INSTANA
an IBM Company

Monitoring Serverless Successfully:

A Comprehensive Guide

Table of Contents

Monitoring Serverless Successfully: A Comprehensive Guide

Since the introduction of AWS Lambda in 2014, 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 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, however, the serverless paradigm has created a series of new challenges for the teams responsible for monitoring and managing serverless functions. Because there are certain aspects of serverless architectures that are fundamentally different from conventional application deployment models, maintaining visibility into the environments that host serverless code, as well as into the functions themselves, is more difficult in some respects.

With that reality in mind, this eBook offers a comprehensive look at the challenges of monitoring serverless functions, as well as tips and best practices for solving them. It starts by discussing the reasons that make serverless monitoring so challenging, and explaining why traditional Application Performance Monitoring (APM) tools come up short when applied to serverless workloads. It then identifies the features necessary to effectively monitor serverless and to fully realize the performance, cost and reliability advantages that serverless can unlock.

Serverless monitoring 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. From these differences arise a number of serious challenges when it comes to monitoring and maintaining visibility into serverless deployments.

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.

Vendor-specific serverless services

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. AWS Lambda does not support all of the same programming languages as Azure Functions and Google Cloud Functions, for example. 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-premises serverless frameworks, such as OpenFaaS, into the mix.

For these reasons, attempting to monitor serverless functions at the level of the service itself leads to an approach that is not portable, and that is dependent on the particular configuration provided by a certain vendor. For teams wary of lock-in and the inefficiency of having to build a new monitoring process from scratch whenever 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 another layer of complexity.

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.

Conventional APM methods such as language-specific tracing and metrics collection are of limited use for gaining visibility into serverless performance and availability.

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 monitoring serverless requires not only understanding what is happening within functions themselves, but also mapping that data to the performance of the larger application. 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 perspective of monitoring, however, end-users' inability to access or modify the host environment represents a significant challenge. It means they cannot 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 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 monitoring can get in the way of continuous delivery – or, worse, that new versions of serverless functions are not properly monitored because the tools used to monitor them aren't yet configured for the new deployment.

Too many functions

The final challenge in monitoring serverless functions is the sheer number of functions that serverless workloads entail. 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 an ongoing basis.

Monitoring functions on this scale, using conventional, manual approaches, is very difficult. It requires tremendous effort and time commitment on the part of admins, and undercuts the ability of teams to continue scaling up their deployments.

Why legacy APM is not enough for serverless monitoring

A variety of legacy APM tools – meaning those designed before the advent of the cloud-native era – may support serverless monitoring to some extent. However, they come up short in several respects.For one, they typically lack the 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 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 APM for serverless monitoring

Although legacy APM solutions are inadequate for serverless monitoring, cloud-native APM tools offer the features required to handle the complexity, dynamism and scale of serverless computing services. Following is a summary of the key functionalities required in an APM solution to monitor serverless effectively.

Comprehensive tracing and analytics

There are two foundations to effective monitoring in cloud-native environments: Tracing and analytics.
To monitor serverless functions effectively, APM tools must be able to not just perform both of these features on individual parts of the application, but also to perform them comprehensively – meaning, across the entire application.

In other words, APM solutions require the ability to trace and analyze individual application requests across every component of the application, including serverless functions and other services. At the same time, they must be able to perform analytics on aggregate metrics collected from the application as a whole, as well as on collections of traces. It's only through comprehensive tracing and analytics that APM tools can provide 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, being able to map and interpret dependencies is critical. 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.

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.

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 APM tools should be able to discover new deployments and updates automatically, then begin monitoring them without manual intervention by human engineers.

Cloud-agnostic serverless monitoring

As explained above, serverless monitoring tools that work only with certain cloud services lack portability. They depend on specific services or configurations in order to monitor functions at the environment level.

A more flexible and portable approach is to choose an APM tool 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 monitoring requires the ability to visualize analytics and traces clearly. Visualization is critical for enabling monitoring 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.

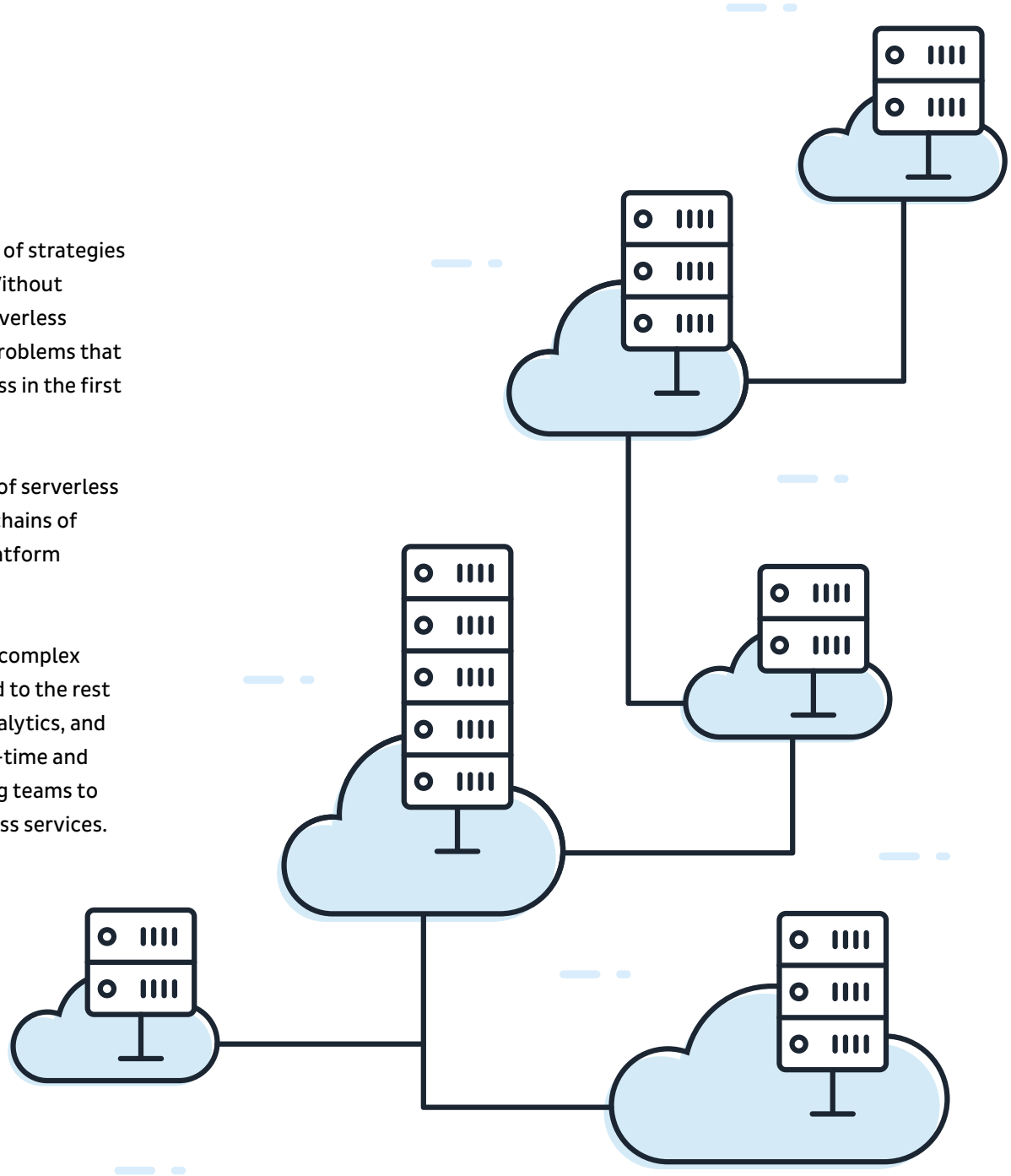
In order to be most effective, APM tools should provide visualization both for real-time and historical data. Real-time visualizations give teams visibility into the application as it currently exists, while historical visualizations enable them to research and issue or gain crucial historical context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

Conclusion

Serverless monitoring requires a fundamentally different set of strategies and tooling than monitoring for conventional applications. Without an APM solution 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. But Instana, an APM platform designed from the start for the cloud-native age, does.

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.**



About Instana, an IBM Company

Instana provides the only **Application Performance Monitoring** (APM) solution that automatically discovers services, deploys agents and monitors component health for microservice and containerized applications.

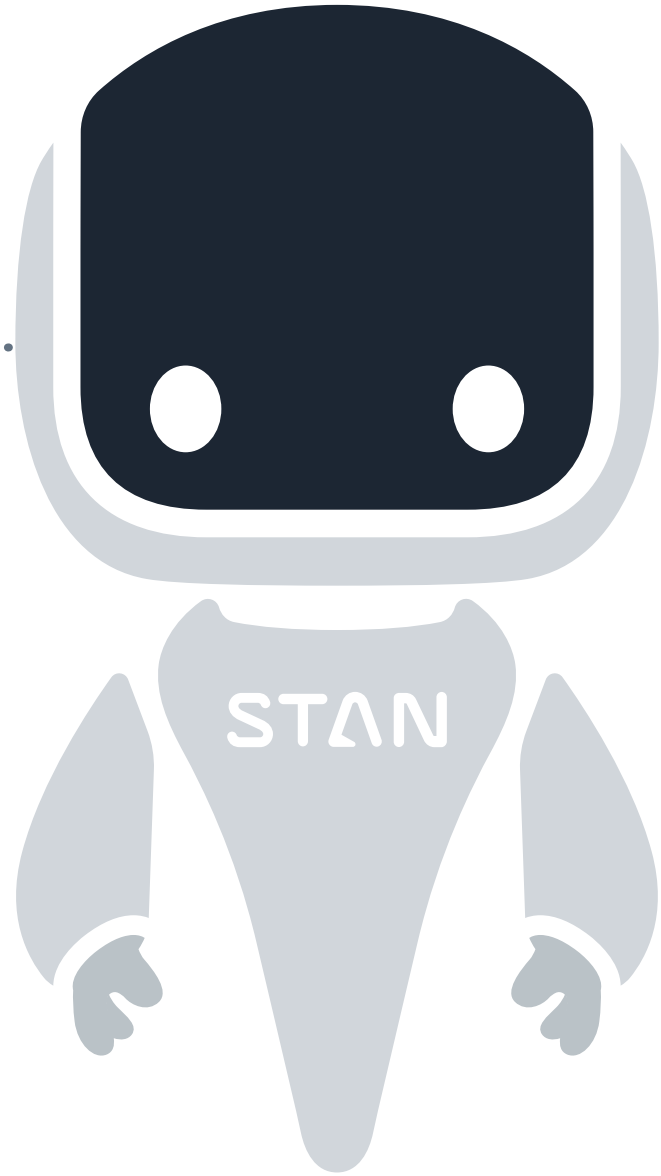
Built to handle the demands of agile organizations, Instana continuously aligns application maps with any changes, detects behavioral anomalies and automatically provides a health score for each technology component. Organizations benefit from real-time impact analysis, improved quality of service, and optimized workflows that keep applications healthy. The solution notifies DevOps teams when service quality is at risk, providing precise information identifying the triggering event and potential root cause.

Visit **https://instana.com** to learn more.

Try It For Yourself

Don‘t just take our word for it. Try Instana‘s Automatic Monitoring for yourself. You can be monitoring your applications in just a few minutes.

Start Your Trial Today



Stan
Your Intelligent
DevOps Assistant

INSTANA
an IBM Company

INSTANA

an IBM Company