platform shows you the what, where and why of any event and what this could mean to application performance, guiding how DevOps teams perform application instrumentation, debugging and performance fixes.

Observability platforms also use telemetry, but in a proactive way. They automatically discover new sources of telemetry that might emerge within the system, such as a new API call to another software application. To manage and quickly gather insights from such a large volume of data, many platforms include machine learning and AIOps (artificial intelligence for operations) capabilities that can separate the real problems from unrelated issues.



# The evolution of APM to observability

Observability and application performance monitoring (APM) are often used interchangeably; however, it's more accurate to view observability as an evolution of APM.

APM includes the tools and processes designed to help IT teams determine if applications are meeting performance standards and providing a valuable user experience. APM tools typically focus on infrastructure monitoring, application dependencies, business transactions and user experience. These monitoring systems aim to quickly identify, isolate and solve performance problems.

APM was the standard practice for more than two decades, but with the increased use of agile development, DevOps, microservices, multiple programming languages, serverless and other cloud-native technologies, teams needed a faster way to monitor and assess highly-complex environments. APM tools designed for a previous generation of application infrastructure could no longer provide fast, automated, contextualized visibility into the health and availability of an entire application environment. New software is deployed so quickly today, in so many small components, that APM had trouble keeping up.

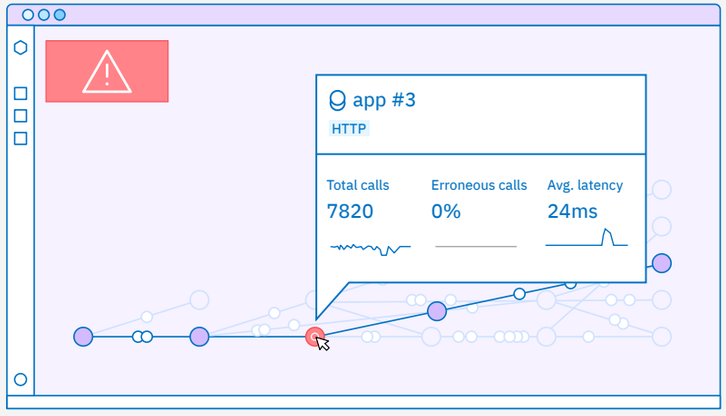
Observability builds upon APM data collection methods to better address the increasingly rapid, distributed and dynamic nature of cloud-native application deployments, making it easier to understand a system and then monitor, update, repair and, ultimately, deploy it.

# Observability tools and automation

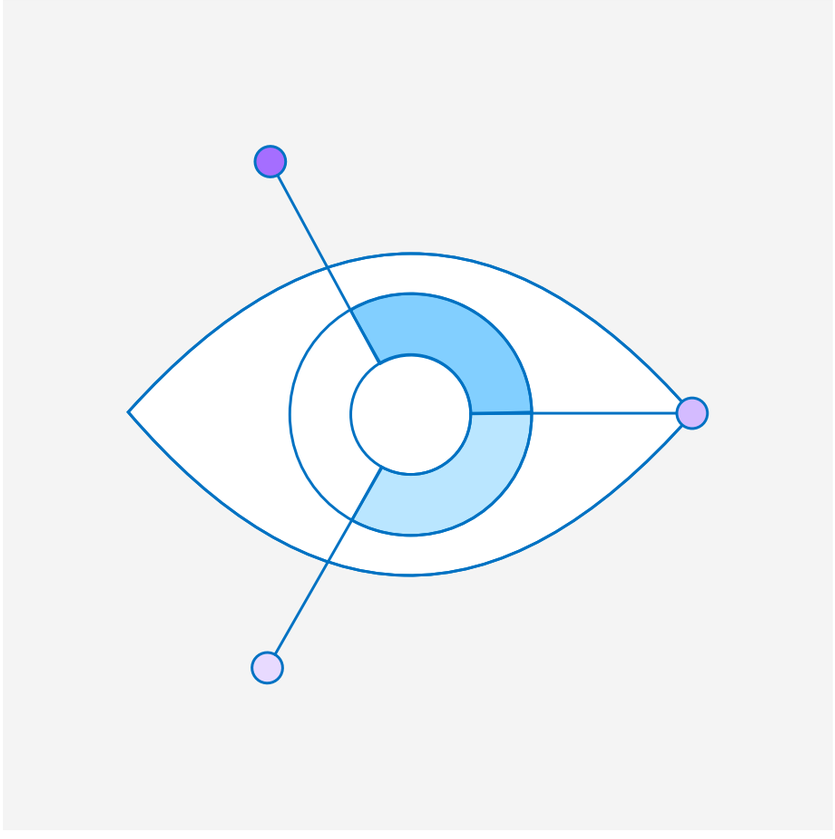
Observability and monitoring tools go deeper than monitoring internal states and troubleshooting problems. These platforms help teams solve problems faster, which in turn, optimizes pipelines and gives more time for core business operations and innovation.

Here, let's dive deeper into some types of tools and approaches to observability and monitoring:

- **Observability platforms:** These platforms provide a way for teams to integrate monitoring, logging and tracing throughout the IT environment to provide a full view of the system's state, even across distributed systems. Some platforms also include user experience and business context to provide a more robust picture of performance health. Depending on the platform, they are designed to provide visualization of both on-premises systems and complex, multicloud environments.
- **Open source:** Open-source data observability tools, like OpenTelemetry, help teams monitor and debug apps, collect log and metric data and perform tracing. These tools offer the ability to perform some, but not all observability functions, and they are often used in some combination.
- **Automation:** Observability automation is simply an extension of existing automation within the CI/CD pipeline, further freeing up DevOps to focus on core tasks. For example, IBM Instana Observability offers state-of-the-art intelligent automation capabilities that accelerate the CI/CD pipeline by automating the discovery of applications, infrastructure and services. This capability means developers don't need to hard-code application and service links every time an update is made. With AI-assisted troubleshooting, Instana can predict incidents and automate remediation. A fully automated application performance management system monitors every service, traces every request and profiles every process.



# Observability and IBM



With Instana, IBM provides a fully automated enterprise observability platform that delivers the context needed to take intelligent actions and ensure optimum application performance. For example, Instana offers the following features and benefits:

- **Automation:** Gain full observability in dynamic environments with auto-discovery. Be able to trace every request, record all changes and get one-second granularity metrics.
- **Context:** Understand all application inter-dependencies to diagnose issues and determine impact. Instana contextualizes raw data into meaningful information, providing an interactive model of relationships between all entities in real-time.
- **Intelligent action:** Proactively detect and remediate issues with an understanding of contributing factors. Analyze every user request from any perspective to quickly find and resolve every bottleneck.

Enhance your application performance monitoring to provide the context you need to resolve incidents faster.

Learn more →



IBM Corporation
Route 100
Somers, NY 10589

Produced in the United States of America
November 2022