Gaining Observability in Cloud-Native Applications
# Table of Contents

- **03** Introduction
- **04** Cloud-Native Comes of Age
- **06** A Standardized Definition of Observability
- **07** Cloud-Native Technology and Infrastructure Create Observability Challenges
- **11** Critical Characteristics of Cloud-Native Observability
- **12** Risks in Not Adopting Cloud-Native Observability
- **13** Benefits of Cloud-Native Observability
- **14** From Observability to Enterprise Observability
- **16** About Instana
Since 2015, the introduction of various Cloud-Native technologies has led to massive shifts in application development, delivery, and operations. This has led to a new competitive dynamic — rather than larger companies beating smaller companies, faster companies are beating their slower competitors.

The changes to the application tech stack came fast and furious — public and hybrid cloud environments 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. The concept of DevOps matured as the operational response to speed up pipelines.

As software and infrastructure blurred, two competing constants emerged:

- Monitoring application performance in production is an absolute requirement to ensure business success
- Legacy Application Performance Monitoring (APM) tools face challenges gaining visibility into modern applications

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 Ops teams had come to expect from APM tools. But Observability has its own challenges, including many of the same roadblocks faced by the legacy APM tools. Speedy, flexible application delivery organizations (Dev, Ops, DevOps, SREs, etc.) needed the entire world of cloud-native technologies. To get it, they needed a comprehensive approach to Observability that leverages the best practices of developer-driven visibility and modern APM solutions.
Since 2013, when Docker created a containerization standard that would be adopted worldwide, information technology 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 the leading orchestration tool for containers just two years later, the same year VMTurbo announced support for PaaS. By 2016, Google, Amazon, and Microsoft developed cloud hosting platforms that continue to dominate the market. In 2017, 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 made it easier and more automated to migrate applications to the cloud and orchestrate them.

During this period, the concept of Cloud-Native had a myriad definitions. The basic definition was an application built to leverage cloud-based technologies. To some, it meant leveraging 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 microservices 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. Where cloud-based applications have been retrofitted for operation in the cloud, “Cloud-Native” applications were developed for the cloud. But whose cloud? Companies are putting their applications in multi-cloud 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-premise hosting, and modern technology stacks can spin heads.

Monitoring the performance of Cloud-Native applications has proven challenging because legacy APM and Observability tools cannot easily see inside cloud-native elements like containers. By contrast, Cloud-Native Observability platforms track all the metrics, event traces, and logs to provide that visibility.

Technology throws some unique roadblocks at legacy 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 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.

- **Orchestration information correlation**: Kubernetes blurs the lines between software and infrastructure to the point that legacy tools had to use multiple agents just to extract basic data, eliminating the ability to directly correlate information.

- **Challenges keeping up**: Ephemeral operational realities required a level of programmatic modeling and real-time detection to keep up with the reality of operating tech stacks.
A Standardized Definition of Observability

As application performance monitoring tools struggled 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 from a performance perspective
- Cloud-Native tech stacks enabled accelerated development and application pipelines, meaning more pieces were operating with visibility gaps from legacy tools

But like the overall concept of Cloud-Native, Application Observability faced the issue of multiple definitions, depending on organizational biases (Dev vs. Ops), available skills, resource availability, and vendor participation.

Observability provides external indicators of the health and status of applications inside Cloud-Native elements like containers, microservices, orchestration tools like Kubernetes, and serverless.
Cloud-Native Technologies Create Observability Challenges

From black boxes to massively ephemeral deployments, the Cloud-Native stack contains a number of challenges for observability and APM tools:

- Traditional monitoring tools struggle with distributed microservice environments
- Containers host a wide variety of workloads
- Lack of built-in Observability or monitoring
- Kubernetes is not a performance monitoring tool
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.

Traditional infrastructure and application monitoring tools struggle to visualize and understand distributed microservice environments. Their base technology was designed to monitor monolithic applications that are mapped to static individual servers, and they focus on providing visibility at the language level. These traditional monitoring approaches break down when faced with microservices distributed across a large cluster of servers (composed of multiple languages, many middleware components, and multiple database systems).
Containers host a wide variety of workloads

There is no single type of workload associated with containers. Some containers deploy monolithic applications, like 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 is challenging because your monitoring tools must support an unpredictable set of technologies, architectures, and configurations.

This workload variability makes it difficult to configure traditional monitoring tools to handle containers because every container is different, and it is hard to determine which thresholds to set in advance for each one. In addition, monitoring tools need to be automated because configuring and enforcing monitoring policies manually for unpredictable types of workloads is not feasible.
Containers have elevated monitoring needs

Container platforms 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 like 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 like 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 while they have tremendous focus on container and host resources, there is no aspect of user experience or application performance included in orchestration management.

Furthermore, if either the applications or the infrastructure is set up in a hybrid environment, orchestrators can actually get fooled into thinking everything is okay 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 means the infrastructure and services running in containers. In a hybrid environment, container orchestrators cannot help to monitor resource usage of any other resources.

This creates a second major Observability and monitoring gap for distributed applications. And that leaves a large part of your environment prone to 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.
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 where tracing and dependency maps are 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. Containerized microservices-based agents 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, every platform, and every 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.
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 even 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.
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 enable engineers and SREs to prevent spending time with eyes on glass, reduce toil, and get back to the jobs they are paid for: producing new product features for customers.

Faster CI/CD processes, less wasted work, fewer errors, and happy customers save time and money. But there’s much more.

- **Reduce 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 is 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?
Introducing Enterprise Observability

We’ve established that Observability can deliver the signals you need to track application performance. But for a truly comprehensive understanding of your entire system, you need what we call 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, and intelligent action.

**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 optimization.

**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.
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 rather on optimizing the entire IT environment. To do this, an Observability platform must be able to map and contextualize interactions between all of 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 a variety of ways. Enterprise Observability tools must support all of them. Whether applications expose data as standard output, conventional logs, or through open source monitoring APIs like OpenTracing, Enterprise Observability means being able to ingest and contextualize every data source.

- **Observability across the pipeline:**
  Understanding what is 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 should not 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.
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 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](http://instana.com).

**Sign Up for Your Trial Today**

Stan
Your Intelligent
DevOps Assistant