IBM Instana

# Application performance management in a containerized world

Achieving visibility in orchestrated environments

## Contents

01

# Executive summary

To say that containers have revolutionized software delivery and deployment might be an understatement. In a few short years, container platforms have evolved from being a novel experimental technology to becoming a core part of the infrastructure of organizations, large and small, across a wide range of industries.

Today, containers are an essential component of the technology stacks at companies as diverse as tiny tech startups to the top of the Fortune 500.

# Explaining container adoption

The biggest drivers of this meteoric rise go together: agility and speed. Containers became hugely popular because they enable organizations to achieve a level of agility that was more difficult using traditional virtualization and software deployment technology. This agility makes it possible to build and deploy new business services faster.

Containerized applications scale quickly, and container environments make it easier for software delivery teams to switch seamlessly between different development frameworks. Containers provide ideal building blocks for creating continuous delivery pipelines, as well as deploying modular, flexible microservices applications.

Containerized environments are also much easier to duplicate, creating tighter parity between development and production, enabling developers and quality assurance (QA) teams to work in environments similar to the actual production application environment.

# The container monitoring challenge

Yet with the revolutionary advantages of containers comes a new type of challenge. Application monitoring, visibility and performance tuning within containerized environments are much more challenging than they were when deploying applications using virtual machines or bare metal servers. Containerized environments are more challenging because they're composed of many components—an orchestrator, registry, runtime and more—whereas traditional environments are less complex. For this reason, organizations that fail to overhaul their approach to monitoring when they adopt containers risk sacrificing quality and efficiency.

If you attempt to monitor your containerized applications and environments in the same way that you monitor traditional applications, you could undercut the core advantages of adopting containers in the first place. Traditional monitoring tools often struggle with the accelerated velocity of containerized operations, especially if it takes too much continuous manual effort to derive understanding from the tools.

Fortunately, there are newer methods for achieving effective monitoring and visibility in a containerized world. Meeting the container application monitoring challenge is easier if you take a fundamentally new approach to monitoring.

↓

Organizations that fail to overhaul their approach to monitoring when they adopt containers risk only harming themselves.

# The visibility challenge of containerization

**Containers host a wide variety of workloads**
Containers can be used to deploy monolithic applications, and to host virtual machines and databases—so the types of workloads that can run in containers varies widely. Typically, data storage is outsourced to permanent servers not hosted inside containers, creating a hybrid environment.

This workload variability makes it difficult to configure traditional monitoring tools to handle containers because every container can be different. These differences from container to container make it difficult to determine which alerting thresholds to set in advance. Configuring and enforcing monitoring policies manually for unpredictable types of workloads can be unfeasible, making monitoring automation more important for continuous, effective performance monitoring.

Additionally, since each container instance can also be different, the same difficulty of pre-setting traditional monitoring thresholds exists at the infrastructure layer. Thus, configuring and manually enforcing monitoring policies is far from feasible.

↓

Monitoring tools must be able to support an unpredictable set of technologies, architectures and configurations.

**Containers are dynamic and unpredictable**
By design, containerized environments are in constant flux. This is one of the reasons containerized environments are helpful in achieving increased agility and development velocity. But traditional monitoring tools can't keep up with these dynamic environments.

For example, if a container initiates a web server instance that handles 150 requests per minute and then shuts down due to a performance issue, a traditional monitoring tool that checks data every 15 minutes will probably not include that server in its list of running systems. And what's more, the problem is not solved, since the technological instance with the problem no longer exists.

**Diverse technologies and languages**
Containerized environments can use a variety of orchestrators, registries and runtimes; they commonly run multiple microservices—written in different types of programming languages—and multiple database servers. If we consider other technology components, such as security, storage or search, the conclusion is that traditional monitoring tools will perform with serious visibility gaps.

**Microservices can be difficult to monitor**
Containerized environments are composed of services that are hosted on clusters of servers. Also, containerized applications usually employ a microservice architecture, where multiple independent services combine to create a complete application. Traditional infrastructure and application monitoring tools were originally designed for monolithic applications that are mapped to static individual servers, and they tend to focus on providing visibility at the language level. These traditional monitoring approaches can break down when you need to support microservices distributed across a large cluster of servers that are composed of multiple languages, many middleware components and multiple database systems.

**Lack of built-in monitoring**
Container platforms tend to include only basic monitoring functionality. They offer a limited amount of performance information about running containers, but this can't ensure efficient monitoring of large-scale production environments.

Adding complexity, containerized environments have more urgent monitoring needs due to the speed and frequency of application updates. And while operating systems can provide significant data about system performance, they don't usually include application-level data.

Traditional virtual machine platforms include more sophisticated monitoring tools, either on their own, or through third-party tools. When containers are in the mix, however, it's much more difficult to collect sufficient monitoring and performance information from the container itself.

**Container orchestrators aren't performance monitoring tools**
Container orchestrators can provide infrastructure management, but that's not performance monitoring, especially at the application layer. Their focus on container and host resources can't offer a reasonable understanding of the performance and quality of microservices, APIs, middleware and applications.

In hybrid environments, which merge containers with other types of infrastructure, the resource usage of applications and servers is not under container orchestrators' control. This situation can create a second large monitoring gap for distributed applications, leaving a large part of your environment open to performance problems.

Orchestrators are excellent provisioning tools, capable of finding and restarting failed containers in some situations, but it's not the same thing as performance monitoring.

# Why container visibility matters

If there's anything IT administrators have learned over the years of platform innovation, it's that maintaining visibility is essential in any type of environment.

In a containerized environment, visibility and monitoring are crucial not only for maintaining application health, but also for maximizing agility and efficiency.

A lack of visibility in a containerized environment leads to performance visibility gaps in ways that don't occur within traditional environments. These gaps create technical and business challenges, some of which are outlined on the following pages.

**Inability to pinpoint the root of a problem**
Without visibility into your containerized application, you probably won't be able to determine whether the cause of a service problem lies on the host server; with the container runtime; poorly operating middleware, running inside or outside of a container; in the application code, inside or outside of a container; or anywhere else.

Traditional monitoring tools won't help you solve these types of problems. DevOps teams will need to spend a lot of time fixing issues manually when dealing with containerized microservices. As that environment grows in size and complexity, small applications can also have hundreds of containerized software parts and thousands of interactions.

**Poor performance metrics**
Without the proper data from inside containers, it's extremely difficult to determine if your containerized applications are meeting your service quality goals. This issue can lead to a bigger risk of not understanding the return on your containerized infrastructure in the first place. Without application performance visibility, it's almost impossible to measure or maximize operational efficiency and development agility.

Having application performance data helps determine whether a new microservice or application deployment is meeting latency goals. A traditional monitoring tool won't be efficient in measuring the performance of a containerized application.

Without this information, it's difficult to see whether the time and money invested in a new deployment is being paid back.

**Hindered scalability**
The scalability to increase or decrease the size or number of instances of an application or microservice in response to fluctuations in demand is a key benefit of using containers. Containers enable easy scalability because they can be started, stopped or migrated in a matter of seconds, compared to minutes for classic virtual machines or days for bare metal servers.

Containers make scaling easy, but intelligent analysis of accurately monitored data is the most effective way to know when to scale. You need to know when an increase in demand for your application merits the scaling up of your deployments, meaning rapidly allocating new containers and hosts.

Just as important, if demand has decreased, you can scale down to avoid paying for under-utilized resources. However, misconfigured container environments can result in problems: insufficient allocation of resources to containers results in poor application performance and scalability issues, while over-allocation leads to wasted money and resources.

Without precise, real-time visibility into container details, including specific information about the platform and workload running in the container, it can become more difficult to effectively scale.

# How not to solve container monitoring issues

Faced with the monitoring and visibility challenges described earlier, organizations that have migrated to containers are sometimes inclined toward solutions that appear to resolve the problem but, in reality, constrain their agility and negate the advantages of containers. The following anti-patterns represent the wrong way to think about how to solve the container visibility challenge.

**Trusting orchestrators to handle service quality**
As discussed previously, orchestrators, by default, aren’t performance monitoring tools. While you should probably use an orchestrator to help provision your containerized environment, you might need a separate performance monitoring solution to provide the deep visibility that can help guarantee service quality and application performance.

**Limiting technology choices**
In some cases, modifying the applications that run inside containers can make monitoring easier. For instance, you might decide to write all your application code in a language that you know your existing monitoring tool supports.

While strategies like these might help simplify the monitoring problems, they defeat one of the purposes of container adoption—increased agility. If you force your development teams to use only certain languages because your monitoring tools don’t support other options, you rob yourself of the flexibility that containers and microservices offer.

**Avoiding monitoring agents**
Some operations (Ops) teams believe that avoiding monitoring tools that require agents to run inside containers helps make monitoring easier because those teams are concerned with the difficulty of deploying and managing an agent within a containerized application. This mindset moves the onus of knowing what to monitor to the development team prior to going live and can lead to a decrease in development productivity. That dilemma is in addition to the same limiting issue of only selecting certain types of technologies, eschewing any benefits from other technology choices.

If you force your development teams to use only certain languages because your monitoring tools don't support other options, you rob yourself of the flexibility that containers and microservices offer.

**Using traditional infrastructure and application monitoring tools**
Put simply, container-based applications have a multitude of ways that they are designed, built, deployed and executed. Because traditional infrastructure and application monitoring tools were built years before the high-speed agile development methodology and containers existed, they're not designed with the speed needed for containerized operations—and can require too much continuous manual effort to derive understanding from the tools.

**Keeping applications monolithic**
Maintaining a monolithic architecture instead of migrating or refactoring your applications to run as microservices can also simplify monitoring needs. However, ignoring the operational and cost benefits of containerized microservice applications can lead to your applications becoming more difficult to maintain and eventually slow down the velocity of your agile development processes.

# Key capabilities for effective application monitoring

Remember, containerized environments can be quite dynamic. To maintain the pace of change, operational teams can use situational awareness and automated real-time visibility as their foundation to understand how applications are performing. It's helpful to also get suggestions and advice on how to optimize the complex technical structures found in today's systems.

Containers and the architectures they enable might present too many differences and blockers to use conventional monitoring tools. To ensure proper service levels are continuously met, a new approach could be helpful—one built from the ground up to deal with the unique characteristics of these dynamic environments.

Here are are some best practices:

**Handle the dynamism by automating everything**
Likely your team doesn't have the time or knowledge needed to manually configure a monitoring tool. To deal with all the changes, automation within monitoring would be helpful. In other words, with zero configuration intervention by a human, the tool should be able to visualize the situation and monitor application performance.

Since continuous integration continuous delivery (CICD) processes are about streamlining and automating the deployment of new services, an automated tool would seem to be the best fit within that environment. Then why would you settle for a monitoring tool that wasn't automatic?

**Be container aware**
A monitoring tool that's container aware can automatically look inside containers and understand the context of the environment in which they are running. This feature helps the tool and its users to keep up with and accommodate the constantly changing state of containerized environments.

**Automatically discover, map and visualize the full app tech stack**
In dynamic environments, it's helpful to understand the structure and dependencies of all your technical components over time, allowing you to analyze application efficiency and performance. In these environments, you need a performance monitoring solution that can automatically discover every application component and map the dependencies between them. If the tool can understand and keep its service maps accurate as changes occur—such as when new software is rolled out and containers are provisioned or deleted—it helps the Ops team understand exactly what's going on in their environment in real time.

**High data fidelity, real-time analysis**
Since microservices and containers can be short-lived and dynamic, look for a monitoring solution that can capture and analyze short-lived performance spikes. These capabilities enable operators to understand when a code rollout caused an issue. If the analysis isn't in near-real time, say less than five seconds, it could lead to updates causing service impacts to end users before the Ops team knows there's a problem. One-second granularity for metrics can be the most effective input for a real-time or near-real time analysis tool.

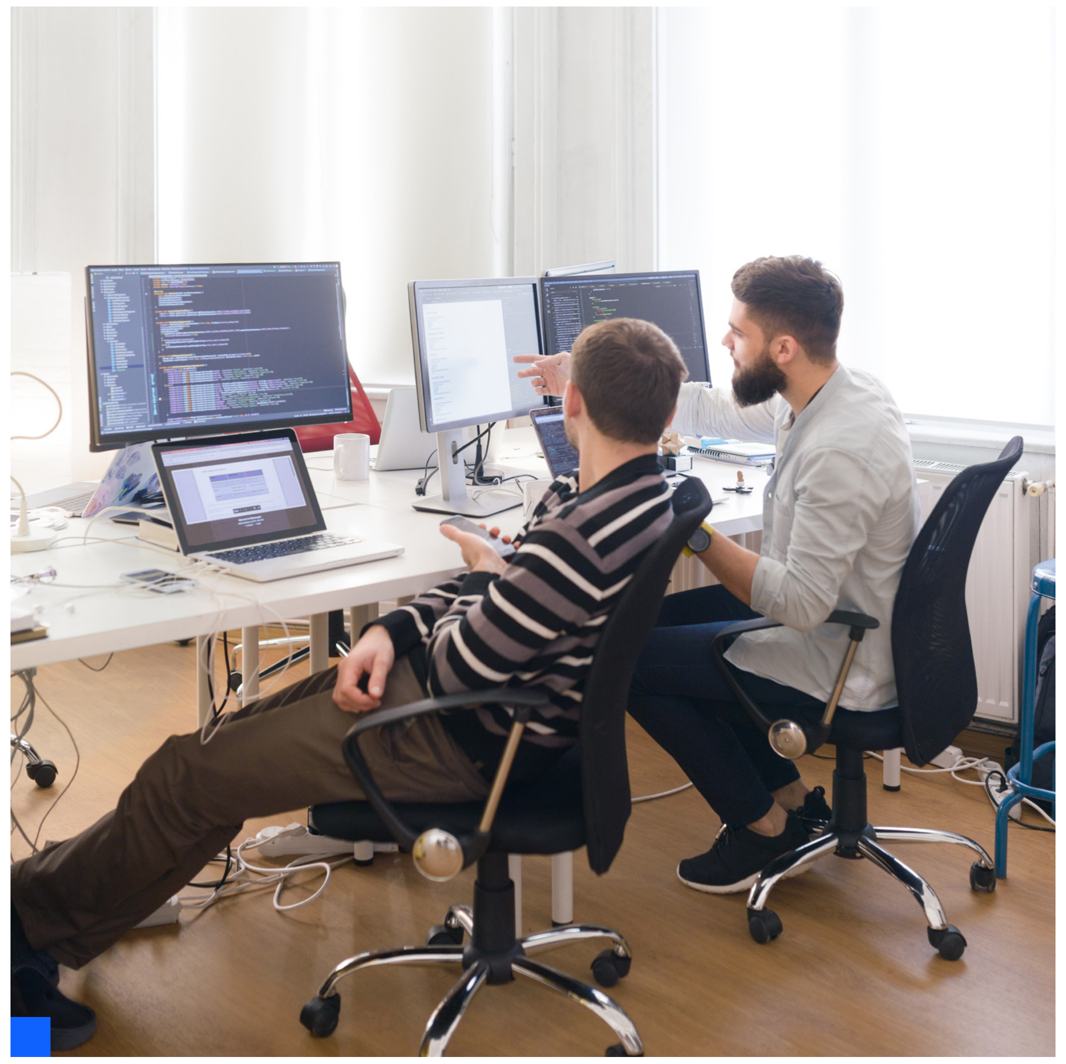
**Capture a distributed trace with every request**
Since containerized application environments are so dynamic and ephemeral, it's possible for every request—even to perform the same function—to follow a different path through the application and infrastructure. Even a completely accurate map might not help troubleshoot issues in such an environment.

Capturing an end-to-end distributed trace helps alleviate this problem by providing the exact details of any call, good or bad. When troubleshooting, the Ops team will have the exact data they need to drill into a slow, problematic request. While sampling traces can be helpful in monolithic environments, this can lead to too many gaps in containerized environments.

**Assist with root cause understanding of microservice issues**
Let's not forget that containerized environments deliver applications that must perform to business expectations. Regardless of your environment complexities, the DevOps team should be able to identify and resolve performance issues as fast as possible. Solutions that can automatically point you to where and why distributed applications break down can be the most effective way to achieve performance goals.

**Stop relying on humans to configure health rules**
As with any process, across application development and monitoring lifecycles, human intervention is prone to errors. With containerized microservices constantly changing, humans can't practically manage application performance on their own. This can lead to large visibility gaps within the application environment.

Look for tools that apply AI and machine learning, advanced statistical analysis, and curated knowledge bases—all of which continue to learn from real-world experiences to automate even more of the solution. It can be helpful if your tool automatically determines which key performance indicators (KPIs) to collect, when measurements indicate service incidents, and the root cause of problems—all without human interaction.



Look for tools that apply AI and machine learning, advanced statistical analysis, and curated knowledge bases—all of which continue to learn from real-world experiences to automate even more of the solution.

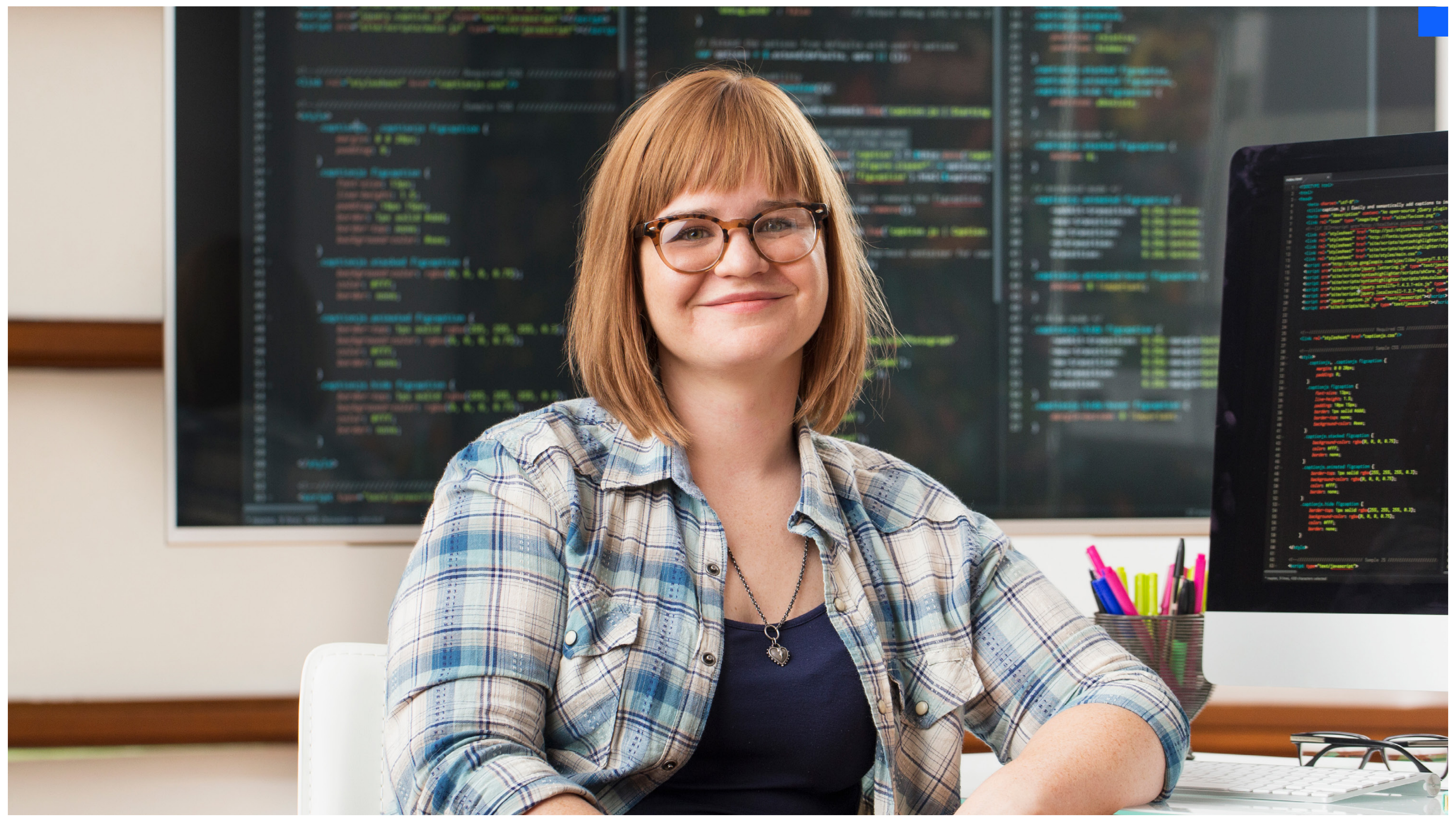
08

# Conclusion

Containers facilitate agility and speed, but they can also create visibility and management challenges. Traditional management tools struggle with these challenges, making it more difficult for IT operations to match the higher velocity of agile development teams. Moreover, the sheer volume of transactions and components make good monitoring even more essential to overall operational success.

The best way to tackle this challenge is to get the required visibility into containerized application environments, especially within the containers. This visibility also includes the ability to see the dependencies between containers and the services that run across them. Because of the ephemeral nature of containerized infrastructure, it's also important to be able to deal with any changes automatically—to ensure that the monitoring tool is continuously accurate, all without any human interaction.

# About IBM Instana



IBM Instana™ provides an enterprise observability platform with automated application performance monitoring 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 IBM zSystems™.

Control modern hybrid applications using IBM Instana for 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 customers 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-level and business-level objectives.

Explore IBM Instana →

IBM Instana free trial →