Microservice observability

INSTANA
an IBM Company

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 About Instana, an IBM Company

01 The brave new world of software design

Microservices have had a big impact on how applications are designed, developed, delivered and managed. Thanks to the microservice architecture model, applications can be developed and updated at a much higher velocity. Applications—and organizations—are more nimble, more scalable, more robust and more resilient against disruptions.

Microservices can be updated quickly and easily. They enable fully continuous delivery and allow DevOps teams to make the most of their skills and resources. But while microservices break through many hurdles associated with traditional software delivery, they also create new types of challenges.

From unique architectures including containers, orchestration, hybrid cloud, multicloud, serverless and so on, to the ephemeral and agile nature of service-based applications, microservices create great opportunities to optimize application delivery. But by disintegrating applications, services and infrastructure, a new level of complexity in application environments is introduced, creating new management challenges.

Complicating matters further, microservices applications are often distributed across a cluster of servers and tend to operate on an ephemeral tech stack that includes containers, which creates constant fluidity regarding application locations and pathways.

In essence, when an organization migrates to microservices to achieve greater velocity and agility, the tradeoff is higher operational complexity. While microservices deliver a clear business benefit by making software faster and more agile, they also significantly complicate management of overall service quality and performance. This tradeoff doesn't mean that the service quality of a microservices environment can't be effectively monitored and managed. It certainly can but doing so requires a new approach to observability and monitoring, as traditional monitoring solutions were designed and developed years before the concept of microservices existed.

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
– DevOps

Continuous delivery
Crafting code at a slow staccato pace, based on plans made far in advance, no longer works. 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 the ability to switch easily between different development frameworks as their needs and the tools available to them change. The ability to maintain a polyglot pipeline is another crucial factor in achieving agility.

Containers
The introduction of production-ready container platforms, especially Docker, over the past several years affords 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 when service discovery is automated, and orchestrators help manage configurations automatically.

Highly distributed environments
It wasn't that long ago that applications and services were rarely distributed across multiple servers. That architectural change began with the introduction of service-oriented architectures (SOAs) in the mid-2000s, along with the move to cloud-based environments. With micro-services, 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 that's distributed across many servers.

Software-defined everything
One of the contributing factors to the difficulty mapping modern applications 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 microservers 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, the consumption of on-demand compute and storage resources has increased, usually as a pay-as-a-service methodology. Operating network, infrastructure, platforms and workloads as on-demand services makes it faster and easier to scale when needed and optimize ongoing costs but operating all these things as an ephemeral service can create a whole new set of observability, visibility and performance monitoring gaps.

Wider DevOps adoption
The embrace of agile development, coupled with the DevOps movement that followed it, has reshaped the way organizations think about and approach software design and delivery, especially observability and monitoring. Today, constant collaboration and communication between all application stakeholders, from business application owners to the operations and development teams, requires flexibility and widespread consolidation of operational roles, especially during performance optimization and troubleshooting. Designing applications to use microservices helps enable more efficient DevOps operations by facilitating production software that's as agile as the teams that create it.

The end result: A modern application deployment process
Well, that's seven different modern concepts about the new application stack, each one presenting its own challenges for observ-ability and performance monitoring. Some of these concepts were going to happen no matter what, as DevOps teams become more agile and accelerate their delivery pipelines.

The introduction of mass virtualization, broad container adoption and the widespread acceptance of Kubernetes as the de facto standard for container orches-tration, have enabled organizations of all sizes to reach new heights of efficiency by redesigning and operating completely revamped application development, delivery 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 is faster, more reliable and more agile than it was just a few years ago. Organizations that embrace modern software delivery techniques are also better positioned to support new business services and cater in a more nuanced way to precise business needs.

However, both the accelerated invention of new application technologies in recent years and the rapid adoption of these technologies comes with some caveats. The innovations that make agile, microservices-based software delivery possible can make observability, monitoring and service quality management more difficult, increasing the overall difficulty of delivering high-performance, highly scalable applications around the clock. Consider the following ways in which the software delivery practices of today can make observability, monitoring and performance management more difficult.

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 extremely important, extremely complex and extremely difficult. In environments like monolithic or SOA-based applications, interservice dependencies are relatively few in number and steady, making them straightforward and fairly easy to identify. Specifically, it’s easy to tell which applications are running on which servers, as well as which storage and data services are 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. Network routes are adjusted constantly by load balancers in response to shifts in demand.

The ports that individual microservices use to communicate with each other can be changed without notice by orchestrators. For these reasons and more, dynamically mapping the ever-changing dependencies within a micros-ervices environment is extremely complex and difficult—if not impossible—to do manually.

This complexity makes the task of 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 the virtual network that connects a storage array to the rest of an application. In such an event, mapping the intricate dependencies of the storage service is the only way to identify the exact cause of the incident to resolve it.

Continuous delivery
Agile code development features services and architectures developed in small increments. While making software delivery faster and more flexible, the use of a large quantity of smaller services means that changes are introduced at a faster pace. Effective observability and monitoring require the ability to see changes in real time, adjusting monitoring configuration and active service maps 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, the number of monitoring objects one would have in a large 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 within the environment as containers spin up and down, increases dramatically. In fact, a microservice-based application could result in exponential growth in the number of objects to observe and monitor.

Microservice architectures
It's easy to see that in a microservices environment, there are more services to track and measure when monitoring application performance. But it's not enough simply to know whether an application is up or down. It's important to also keep track of the many microservices that make up the applications, monitoring them not just for uptime, but also quality of service, so that reliability issues within a specific microservice can be identified and addressed before they degrade the performance of the entire application.

Everything as a service
The embrace of the everything-as-a-service model means many organizations no longer own the underlying infrastructure on which 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 might not be the best option. Modern observability platforms and tools can provide visibility into the infrastructure no matter where the workloads are located—even if they're managed by third parties.

DevOps and fluid roles
Under the DevOps model, individual team member roles are fluid and shifting. Hence, everyone now has a hand to play in application delivery and service quality management, including engineers and administrators who were not specifically trained in application technologies or in the architecture of the apps. For this reason, DevOps teams look for quality management solutions that are sufficiently intuitive and automated for nonspecialists to use effectively.

04 Use intelligent analysis to thrive with microservices

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

Using intelligent analysis to thrive in the brave new world of microservices:

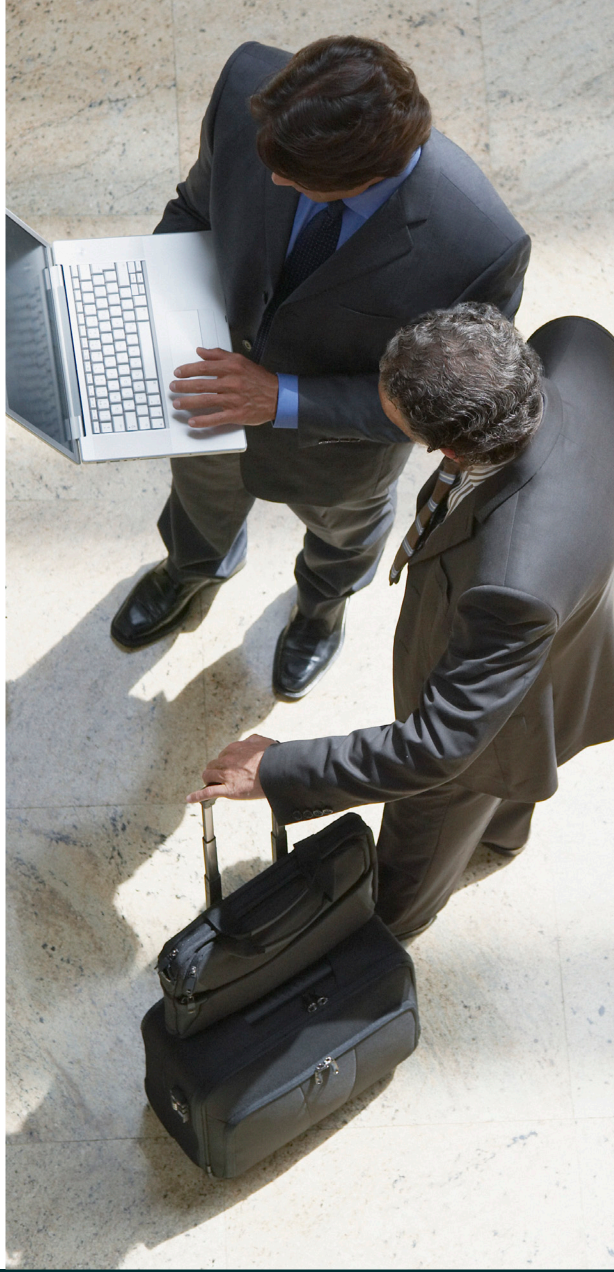
- Separate incidents from noise.
- Make monitoring information actionable.
- Understand incidents and dependencies precisely.
- Share information easily across the team.
- Maintain both real-time visibility and historical visibility.
- Minimize manual configuration.
- Achieve continuous understanding.

All the strategies and capabilities for effective service quality management in a microservices environment, which are outlined here, center on automating workflows and using tools to achieve results on a scale that humans alone can't feasibly produce. Intelligent analysis, machine-assisted learning and automation are at the root of successful service delivery and quality management for microservices.

Three key steps to efficient service quality management are eliminating noise, making monitoring data actionable and precisely understanding incident dependencies.

Separate incidents from noise
Because a microservices environment involves so many moving parts and a high volume of activity, it's important to have the ability to quickly separate incidents that require immediate handling from operational noise, using intelligent analysis that's built into the tools. An incident is any event or issue that negatively impacts the quality of a service or microservice. Incidents happen when a service or microservice fails to deliver within a period of time that's acceptable to the organization.

In contrast, noise is data generated by normal activities. Noise isn't associated with a quality-of-service problem. Separating incidents from noise means being able to glance at an application and service dashboards to recognize which types of data are associated with a potential problem, like a microservice running short of compute resources and which types of data are just natural, unremarkable events, such as



Make monitoring information actionable
For similar reasons, any collected 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, one should look for tools that go beyond collecting data, using intelligent analysis to turn data into actionable information by taking advantage of machine learning and automation. The tools should also be capable of detecting anomalies and mapping service dependencies automatically, allowing DevOps team members to spend their time solving the actual problem identified by the solution, rather than having to invest time in 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 quickly identifying the root cause of the issue and immediately understand how it impacts other parts of the environment, including other applications. Tools that automatically map dependencies can enable this level of insight. Understanding how services interact and how a problem with one component can affect others is essential because, in a fast-moving microservices environment composed of many layers of infrastructure, the source of an incident could lie in many different places—in a host server, in middleware, in application code, in a container, on the network and so on.

Easily share information across the team
In modern DevOps, giving more stakeholders access to real-time performance monitoring and observability data can create a more collaborative and responsive team. There's value in sharing information seamlessly across the entire organization. For example, enabling the Ops team to forward helpful problem information to the assigned programmer is nice, but it would be more valuable if developers simply used the data themselves. Likewise, observability and performance management tools can provide visibility into the full application stack for all team members, enabling a continuous feedback loop. When it comes to information sharing, the central goal of an operations workflow should be the implementation of continuous understanding of events—for all members of the team.

Maintain both real-time visibility and historical visibility
It's important to be able to detect and understand problems in real time as they occur. Real-time reaction requires an ability to separate incidents instantaneously from noise, identify the root of a service problem, and use service quality and performance data to develop an immediate response. It's equally crucial to be able to time-shift—that is, to step back in time by viewing historical data and dependencies at a specified point in the past. Time-shifting enables an understanding of how an incident may have evolved over time, and which initial conditions triggered it, by examining information, such as the initial layout and structure of an application, or previous allocations of containers to hosts. The tools should support both types of scenarios. This knowledge is especially valuable when trying to achieve continuous understanding of both present and past events.

Minimize manual configuration

In a hyperscaled environment, tools that automate monitoring configuration can make a big difference in overall deployment efficiency.

The reason is because, in a microservices environment, updating configuration information or adding services manually takes a great deal of time. This is true both because of the large number of components in a microservices environment and the complex dependencies that can arise from a microservices architecture. Attempting to configure tools for the environment manually undercuts the ability to be agile and scalable and, therefore, defeats much of the purpose of implementing a microservices architecture in the first place.

For that reason, the service quality management strategy should take advantage of tools that automate configuration and service discovery as much as possible. The deployment of new code, new applications, new services, new servers and so on within the environment should be automatically discovered along with their associated dependencies, which is another essential ingredient for achieving continuous discovery and understanding

When monitoring configuration is automated, it can free the operations and development teams to focus their time doing things that humans can do, such as interpreting and acting upon complex health data, rather than the tedium of manually installing and setting up monitoring agents.

As was previously noted, automated anomaly detection, delivered through machine learning and the mapping of service dependencies, also helps minimize the amount of manual configuration that the monitoring workflow requires. When the monitoring tools can detect anomalies and dependencies automatically, the team doesn’t need to spend time configuring its tools to focus on these points of interest.

Achieve continuous understanding

With the constant change in highly dynamic microservices environments, performance and service quality information, and trends are extremely perishable. In some cases, data may cease to be relevant just minutes after it’s identified. That’s why continuous understanding can provide enormous value when implementing workflows for observ-ability, application performance monitoring (APM) and service quality management solutions workflows.

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

05 A new generation of observability

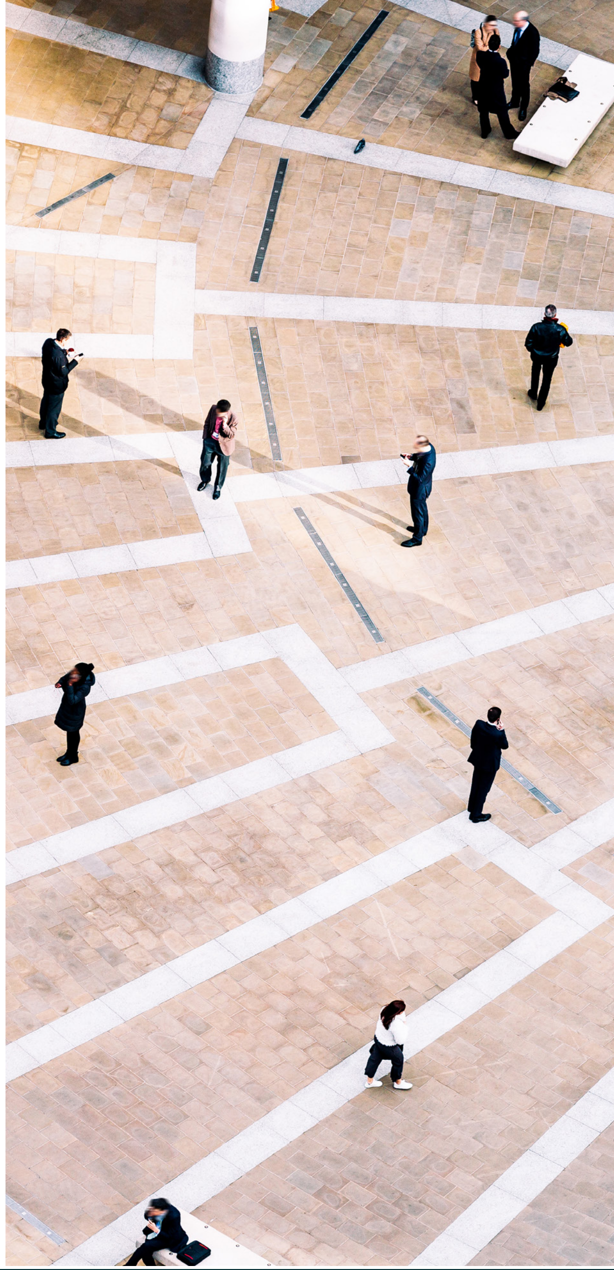
Achieving the complex, flexible, ultra-scalable development and deployment strategy in the most agile development environments requires a completely different approach to observability, monitoring and performance management.

At Instana, while we've always prided ourselves on our automated approaches to monitoring, the next generation of management platforms is enterprise observability. Built on the backbone of a completely automated application monitoring solution, IBM® Observability by Instana APM can help agile organizations deal with the dynamic complexity inherent in microservice applications.

While first-generation and second-generation application monitoring strategies might struggle with the ephemeral nature of cloud-native applications built on orchestrated, containerized microservices, plain old observability solutions aren't necessarily a panacea.

Simple observability tooling can help individual stakeholders, but it won't necessarily have the broader data set that DevOps and IT decision makers involved in modern application operations need to operate applications effectively. That's where enterprise observability comes in.

Automation
Automation is a natural desire in any monitoring solution—but for modern application teams, using agile development methodology and running an always full continuous integration continuous delivery (CICD) pipeline, automation is their life. 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—discovery, mapping, monitoring configuration, alerting, even root cause analysis.



Context
In monolithic applications, transactions always follow the same path through the tech stack and code base, allowing shortcuts like sampling or reverse engineering when deploying monitoring and tracing. But in dynamic microservice applications, the only constant is change.

Understanding how every component or entity interacts with others helps teams understand where to start solving problems. Further, sampling of application response metrics, infrastructure configuration and request tracing is no longer an effective proxy for reality.

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

Intelligent actions
Collecting data without the desire or ability 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.

Since container-based applications are more complex, finding the root cause of problems requires assistance. Enterprise observ-ability 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 to 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 the actionable information we've been discussing. That's why ease of use, which was never a stalwart of APM solutions, is critical to enterprise observability. The more people that can use the information from their observability solution or platform, the better the overall application performance will be. DevOps organizational efficiency will improve, too.

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 not embracing new IT service management solutions that are built on the premise of automated monitoring can range from late nights in the lab to top-line or bottom-line revenue loss. Regular server monitoring and APM monitoring can miss the ability to reflect the reality of microservice environments.

Conversely, manual instrumentation of microservice code can provide some details, but can be less scalable operationally for monitoring performance, especially as software update frequencies increase.

DevOps teams can benefit from a totally new generation of enterprise observability and performance management platforms—platforms that automate setup, discovery, observability, monitoring and tracing. Optimally, these solutions would operate without much, if any, human configuration either up front or after any application updates.

IBM Observability by Instana APM automates application discovery, agent deployment and monitoring configuration across the full microservice technology stack. Instana reduces troubleshooting with automatic root cause analysis and mass correlation of requests, metrics, traces and profiles. The platform identifies the triggering event and most likely cause of each incident. Essentially, Instana does what a human operations team can't—deliver full observability and performance monitoring across modern microservice environments.

07
About Instana, an IBM Company

Instana, an IBM Company, provides an enterprise observability platform with automated APM capabilities to businesses operating complex, modern, cloud-native applications no matter where they reside—on premises or in public and private clouds, including mobile devices or on IBM Z® mainframe computers.

Control modern hybrid applications with Instana's AI-powered discovery of deep contextual dependencies inside hybrid applications. 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, helping DevOps increase efficiency and add value to software delivery pipelines while meeting their service and business-level objectives.

For more information, visit instana.com.

We Invite you to try Instana's automatic monitoring for yourself. You can be monitoring your application in just a few minutes.

Learn more →