DATADOG

The Benefits of End-to-End Observability

Break down silos and correlate front-end and back-end data for full observability

datadog.com

Table of contents

1 Introduction

In an era where software environments are becoming increasingly distributed, dynamic, and complex, robust observability practices play a critical role in maintaining system health and performance. Traditional monitoring solutions that only track predefined metrics or provide point-in-time snapshots can no longer keep pace with the speed, scale, and intricacy of modern applications. Observability strategies have evolved into holistic monitoring approaches aimed at delivering a seamless user experience, diagnosing issues faster, and aligning diverse teams around a single, comprehensive view of application health.

This practice is better known today as **end-to-end observability**. End-to-end observability is the act of monitoring and gaining insights from an application's entire architecture, from its foundational infrastructure to the user-facing frontend, so teams can detect, diagnose, and remediate issues rapidly and effectively.

According to a 2024 Observability Pulse Report, just **one out of ten organizations** are utilizing full end-to-end observability. Within these organizations, unifying and integrating observability data has delivered measurable gains in reducing downtime, cutting costs, and optimizing engineering efforts. These organizations experienced:

- **78% less annual downtime:** 107 hours per year vs. 488 hours
- **11% reduction in engineering time spent on disruptions:** 28% vs. 32%
- **4% higher median ROI:** 302% vs. 290%

The impact was even greater for those organizations that combined their existing observability data with **five or more** business data types:

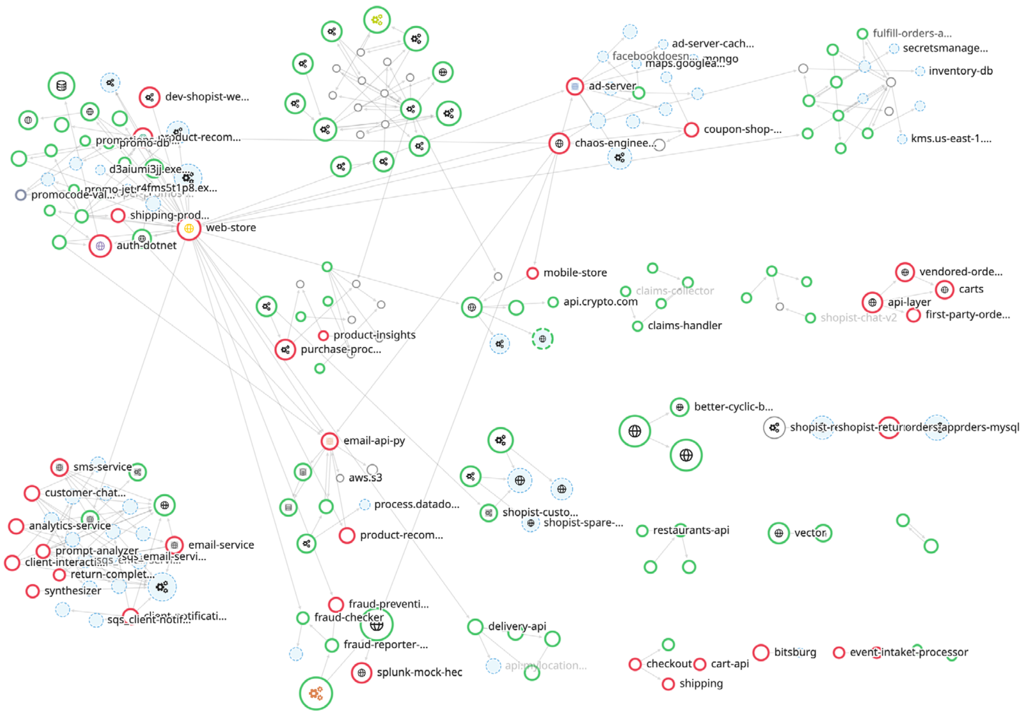
- **63% less annual downtime:** 139 hours vs. 370 hours
- **27% fewer engineering hours lost to disruptions:** 11 hours vs. 15 hours (per 40-hour week)

To achieve this comprehensive level of insight, teams are analyzing backend and frontend data with the combination of Application Performance Monitoring (APM) and Digital Experience Monitoring (DEM) solutions (e.g., Real User Monitoring (RUM), Synthetics, and Session Replay). Together, APM and DEM provide the data and context needed to gain deep visibility into every layer of an application, capturing the interconnectedness of services, infrastructure, and user interactions.

In this eBook, we'll dive deeper into why embracing end-to-end observability is essential and how to do so, ultimately showcasing the best way to push your organization to consistently deliver reliable, high-performing applications while minimizing downtime and optimizing the end-user experience.

2 Complexity of Modern Applications

Building and operating modern applications is a balancing act between innovation and complexity. On the one hand, organizations can now integrate microservices, micro-frontends, and third-party services at unprecedented speed, fueling rapid feature delivery and a frictionless user experience. On the other hand, the intricate web of dependencies, frontend layers, backend services, multiple databases, and managed cloud offerings demands more sophisticated approaches to monitoring and troubleshooting. These challenges only grow as continuous integration/continuous delivery (CI/CD) pipelines accelerate deployment frequency, leaving little time to diagnose issues or ensure optimal performance at every layer.



The Multilayered Challenge

At a high level, modern applications can be divided into three core layers:

- **Frontend Dependencies:** Often consisting of micro-frontends or intricate mobile applications, user interface components, and integrations with analytics or user behavior tracking tools. Here, teams focus on the end-user experience, measuring availability and performance KPIs such as load times, crash rates, conversion rates, and time to completion of critical user flows.
- **Backend Dependencies:** A constellation of microservices, containerized runtimes, managed cloud services, and third-party APIs that handle core business logic, data processing, and system operations. Distributing responsibilities across microservices offers flexibility and scalability but also increases the need for robust communication and monitoring.
- **Databases:** Ranging from traditional SQL databases to NoSQL and cloud-native storage solutions, each of which must remain performant and highly available to support real-time user demands. Managing data across multiple specialized databases can optimize performance but requires careful orchestration and observability to prevent silos.

In traditional monolithic architectures, these layers are often bundled and deployed as one large artifact, with limited separation of concerns. Over time, the demands for increased agility, scalability, and flexibility spurred a shift toward microservice architectures, micro-frontends, and composite mobile UIs, where individual components are designed, developed, and deployed independently.

This evolution allows teams to iterate faster and reduce interdependencies across the application stack, however, each of these layers is often owned by different teams with different expertise, tools, and objectives which introduces new hidden costs:

- **Limited Performance Data:** When data is collected piecemeal, teams lack a clear understanding of how users actually experience the product. Without user-centric metrics, it's impossible to fully validate whether performance issues impact real-world scenarios and hence a businesses bottom line.
- **Fragmented Troubleshooting:** Isolating the root cause of a problem frequently requires manual correlation across multiple tools. This not only consumes valuable time but also increases the likelihood of missed clues or conflicting data, prolonging incidents and causing user frustration.
- **Siloed Teams, Siloed Perspectives:** Different groups may rely on distinct tools and metrics, making collaboration more difficult. Without a single source of truth, each team may interpret performance or reliability differently, creating misalignment on priorities and strategies.
  - At an international media company, each team used different monitoring tools, preventing full visibility across applications. Without unified distributed tracing and real-time monitoring solutions, engineers had to manually manage live interactive shows, relying on "war rooms" and overprovisioning of resources. After implementing their end-to-end solution, they can now avoid "war rooms" and have not only gained a 360-degree view across all applications but have also reduced their costs by 78%.
  - A large real estate firm was experiencing extensive context switching and slow RCA due to disparate tools for tracing, logging, and frontend monitoring. After implementing a unified infrastructure, APM, logs, and UX monitoring solution, their MTTR for high-severity incidents fell from 2h 26min to 16min—a reduction of nearly 90%.

Moving Toward a Unified Perspective

Applications will continue to expand and diversify, and so will the challenges that come with it. How do teams consistently ensure their product is truly performing well end to end across all these distributed components? That's where end-to-end observability comes in.

3 The Power of End-to-End Observability

What Is End-to-End Observability?

At its core, end-to-end observability involves consolidating:

METRICS	Key performance indicators from infrastructure host/resource metrics, microservices, containers, databases, and third-party services.
TRACES	In-depth diagnostic information that maps the journey of requests through various applications and services.
FRONTEND AND USER EXPERIENCE DATA	Real-time analytics on load times, interaction patterns, conversion rates, and more performance indicators from mobile and browser-based clients.

By unifying this data, teams gain an all-encompassing understanding of how their technology stack operates, how users interact with it, and how potential bottlenecks or errors can ripple through the entire system. This broad scope provides a holistic view of system health, enabling teams to:

- **Understand how user requests flow through various microservices and databases.**
- **Correlate application performance with user behavior.**
- **Pinpoint and resolve issues that occur at the infrastructure, application, or user-interface level.**

Why Is this Important?

RUN YOUR BUSINESS WITH CONFIDENCE
Gaining unified visibility into every component instills confidence in your operations. When you know what's happening across the entire stack, you can better anticipate and respond to performance degradations before they affect end users.

RAPID TROUBLESHOOTING AND REDUCED MTTR
By correlating frontend data (e.g., loading times, user journeys) with backend telemetry (e.g., infrastructure, traces, logs), you can quickly identify whether an app crash or other issue originates in the UI, a specific service, or the infrastructure layer. This integrated approach accelerates mean time to resolution (MTTR) by consolidating investigative work into a single, cohesive workflow.

PROACTIVE DETECTION AND INTELLIGENT ALERTING
End-to-end observability tools allow you to create alerts and dashboards based on both frontend and backend performance indicators. Instead of waiting for customers to report issues, your teams receive proactive notifications when a critical service slows down or when user experience metrics deviate from the norm. This real-time insight ensures that minor hiccups don't become major incidents.

DRIVE BUSINESS GROWTH AND ENHANCE BRAND PERCEPTION
A seamless user experience is pivotal for retaining customers, encouraging brand loyalty, and driving revenue. With end-to-end observability, you can identify and resolve the performance bottlenecks that frustrate users and cause churn. Improving the product experience translates directly to higher conversions, better retention, and opportunities for sustainable growth.

UNIFIED VISIBILITY ACROSS MOBILE AND BROWSER APPLICATIONS
Modern applications often span multiple platforms. Whether your users are on mobile devices or desktops, capturing performance data from each interface is critical. End-to-end observability bridges these environments, giving you clarity into how each platform performs and how it impacts the broader system.

4

How to Achieve End-to-End Observability

Understanding the benefits of end-to-end observability naturally leads to the question: how can teams practically achieve it? In order to correlate your backend and frontend data, teams must collect and connect data from Application Performance Monitoring (APM) and Digital Experience Monitoring (DEM) tools.

What Is Application Performance Monitoring?

Application Performance Monitoring (APM) provides visibility into the health and performance of applications across code, infrastructure, and third-party services.

APM	
Feature	**Key Questions to Ask**
By collecting and analyzing telemetry data such as requests, latency, and errors, APM helps teams understand how requests flow through their systems, pinpoint the root cause of performance issues, and optimize application responsiveness.	– Which services or endpoints are seeing elevated error rates, and how is that impacting users? – What is the normal latency for key services or endpoints, and are we seeing deviations from that baseline? – Are dependencies like databases, APIs, or third-party services causing slowdowns or failures in upstream services?

What Is Digital Experience Monitoring?

Digital Experience Monitoring (DEM) prioritizes the user's perspective rather than focusing solely on backend application traces. DEM tools measure the availability, performance, and quality of the user experience of critical applications.

Generally speaking, a DEM suite must include **Real User Monitoring (RUM)** and **Synthetic Testing** tools as they both play a major role in detecting and reducing frontend issues.

Lets take a closer look at the two:

RUM	
Feature	**Key Questions to Ask**
– Captures real user interactions within web and mobile applications, including metrics such as load times, error and crash-free rates, and platform-specific KPIs such as Core Web Vitals, ANRs, and app hangs. – Records user session visuals using Session Replay, providing deeper context for frontend issues.	– Which part of the loading process is contributing most to the overall delay? – Is our page delivering content quickly enough to keep users engaged? – Is there a correlation between LCP delays and user drop-offs?
SYNTHETIC MONITORING	
Feature	**Key Questions to Ask**
Simulates user traffic in order to help teams proactively detect issues with key endpoints and user journeys. This enables them to catch issues such as slow response times, elevated error rates, and broken endpoints earlier in the development process and prevent breaking changes from reaching end users in production	– Are my frontend user workflows behaving as expected? – Are my APIs and endpoints available anytime and from anywhere? – How do I get ahead of faulty behavior like regressions, broken features, high response times, and unexpected status codes?

Integrating these complementary tools ensures teams gain comprehensive visibility across both frontend and backend systems, allowing them to correlate data effectively, optimize application responsiveness, enhance user satisfaction, and ultimately deliver seamless digital experiences.

5

Real World Use Case

booksy

Booksy has transformed the way beauty and wellness professionals connect with their clients. Today, it operates one of the largest appointment scheduling and business management platforms in the industry. With more that 400,000 service providers and nearly 40 million consumers across 20 countries, Booksy's platform processes millions of appointments monthly through its mobile and web applications.

As Booksy continued its rapid global expansion, it faced mounting challenges in maintaining consistent performance and reliability across its diverse tech stack. With three primary platforms—Android, iOS, and web—and mobile usage dominating customer interactions, the need for comprehensive monitoring became increasingly critical. With its previous observability tools, backend teams didn't have access to frontend errors, which led to siloed communication. At the same time, tool sprawl required teams to context switch and manually correlate data across multiple tools. The SRE team needed to reduce the burden of maintaining their in-house monitoring solutions while building a more holistic view of their observability landscape. Most critically, they needed to improve application availability and performance monitoring across their expanding global footprint.

The Booksy team had been using Datadog Application Performance Monitoring (APM) for backend application performance monitoring for four years, but were using another product for frontend monitoring. The SRE team recognized an opportunity to consolidate observability and decided to expand their Datadog implementation to include Real User Monitoring (RUM) and Error Tracking, creating a single point of truth to correlate all metrics.

The SRE team uses RUM daily to address previously undetectable challenges. For instance, before RUM, they were unaware of the prevalence and user impact of random iOS user logouts. **By using RUM, they were able to decrease these occurrences from 100 cases per day to zero.**

By centralizing observability across teams, Booksy has created a single source of truth for monitoring its complex global infrastructure, improving efficiency, minimizing downtime,
and optimizing platform performance. It has also enabled Booksy to scale more effectively by using Datadog metrics to inform infrastructure decisions and optimize resource allocation. **As a result, the company's deployment frequency has increased from once weekly to 5–6 deployments per day.**

In addition, RUM now provides a unified language within the organization so everyone can access and use the same relevant data. For example, C-suite executives and managers use high-level SLOs and metrics to assess business impact, while developers can dive deeper into the same data to investigate specific issues by examining errors, traces, logs, and events.

READ THE FULL CASE STUDY >

6

Better Together: APM & DEM (Conclusion)

The ability to swiftly detect, diagnose, and resolve application issues is no longer just a competitive advantage—it's a business imperative. End-to-end observability addresses the evolving challenges of modern software environments by breaking down silos, providing unified visibility, and ensuring that organizations can proactively manage application performance across all layers of their technology stack.

By integrating Application Performance Monitoring (APM) with Digital Experience Monitoring (DEM), organizations not only gain technical clarity but also foster a culture of collaboration, accountability, and continuous improvement. Ultimately, embracing this comprehensive approach positions your organization to deliver consistently exceptional user experiences, minimize costly disruptions, and drive sustainable business growth.

The time to elevate your observability strategy is now, and your customers, teams, and bottom line will thank you for it. To learn more about how APM and DEM work together, check out Datadog's APM x RUM and APM x Synthetics blog posts.

SIGN UP FOR A FREE TRIAL

DATADOG