5 Things to Consider When Choosing an APM Tool

Make informed decisions about APM in an increasingly cloud-dominated world

splunk> turn data into doing™

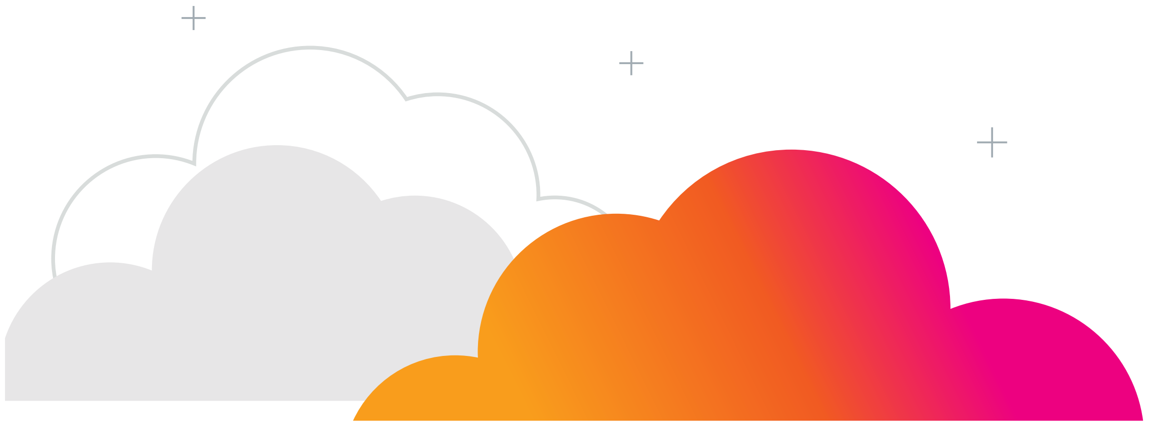
Monitoring and availability has long been part of the IT and development process. But where once you pinged a system to see if it was up or down, it's now become imperative that IT operations and DevOps understand the "why" at a more granular level. Organizations undergoing digital transformation or marching forward with new digital initiatives need to effectively evolve to meet the operational scale, speed and complexity they're facing with modern tech stacks and development cycles that are cloud-native.

Infrastructure is now elastic to enable more apps on demand. Architectures are distributed from physical and virtual servers to Kubernetes and serverless functions. Microservices are now entrenched in modern application development and delivery.

To accelerate innovation and protect digital and cloud investments, organizations need to invest in modern monitoring and application performance monitoring (APM) solutions that are built for the new data age powered by the cloud. And, they 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. 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.

5

Here are five things to consider when choosing an APM solution, regardless of what stage or maturity level you are in your cloud journey.



1 Trace Sampling vs. Full-Fidelity Tracing

Trace data has become a keystone in modern monitoring systems. It provides visibility into both the path and structure of a request. The former allows developers and SREs to know the services involved and the connection between them while the latter helps them understand the mismatches that take place in executing a request.

There are two common forms of sampling: head-based and tail-based sampling. Head-based sampling is the most commonly used method. A trace is sampled before it has gone through its entire path. It's simple and has little to no impact on application performance, but it samples randomly and — because it samples at the outset — cannot determine with certainty which trace will encounter an issue as it makes its way through all the services and touchpoints. In short, it misses the bigger picture.

In contrast, tail-based sampling takes a sample after traces have fully completed. It can catch errors or latency but still encounters its own problems. For instance, despite it being inherently more comprehensive than head-based sampling because it analyzes 100% of traces, it only samples and stores some of those traces. The traces sampled and stored are usually chosen by criteria that's predetermined by the vendor providing the solution, which although helpful cannot account for unknown or unplanned issues. Essentially, some requests provide bad results without triggering a code error, allowing them to go undetected, because organizations use their solution in ways that the product designers did not intend or experience problems the same designers didn't think about. Then there's the issue of cost. It is often executed on "satellites" running in the customer environment that consume compute resources — leading to higher cloud spending and management.

Both sampling methods also run into issues with code that is executed on the user side (e.g., on the browser) that's monitored by products called Real User Monitoring (RUM). Similar to the backend, this code has its own spans. Many of these spans invoke backend code execution monitored by APM. Any APM that samples (whether head- or tail-based) has issues providing end-to-end visibility, requiring developers and SREs to manually stitch together browsers spans with backend traces. This task is difficult, time consuming and impossible if the relevant backend traces have not been sampled.

On the other hand, full-fidelity tracing provides developers with unmatched levels of visibility into their applications, enabling them to find and understand any issue — even unforeseen ones — through analysis that overcomes the limitations of typical head- and tail-based sampling. Developers are used to looking at full-fidelity logs as part of the development process, in order to get feedback on any new code push or fixes to issues that they released. Traces provide another easy lens to do that, and provide more information about the interaction between the different services, something that is difficult to do with logs. If those traces are sampled, it is very difficult for developers to use them — they would need to wait an unspecified amount of time to see if the fix or the new feature works as intended. Full-fidelity offers the entire picture and can help account for all aspects of the request flow, from the front end to back end.

To ensure the best results, look for a solution that not only offers full-fidelity tracing but also builds atop AI-driven directed troubleshooting, real-time alerts and infinite cardinality so you can easily make sense of all your data.

2 Real-Time Monitoring — Minutes or Seconds?

Most solutions promise real-time monitoring, but not all real-time capabilities are equal. When it comes to operations in general, but especially DevOps, the difference between effective, secure operations and broken ones can be determined by a function or service that spins up and down in seconds. If you can't have that level of visibility and in that time window, you could be in store for a rude awakening.

That's because modern, cloud-based infrastructure is complex and fast. Technological pivots to cloud-native solutions like microservices and serverless functions require a completely different approach to monitoring that is granular and timely. Traditional solutions have lost pace with the speed of modern day architectures and applications. For instance, slower analytics translates into slower problem resolution. Many APM solutions process and surface alerts in minutes, not seconds, which is too slow. And with batch analytics and sampling, developers need to wait several minutes to see whether a change actually solved the problem. It's a massive shortcoming in a world that is all about reducing mean time to detection (MTTD) and mean time to resolution (MTTR).

Think about e-commerce platforms. Consumers expect a simple, frictionless experience. But too often, there are issues with the cart, errors are produced when processing a financial transaction, and more.

One misfire and churn happens — the consumer abandons the cart and does not complete the purchase. Operations must be able to identify any issues in seconds and remediate them before they affect the end user. This can be complicated because the e-commerce experience is an amalgamation of many apps and services working in concert to produce the right experience. One kink in the process and it can get derailed. It's like driving. You need immediate feedback from the car when you turn, in seconds not minutes, or you'll end up going off the road. But that can all be handled with monitoring that is constant and exhaustive.

As such, it's important that the APM solution you employ to monitor the applications in your tech stack, provides the insights you need in seconds. With thousands of containers and millions of data points, you need a solution that leverages AI-driven analytics for results at the speed the business needs.

3 You Can't Forget About Scale

Where once monolithic set ups worked — producing thousands of lines of code written in the same programming language, operating from a single piece of software — organizations are now using a distributed architecture dependent on the cloud. It includes tens to hundreds of distributed services, multiple hosts/clouds, several languages and frameworks and frequent code pushes under new DevOps standards. The switch has been instigated by the need to innovate and compete, but now that organizations are there, many struggle to keep up with all the change. Older monitoring solutions simply can't scale.

These 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. They need a monitoring solution that can account for that complexity at scale. Not having that ability can limit your toolset in the worst possible moment (e.g., a flash sale or busy season). Not accounting for the whole picture also slows innovation, as developers have to constantly think about whether they're going to hit those limits or work around them.

The only way to keep up with the scale and diversity of today's modern architecture is to have a solution that takes all data into account regardless of form or source, with no performance degradation. Additionally, it should be able to surface insights through integrated workflows, centralized management and new best practices. The right tool should also accelerate development cycles through dependency analysis that immediately shows how other services are impacted by new releases or changes.

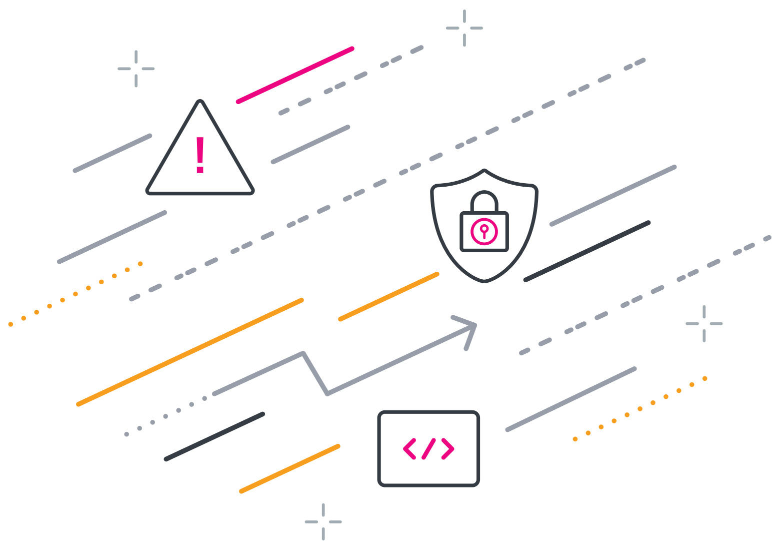
4 What Is Open Source in APM?

In a space where scale and flexibility are important, a solution that is open source and flexible is the best choice. There are many APM solutions that will promise integrations and malleability beyond that of their competitors but will lock in customers with proprietary instrumentation — limiting control over what data is collected and how.

But those that employ open standards are out there. OpenTelemetry is one such project. It lends itself to an open ecosystem that avoids vendor lock-in. It is quickly becoming the standard for instrumentation and data collection in the observability space by providing a single set of APIs, libraries, agents and collector services to capture distributed data from applications. In fact, over 160 businesses ranging from cloud providers to observability vendors are employing it, according to CNCF's Devstats. It's also the most active project for the Cloud Native Computing Foundation, only behind Kubernetes.

OpenTelemtry removes barriers to data acquisition and ingest, granting organizations a choice to freely move between vendors that best suit them without having to redo their instrumentation. It also breaks down artificial barriers between tools, which often require manual effort and repetition of forensic efforts across toolsets. In short, only through true open-source development models can organizations meet the demands of modern infrastructure.

Breaking free from vendor lock-in requires a tool that lets you see the code, suggest changes to it, and then implement those changes. Many vendors will present the semblance of choice with none of the benefits. Don't fall for it.



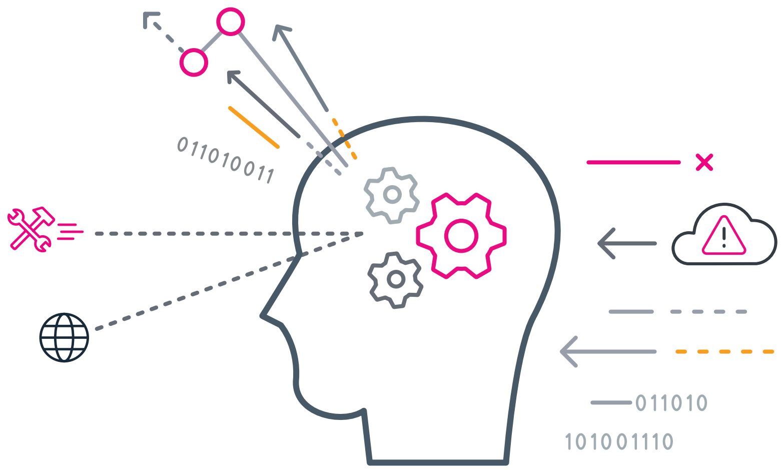
5 Integrations and Collaboration for Total Observability

The right APM tool should be one aspect of a tightly combined observability solution. It should be able to integrate seamlessly with today's open source, highly volatile observability technology. It needs to be able to lend and receive the right support in order to provide the best results.

The risk of not having an APM tool that is a seamless part of a larger observability suite is having to switch across different tools and data sources in search of the whole picture. And you can't completely trust vendors when they say that's what they offer. Players in the APM space may offer all their observability tools under a single UI, but under the hood these tools are disjointed and users still need to manually stitch together data from different sources before they can know exactly what's going on.

With a combined monitoring and observability platform built on a common user experience, you'll be able to have simple and seamless workflows across monitoring, troubleshooting and investigation, making it easy to go from problem detection to resolution in time with business needs — no matter how complex the problem.

You'll be able to pull all the data, analyzing and storing all traces in fine detail for infinite cardinality that lets your developers break things down by container, version, user, or any other attribute. AI and machine learning are key in helping process all this information. Domain-centric AI within observability solutions can detect patterns and surface relationships between interdependent services. A streaming architecture is also important in this context for analysis that discovers services, detects patterns, alerts on problems and automatically surfaces recommendations in seconds.



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.

Visit splunk.com/devops and learn more about how you and your organizations can maintain the highest levels of business performance, minimize downtime and deliver world-class digital experiences.

splunk® > turn data into doing™

20-15973-SPLK-5 Things to Consider When Choosing an APM Tool-108-EB