The Guide to **Modern APM**

splunk>
turn data into doing™

APM essentials for your cloud-native journey

To accelerate innovation and protect digital and cloud investments, organizations need to prioritize modern infrastructure monitoring and application performance monitoring (APM) solutions that are built in and for the cloud.

A supplement to infrastructure monitoring, APM solutions allow understanding of the application layer by monitoring, analyzing and driving response to issues therein — be they rate concerns, errors or duration problems. The ability to map the progress of a request through a services architecture with the right APM solution is key when “slow” is the new “downtime” in our always-on world.

While infrastructure monitoring can help identify if there’s a problem, application performance monitoring helps teams locate where the problem is occurring. APM tools are designed to ensure applications provide the right level of service without interruption. Application speed and uptime — for internal enterprise apps and consumer apps — is directly tied to an organization’s profitability. Knowing where in the environment an outage originates can result in much faster incident resolution, reducing the consequences of the outage by a significant margin.

But organizations can’t do it alone. They require an ecosystem of reliable partners that provide comprehensive approaches to making effective use of all data, instead of partial solutions that can slow down the pace of innovation or even give a false picture of the current state. The goal: to reach a state of observability where the whole picture is detailed and constantly up-to-date — with monitoring and alerts that take it all into consideration.

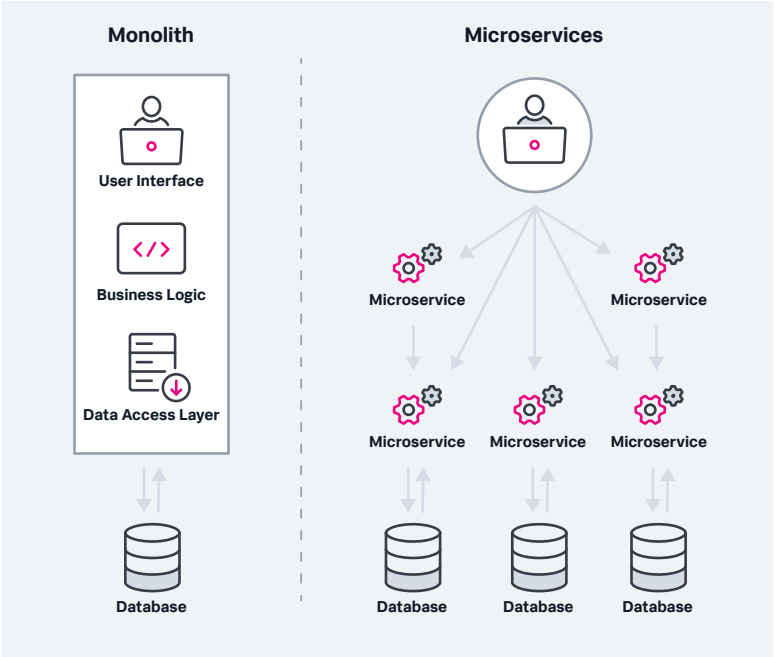
What's changed?

Although most new and advanced companies are accelerating to the cloud, many enterprises still operate with a monolithic architecture. Monolithic architecture is like a thousand-ton boulder with sedimented layers of features and redundant logic translated into thousands of lines of code — written in a single language. It can be difficult to separate a monolithic architecture into smaller components, so scaling is challenging, as you must scale the entire environment rather than scaling for a specific need. However, code-pushes are infrequent, so monitoring is fairly easy.

Microservices are like a bunch of tiny pebbles carved out of the boulder. They have distributed components, each described by a set of specific characteristics of business functions. Because microservices architectures often use containers to encapsulate distinct functions, you can easily scale up each service as your demands on it change.

These loosely-coupled distributed services perform the overall functionality of the original monolith and are aligned with event-driven and service-oriented architecture principles, where complex applications are composed of independent processes that communicate with each other through APIs over a network.

Applications built with a microservices architecture approach are commonly written in many different languages and each service can scale independently. Because of the independent nature of each service, they can be deployed on separate schedules. Organizations thus gain added agility for more frequent code pushes, allowing them to incorporate user feedback more regularly. With frequent code pushes and independent components updated frequently, organizations must rely on more involved monitoring processes.

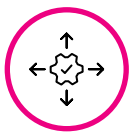


Many companies have evolved from a monolith architecture to a microservices structure, including Amazon, Spotify, Uber and Groupon. Using microservices, developers at Netflix deploy thousands of code sections every day to support more than 139 million subscribers and 10 billion hours of movies and TV series.

A primary reason to switch to microservices is to better focus on business priorities due to the increased speed of innovation. The rise of DevOps, similarly focused on speed and outcomes, has also fueled interest and adoption of microservices.

Why is this transformation happening?

Scalability on Demand



Applications can scale to support increased user demands instantly

Increased Productivity

New products in the hands of their customers, faster

Faster Issue Detection

Engineers can find problems within their services faster

Increased Availability

24/7 application availability

What are the challenges?

Traditional monitoring solutions can't address the new operational challenges that come with modern technologies — containers, Kubernetes, serverless functions and microservices as well as other supporting DevOps practices.

So why are organizations and DevOps teams bothering with all this complexity?

They're moving to the cloud and migrating to microservices because it helps them improve scalability, increase productivity, detect and resolve issues faster, improve application availability — and ultimately provide the best possible user experience.

Monitoring is a vital component of a microservices architecture. While breaking applications into component microservices offers many benefits, it also creates complexity. Microservices must communicate with one another, and each individually created and updated component must work with other components, with a minimum of latency. So when managing an application composed of microservices, you're managing a network of interrelated components. Effective monitoring of the communication interactions of services is essential to overall reliability.

Helping achieve that overall reliability is observability. Observability uses new approaches to generate insights into how applications are performing, most often by providing highly detailed data in three classes: metrics, traces and logs. Observability, with its detailed data, allows us to investigate the “unknown unknowns” of our applications, helping us to answer questions we have never thought of or identifying issues whenever they arise. And observability needs to be partnered with monitoring. Monitoring and observability are easier for developers who already have a DevOps mindset. In alignment with those DevOps practices, microservices rely on automation and collaboration across all facets of the software development life cycle (SDLC). Config management, CI/CD servers, APM, dashboards, alert automation and incident management are basics for teams running microservices.

Tradional APMs can’t get the job done

There are multiple issues that legacy APMs can’t resolve:

- Partial trace ingestion leads to missed anomalies and lower developer adoption/productivity
- Batch-based analytics means slow problem detection
- Complicated pricing and performance issues limit necessary scaling and growth
- Heavy, proprietary agents hinder innovation
- Multiple disjointed observability tools require repeated steps

Full-fidelity tracing, metrics and logs

There are two major questions we need to be able to answer about our applications. First, is the application working correctly? And second, has the application ever worked as expected? This feedback should drive almost all underlying efforts.

Without consistent and complete feedback, there's no way of knowing whether a change worked or when a particular system encounters problems. Whether you're developing code or delivering infrastructure, feedback is a vital part of being successful.

How can you consistently get to the heart of a problem through the right level of feedback, irrespective of organizational scale? Just as importantly, how can you identify and fix issues in real time?

Trace sampling vs. full-fidelity tracing

The transition to public cloud and cloud-native applications unlocked new capabilities that bring many advantages to organizations — easier scaling, faster compute capacity adjustment and reduced need to operate data centers, to name a few — but it also introduced a new set of challenges. APM can help address these challenges, but only if done correctly. Most APM solutions use sampling to reduce the amount of data they analyze in order to detect slow performance and errors, but unless a solution ingests and analyzes all of the data, it will inevitably fail to provide insights into the erroneous behavior.

Only APM solutions that ingest all transaction data in real time can help organizations elevate user experience, uphold brand image and accelerate time to market. With open standards such as OpenTelemetry, Splunk® APM helps you free your code from the constraints of any single vendor, enabling you to use the languages and frameworks that work best for you. OpenTelemetry also supports all three classes of observability data, and helps democratize data, avoiding vendor lock-in.

Metrics

Numbers that give us insights about a process or an activity, or the status of an underlying system, network or storage. Generally, metrics are measured over time — often referred to as a time series.

- **System metrics** (CPU usage, memory usage, disk I/O)
- **App metrics** (rate, errors, duration)
- **Business metrics** (revenue, customer signups, bounce rate, cart abandonment)

Traces

A trace is the record of the progression of a request through an application, including all of its myriad services.

A single trace typically captures data about:

- **Spans** (service name, operation name, duration and other metadata)
- **Errors**
- **Duration of important operations within each service**
- **Custom attributes**

Logs

Immutable records of discrete events that happen over time. Event logs exist in plain or structured text, or binary.

- **System and server logs** (syslog, journald)
- **Firewall and intrusion detection system logs**
- **Social media feeds** (Twitter, etc.)
- **Application, platform and server logs** (log4j, log4net, Apache, MySQL, AWS)

Metrics

Metrics are the numbers that give us insights about a process or an activity, or the status of an underlying system, network or storage. Generally, metrics are measured over time and often referred to as a time series. It's measured over time because systems are not stagnant, and are instead constantly changing.

For metrics, a service provides a metric key (the what) and a value. This is combined with a timestamp (the when) to make time series data, so that values can be charted over a time interval as a set of data points.

For both logging and metrics, though, it's not just the application that provides insight: fabric (like cloud infrastructure), databases, caches, queues, servers, and all sorts of things will generate telemetry (output data from the applications and infrastructure), providing varying degrees of insight.

The right APM tool takes everything into account, giving you a precise and comprehensive picture, with fewer false positives and more context around every alert.

Traces

A trace is the record of the progression of a request through an application, including all of its myriad services. Applications are complex. It's no longer the case that any one person can understand the complete flow of the system. That means we are dependent on traces to help us understand how a request goes through the application.

Traces also give us metrics. In general, these metrics are presented in a modern monitoring practice called RED. RED (Rate, Errors, Duration) gives a rapid and insightful view of our independent services and how our applications are performing. Using RED, we can use AI/ML techniques to help identify issues and respond appropriately.

Logs

A log entry is the record of a discrete event in a system, application, environment or other element. Logs can be in different formats, like raw or structured, and can come from a variety of sources, like system software, third-party software or proprietary software.

To gain observability, you need insights into metrics, traces and logs in tandem. Together, they form the three pillars of observability. Metrics let you know if you have a problem. Traces help you troubleshoot the problem. And logs are how you find out what's causing the problem.

With an observability solution that incorporates all three, you can detect, troubleshoot and find the root causes of issues, reducing your MTTR and keeping systems up and running.

Beyond monitoring — the move to observability

As you progress on your cloud-native journey, traditional monitoring begins to fall short. You're absorbing more data from more sources, with frequent changes and elastic demands on services. Distributed microservices with complex dependencies, ephemeral infrastructure and more frequent code pushes result in more complex monitoring challenges.

Only by establishing a culture of observability and leveraging a modern observability platform will you progress on your cloud-native journey and achieve the velocity needed for business success. The right solution should provide end-to-end full-fidelity visibility, give you streaming, real-time results, deliver business insights and use AI/ML techniques to reduce your troubleshooting workload.

Better Customer Experience

Catch problems before they impact customers with improved MTTR

More Predictable Operations

Alert in seconds for faster MTTD and incident response

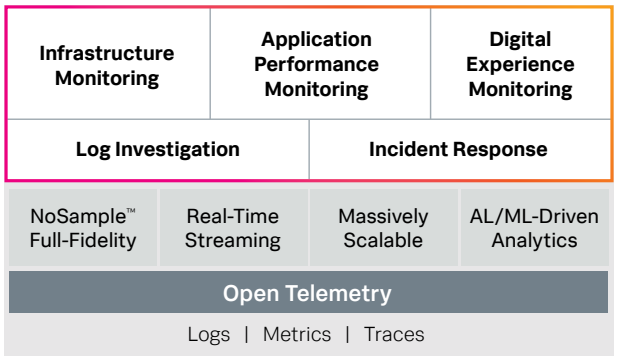
Greater Resource Efficiency

Improve visibility and correlate system and business health metrics

Higher Developer Productivity

Improve engineering agility and effectiveness to release code faster

Splunk Observability Suite



Single, tightly integrated user experience

Seamless workflow for monitoring, troubleshooting, investigation and resolution

The benefits of modernizing your APM

Observability

NoSample™ full-fidelity tracing detects all anomalies, and our streaming architecture means that it happens in real time, before users are impacted. Furthermore, trace data is collected for multiple programming languages and open source frameworks via automatic or manual instrumentation. Having trouble with that? Splunk is a major contributor to the OpenTelemetry project — so you have world class experts to help you. And since Splunk® Infrastructure Monitoring is designed to handle and analyze scale, thanks to dependency analysis the system can immediately show how a new release affects other services, allowing your team to easily identify the impact on individual users or components.

Control

Your data, your choices. Avoid vendor lock-in and expedite time-to-value with open source, lightweight agents and open standards-based instrumentation. Hundreds of ready-to-use integrations with popular OSS, cloud infrastructure and services are at your fingertips, so you can automatically pull standard metrics from services and feed them into pre-built dashboards to see things clearly and quickly. Dynamic point-and-click alert conditions and a smart agent for service auto-discovery allows for adaptability — as does fully-automated Kubernetes monitoring, which empowers your team to accelerate troubleshooting times and root cause analysis.

Speed

Resolve issues fast with AI-driven troubleshooting and instant alerting that identifies problems in mere seconds — the same time it takes to monitor serverless functions. High resolution, easy-to-use dashboards and charts update in real time with the metrics that matter most to you; you can even see a live heatmap of your entire infrastructure in one unified view. The ultimate result: Performance degradation is minimal, so users can have the best experience the applications can offer.

Get **Started.**

All of the outlined considerations mark the difference between an okay APM tool for your needs versus the right APM tool for your needs — now, and in the future. Only when you ensure that your solution will cover all of the above will you be able to make swivel-chair operations a thing of the past.

Find out how you and your organizations can maintain the highest levels of business performance, minimize downtime and deliver world-class digital experiences.

Learn More

splunk>

20-16346-SPLK-The Guide to Modern APM-106