Better Apps, Happier Customers:

Performance Monitoring Decoded

SMARTBEAR

Contents

Introduction

What is Performance Monitoring?

Given the increasingly complex and distributed nature of modern software, many businesses struggle to meet the growing expectations of their users. Keeping up with identifying and fixing errors and performance issues can make these challenges even tougher.

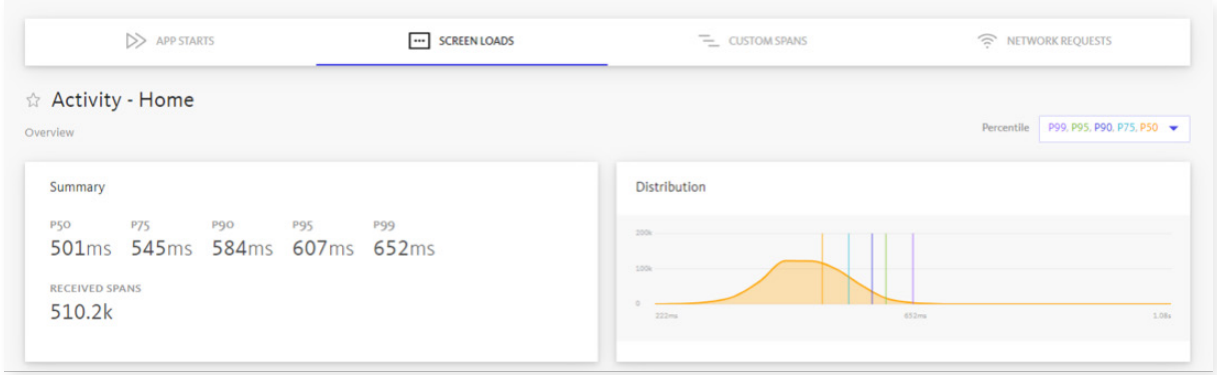
To handle these problems, many companies use application performance monitoring (APM) tools to check the stability of their applications and find any errors. This allows developers to prioritize the problems that need fixing right away. Server-side or infrastructure monitoring looks at the back-end services, while real user monitoring (RUM) focuses on the front-end user experience.

Many of the APM tools available today are made with DevOps professionals in mind, providing solutions like application architecture insights, container monitoring, and service monitoring. Developers, on the other hand, typically need deeper insights into specific areas of their applications so they can track down the root cause of any issues. This eBook primarily focuses on the observability needs and practices of software developers; read on to learn more about real user monitoring, server-side monitoring, and distributed tracing.

Benefits of Utilizing Performance Monitoring

Performance monitoring is critical for modern businesses offering SaaS or consumer applications. There are numerous benefits of implementing a performance monitoring solution for your team:

- **Quickly Surface Issues:** Performance monitoring enables organizations to rapidly identify performance issues impacting users – even if they don't report them. With performance monitoring, you no longer need to rely on support tickets to learn about poor end-user experiences.
- **Monitor Key Operations:** With performance monitoring, engineering teams can monitor start times, screen loads, network requests, database queries, and any other operations chosen to instrument.
- **Detailed Actionable Data:** When using performance monitoring, the unit of work in distributed systems is called a span. Span details are captured and displayed intuitively, showing engineers exactly where time was spent in an operation, helping them get to the heart of the problem as soon as possible.



Implementing performance monitoring helps teams monitor key operations in their applications, like screen loads and app starts.

Real User Monitoring

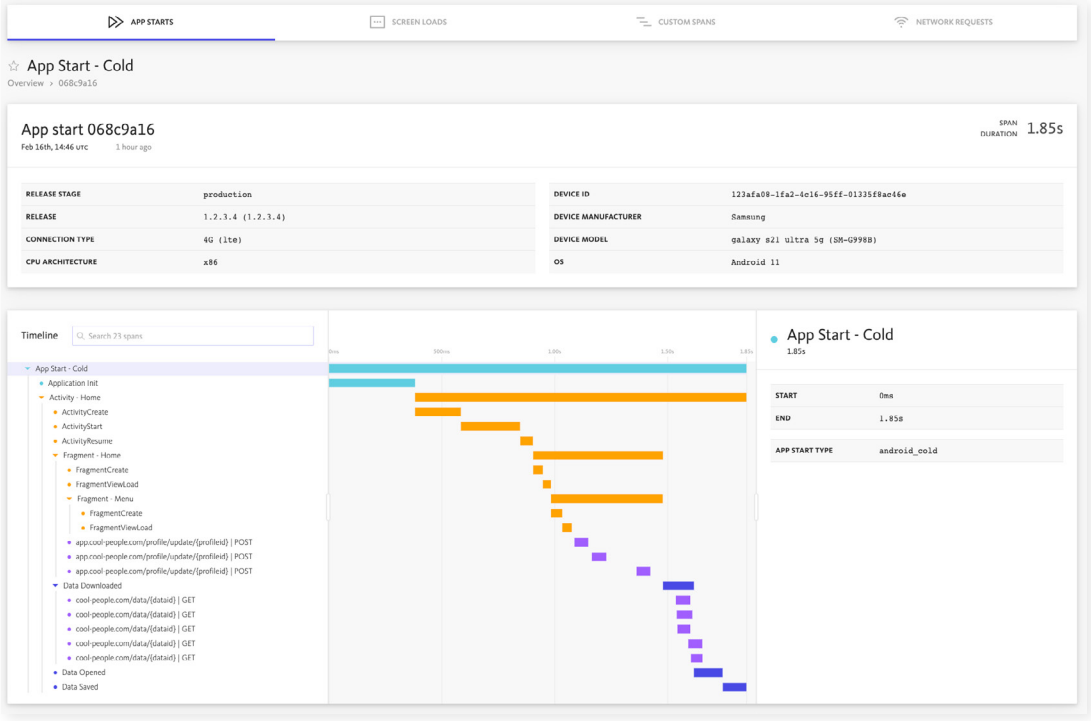
What is Real User Monitoring?

Real user monitoring (RUM) involves collecting detailed data about a user's interaction with an application. Under the hood, RUM involves measuring the time taken for key operations within a user interaction. A developer might use RUM to monitor an online banking site to detect any increase in page load times or to find out why a shopping app's payment screen isn't converting.

RUM provides developers with critical insights into their users' experience. Understanding the time that users spend within an application allows teams to identify the most important areas and features and prioritize problems accordingly. From there, developers can dig into specific issues to find out why they happened and focus on prevention.

Example Problem: Let's imagine a user is trying to complete a purchase on a shopping application on Android. The payment screen is the most important on the app – that is where conversion happens. If this area of the app is lagging or loading incorrectly, the user will likely give up and cancel their purchase.

Solution: By using a RUM tool, developers will be able to pinpoint the exact area of the application that is performing poorly – like a slow network call causing the tax calculation to lag. The developer can then draw a direct line between the problem and a reduction in revenue.



BugSnag real user monitoring allows developers to drill down into specific performance issues to pinpoint the exact issue and fix it faster.

Why is Real User Monitoring Important?

Software users are more impatient than ever, and applications have become increasingly more complex. This combination can cause chaos for your app's performance. A slow, glitchy app isn't just a bad user experience, it can also damage your bottom line.

Does your mobile app start up super fast, or is it stuck buffering? Do important operations take more than 3 seconds to load? Does your web app load slowly on popular browsers?

Over time, these frustrating user experiences snowball into a profoundly negative impact on your business:

70% 70% of mobile users quit an app if it takes too long to load[1]

40% 40% of users will switch to a competitor after a frustrating mobile experience[2]

1s Every 1 second delay in page load time may lead to 20% lower conversion[3]

[1] https://techjury.net/blog/app-usage-statistics/
[2] https://outerboxdesign.com/web-design-articles/mobile-ecommerce-statistics
[3] https://thinkwithgoogle.com/marketing-strategies/app-and-mobile/

But have no fear – just as a poor performing app leads to negative business outcomes, a great performing app boosts them. Because consumers are impatient, speed is everything. When users have a seamless, easy, and quick experience using your application, they are more likely to return.

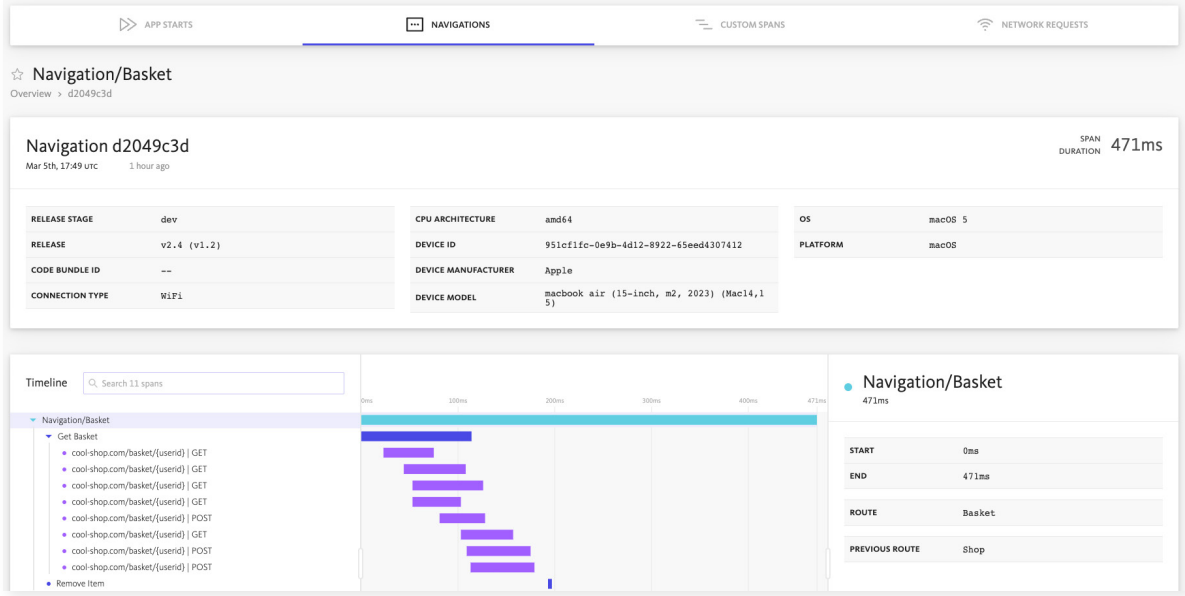
Over time, these satisfying user experiences will help your business thrive:

65% 65% say a positive user experience is more influential than great advertising[4]

86% 86% of mobile buyers are willing to pay more for a great customer experience[5]

5% 5% increase in customer retention can lead to a 25%-95% increase in profits[6]

Implementing real user monitoring allows developers to see exactly what their users see, making it easier to provide a seamless app experience. Happy users produce better app store ratings and search engine trust, skyrocketing your brand reputation and credibility.



Real user monitoring helps developers identify critical issues in key areas of an application, such as the shopping cart on an ecommerce app.

[4] https://pwc.com/us/en/services/consulting/library/consumer-intelligence-series/future-of-customer-experience.html
[5] https://superoffice.com/blog/customer-experience-statistics/
[6] https://businessnewsdaily.com/16027-customer-retention-rate.html

Server-Side Monitoring

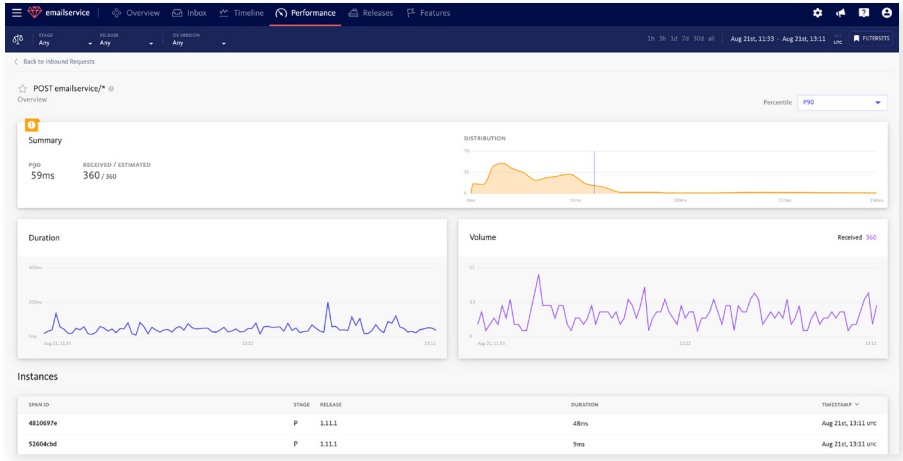
While RUM is a critical tool to monitor the front-end side of an application, developers also need a way to ensure that their back-end systems are up and running at full capacity. After all, back-end systems feed data into your front-end applications, so a server slowdown will ultimately impact your users' experience.

Server-side monitoring is a critical aspect of performance monitoring that focuses on the health and performance of server-side infrastructure. Using it, developers can monitor and analyze data from the back-end, such as CPU usage, memory consumption, disk I/O, network activity, and database calls.

As a result, server-side monitoring helps identify potential performance problems in real-time. This visibility is essential for developers working to maintain optimal performance and provide a seamless user experience. Utilizing server-side monitoring tools can help developers and IT teams quickly detect and respond to issues using dashboards and alerts. As a result, downtime is minimized, and high-performance standards are maintained.

Much like RUM, server-side performance monitoring's importance increases as applications become more complex and distributed. By proactively monitoring server health, teams can prevent minor issues from snowballing into major outages. This proactive approach reduces the risk of downtime and helps optimize resource utilization.

Server-side monitoring also contributes to improved security and compliance by detecting anomalies that may indicate security breaches or vulnerabilities. By addressing these issues promptly, organizations can protect sensitive data and maintain regulatory compliance.



BugSnag's server-side performance allows developers to drill down into inbound requests like this POST request to an email service written in Ruby.

Distributed Tracing & OpenTelemetry

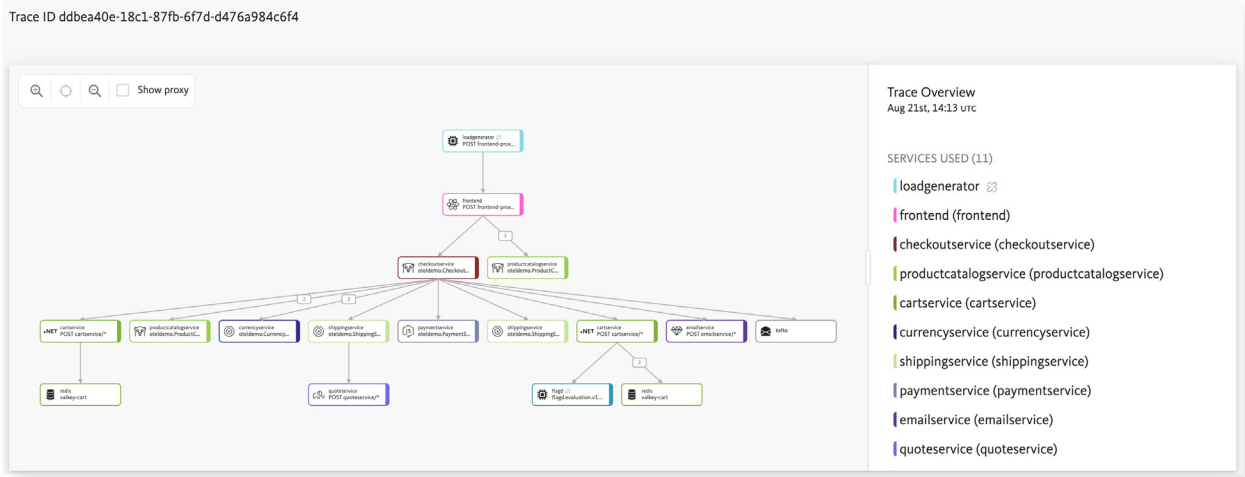
Connect the Dots with Distributed Tracing

Distributed tracing is a solution for tracking requests moving through your system, end-to-end. It allows you to track which services a request interacts with, how data flows between these services, and records any errors encountered along the way.

In distributed systems, requests span multiple services that can exist in various environments, from containers to cloud infrastructures. This complexity often leads to performance challenges and makes root-cause analysis a daunting task. Distributed tracing is the remedy.

Think of traces as detailed call stacks for microservices. They can gauge the duration of each request, the components and services they engage with, and the latency introduced at each step.

With tracing data, engineers gain invaluable insights into request journeys, service relationships, and potential bottlenecks. This newfound visibility empowers them to not only identify and resolve issues but also to optimize system operations.

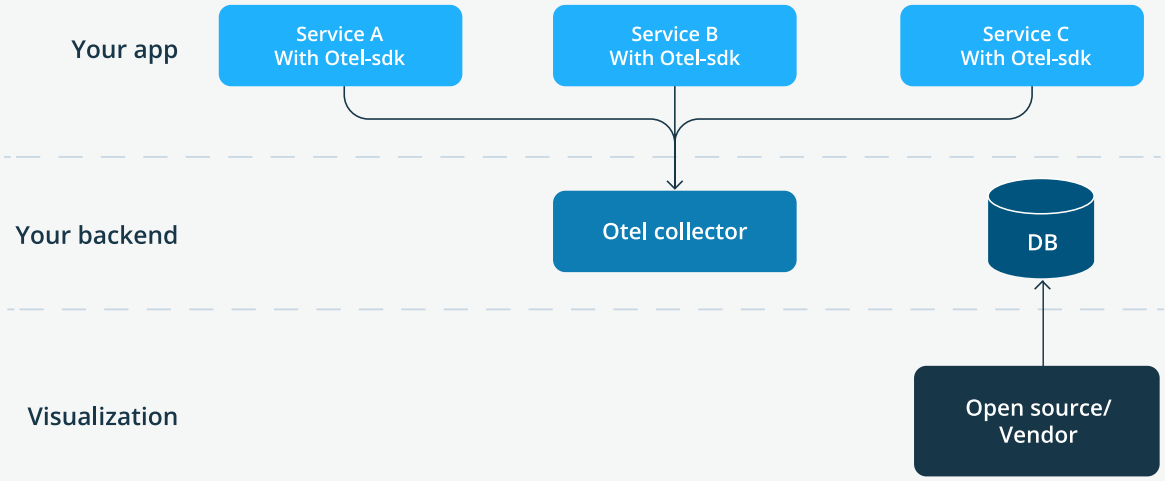


BugSnag's distributed tracing solution is powered by OpenTelemetry, and allows developers to bridge the gap between front-end and back-end, making it easier to pinpoint root causes.

OpenTelemetry

Distributed tracing is only as effective as the data you have and whether it is compatible across all systems. OpenTelemetry (OTel) solves that problem. As a dynamic ecosystem of APIs and SDKs, it allows you to capture and export the essential elements of observability: traces, logs, and metrics.

Supported by the Cloud Native Computing Foundation (CNCF), the driving force behind Kubernetes, OTel powers you to instrument your cloud-native applications. This instrumentation is your ticket to gather invaluable telemetry data, offering profound insights into your software's performance and behavior.



OpenTelemetry basic stack architecture

OpenTelemetry captures data from each service it's installed on, creating spans that are aggregated into traces. Analyzing these traces using external tools provides insights into system performance and service behavior. Engineers no longer need to manually add traces, making it a valuable addition to complex architectures with diverse data types.

OpenTelemetry stands out for three pivotal reasons:

- **Open-Source Community:** Thriving on collaboration, OTel boasts a vibrant open source community committed to transparency and innovation.
- **Unified Telemetry:** It seamlessly consolidates logs, metrics, and traces, acting as the cohesive force bringing them together.
- **Standardization:** OTel adheres to a single specification, ensuring consistency across vendors and platforms and establishing a standard framework for observability.

OpenTelemetry is gateway to unparalleled insights and observability in modern software systems. OTel is currently transforming the way that teams practice observability, and the results are far-reaching.

Implementing a Performance Monitoring Strategy

Performance monitoring is relatively easy to implement using today's solutions. The real challenge is converting the troves of data you'll receive into actionable insights. You'll also need to integrate RUM into your existing quality assurance and development workflows to be effective.

Here are some best practices to implement performance monitoring for your applications.

- **Start with Business Objectives:** Determine the business KPIs you'd like to improve (e.g., reducing user churn or reducing abandoned carts by 10%). Link these business goals to technical goals you can measure with real user monitoring solutions and metrics.
- **Test New Product Releases or Features:** Performance monitoring is an excellent way to test new product features. Use staged rollouts to test new releases or features (e.g., feature flags) with a limited set of users before rolling them out to everyone, and use your monitoring tools to determine when they're ready.
- **Foster Open Communication:** Performance monitoring works best with open lines of communication between business and technical teams. Technical teams need business metrics to know what to test, while business teams, especially marketing departments, can glean helpful insights from technical data.
- **Don't Rely Exclusively on Performance Monitoring:** It should be part of a more comprehensive quality assurance strategy. For example, performance monitoring isn't a substitute for automated or ad hoc testing, since it generally only uncovers issues after deployment. Instead, it's a valuable addition to an overall observability strategy.

What's next for performance monitoring?

Traditional observability approaches, like performance monitoring and digital experience monitoring, may struggle to keep up with the increasing scale and complexity of future applications. As the amount of telemetry data grows, it becomes increasingly expensive and complex to navigate.

Enter Artificial Intelligence (AI) and its potential to revolutionize these practices. The concept of AI-driven observability explores how AI capabilities can enhance and transform traditional software observability tools and practices. This approach leverages AI techniques to improve various aspects of observability, from data collection and analysis to visualization and insights.

As with any technological advancement, there will be challenges to AI-driven observability. From data privacy and security to trust and data ownership, there are many facets of AI-driven observability that developers will need to work through while unlocking the true potential of this technology.

The BugSnag Difference

BugSnag is an observability tool built by developers, for developers, that offers performance monitoring, error monitoring, and application stability management.

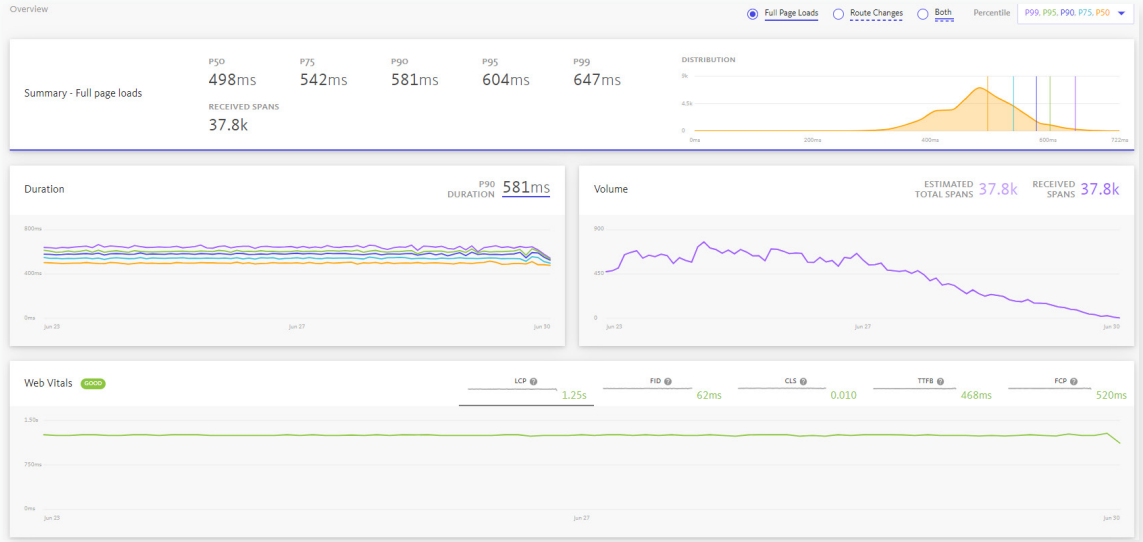
BugSnag's performance monitoring solution spans the front-end to the back-end. With real user monitoring, developers get the insights they need to quickly identify and prioritize issues impacting your users on the front-end. On the back-end, developers can monitor the speed and efficiency of their server-side activity.

As an OTel native solution, BugSnag provides users with the ability to fully own and instrument their data. OTel powers BugSnag's distributed tracing capability, providing developers with the entire picture of what occurred during an error or performance issue.

The uniqueness of BugSnag's distributed tracing offering lies in its true end-to-end nature, which is increasingly hard to achieve in modern architectures. By tracing issues all the way back to the user and tying everything together, BugSnag provides a comprehensive view of an application's performance. This is essential for developers and enterprises who need to ensure that every aspect of the user experience is optimized

"BugSnag gives valuable insight into the real-world performance of Concepts, our award-winning app, across the many devices and platforms we support. Ensuring that our app is fluid and responsive is important to our customers and this is a powerful tool to help deliver on that promise."

– David Brittain, CEO, TopHatch



BugSnag's intuitive UI makes it easy for developers to get started monitoring the performance of their applications.

Take control of your user experience with performance monitoring from BugSnag – try it now with a 14-day free trial.

Start 14-Day Free Trial

Request a Demo