INSTANA
an IBM Company

Serverless Observability

Table of Contents

A Comprehensive Guide to Serverless Observability

Since the introduction of AWS Lambda, serverless computing has grown into one of the core building blocks of cloud-native infrastructure. By allowing organizations to run resource-intensive application code, on-demand, in a cloud environment, and pay only for computing when the code is actually running, serverless functions have unlocked new opportunities for optimizing application performance, availability and cost-efficiency.

In each respect, serverless computing delivers critical benefits that are not available from traditional physical servers or virtual machines, which incur costs constantly (even when they are sitting idle) and have limited resource allocations for handling high-intensity workloads.

At the same time, the serverless paradigm has created a series of new challenges for the teams tasked with monitoring and managing serverless functions. Because there are certain aspects of serverless architectures that are fundamentally different from conventional application deployment models, obtaining visibility into serverless environments, as well as the functions themselves, is more difficult in some respects.

With that reality in mind, this eBook offers a comprehensive look at the challenges of gaining observability and monitoring the performance of serverless application functions, as well as providing tips and best practices for solving them. We'll discuss the reasons that serverless observability is difficult and looking at why traditional Application Performance Monitoring (APM) tools struggle when attempting to monitor serverless workloads. We'll conclude with a look at the capabilities necessary to effectively gain observability into serverless workloads and to fully realize the performance, cost and reliability advantages that serverless technology can unlock.

Serverless Observability challenges

Although the code inside a serverless function may not differ fundamentally from that inside a traditional application, the architecture, host environment and deployment patterns associated with serverless functions are profoundly different. These differences crate serious challenges when it comes to observability, performance monitoring. Tracing and maintaining visibility into serverless deployments.

Vendor-Specific Serverless Services

A conventional application operates in the same essential manner regardless of where it is hosted. An NGINX Web server or WordPress instance hosted on AWS EC2 virtual server works – and, by extension, can be monitored – in the same way as it would if it were hosted on the Azure Virtual Machines service.

When it comes to serverless, however, workload cross-compatibility breaks down. Although all cloud-based serverless services deliver the same basic features, each one is configured in unique, vendor-specific ways. For example, AWS Lambda does not support all of the same programming languages as Azure Functions and Google Cloud Functions. Each serverless service also comes with different environment configurations, which for the most part cannot be modified by end-users. Configurations become even more diverse and inconsistent if you add on-prem serverless frameworks, such as OpenFaaS, into the mix.

For these reasons, observability methods for monitoring serverless functions at the level of the service itself leads to different approaches that aren't necessarily portable (in fact, it's probably not). Further, it is likely dependent on the particular configuration provided by the cloud infrastructure vendor. For teams wary of lock-in and the inefficiency of building new monitoring processes and custom observability from scratch every time they migrate to a different serverless platform, this is a serious limitation.

Serverless functions come in many languages

Similarly, serverless functions can be written in a variety of different languages. And, as noted above, some serverless services support different languages than others. In some cases, serverless platforms require functions written in one language to be "wrapped" in another in order to execute them, adding yet another layer of complexity (or another hurdle to complete observability).

What all of this means for monitoring is that there is no easy or consistent way to monitor serverless functions by hooking into specific programming languages.

Serverless is one piece of larger deployments

It is rare to deploy a workload that is composed solely of serverless functions. In many cases, serverless functions comprise one part of a larger application. For example, most of the components of a Web application might be hosted using traditional virtual machines and databases, while serverless functions are used to handle certain resource-intensive processes (such as resizing images or performing optical character recognition), on-demand.

For this reason, effectively observing serverless requires not only an understanding of what is happening within functions themselves, but also mapping that data to the performance of the larger application. Observability and monitoring tools must be able to interpret the complex relationships and dependencies between each serverless function and the various non-serverless microservices that power the rest of the workload.

Lack of control over serverless environments

Serverless is so-called because it eliminates the need for the teams who deploy serverless functions to set up or manage the servers that host them. The provider of the serverless server delivers a prebuilt, preconfigured environment in which end-users can quickly deploy and execute functions. This is one of the features that makes serverless so valuable.

From the observability and monitoring perspective, though, the operators' inability to access or modify the host environment represents a significant challenge. Most critically, there is no way to modify the way that the servers hosting their serverless functions log data or other metrics. Most serverless services do provide facilities for forwarding log data to other cloud services (such as AWS CloudWatch in the case of Lambda), but there is little or no ability for teams to customize the way that data is generated or structured. Nor can they deploy monitoring agents on host servers in a conventional way to collect and aggregate metrics.

Highly dynamic functions

Serverless functions are often deployed as part of continuous delivery pipelines. Updates are rolled out on an ongoing basis, meaning that new versions of functions are frequently deployed. Static approaches to serverless observability and monitoring can't keep up with this rapid change. If monitoring tools must be manually mapped to a specific serverless function deployment, they must be reconfigured manually whenever the functions are updated. This dependency means that observability not only can get in the way of continuous delivery, but can actually bring it CI/CD pipeline grinding to a halt. Ot worse, updates to serverless functions are not properly monitored because the observability tools aren't configured for the updated deployment.

Too many functions

The final challenge in serverless observability is the sheer number of functions that serverless workloads entail. Traditional APM / Application Monitoring tools just can't handle the changes. Although there is no minimum number of functions required to use a serverless service, teams that leverage serverless computing often deploy a dozen or more functions at the same time. They introduce new functions and retire old ones on a constant basis.

Attaing observability on this scale with conventional and/or manual approaches is not just difficult, it is practically impossible. It WOULD REQUIRE (if it were possible) tremendous ongoing and continuous effort and time commitment on the part of developers and admins, undercutting the ability of teams to continue operating their applications at scale.

Why legacy APM is not enough for serverless monitoring

One critical nuance of legacy APM tools is that they were all – ALL – designed before the Cloud-Native era was even a futuristic though. While some of these tools may include some aspect of observability for serverless environments (such as showing CloudWatch metrics), they struggle to provide the same level of visibility that their users are accustomed to.

One reason for this is the lack of ability to map and interpret the complex application architectures, of which serverless functions are a part. Legacy APM tools were designed for monitoring monoliths, and they are ill-equipped to understand inter-service dependencies or distinguish normal activity from anomalies within highly dynamic environments.

Legacy APM tools are also usually vendor-specific when it comes to serverless monitoring. They may support the monitoring of serverless functions on certain cloud services, like Lambda and perhaps Azure Functions. But they cannot really observe / monitor serverless functions in a vendor-agnostic way, and they are not portable from one cloud to another.

Ultimately, legacy APM solutions were designed for fundamentally different types of infrastructure. If they can work with serverless functions at all, it is because facilities for monitoring serverless under certain conditions were grafted onto tools that were originally created for different purposes. They simply lack cloud-native monitoring functionality.

Selecting an Observability Platform to Enable Serverless Monitoring

Because legacy APM solutions are inadequate for serverless monitoring, many operations, SRE, DevOps and development teams try their hand at manual observability methods. While each represents unique challenges, at the end of the day, Ops teams dealing with serverless workloads want an automatic way to observe performance data and monitor applications.

Following is a summary of the key functionalities required in an APM solution to monitor serverless effectively:

- Comprehensive Distributed Tracing and Analytics
- Automatic Real-time Dependency Mapping
- Auto-Discovery and Monitoring Configuration
- Cloud-Agnostic Serverless Monitoring
- Real-time and Historical Monitoring, Visualization and Reporting

Comprehensive tracing and analytics

There are two foundations to effective observability in cloud-native environments: Tracing and analytics.

For effective Observability and performance monitoring for Serverless functions, the observability platform must be able to capture distributed traces that run through serverless and traditional workloads – AND those traces must be part of any analytics engine. The key aspect of this combination, though, is ACROSS THE ENTIRE APPLICATION!

In other words, your Observability platform should have the ability to trace and analyze individual application requests across every component of the application, both serverless functions and hosted / virtual services. At the same time, they provide deep analytics on aggregate metrics collected from the application as a whole, in addition to individual trace(s). Only through comprehensive tracing and analytics can your Observability platform guarantee the visibility into all layers of a complex application and give teams multiple vantage points for gaining the insight they need to address performance, availability or cost-optimization issues.

Dependency Mapping

Because serverless functions depend on each other, as well as other application components that are external to the serverless environment, automatically finding these dependencies and mapping them are critical capabilities. Observability and/or APM tools that are designed only to monitor each part of the application individually, without understanding how relationships between each part form a larger whole, are insufficient for production monitoring.

Instead, performance and cost optimization in a serverless workload requires the ability to determine how a problem with one serverless function impacts other functions or services within the application – and of course, the application as a whole (and any end users).

Auto-Discovery

If monitoring instrumentation has to be configured manually each time a new serverless function is deployed, or a new version of an existing function is released, it is virtually impossible for monitoring tools to keep pace with continuous delivery chains, at scale. For that reason, serverless Observability solutions must be able to discover new deployments and updates automatically – and delivering the appropriate levels of observability and monitoring automatically, without any human intervention, special developer instrumentation, or configuration.

Cloud-agnostic serverless monitoring

As explained above, observability platforms that can work only with specific certain cloud services lack portability. They depend on specific services or configurations and their monitor APIs at the environment level. A more flexible and portable approach is to choose an Observability Platform that can monitor functions regardless of which serverless service hosts them. This requires the ability to perform tracing and analytics within functions themselves, rather than hooking into the host environment.

Real-Time and Historical Visualization

Finally, successful serverless observability requires the ability to visualize analytics and traces clearly. Visualization is critical for enabling Dev and Ops teams to make sense of and act on monitoring data. Without rich visualizations, teams are hard-pressed to interpret the complex, rapidly-changing metrics generated by serverless functions and the applications of which they are a part.

To be most effective, Observability and monitoring tools must provide visualization for both real-time and historical data. Real-time visualizations give teams visibility into the application as it currently exists, while historical visualizations enable them to research and issue or gain crucial historical context when troubleshooting a problem.

The Next Generation of Observability

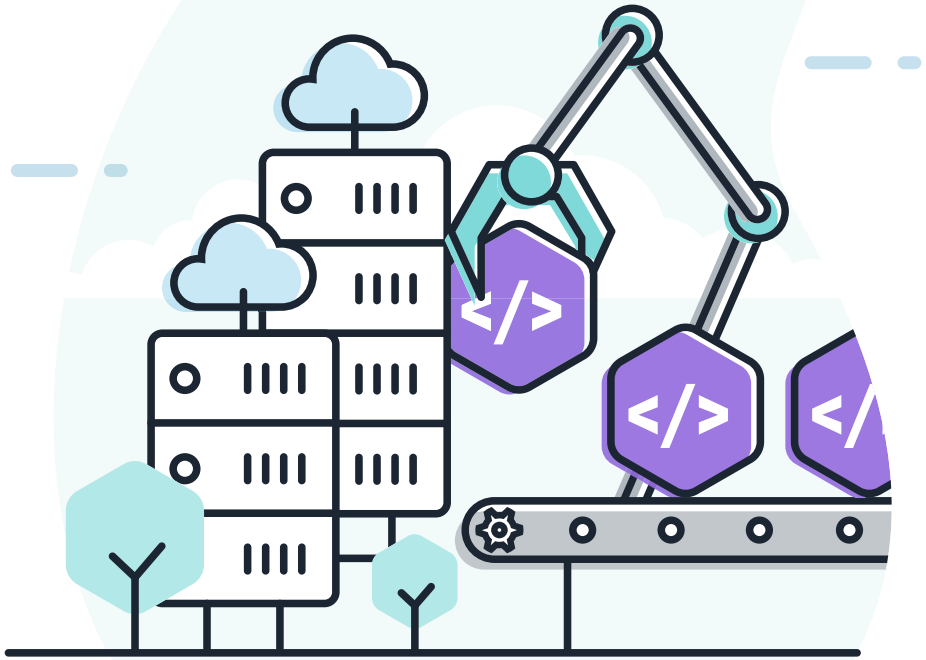
Achieving the complex, flexible, ultra-scalable development and deployment strategy in the most agile development environments requires a completely different approach to observability, monitoring and performance management.

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, Instana's Enterprise Observability Platform is just what agile organizations need to deal with the dynamic complexity inherent in microservice applications.

Traditional monitoring can't handle the ephemeral nature of microservices, containers, orchestration or other critical Cloud-native infrastructure and application environments. Meanwhile, simple Observability tooling can help individual stakeholders, but not the broad set of Dev, Ops and IT Decision Makers involved in modern application operations. That's where Enterprise Observability is required.

Automation

Automation is a natural desire in any monitoring solution – but for modern application teams using agile development methodology and running an always-full CI/CD pipeline, automation is their life. But 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, each transaction runs through the same stack and code base as every other transaction, allowing certain functional shortcuts like sampling or reverse engineering when deploying monitoring and tracing; but in dynamic microservice applications, the only constant is change.

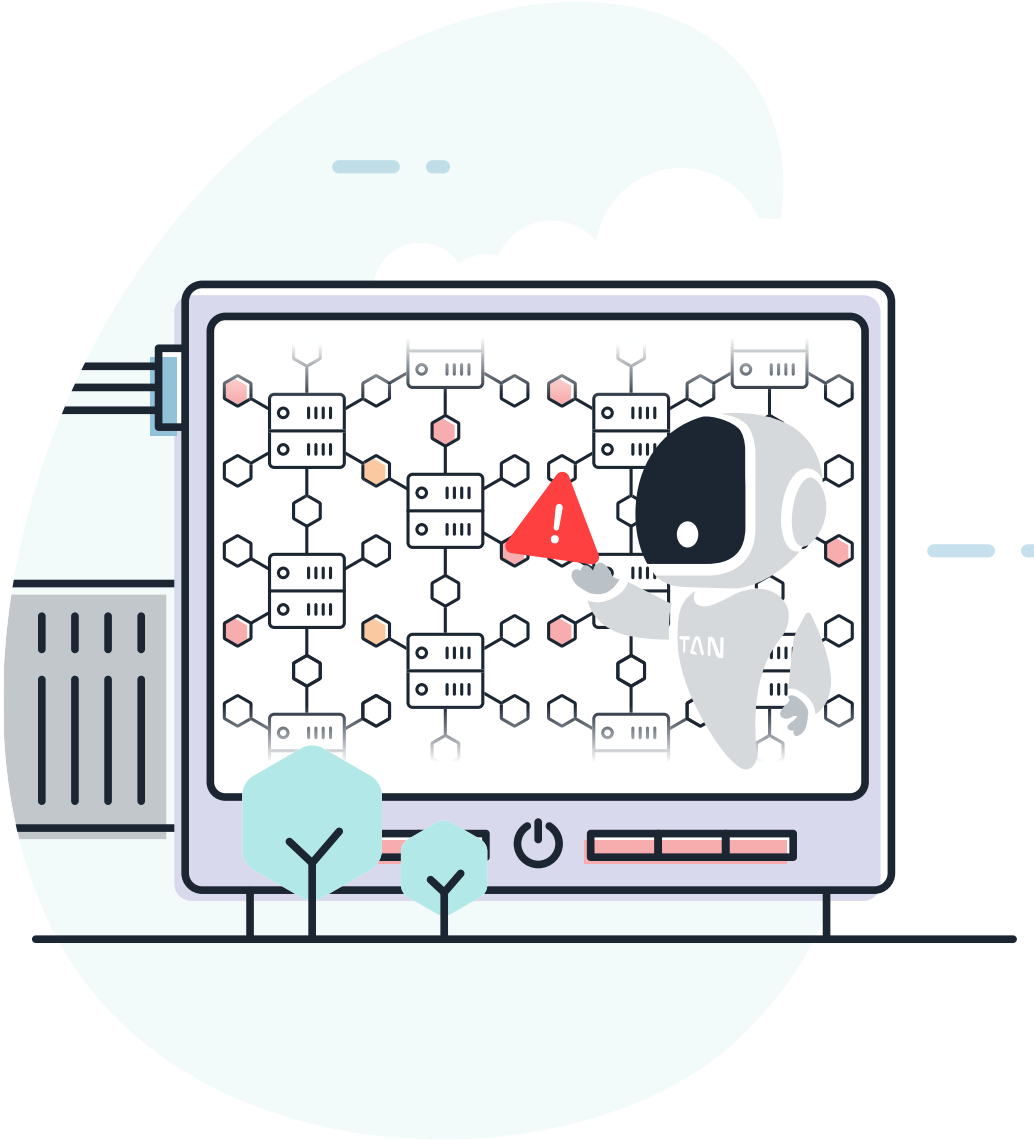
Understanding how every component (or entity) interacts with others is critical to understanding where to start solving problems. Further, sampling of application response metrics, infrastructure configuration and request tracing is no longer an effective proxy for reality.

Enterprise Observability ties everything together – service maps, infrastructure configuration, request traces, open source protocols, even profiles – delivering understanding, both programmatically and to end users (Dev+Ops 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 to find and fix problems when they occur.

Since container-based applications are more 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 (traces) to identify triggering events and inter-component 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 CI/CD 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 Enterprsei Observability. The more people that can leverage an Enterprise Observability platform, the better overall application performance will be, and the more efficient the entire DevOps organization will be.

Conclusion – Enterprise Observablity is Best for Monitoring Serverless Performance

Serverless observability / monitoring requires a fundamentally different set of strategies and tooling than solutions for monitoring conventional applications. Without an Observability Platform that can handle the unique challenges of serverless computing, organizations risk performance and availability problems that bloat their costs and undercut the value of adopting serverless in the first place.

Legacy APM tools are not designed to handle the complexity of serverless environments, or to keep pace with the continuous delivery chains of which serverless functions are a part. Meanwhile, manual observability strategies run the risk of quickly becoming obsolete.

But the Instana Enterprise Observability Platform is designed to handle the needs of Cloud-Native microservice applications, orchestrated applications and serverless application workloads – all in the same way—and all automatically.

Using data analytics and machine learning, Instana maps the complex dependencies that link serverless functions to each other and to the rest of the application. It performs comprehensive tracing and analytics, and offers rich visualizations to help teams understand both real-time and historical data. And it works in a cloud-agnostic way, allowing teams to monitor their serverless workloads across a range of serverless services.

We Invite you

...to sign up for a free Instana trial.

Start Your Trial Today

INSTANA

an IBM Company