Gaining Observability in Cloud-Native Applications

INSTANA
an IBM Company

Start here

Contents

01
The history of cloud native and observability

The introduction of various cloud-native technologies has led to massive shifts in application development, delivery and operations. This created a new competitive dynamic: rather than larger companies beating smaller companies, faster companies are outmaneuvering their slower competitors.

The changes to the application tech stack came fast and furious. Cloud environments, public and private, took hold. Virtual servers evolved into containers, and Kubernetes became the orchestration standard. Container adoption exploded, and workloads began moving to serverless environments. Containers evolved into serverless workloads, all held together by a convergence of resource orchestration. Dev teams became more agile. As software and infrastructure blurred, two competing constants emerged:

– Application performance monitoring (APM) became an important contributor to business success.
– It was difficult to gain visibility into modern applications using traditional APM methodologies.

Thus, the concept of observability—in many forms—emerged as organizations grappled with the need to accelerate development and delivery while maintaining the levels of visibility IT operations teams had come to expect from APM tools. But observability has its own challenges, including many of the same roadblocks faced by traditional monitoring tools. Speedy, flexible application delivery organizations (Dev, Ops, DevOps, SREs and so on) want to use the entire set of cloud-native technologies. To do that, they would benefit from a comprehensive approach to observability that takes advantage of the best practices of developer-driven visibility and modern APM solutions.

Ever since Docker created a containerization standard that would be adopted worldwide, IT workers have debated what a cloud-native application means. Is it the new gold standard for operating applications, or just the latest buzzword?

Kubernetes became a leading orchestration tool for containers just two years later, the same year VMTurbo announced support for platform as a service (PaaS). By 2016, Google, Amazon and Microsoft developed cloud hosting platforms. Serverless services became ubiquitous, enabling organizations to use containers without needing to manage the layers within. In 2018 and 2019, Google Cloud Platform, Amazon Web Services and Microsoft Azure automated and eased the migration and orchestration of applications to the cloud.

During this period, the concept of cloud native had myriad definitions. The basic definition was an application built to leverage cloud-based technologies. To some, it meant using containers. Others defined it as using some level of PaaS infrastructure. Still others simply thought of it as any workload running in a cloud.

In 2019, all these definitions fell by the wayside as a de facto definition emerged and stuck:

Distributed, orchestrated, containerized micro-services operating in a wide array of cloud-based architectures. Cloud native breaks down applications into microservices, small individual parts that still work together, enabling faster deployment and orchestration.

You may have heard of cloud-based applications too. Cloud-based applications have been retrofitted for operation in the cloud, whereas cloud-native applications were developed for the cloud. But whose cloud? Companies are putting their applications in multicloud environments for more power and greater redundancy. Hybrid clouds add to the complexity, with a combination of public and private clouds. Add a layer of on-premises hosting, and modern technology stacks can spin heads.

Monitoring the performance of cloud-native applications can prove challenging because it can be difficult for traditional APM and observability tools to see inside cloud-native elements such as containers as easily as they see into other application stacks. In contrast, cloud-native observability platforms tend to track all the metrics, event traces and logs across the full stack to provide broader visibility.

Cloud-based applications have been retrofitted for operation in the cloud.

Cloud-native applications were developed for the cloud.

The technologies that make up this new cloud-native stack throw some unique roadblocks at traditional monitoring tools:

- **Lack of backward compatibility**
  Containers are purposely built to contain only required services to keep them lightweight, which means older instrumentation engines may not have access to required services needed to work.

- **Serverless elimination**
  Serverless architectures take the service elimination ideal to an even greater level by taking the servers from local control to the cloud platform, leading to the same service availability issues.

- **Orchestration information correlation**
  Kubernetes blurs the lines between software and infrastructure to the point that existing tools may have to use multiple agents just to extract basic data, which would eliminate the ability to directly correlate information.

- **Accelerating development and deployment rates**
  As changes occur in the application environment due to the ephemeral nature of containers, it can be difficult for traditional monitoring tools to reach their accustomed level of visibility.

As APM tools proved insufficient with cloud-native technology stacks, alternatives for application observability emerged thanks to other factors:

- Developers were being asked to take a larger role in assuring and operating production workloads, especially from the perspective of application performance.
- Cloud-native tech stacks enabled accelerated development and application pipelines, meaning more pieces could be operating with visibility gaps from existing tools.

Similar to the moving target of defining cloud native, application observability brought the issue of multiple definitions, depending on organizational biases (Dev versus Ops), available skills, resource availability and vendor participation.

Observability provides external indicators of the health and status of applications inside cloud-native elements such as containers, microservices, orchestration tools such as Kubernetes, and serverless, from a performance perspective.

02 Cloud-native technologies create observability challenges

From black boxes—the hidden internal workings of a product or program—to massively short-lived deployments, the cloud-native stack contains numerous challenges for observability and APM tools:

- Traditional monitoring tools are ill-equipped to handle distributed microservice environments.
- Containers host a wide variety of workloads.
- There is a lack of built-in observability and monitoring.
- Kubernetes is not a performance monitoring tool.

Traditional monitoring tools can't handle distributed microservice environments
Environments for container-based applications tend to be dynamic, distributed microservices, deploying those services across clusters of servers that are also dynamic. One complete application could cross different kinds of infrastructures, platforms, languages and technologies. Requests between those services accomplish the business processing.

It can be challenging to visualize and understand distributed microservice environments using traditional infrastructure and application monitoring tools. Their base technology was built when the world was dominated by monolithic applications mapped to static individual servers, and they tend to focus on providing visibility at the language level. These traditional monitoring approaches can break down in the face of microservices distributed across a large cluster of servers composed of multiple languages, middleware components and database systems.

Containers host a wide variety of workloads
There is no single type of workload associated with containers. Some containers deploy monolithic applications, such as an Apache HTTP server. Some host Java virtual machines. Others host databases, which makes monitoring even more complicated. Why? Persistent data storage is typically outsourced on permanent servers, not hosted inside containers, creating a hybrid environment.

From a monitoring perspective, the variety of workloads that might run inside a containerized environment can be challenging because your monitoring tools must support an unpredictable set of technologies, architectures and configurations.

This workload variability can make it difficult to configure traditional monitoring tools to handle containers because every container is different, and it can be hard to determine which thresholds to set in advance for each one. In addition, monitoring tools must be automated because it's not feasible to manually configure and enforce monitoring policies for unpredictable types of workloads.

Containers require elevated monitoring
Container platforms typically include only basic monitoring functionality. For example, the Docker stats command provides a limited amount of performance information about running containers, but the stats tool is insufficient for monitoring large-scale production environments. The lack of built-in monitoring beyond basic data separates containers from other application infrastructure technologies.

Ironically, this observability and monitoring challenge occurs in an environment that has elevated monitoring needs due to the speed and frequency of application updates. Similarly, operating systems provide significant data about system performance, but they don’t include application-level data. Even advanced virtual machine platforms such as VMware, which include relatively sophisticated monitoring tools, fail to provide application layer data or distributed tracing.

Kubernetes is not a performance monitoring tool
Container orchestrators such as Kubernetes are great for organizing, provisioning and managing the deployment of their production container environments and applications. There’s hardly a containerized application left that isn’t orchestrated, whether by Kubernetes, D2iQ or some other orchestration engine.

But orchestrators are neither intended nor designed for sophisticated monitoring, and although they have tremendous focus on container and host resources, orchestration management includes no aspect of user experience or application performance.

Furthermore, if either the applications or the infrastructure is set up in a hybrid environment, orchestrators can actually get fooled into thinking everything is OK when, in fact, the overall system is crashing or hung up. This is because container orchestrators can only manage things that they are aware of—which mean the infrastructure and services running in containers. In a hybrid environment, container orchestrators, by definition, don’t monitor resource usage of any other resources.

This creates a second major observability and monitoring gap for distributed applications. And that can leave a large part of your environment at risk for performance problems. So remember, while orchestrators are excellent provisioning tools capable of finding and restarting failed containers, don’t confuse these management tools with performance monitoring or observability.

03 Critical characteristics of observability for cloud-native environments

Visibility is the heartbeat of cloud-native observability platforms. The ability to monitor containers from within is a critical feature for dynamic microservices-based architectures in which tracing and dependency maps can be a convoluted, indecipherable mess.

Even with the ephemeral and sometimes incomplete nature of containers, cloud-native observability provides actionable insights to turn your people from passive observers into active participants who can resolve and even prevent issues. Agents knowledgeable in containerized microservices can map the resources in an IT architecture, even if those resources are barely related and constantly changing.

Other critical characteristics include:

- **All the data**
  DevOps must have complete analysis and understand exactly how, when and where problems exist. The only way to achieve this is to capture a distributed trace of every request—no gaps, no sampling and no proxies. Every piece of infrastructure, platform and line of code deployed into the containerized environment must support tracing.

- **Cloud-native deployment**
  Cloud-native observability tooling integrates seamlessly into cloud-native application environments, fully automated deployments and instrumentation processes.

- **Root cause analysis**
  No matter how complex your environment, DevOps teams must identify and resolve performance issues as fast as possible. An observability solution must be able to automatically point you to where and why distributed applications break down.

04 Risks in not adopting observability for cloud native

It's tempting to wait on deploying observability for cloud-native applications. IT workers must learn a new platform. Maybe you don't see the urgency because your APM solution isn't presenting an alarming number of errors.

But if your traditional monitoring solution is missing errors, transaction latency and opportunities for optimization, you might not be aware that you need an enterprise observability platform. Delay comes with risk:

- An undetected error could result in customer-facing service degradation or an outage.
- Delays from time of incident detection to the time of discovery by a human could be the difference between incident prevention and major incident management.
- Your monitoring tool could find an issue but not provide enough contextual information for you to react.

All these scenarios represent potential damage to your reputation, loss of revenue due to downtime, and customer attrition.

05 Benefits of observability for cloud-native applications

Of course, the primary benefit of a more comprehensive monitoring solution is to find more stuff. With observability, transparent and observable systems help enable engineers including SREs to prevent spending time on continuous monitoring and get back to the jobs they are paid for: producing new product features for customers.

Faster continuous integration/continuous deployment (CI/CD) processes, less wasted work, fewer errors and happy customers save time and money. But there's much more:

Reduce mean time to repair (MTTR)
Event correlation can be the difference between triaging an incident for hours and resolving an impending issue before it can even happen.

Shift left
The earlier your engineers and developers can become aware of an issue, the more proactive and less reactive they can be, and the less impact every issue will have.

Reduce CI/CD impact
Observability makes it easier to track the success of a canary release or a blue/green deployment in progress. Once unsuccessful code is released into production, damage is done.

Healthier systems
The natural output of an observability tool in a cloud-native environment with DevOps means systems with fewer errors and more efficient processes.

Happier customers
As a customer, how long would you continue to pay for an application or service that has errors, responds slowly, experiences downtime and threatens your business?

06

Introducing enterprise observability

Although observability can deliver the signals you need to track application performance, getting to a truly comprehensive understanding of your entire system is better served by enterprise observability.

With enterprise observability, you can do more than just monitor individual systems. You can contextualize data about those systems and correlate interactions between discrete applications and systems across your entire IT environment. The foundations of enterprise observability are automation, context, intelligent action and ease of use.

More specifically, enterprise observability is founded on several key practices and principles:

- **Systematic optimization**
  Enterprise observability focuses not on managing the health and performance of individual applications or systems but on optimizing the entire IT environment. To do this, an observability platform must be able to map and contextualize interactions between all the resources that exist within a business's IT architecture, even if those resources are loosely coupled and constantly changing.

- **Complete contextualization**
  To achieve enterprise observability, every unit of observability data must be delivered with complete context. Teams cannot rely on sampling to make informed guesses about what is happening; they need end-to-end tracing and contextualization of every unit of work.

- **Cloud-native deployment**
  Enterprise observability tooling must be able to integrate seamlessly into the cloud-native application environments that it supports. The deployment and instrumentation processes are fully automated.

- **Comprehensive support for data ingestion**
  Modern enterprise application environments expose data in various ways. Enterprise observability tools must support all of them. Whether applications expose data as standard output or conventional logs or through open source monitoring APIs such as OpenTracing, enterprise observability means being able to ingest and contextualize every data source.

- **Observability across the pipeline**
  Understanding what's happening within production application environments is not enough to achieve enterprise observability. Instead, teams must be able to monitor and contextualize application behavior starting at the beginning of the CI/CD pipeline and continuing through to deployment. You shouldn't have to wait until a new application release is in production to be able to understand how it interacts with other systems and optimize its behavior.

Automation
When new code is deployed or system changes are made, Enterprise Observability automatically and continuously discovers changes and provides immediate feedback.

Context
Enterprise Observability explains how every application component and service interrelates with every other component and service to optimize performance and availability of resources.

Intelligent action
When changes occur, Enterprise Observability proactively provides deep analysis with context and suggests steps for system optimizations.

Ease of use
As organizations roll out agile development processes, increase the frequency of their application updates and fill their CI/CD pipelines, more stakeholders require actionable information.

07

About Instana, an IBM Company

Instana, an IBM Company, 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 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 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.

Learn more →