Three Application Performance and Visibility Approaches to the Modern Application

Introduction ................................................................. 2
Monitoring and Troubleshooting Monolithic Applications .................. 3
Application Performance Monitoring/Management (APM) ................. 4
IT Infrastructure Monitoring (ITIM) ........................................ 4
Log Aggregation ............................................................. 5
New Monitoring and Troubleshooting Requirements ......................... 6
Introducing Epsagon for the Modern Application ................................. 8
Summary ........................................................................ 11
INTRODUCTION

Software development based on cloud microservices results in highly distributed, modern applications that are easier to manage. This type of development requires a modern architecture: a suite of stand-alone components such as containers, managed services, and serverless functions. The components each run in their own process and work together in a well-defined interface across various communication channels, including both synchronous and asynchronous.

The development community has embraced microservices and modern applications for their flexibility, scalability, and maintainability. Once the application design is in place, delivery times can be accelerated by leveraging existing managed services, as well as through the parallel development and testing of components prior to integration. Because each microservice runs in its own process, scalability can be implemented on a task-by-task basis—far more efficient and cost-effective than having to scale an entire monolithic application. The modular approach also makes it easier to maintain system health with little or no impact on the end-user experience.

Despite its many benefits, however, the complex architecture of modern applications can make them hard to manage, orchestrate, and troubleshoot. Monolithic application monitoring and troubleshooting paradigms, for example, are no longer relevant or effective. With the ephemeral and self-contained nature of microservice instances, CPU and network metrics are no longer an indicator of application performance. And with heavy reliance on managed services, the microservice host is often not accessible for deploying an automated monitoring agent. It is also a challenge to trace the high volumes of asynchronous requests across many distributed components.

⚠️ The complex nature of modern applications can make them hard to manage, orchestrate, and troubleshoot.
In this white paper, we explore why traditional application performance monitoring and troubleshooting approaches are no longer effective, what needs to change to meet the development and troubleshooting needs of the modern app, and how Epsagon has been built from the ground up to provide an automated application performance monitoring and troubleshooting solution for modern cloud applications.

MONITORING AND TROUBLESHOOTING MONOLITHIC APPLICATIONS

As opposed to highly distributed modern apps, legacy apps have a monolithic architecture built on a single codebase that is shared by all modules. Communication among the application modules is via the code itself, with one function calling another function, or synchronous HTTP communication across services that are relatively easy to trace. In fact, monolithic apps can be instrumented quite easily for high observability—manually by DevOps engineers or by using popular platforms.

Whether deployed on-prem or on the cloud (or both), monolithic applications also run on a known infrastructure, with the DevOps team being responsible for latency, availability, and scalability. If CPU and network bandwidth are not managed properly, the application’s performance is impacted directly.

There are three leading approaches to monitoring and troubleshooting monolithic applications, which we present here briefly along with their shortcomings.
APPLICATION PERFORMANCE MONITORING/MANAGEMENT (APM)

Traditional APM tools detect and get to the root cause of issues that can negatively impact an application’s performance from the perspective of the end user. They monitor infrastructure metrics over time on performance-related processes, such as CPU utilization, memory requirements, data I/O, and bandwidth consumption, as well as application metrics. A traditional APM platform integrates and correlates information from various monitors and logs, generating reports and dashboards to help IT teams more quickly understand and resolve anomalies.

Downside: The traditional APM approach is problematic for modern applications since the metrics that they track do not go beyond a single monolithic service. In addition, most traditional APM platforms deploy their own agents to monitor key processes—an approach that is difficult to implement when DevOps has little or no access to process hosts. Even newer APM companies require installation of a “sensor” on the infrastructure.

IT INFRASTRUCTURE MONITORING (ITIM)

ITIM tools monitor for and identify IT infrastructure problems before they adversely affect business-critical processes. By aggregating data from numerous sources, they provide single-pane visibility into infrastructure health across even the most complex environments—on-premises, multiple public clouds, and hybrid clouds.

One of the key metrics that ITIM tools capture is infrastructure availability. They also track and analyze resource utilization of servers, networks, database instances, hypervisors, and storage. Many APM vendors are also considered ITIM vendors.

The traditional APM approach is problematic for modern applications since the metrics that they track do not go beyond a single monolithic service.
**Downside:** Although ITIM vendors are scrambling to adapt their platforms to the needs of today’s highly distributed application architectures, monitoring infrastructure alone is not going to provide meaningful insight into application performance. In addition, manual annotations to code requires time and effort by the development team.

*Monitoring infrastructure alone is not going to provide meaningful insight into application performance.*

**LOG AGGREGATION**

Log aggregation tools gather the many disparate and voluminous logs being produced by IT systems and applications and transform them into data that can be parsed, searched, and indexed. Although an IT team can implement a home-grown log aggregation solution, today there are many platforms and services that discover and aggregate logs automatically, with value-added features for analyzing them, creating alerts, generating reports, and so on.

In addition to log aggregator companies, public cloud providers offer log aggregation services, such as Amazon CloudWatch on Amazon Web Services (AWS).

**Downside:** Because each log is written individually from a distributed service with no correlation among the logs, the logs do not capture the essence of the distributed application. They also must be written manually. The number and size of the logs to be aggregated in order to get actionable insight into the performance of a modern microservice application is a challenge in and of itself.

In summary, the legacy approaches to monitoring and troubleshooting monolithic applications are not appropriate for today’s highly distributed applications. This new development paradigm calls for new monitoring and troubleshooting paradigms as well.

*Log aggregation tools are inadequate for providing actionable insight into the performance of modern applications.*
NEW MONITORING AND TROUBLESHOOTING REQUIREMENTS

The new application performance monitoring and troubleshooting approaches must address a number of key requirements.

First, they have to keep pace with very rapid, iterative, and automated development, testing, staging, and deployment cycles. Meeting this requirement demands very high levels of monitoring and troubleshooting automation, including the ability to self-discover the many components that make up the application architecture and the dependencies between them.

Modern applications function through asynchronous requests that often traverse many components in order to carry out a transaction. It is virtually impossible to anticipate the exact timing and sequencing of these requests. Yet, it is critical to monitor them closely since latency or availability issues anywhere along the request route can have a significant impact on application performance and the end-user experience. Hence, the new monitoring and troubleshooting solutions must be able to automatically map and trace these distributed requests in real time.

Another new requirement for monitoring and troubleshooting modern cloud applications is agentless visibility. Monolithic applications typically make use of vendor-specific agents to provide a wealth of valuable real-time data on the networking devices, servers, storage devices, and virtual machines on which the application relies. Abstraction of the infrastructure layer, however, is one of the key tenets of modern microservice applications. While this abstraction frees up DevOps engineers from directly worrying about latency, scalability, and availability, it also means that they have little or no accessibility to deploy agents on infrastructure components. Thus, the new monitoring and troubleshooting approaches must be agentless, relying on protocols such as SNMP, WMI, or SSH to relay infrastructure data to the central monitoring system.
In any case, the classic infrastructure-oriented metrics of legacy monitoring and troubleshooting approaches are not relevant to modern applications.

Google, one of the pioneers in distributed systems, has identified the following four golden signals of monitoring that must be measured at the component level in modern applications:

- **Latency** to detect task completion bottlenecks and anomalies through measuring metrics such as processing, response, and travel times
- **Traffic** to measure the load on components (such as number of requests per second) and determine if more resources are needed or a problem is adversely affecting load balancing
- **Errors** to see how frequently a component is not responding as designed to requests, issuing alerts for impactful issues
- **Saturation** to assess whether a given resource is operating efficiently and what impact that has on overall system health.

While even legacy monitoring and troubleshooting solutions must provide ways to aggregate the high volume, velocity, and variety of log data, this issue in modern applications takes on an even greater order of magnitude and complexity. In the modern application, incoming log data comes from many distributed components. New monitoring and troubleshooting solutions, therefore, must be able to make actionable sense in real time out of growing volumes of diverse log data.

In short, new monitoring and troubleshooting solutions must be highly automated and must provide a holistic, intuitive view of all the “moving parts” as well as the real-time interactions among them.
INTRODUCING EPSAGON FOR THE MODERN APPLICATION

Epsagon has been built from the ground up to meet the automated monitoring and automated troubleshooting requirements of the modern cloud application. Installed in minutes, Epsagon’s cloud-native SaaS solution is compatible with OpenTracing, Amazon CloudWatch, and all Java, Go, Python, .NET and Node.js libraries. It automatically discovers and creates a visual map of an application’s distributed components—containers, VMs, serverless functions, on-premises—and the messaging interactions among them across multicloud, on-premises, and hybrid environments (see Figure 1 below).

Figure 1. Epsagon automatically maps app components and the messaging among them
Using an agentless approach that is agnostic to the workload format, Epsagon combines full distributed tracing and logging that never misses a single trace. Aggregating all metrics in a single pane, IT teams can query the data as well as drill down into operations, payloads, and logs to fix complex issues in seconds (see Figures 2 and 3).

**Figure 2.** Automated tracing and logging speeds time to resolution
Figure 3. Conduct smart queries across aggregated tracing and logging data
To ensure that you don’t miss issues as they occur, Epsagon also empowers you with the Issue Manager where you can see the issue type (see Figure 4). Then you can create alerts for every stage of the request path and determine which channel to use to broadcast the alert.
By incorporating Epsagon’s automated monitoring and troubleshooting software as a service into their DevOps stack, companies can improve their business outcomes with:

- 90% reduction in troubleshooting time
- 75% reduction in error rates
- Enhanced developer velocity and faster time to market
- Improved end-user experience through quick visual troubleshooting of production issues
- High availability and minimal loss of business through real-time identification and remediation of performance issues
- Seamless scalability and ongoing improvement of system health and performance.

**Improve Business Outcomes with Epsagon:**

- 90% reduction in troubleshooting time for modern applications
- 75% reduction in error rates
- Enhanced developer velocity and faster time to market
SUMMARY

Modern applications cannot be effectively monitored by the three prevailing traditional monitoring approaches: traditional Application Performance Management (APM), IT Infrastructure Monitoring (ITIM), and centralized logging. With their complex microservice architectures, the sheer number and diversity of logs that would have to be maintained and correlated make it virtually impossible to get a holistic real-time view of application health. In addition, their ephemeral infrastructure resources do not provide a foothold for the agents that the traditional monitoring solutions rely on.

Epsagon provides agentless and automated real-time monitoring and troubleshooting solution for modern applications. See for yourself how Epsagon can accelerate your business outcomes by dramatically reducing error rates and troubleshooting times while significantly shortening the time to market for your modern distributed applications.
EPSAGON

Epsagon delivers automated, cloud-native application performance monitoring and troubleshooting for modern applications, including microservices and serverless. It’s the only solution that provides distributed, automated tracing of every request in a transaction and integrated logging and tracing in a single interface. Epsagon is agentless so you can run across any production workload – AWS ECS, AWS Lambda, Kubernetes, AWS Fargate, or just a VM. Everything about Epsagon is automated for DevOps/Engineering.

Epsagon, headquartered in San Francisco, CA., is an AWS Advanced Technology Partner with DevOps and Data and Analytics competencies and is sold in the AWS marketplace, including enterprise contracts, as a subscription-based SaaS service. Epsagon is also an AWS Lambda Layers Launch Partner. Epsagon provides tracing support for Adobe Apache OpenWhisk development. For a 1:1 demo, click here. You can also start a free trial now, or join our live weekly demo.

AMAZON WEB SERVICES (AWS)

Amazon Web Services (AWS) offers a serverless platform that lets you build and run virtually any type of application or backend service at scale, without having to provision, manage, and scale servers. Born in and made for the cloud and focused on AWS, Epsagon enhances product integration with a multitude of AWS services including AWS Lambda and Amazon CloudWatch, along with other management services, programming languages, and API-level integrations.