



Search. Observe. Protect.

An Introduction to APM

The what, why, and how.

elastic.co

Table of Contents

Why do we need application performance monitoring (APM)?	3
Applications power businesses	3
Poor performance equals lost revenue	4
Modern, cloud-native apps are complex	4
What exactly is APM?	5
End-to-end visibility	5
Surfacing latency and errors	6
APM beyond production	6
How APM works	7
APM agents	7
Instrumentation and configuration	8
Analysis	8
Key terms	9
Distributed tracing	9
Spans	10
Transactions	10
Traces	11
Real user monitoring	11
Choosing an APM tool	12
Technical capabilities	12
Ease of use	13
Deployment options	13
Support for open standards / open data	13
Architecture and scalability	14
Security	14
Capabilities beyond APM	15
Pricing	15
About Elastic	16



Why do we need application performance monitoring (APM)?

A quick note: Throughout this guide, we'll discuss [application performance monitoring \(APM\)](#) as one pillar of [observability](#). Together with logs and metrics, APM plays a critical role in building observable systems.

Applications power businesses

Applications are the public faces of modern organizations. They're how we engage with products and services, whether that's an ecommerce storefront, a ride-sharing app, or the various collaboration and productivity tools we use daily. When we're adding a pair of headphones to our digital shopping carts or creating the perfect playlist for a dinner party, we're interacting with — and forming opinions of — applications. Brand perception, loyalty, and ratings are based on these experiences.

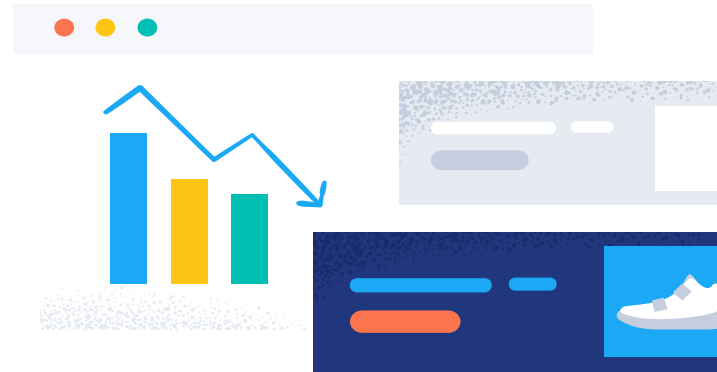


Poor performance equals lost revenue

Think of the applications you use for work — or on your personal devices at home. What comes to mind? How's the ease of use? How quickly do they respond? How often do they error out? Do the answers to these questions influence your perception of the companies behind those offerings?

When applications don't perform as expected, there are direct impacts to the business. Revenue and customer confidence are at stake.

If a digital shopping cart experience is broken, users will head to another store, resulting in lost sales. If response times are consistently slow or errors are frequent, even loyal users may decide to stop using the product or service altogether. Every organization knows what downtime means to their bottom line. Mean time to detect (MTTD) and mean time to respond (MTTR) have price tags.



Modern, cloud-native apps are complex

The evolution of how applications are developed and delivered, and the complexity resulting from that shift, makes the need for comprehensive performance monitoring even more pressing.

The adoption of microservices architectures effectively changed the way applications were built and maintained, offering continuous delivery and scale that was previously unimaginable. But the distributed and polyglot (written using different languages/frameworks) nature of these architectures adds complexity. Continuous delivery and automation principles enabled frequent code pushes, but made the ability to track performance impact more critical so that issues could be quickly patched or changes reverted if a deployment didn't go as planned.

Cloud-native development principles allow teams to continuously deliver great digital experiences to their users, react quickly to feedback, and pivot as needed. What users don't see are all of these gears humming away in the background, powering their every request. So how do teams keep an eye on these applications?

As humans, we can't process this volume of data in a meaningful way. We need an organized view with a clear path to insight. Enter APM.

What exactly is APM?

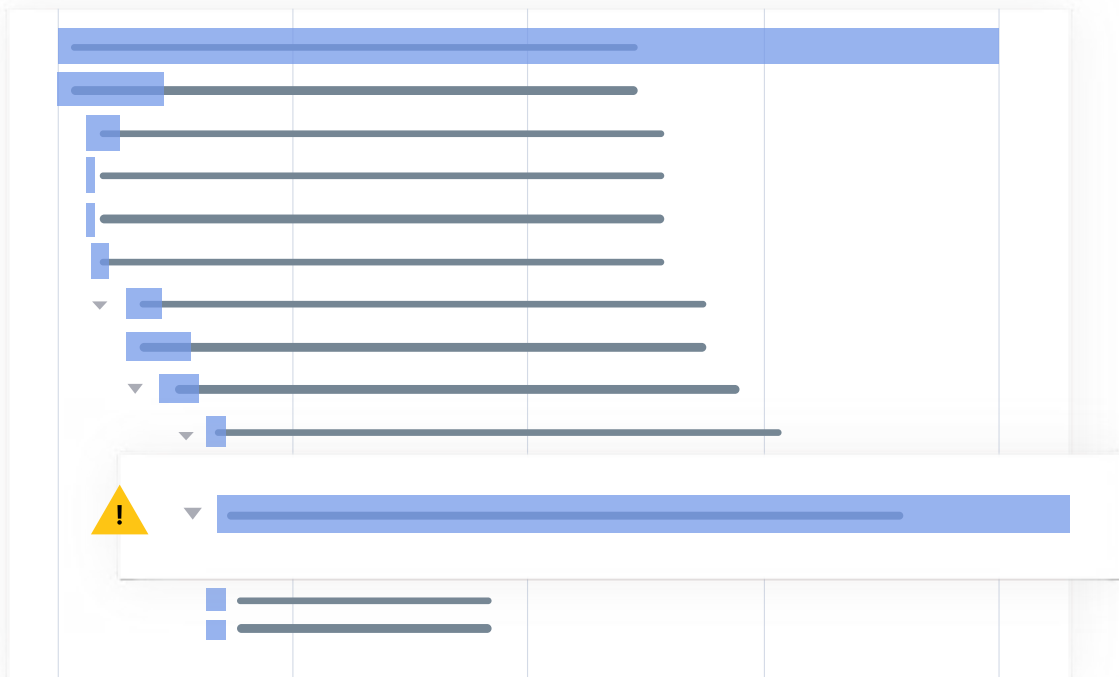
APM is the solution for collecting, monitoring, and analyzing the end-to-end performance and behavior of transactions through your applications (and the services they depend on). From the moment a user starts interacting with the application until they have achieved their desired results, APM tracks their experience.

Because of its direct ties to user experience, APM helps connect IT to business goals. The relationship might not be exactly 1:1, but it has a positive correlation. By providing continuous insight into how your applications are performing, APM allows you to be proactive, rather than reacting to issues after an onslaught of customer complaints via your ticketing system or even social media.

End-to-end visibility

Once an application is instrumented, the resulting data tells you exactly what's happening inside of it. APM tracks transactions all the way through their journey, so every request and response is recorded and measured, no matter how complex your architecture is.

Linking all of these traces together, APM gives you the complete picture of your app's performance — from a bird's-eye view of your services (and how they are interacting) all the way down to code-level insights.



But wait — don't we have logs that tell us what's happening inside our applications? Logs do give us valuable data with contextual information about the types of errors or other events that occur. What logs don't show is the end-to-end journey of the user experience, which is where APM traces (and distributed tracing) fill in the details. Pairing traces with logs and metrics will give you unified visibility into your ecosystem.

Surfacing latency and errors

APM monitors two attributes surrounding user experience: transaction duration and errors. How long do HTTP requests usually take? How many 500 errors do we usually see over the course of an hour?

Latency issues occur when a response to a request is taking longer than usual. An example is a user clicking to see a product and the response taking more than a few seconds to load.

Errors, of course, signify when an unintended result has occurred. This occurs when the request isn't completed successfully.

The end goal of any investigation is to fix issues. To do this effectively, teams perform root cause analysis to determine exactly what component is causing the problem. Can we quickly identify which service is the performance bottleneck? Can we pinpoint the exact function call or method responsible for this issue? With APM, the answer to both questions is “yes.”

APM beyond production

We often talk about APM in the context of user experience with applications in production, but APM comes into play earlier with impacts across multiple teams and environments. For instance, APM provides developers with the feedback they need to create code that leads to a speedier develop-test-deploy loop.

Businesses are using APM across stages to help teams build, QA, deploy, and monitor apps more efficiently. If organizations use APM across many teams, they can improve processes and create more efficient workflows, leading to more time for innovation (and less time spent putting out fires).

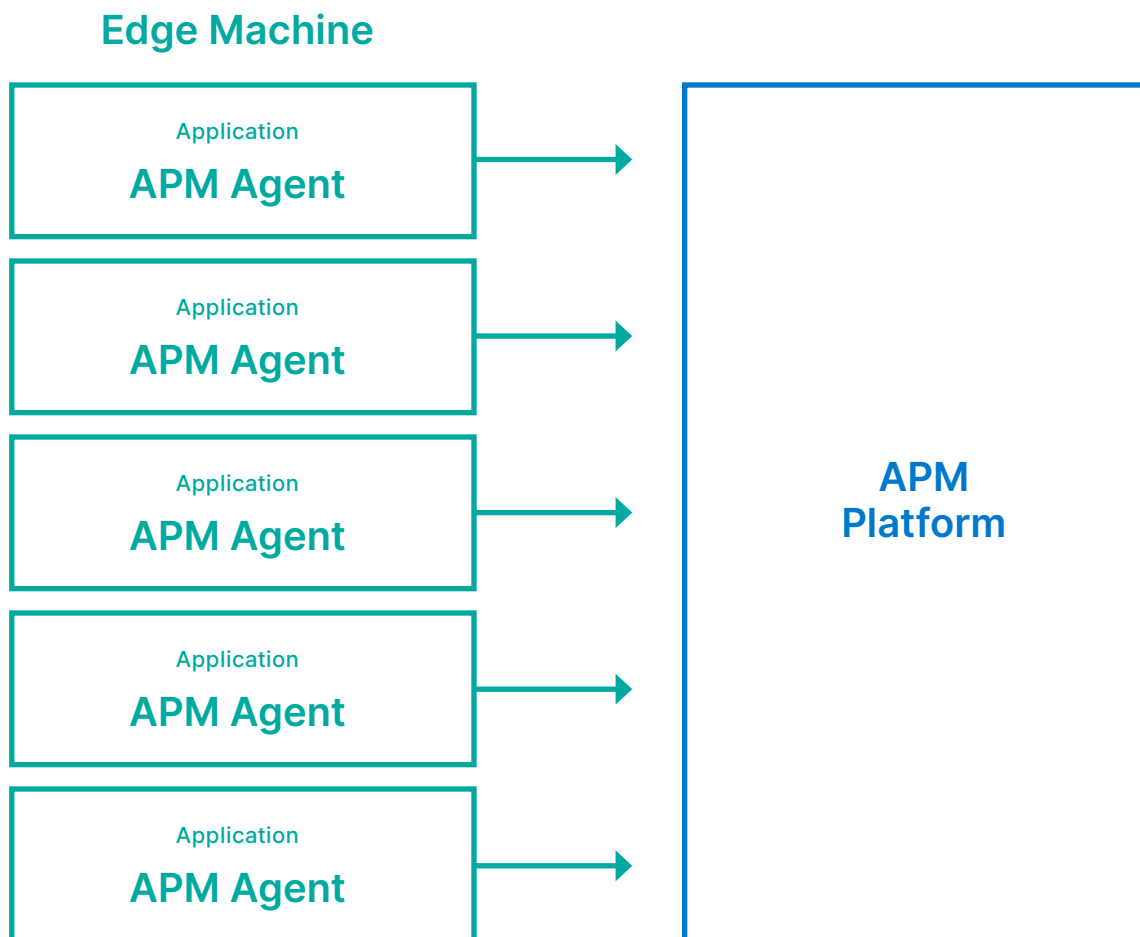
We've seen the benefits; now let's dive into how APM works.

How APM works

APM works by instrumenting every aspect of your system that you'd like to monitor (hosts, containers, applications, etc.) and then sending the collected performance data to the location of your choice for analysis.

APM agents

An APM agent is a library, plugin, or extension that monitors the performance metrics we described above. Depending on what you need to monitor (and the language it's written in), you may need one or multiple agents. Once you've identified everything that you'd like to monitor, you'll deploy APM agents to each of these pieces. While agents vary per vendor, most agents instrument your code, collect performance data, and then send your data to a server or collector.



Instrumentation and configuration

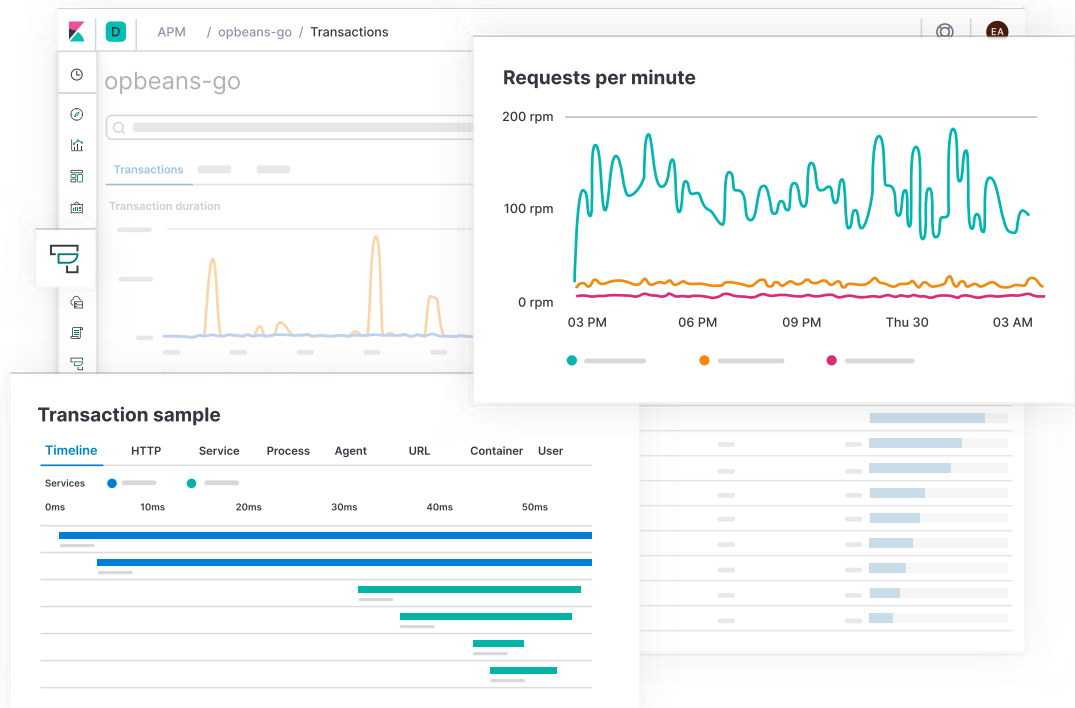
Instrumentation is the process of extending your application's code to report trace data. After installing the required agent(s), you can instrument the specific pieces of your application that you'd like to analyze to collect and send performance data to your location of choice, usually an endpoint determined by your APM tool.

You'll likely want to configure things like environment names, sampling rates, instrumentations, and metrics to help your teams easily identify and analyze the data that's streaming into your APM tool. Generally this is done using the tool's UI and API, or directly in the environment variables.

Analysis

Once the performance data has been sent to the location of your choice, you're ready to analyze the data. Most tools have a UI that will help you identify errors, latency issues, and other anomalies that are impacting your users.

This is a great place to start your investigation about reported issues, or even better, catch issues before your users are impacted by them. Identify precisely which services (down to the code level) are experiencing issues to speed up root cause analysis and MTTR.





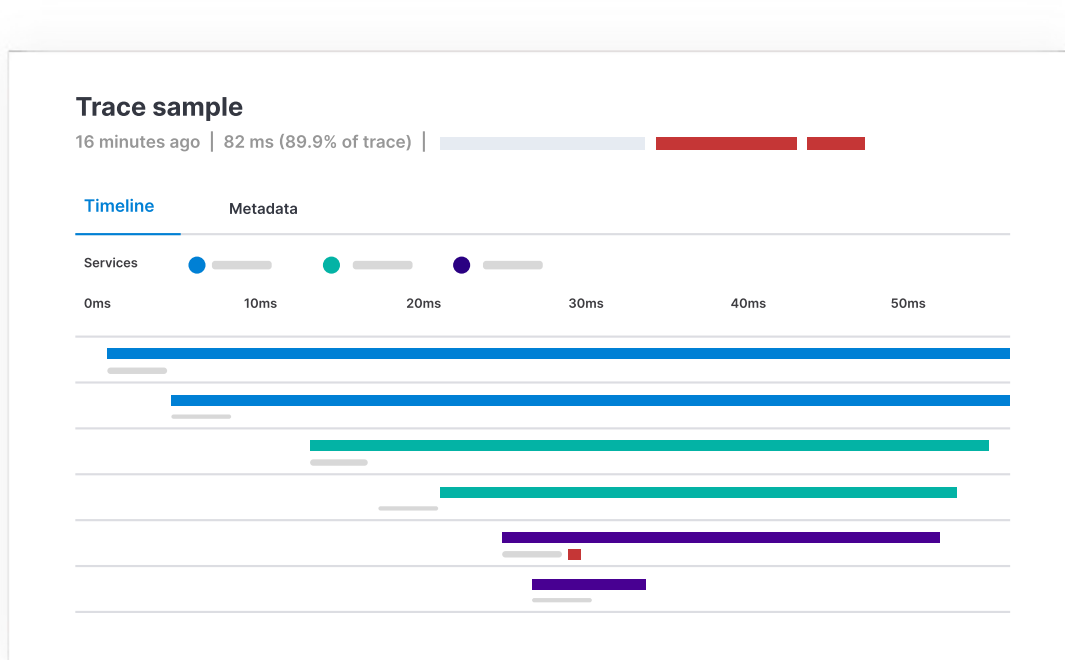
Key terms

Each APM tool might have slightly different definitions for these terms, but in general they capture the same information to provide the user insight into the performance of their applications.

Distributed tracing

By tracing all of the requests, from the initial web request to your frontend service to queries made to your backend services, distributed tracing enables you to analyze performance throughout your microservices architecture all in one view.

Distributed tracing makes it easy to spot bottlenecks by displaying complete events by service, and then by each request within that service. Surfacing errors and other issues in an actionable manner at the code level makes investigations and MTTR faster.



Spans

Each unit, or piece of the workflow, is called a span. Spans are what you would typically see in the waterfall view of an APM analysis tool, usually depicted in horizontal bars. These segments are the core of distributed tracing. Spans measure from the start to the end of an activity and contain information about the execution of a specific code path.

Common attributes of a span include:

- Start time
- Finish time
- A name
- A type

Transactions

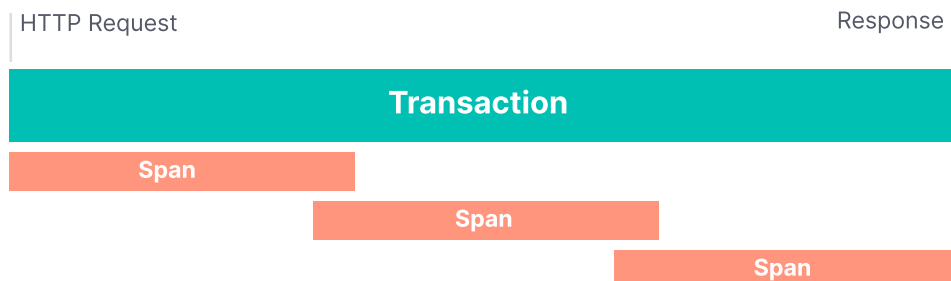
Transactions are a type of span that describe an event and can also include multiple spans.

A few examples of transactions include:

- A request to your server
- A batch job
- A background job

Transactions have additional attributes associated with them, like data about the environment in which the event is recorded:

- Service: environment, framework, language, etc.
- Host: architecture, hostname, IP, etc.
- Process: args, PID, PPID, etc.
- URL: full, domain, port, query, etc.
- User: (if supplied) email, ID, username, etc.

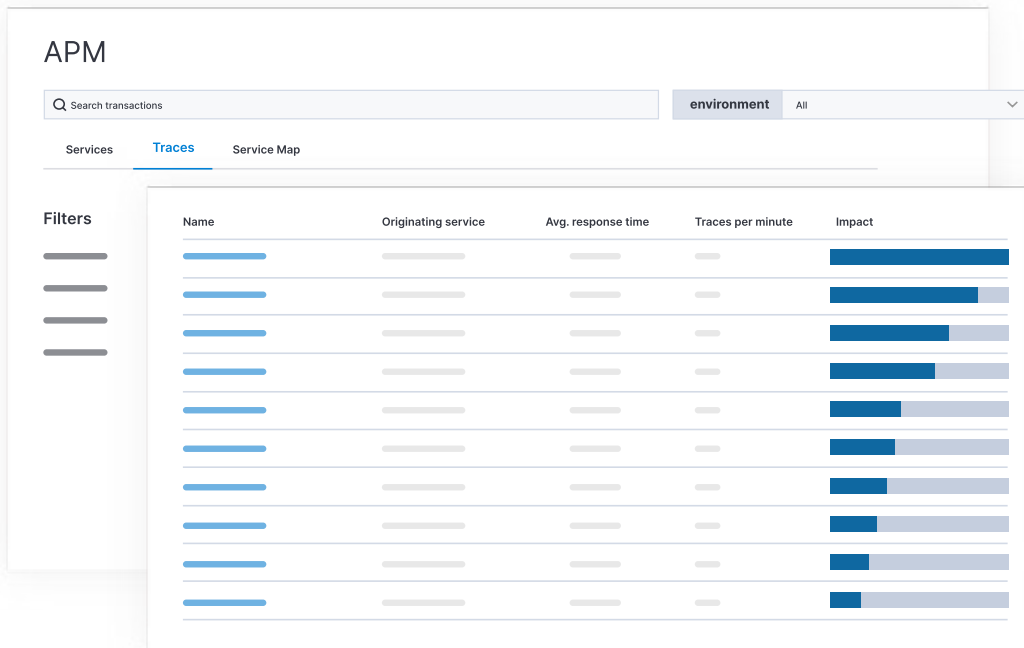


Traces

Traces measure the status and duration of a request made through your application. Spans and transactions together form a trace, connecting the dots from a user request all the way through to response.

You'll see individual traces for complete actions like processing a payment, processing a completed order, and updating shipping status.

When paired together with the logs and metrics from the application (and other aspects of your infrastructure), traces provide complete visibility into your entire ecosystem.



Real user monitoring

Real user monitoring (RUM) captures direct user interaction with clients in production environments. For example, RUM shows you exactly how a user interacted with your website along with the response times and errors incurred. This is performed by an agent on the user's browser that collects the performance data and sends it to the APM tool to be analyzed.

Choosing an APM tool

As with any tool selection process, start with clearly defining your requirements. As you list your requirements, take your growth and future needs into account. Plan a proof of concept (POC) to make sure that the tool truly fits your needs. Do the capabilities demonstrated and marketed by the vendor work as expected in your environment? A POC helps avoid surprises and hidden costs later in the process.

Technical capabilities

Create a checklist of required features with your specific technical needs in mind. Be thorough and granular with your requirements. While not an exhaustive list, here are some areas to think about:



Language and framework support

- Does the APM tool support the language(s) and frameworks widely deployed in your environment?



Breadth and depth of visibility

- Does it provide code-level visibility into performance?
- Does it give you end-to-end visibility into your tech stack?



Search, analytics, and visual exploration capabilities

- Does it include a curated UI to support proactive and reactive investigative workflows?
- Does it have visual tools like waterfall charts and dependency maps to look into the performance of distributed apps?
- Does it provide a flexible (and fast) query language to enable ad hoc investigation?
- Does it offer a flexible visualization framework that goes beyond standard vendor-provided dashboards?



Integration into your broader ecosystem

- Does it integrate seamlessly into other standard tools in your environments — for example, PagerDuty for incident response, ServiceNow for case management, or Slack for notifications? Or your CI/CD frameworks to help you automatically track the performance impact of code pushes and new deployments?

Ease of use

As you work through the technical feature requirements, evaluate the ease of use and accessibility of those features. Think about the various personas who will be interacting with the tool, and take their experience, expectations, and workflows into consideration as you answer the ease-of-use question. Expand your evaluation beyond the getting started experience into continuous operational overhead.

- How quickly can you instrument an application and go from zero to insight?
- Is the UI intuitive and easy to use?
- How much effort is needed to deploy the solution?
- How easy is it to upgrade the platform? What about the agents?
- Can the agents be centrally managed from a single place?
- What is the effort to scale up (or down)?

Deployment options

It is important that the chosen APM tool supports your software consumption preferences.

- Do you prefer a SaaS option to reduce operating and administrative costs?
- Do you want an option that can help draw down annual committed spend on your chosen cloud provider (AWS, Azure, Google Cloud, etc.)?
- Do you need a self-managed offering because cloud is not an option for cost or compliance reasons?
- Do you have a multi-cloud or hybrid strategy and want to run your APM solution closer to your workload to reduce data transfer costs or latencies?

Support for open standards/open data

The observability space is beginning to develop and embrace open standards such as OpenTelemetry (formed by merging two other open standards, OpenTracing and OpenCensus) as a standard, vendor-neutral instrumentation framework. The goal of these initiatives is to help developers easily port their application to a different APM solution without needing to reinstrument.

If you have existing apps instrumented using an open standard like Jaeger, switching to a tool that supports it can speed up and simplify migration. Open standards also help future-proof your investment.

Architecture and scalability

Evaluate whether the APM tool you select has a robust foundation and architecture to handle your current and future scale. Your ability to quickly and effectively investigate and resolve application performance issues relies on the performance of your APM tool. Make sure to load test your APM tool during the evaluation process (or review benchmark data if that's not possible) to make sure that it can handle the anticipated ingest and query volume in your environment without buckling under the load.

- Is it built on a simple architecture? Or is there a patchwork foundation under the hood that will eventually crack?
- Does it support high availability?
- Can it be scaled up easily to handle bursts in monitoring data volume?
- Are there any limits on the volume (apps, metrics, queries, etc.) it can handle?

Security

The security review process should be a core part of your tool evaluation process. Be sure to consider the following two angles:

1. The APM vendor's commitment to security in how the tool is built and delivered
 - Do the APM agents deployed in your applications need excessive privileges?
 - Is the traffic between internal components properly encrypted and secured?
 - Does the tool have the required certifications (especially in SaaS)?
 - Does the tool use third-party extensions? Are they secure?
2. The ability to secure and control access to the APM tool
 - Does the tool integrate with your corporate identity access management system?
 - Does the tool support role-based access control, with a granular permissions model?

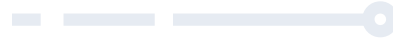
Capabilities beyond APM

When assessing tools, think beyond the typical boundaries of APM. The need for tool consolidation and unified visibility are motivating many organizations to unify APM with logs and metrics — the other two “pillars of observability” — to help streamline investigation and reduce MTTR. Evaluate whether the APM tool supports this unified vision by surfacing logs and metrics in the context of the application trace data. What about ingesting data outside of the typical IT landscape, such as social sentiment or customer support requests?

Pricing

Finally, it’s important to carefully weigh pricing options to make sure that your tool of choice doesn’t force you to compromise on your visibility or monitoring goals. As for many of the other criteria, you’ll need to take both current and future usage (and architecture) into account in the process. There is a lot of variance in how APM solutions are priced — by number of agents, number of hosts, hardware resources, etc. — with some vendors also imposing additional costs when you cross certain thresholds (for example, the number of containers or metrics). Here are a few questions to ask about pricing models:

- Is the pricing model aligned with your business needs and architectural choices?
- How will the costs scale with planned growth and architectural evolution (e.g., monolith to microservices)?
- Is there a free tier? What is included in the free tier? Are there any usage limitations?
- What level of support is included?



About Elastic

As the most popular open source logging platform, Elastic brings speed, scale, and relevance to APM.

When you're investigating an issue that's impacting users, every second is valuable. With Elastic APM, your performance data is saved as an index in Elasticsearch, enabling teams to search for and find bottlenecks in real time. Elastic APM also features service maps powered by machine learning, custom alerting options, and more so you can create better digital experiences for your users. Visit elastic.co/apm to learn more.

APM is one piece of the puzzle. Unify your logs, metrics, and APM traces in one platform for true observability of your entire ecosystem. Visit elastic.co/observability to learn more.

[Start your free trial today](#)

