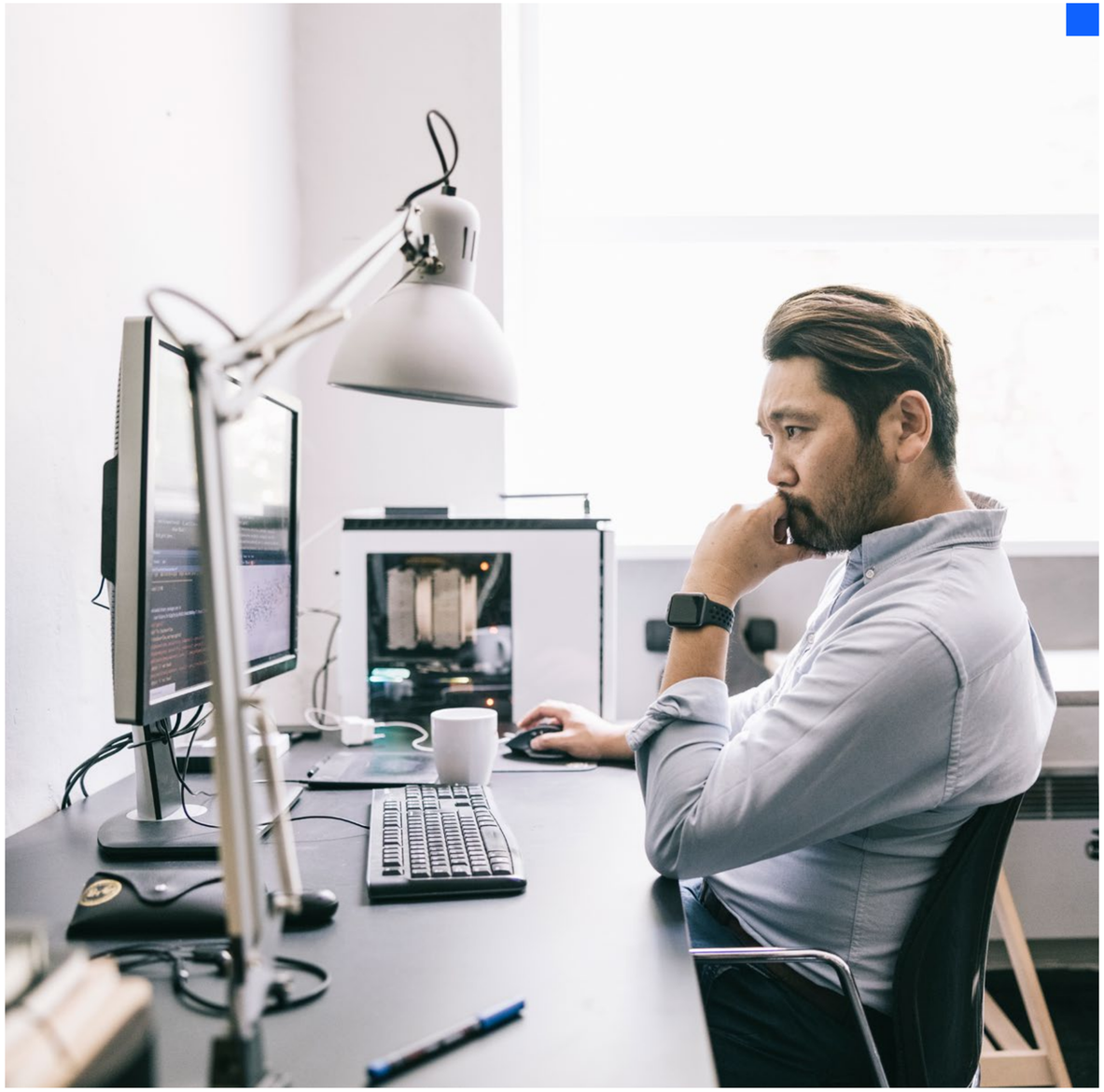
IBM Instana

The critical role of observability in microservice environments



Contents

01 →
The brave new world of software design

02 →
The technologies of the microservices age

03 →
The challenges of operating microservices

04 →
Use intelligent analysis to thrive with microservices

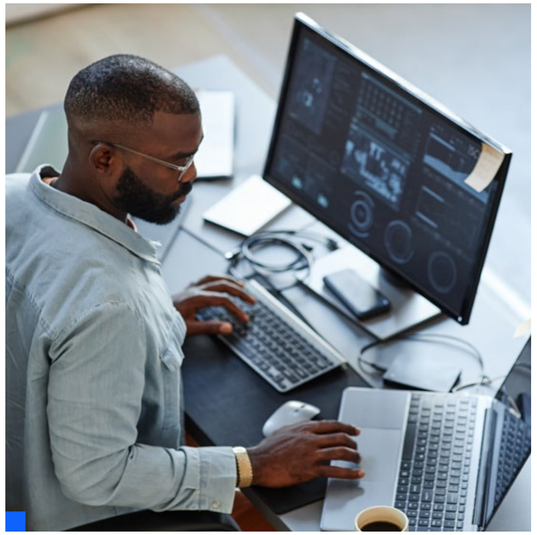
05 →
A new generation of observability

06 →
Conclusion

07 →
Is IBM Instana right for you?

01

The brave new world of software design



The adoption of the microservices architecture has revolutionized the landscape of application development, delivery and management. By embracing this model, organizations can expedite the pace of application development and updates, enabling businesses to become more agile, scalable, robust and resilient against disruptions. Microservices empower DevOps teams to leverage their skills and resources, facilitating continuous delivery and efficient application updates.

However, while microservices overcome many challenges associated with traditional software delivery, they also introduce new complexities. Unique architectures like containers, orchestration, hybrid cloud, multicloud and serverless bring a heightened level of intricacy to application environments. By breaking down applications, services and infrastructure, microservices create novel management hurdles.

Furthermore, microservices are often distributed across a cluster of servers and operate on a dynamic tech stack that includes containers. This constant fluidity in application locations and pathways adds to the operational complexity. While migrating to microservices achieves greater agility and velocity, it also increases operational intricacies.

While microservices offer clear business benefits, they significantly complicate overall service quality and performance management. However, this tradeoff does not imply that the service quality of a microservices environment cannot be effectively monitored and managed. It necessitates a fresh approach to observability and monitoring since traditional solutions were developed before the advent of microservices.

In summary, microservices have brought about substantial changes in application development, delivery, and management. While they offer benefits such as increased agility and scalability, they also introduce new challenges that demand a novel approach to monitoring and observability. By adopting appropriate tools and practices, organizations can effectively manage these complex microservices environments while optimizing application delivery.

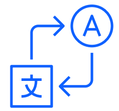
02

The technologies of the microservices age

Let’s begin by defining the new technologies and design philosophies that continue to shape the way software is conceived and delivered in the age of microservices. Today, the following concepts and technologies define the production of software:

- Continuous delivery
- Polyglot pipelines
- Containers
- Highly distributed environments
- Software-defined everything
- Everything as a service
- Wider DevOps adoption

Continuous delivery
Crafting code slowly based on plans made far in advance is no longer feasible. Today’s software engineering teams strive to release code on a rapid, continuous basis by reacting to user feedback as quickly as it arrives. Continuous delivery helps make software delivery agile and nimble. It also makes it easy to adjust or add a new microservice because changes to a single microservice don’t disrupt the larger application.



Polyglot pipelines
Software engineering teams also place a premium on switching easily between different development frameworks as their needs and the tools available change. The ability to maintain a polyglot pipeline is another crucial factor in achieving agility.

Containers
Production-ready container platforms, especially Docker, introduced over the past several years give software delivery teams a leaner, faster way to write, stage and deploy code. Containers lend themselves well to microservices apps because the various microservices that comprise an app can be rapidly deployed across different groups of containers. Applications also scale easily within a containerized environment, especially with automated service discovery. And orchestrators help manage configurations automatically.

Highly distributed environments
Not long ago, applications and services were rarely distributed across multiple servers. But with the introduction of service-oriented architectures (SOAs) in the mid-2000s, along with the move to cloud-based environments, single server architecture changed. With microservices, the transition is complete; production environments are commonly distributed across multiple servers.

This distributed design adds a great deal of scalability and resiliency because it allows hosting infrastructure to be added or subtracted from an environment without downtime. Distributed environments also ensure that the failure of an individual server won't disrupt the continuity of a running application when it's distributed across many servers.

Software-defined everything
One of the factors that makes mapping modern applications difficult is software-defined environments. Now widespread, software-defined entities such as infrastructure and servers create layers of abstraction between application services and their underlying infrastructure.

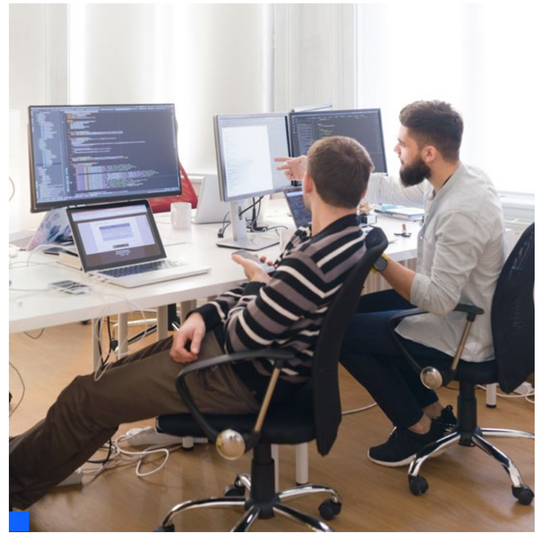
While these layers make microservices easier to deploy by removing the need to tie individual services to specific hosts, this dynamic architecture can create several observability issues.

Everything as a service
As more workloads move to the cloud, teams are using on-demand compute and storage resources more often, usually as a pay-as-a-service method. Operating network, infrastructure, platforms and workloads as on-demand services makes it faster and easier to scale when needed in order to optimize ongoing costs. But operating all these things as an ephemeral service can create gaps in observability, visibility and performance monitoring.

Wider DevOps adoption
The convergence of agile development practices and the rise of the DevOps movement has revolutionized how organizations approach software design and delivery. This transformation has had a profound impact on observability and monitoring, underscoring the importance of continuous collaboration and communication among all stakeholders involved in the application lifecycle—from business application owners to operations and development teams. This cooperation requires flexibility and widespread consolidation of operational roles, particularly during performance optimization and troubleshooting.

By adopting a microservices architecture for application design, you can enhance DevOps efficiency by producing software that is as agile as the teams developing it. Breaking down applications into smaller, more manageable components empowers development teams to work more autonomously—which leads to faster iteration and deployment of new features, enhanced scalability and resilience, and can even boost overall innovation and productivity. Concurrently, this approach optimizes observability and monitoring capabilities, allowing for improved insights into application performance and health.

The end result: A modern application deployment process
These seven modern concepts about the new application stack present their own challenges for observability and performance monitoring. Mass virtualization, broad container adoption and the widespread acceptance of Kubernetes have helped organizations of all sizes reach new heights of efficiency by redesigning and operating completely revamped application development and deployment processes.



03

The challenges of operating microservices

Overall, the microservices technology stack has had a positive impact on software delivery in most respects. Software production and deployment today are faster, more reliable and more agile than just a few years ago.

Organizations employing modern software delivery techniques are also better positioned to support new business services and cater to precise business needs. However, both the accelerated invention and adoption of new application technologies come with some caveats. The same innovations that make agile, microservices-based software delivery possible can also make observability, monitoring and service quality management more difficult. Here's why.

The challenges of microservices:

- Complex dependencies
- Continuous delivery
- Containerized environments
- Microservices architectures
- Everything as a service
- DevOps and fluid roles

Complex dependencies
Mapping dependencies in a microservices environment is critical but extremely complex and almost impossible to do manually. Previous monolithic or SOA-based applications had only a few steady interservice dependencies. So it was fairly easy to identify which applications were running on which servers and what storage and data services were linked to which applications.

In a microservices architecture, however, the dependencies and performance patterns that link services and infrastructure together are much more complicated. Microservices are often deployed using containers that are distributed across a cluster of servers with network routes being constantly adjusted by load balancers responding to demand. The ports that individual microservices use to communicate can be changed without notice by orchestrators.

This complexity also makes incident resolution and root cause analysis difficult. For example, a storage service problem could be the result of a failed server, a coding mistake, a disk failure or a slowdown on a virtual network. When this happens, the only way to identify the cause and resolve the issue is mapping the intricate dependencies of the storage service.

Continuous delivery
In agile code development, services and architectures are developed in small increments. While this makes software delivery faster and more flexible, using a large quantity of smaller services means that changes are introduced at a faster pace. Effective observability and monitoring requires seeing changes in real time so that monitoring configuration and active service maps are adjusted every time a service is updated.

Containerized environments
In containerized environments, there are significantly more components to monitor within an application. Instead of a few dozen servers and applications that you might have in a traditional environment, a containerized environment could be made up of thousands of individual ephemeral containers. As a result, the number of objects that require observability and monitoring, as well as the rate of change, increases dramatically.

Microservice architectures
Plainly, in a microservices environment, there are many more services to track and measure when monitoring application performance. But it’s not enough simply to know whether an app is up or down. It’s important to also understand the quality of service of the many microservices that make up the applications. This method helps you identify and address reliability issues before they can degrade the performance of the entire application.

Everything as a service
The everything-as-a-service model means many organizations no longer own the underlying infrastructure where their applications, services, microservices and containers live. But it also means there's no easy way to map any dependencies between microservices. When the separation of physical servers from the application workloads prevents DevOps teams from touching the production application servers, conventional monitoring software is likely the best option. Modern observability platforms and tools can provide visibility into this complex infrastructure, no matter where the workloads are located or if they’re managed by third parties.

DevOps and fluid roles
Under the DevOps model, individual team member roles are fluid and shifting. This means everyone now has a hand to play in application delivery and service quality management—including engineers and administrators not specifically trained in application technologies or app architecture. Ever-shifting roles mean DevOps teams should look for quality management solutions that are sufficiently intuitive and automated for nonspecialists to use.

04

Use intelligent analysis to thrive with microservices

Now that we've outlined the challenges associated with monitoring and ensuring quality in the microservices age, let's examine the strategies software delivery teams can adopt to meet the challenges.

Using intelligent analysis to thrive in the world of microservices:

- Disconnect incidents from noise.
- Make monitoring information actionable.
- Understand incidents and dependencies precisely.
- Share information easily across the team.
- Maintain both real-time visibility and historical visibility.
- Minimize manual configuration.
- Achieve continuous understanding.

Effective intelligent analysis centers on automating workflows and using tools to achieve results at scale. Intelligent analysis, machine-assisted learning and automation are at the root of successful service delivery and quality management for microservices.

Disconnect incidents from noise
Because a microservices environment involves so many moving parts and a high volume of activity, it's important to quickly separate incidents that require immediate handling from operational noise using intelligent analysis. An incident is any event or issue that negatively impacts the quality of a service or microservice. Most often, this involves a failure to deliver within a predefined time.

In contrast, noise is data generated by normal activities not associated with a quality-of-service problem. Separating incidents from noise means glancing at an application and service dashboards to recognize quickly which data is associated with a potential problem, like a microservice running short of compute resources. And which types of data are just natural, unremarkable events, like containers spinning down as demand decreases.

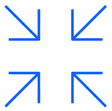
Make monitoring information actionable
Monitoring data should be immediately translatable into action. Both the monitoring software and its users should be able to quickly use monitoring information to understand and fix a problem. To achieve this goal, look for tools that go beyond simply collecting data. Tools that take advantage of machine learning and automation can use intelligent analysis to quickly turn data into actionable information. The tools should also be able to detect anomalies and map service dependencies automatically. This capability lets the DevOps team spend their time solving the actual problem rather than wasting time interpreting data manually.

Understand incidents and dependencies precisely
Knowing that a problem exists within an application service is only the beginning of the battle. The harder problem is to quickly identify the root cause of the issue and immediately understand how it impacts the entire environment, including other applications. Tools that automatically map dependencies can provide this level of insight. Understanding how services interact and how a problem with one component can affect others is essential. That's because, in a fast-moving microservices environment with layers of infrastructure, the source of an incident could lie in many different places—in a host server, middleware, application code, a container, on the network or somewhere else.

Share information easily across the team
In modern DevOps, giving more stakeholders access to real-time performance monitoring and observability data can create more collaboration and faster responses. There's tangible value in sharing information seamlessly across the entire organization. For example, letting the Ops team forward problem information to an assigned programmer is nice. But it could be far more valuable if developers simply used the data themselves. Likewise, observability and performance management tools can provide visibility into the full application stack for the entire team to support a continuous feedback loop. When it comes to information sharing, the central goal of an operations workflow should be implementing continuous understanding of events for all team members.

Maintain both real-time visibility and historical visibility
Real-time problem detection is vital to organizations. Real-time requires that you separate incidents from noise instantaneously, identify the root of a service problem, and use service quality and performance data for an immediate response. It's equally crucial to be able to time-shift.

Time shifting means stepping back in time by viewing historical data and dependencies at a specified point in the past. This technique gives you an understanding of how an incident may have evolved over time and which initial conditions triggered it. You can do this by examining information such as the initial layout and structure of an application or looking at previous allocations of containers to hosts. The tools you choose should support both types of scenarios.



Minimize manual configuration
In a hyperscaled environment, tools that automate monitoring configuration can make a big difference in efficiency. Because, in a microservices environment, updating configuration information or adding services manually takes time—time that could cost your organization revenue or reputation.

This is due both to the large number of components in a microservices environment and the complex dependencies that can arise in a microservices architecture. Attempting to configure tools for this environment manually can weaken your efforts to be agile and scalable. And that undermines the purpose of implementing a microservices architecture in the first place.

That's why the service quality management strategy should maximize tools that automate configuration and service discovery. Deploying new code, new applications, new services, new servers and so forth should be automatically discovered along with their associated dependencies. It's another essential ingredient for achieving continuous discovery and understanding.

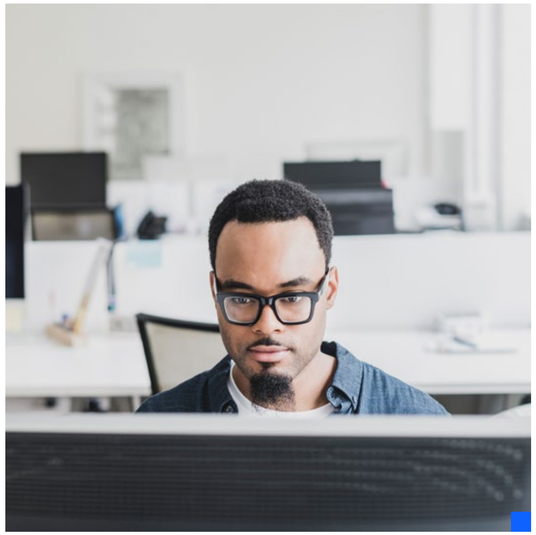
Automating monitoring configuration helps free the operations and development teams to focus on activities that require human input. For example, DevOps teams should spend time on interpreting and acting on complex health data, not manually setting up monitoring agents.

As noted previously, automated anomaly detection delivered through machine learning and the mapping of service dependencies can minimize manual configuration. Using monitoring tools that detect anomalies and dependencies automatically reduces DevOps team involvement.

Achieve continuous understanding
In highly dynamic microservices environments, performance and service quality information, and trends are extremely perishable. Data can cease to be relevant just minutes after it's identified. That's why continuous understanding can provide enormous value when implementing workflows for observability, application performance monitoring (APM) and service quality management solutions.

Continuous understanding is facilitated by ongoing, automatic rediscovery of system and monitoring configurations, environment data and dependencies. Automation and intelligent analysis enhance continuous understanding because automated processes can help you gain insight into microservices.

A new generation of observability



Achieving a complex, flexible, ultra-scalable development and deployment strategy in the most agile development environments requires a completely different approach to observability, monitoring and performance management.

IBM Instana™ is an industry leader in providing automated approaches to monitoring and delivering the next generation of management platforms to support enterprise observability. The platform helps agile organizations deal with the dynamic complexity inherent in microservice applications.

Automation
For modern application teams who are using agile development methodology and running an always full continuous integration continuous delivery (CICD) pipeline, automation is their lifeblood. If observability and monitoring are manual, the entire process can get stalled or derailed at the monitoring step. Enterprise observability automates the entire monitoring lifecycle including discovery, mapping, monitoring configuration, alerting, and root cause analysis.

Context
In monolithic applications where transactions always follow the same path through the tech stack and code base, the use of shortcuts like sampling or reverse engineering for monitoring and tracing might be tolerated. But in dynamic microservice applications, the only constant is change.

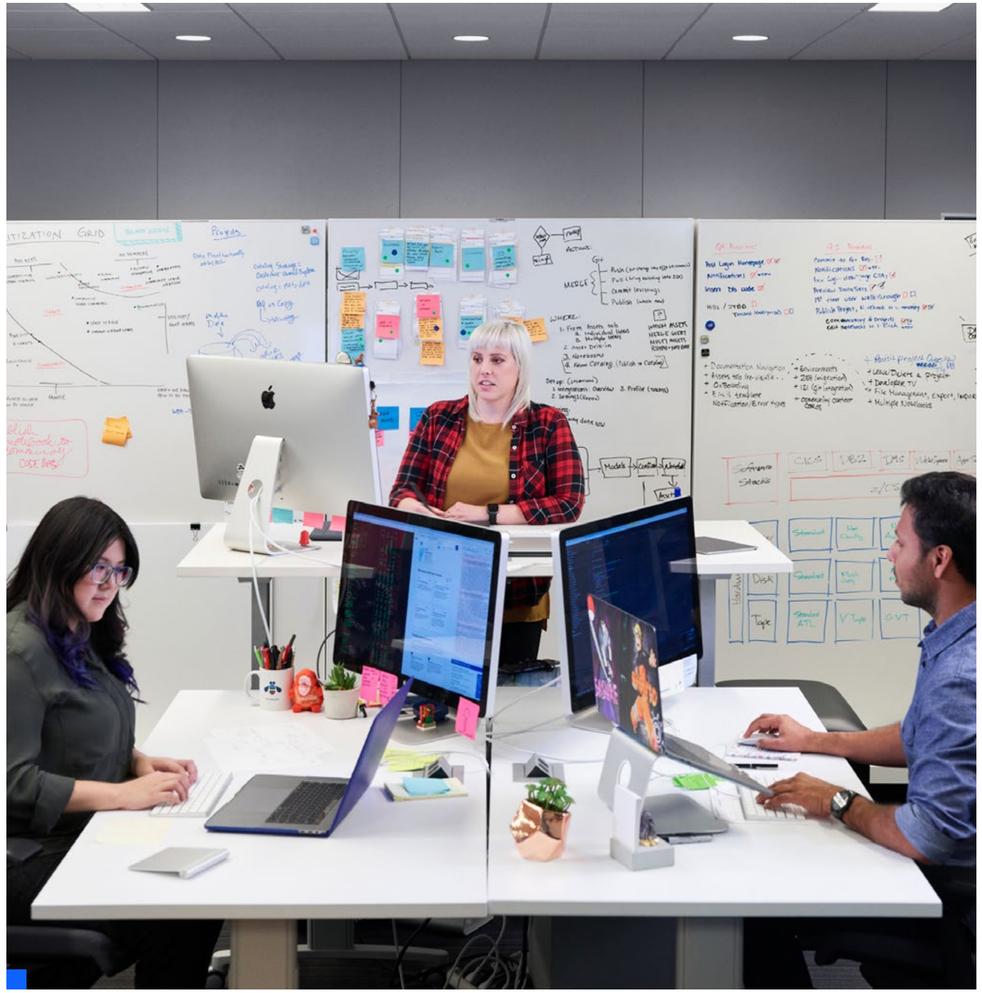
In this space, understanding how every component or entity interacts with others helps teams understand where to start solving problems. But merely sampling application response metrics, infrastructure configuration and request tracing is no longer adequate.

Enterprise observability ties everything together—service maps, infrastructure configuration, request traces, open-source protocols, even profiles—to provide understanding, both programmatically and to end users, from DevOps to decision makers.

Intelligent actions
Collecting data without being able to do anything with it is simply a waste of time and money. The purpose of observability in container-based applications remains the same as application monitoring or APM solutions: to validate application performance and find and fix problems when they occur.

Because container-based applications are so complex, finding the root cause of problems requires assistance. Enterprise observability uses its knowledge of application and infrastructure architecture, its immense collection of events and understanding of requests, or traces, to identify triggering events and intercomponent relationships. In today's IT environments, it's the only way to effectively highlight where attention is required by DevOps to fix problems and optimize application performance.

Ease of use
As organizations roll out agile development processes, increase the frequency of their application updates and fill their CICD pipelines, more stakeholders require actionable information. That's why ease of use, which was never a strength of APM solutions, is so critical to enterprise observability. The more people that can use the information from their observability solution or platform, the better overall application performance will be.



06

Conclusion

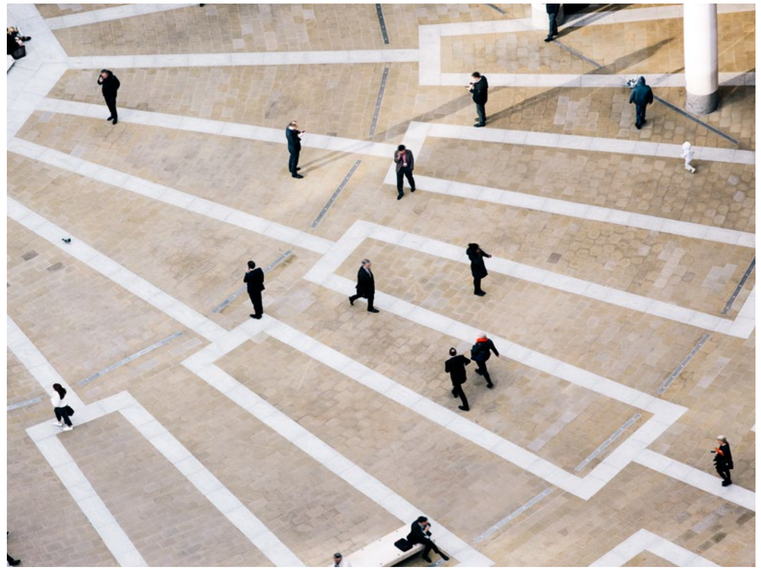
While organizations tend to think of observability issues as a new set of problems unique to microservice environments, enterprise observability expands modern APM, or APM 3.0, to provide better support for cloud-native applications.

The dangers of ignoring new IT service management solutions built on automated monitoring can range from late nights in the lab to significant revenue loss. Regular server monitoring and APM monitoring simply cannot reflect the reality of microservice environments.

While manual instrumentation of microservice code can provide some details, it is less scalable operationally for monitoring performance, especially as software update frequencies increase. DevOps teams can derive substantial benefits from a totally new generation of enterprise observability and performance management platforms that automate setup, discovery, observability, monitoring and tracing. Optimally, these solutions operate without much, if any, human configuration—either up front or after any application updates.

07

Is IBM Instana right for you?



IBM Instana™ is an enterprise observability platform that includes automated application performance monitoring capabilities. It's designed for businesses operating complex, modern, cloud-native applications no matter where they reside—on premises, in public and private clouds, on mobile devices or in an IBM zSystems™ environment.

IBM Instana helps you control modern hybrid applications with AI-powered discovery of deep contextual dependencies inside hybrid applications. IBM Instana also provides visibility into development pipelines to help enable closed-loop DevOps automation.

These capabilities provide actionable feedback needed for clients as they optimize application performance, enable innovation and mitigate risk. These features help DevOps increase efficiency and add value to software delivery pipelines so they can meet their service and business-level objectives.

See the power of IBM Instana for yourself. Sign up today for a free 14-day trial of the full version of the product. No credit card required.

IBM Instana free trial →

Explore IBM Instana →

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
May 2023