Real-World Superheroes Needed: Modern APM for Developers

An Intellyx Whitepaper for Sentry
by Jason English, Principal Analyst
March 2023

Have you noticed that lately all major motion pictures follow a comic book formula?

You know that story, where an ordinary person is transformed into a superhero? It seems like that's the only movie plot the studios will greenlight these days. Modern application development shares this theme.

Today's conventional APM (application performance monitoring) and observability tool vendors aren't providing devs with the powers they need to maintain the health of their apps.

Therefore, devs need to transform themselves, and gain 'superpowers' of laser-vision error awareness and supersonic SRE response times to keep customers and the business happy. Is this plot line realistic, or have we jumped the shark?

The pitch? Our humble developer is just coding an app, when they are suddenly bombarded with metrics and alerts from a server-side or application-focused APM tool. They must then survive a painful training montage to gain IT operational superpowers over system and networking infrastructure to defeat the villain before people's lives are ruined.

In the real world, investing so much developer time and energy on the steep learning curve of IT operational awareness may offer some limited performance improvements. But our story won't end happily if we think doing observability the old way will save the world.

This paper will discuss how modern APM (application performance monitoring) can meet developers where they are and improve developer experience, with realistic superpowers that make their daily coding and integration work fit into the quality, performance, and resiliency objectives of the software they are building.

Can we flip the script on observability for app developers?

The developer challenges of reading operational signals

User experiences are notoriously hard to performance test with old server-oriented monitoring tools, especially within highly dynamic front-end websites and mobile applications, where there are infinite combinations of data and interactive elements appearing as on-screen symptoms for end user delays and interruptions.

Ops teams are shorthanded.

Since reliable testing results are so difficult for developers to replicate in modern application environments, organizations started leaning on Ops teams and SREs to use APM tools to produce more post-release infrastructure monitoring data and alerts.

This created a real logjam for new software releases that are valued by customers. Since most companies have already cut bottom-line IT costs to the bone, there's never enough Ops people to validate the results of a firehose of APM data. Perhaps developers can use the same tools?

Legacy APM tools don't suit the needs of today's developer.

Developers are being pushed to build more often and ship more features, faster, in multiple form factors, often without adequate testing time or resources. Executive and customer imperatives for agile software delivery impact hiring decisions, and therefore devs responsible for coding software usually outnumber ops professionals by a factor of 10-to-1 or more.

Despite this imbalance, developers still don't have an inclination to use legacy system-level monitors, network traffic analyzers and real user monitoring tools that don't have their needs in mind. These tools were built for DevOps engineers and SREs to get reports of application flaws and misconfigurations in the infrastructure, but they don't point out the root cause of errors within actual code.

Devs need tools that integrate with their workflow. They scarcely have time to look away from coding and validating what is in the pipeline, in order to chase down error logs and guess at root causes within antiquated APM tools.

Which dev owns the error?

Every CTO or VP of Engineering wishes they could keep the developer and approving manager of every critical software feature on automatic speed dial. While containers, automated cloud infrastructure-as-code and distributed API services make software delivery way faster, these advances also muddy the waters for who owns errors.

Once a problem is discovered, and if its root cause is identified, what happens next? Code that gets delivered to production doesn't come with baggage tags. It may take several Slack messages or phone calls for an incident manager to find out exactly which dev on a team made the pull request that created the error, and only then they can put a fix request into that developer's queue.

All of this investigation and communication eats up valuable time while the error is potentially impacting the end user. Modern APM should automate the mapping of code to deployment, so the true owner of an error can be notified using the company's collaboration or service management tools of choice.

Development teams have problems using production monitoring to trigger automated alerts and awareness within the CI/CD lifecycle, especially when they can't exercise a finished production environment.

Operations teams are also getting bogged down trying to pinpoint who is responsible for issues in production, when errors are baked in and they were never involved in coding the application. This double-edged sword of accountability causes failures, post-mortem incident blaming, and ultimately poor morale and attrition.

Front-end performance where it counts

Take any brand or company that depends on applications for improving revenue, customer satisfaction, or employee retention. The quality and responsiveness of the application's user experience will define the value of every interaction with that company.

To meet this challenge, we've seen a lot of hype around the ***observability*** space, especially over the last five years or so. It's hot because there is clearly a lot of value to be gained from understanding the inner workings of application performance, and making them transparent or self-reporting in some way.

However, observability still resides largely in the infrastructure world, and lacks cohesion as a 'space.' While observability tools can gather telemetry on everything from server-side agents streaming logs and metrics, to data layer observability, to extremely slick mission-control-style management dashboards of multi-cloud networks with AI-filtered analytics, what does this flood of back-end data mean to the developer?

What aspects of observability can help app builders?

There are many different kinds of observability on the market today, but when it comes to application developers, they are primarily looking to ensure both code quality and an excellent user experience.

- **Front-end observability** concentrates developer attention on the consequences of changes that impact front-end performance and responsiveness of an application as it is displayed in the client in front of customers. Feedback and telemetry from front-end observability influences development and design tradeoffs and considerations.

- **Back-end observability** has its place in considerations of database and network optimization, cloud provisioning and infrastructure provisioning, which can impact front-end performance. While back-end tools are great at grabbing events and metrics that can point out improvement opportunities for operations teams, making complex queries into warehouses full of log data is not where app developers will likely spend most of their time.

- **User experience observability** can include real user monitoring (RUM) and monitoring session data in an app, but the range of influences on UX is way

bigger than that. Our means of delivering UX have become as fragmented as the hybrid cloud backends our UIs must invoke. Devices and platforms are added and dropped daily, along with new development tools and libraries, new .JS frameworks, and new protocols, so the ability to capture and simulate real-world usage models is critical for success.

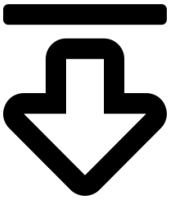
- **Code observability.** Developers need to abstract away the complexity of assuring quality and prioritizing changes within code when building or transcoding apps for so many different device/OS combinations. Observability must go beyond simple unit tests, code scanning, and linting – to pinpoint the impact of changes on supported systems while devs are coding – to head off issues even before a build happens in many cases.

Everyone should care about performance

User experience and performance is not the exclusive domain of front-end developers, back-end teams, or operations engineers. Since everything about a company's brand is on the line in front of customers, performance is everyone's job.

How can development and delivery teams think about APM differently?

- **Operations engineers** have had APM agents running on servers and in VMs for decades, as well as load balancers, ADCs and gateways that feed a steady stream of millions of system-level metrics into operational and security control center dashboards.

  Since the production environment is ultimately what matters most, why discard all of the utility of existing Ops-oriented APM tools for developers? Downtime, back-end network and database latency, and security events become another dimension that gets fed into the organization's service management and incident management collaboration tools.

  Ops teams are always short-handed, so the key to success is automating as much of this data stream as possible, and tagging which events are most relevant to builders, so they aren't left to self-search across the infinite points in time between when an error was introduced, and when it appeared as an exception.

- **Back end developers** also place monitoring agents in test and staging environments, as IaC (infrastructure as code) became a common practice in CI/CD software delivery pipelines.

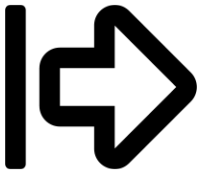
  

  These core developers led the way in embedding automated code checks and telemetry (logs, metrics, traces) into each release cycle, as they write the Java or .NET integration code behind an application that talks to an accounting system, or APIs for a third-party ad server, or a cloud-based data warehouse to privately write and read customer data records.

  Back-end devs also popularized automated unit and regression testing, code checking and linting tools within delivery pipelines that can point out structural

flaws and syntax errors in code. This is also worth keeping for front-end developers as it starts to shift-left application quality, but it may not be useful for finding the source of specific customer problems that appear on screen.

- **Application builders**, or **front-end developers** are now moving into prominence as the primary creators of multi-modal, customer-and-employee facing applications. Companies hire more builders because they are investing in dev teams that can own the total application experience.

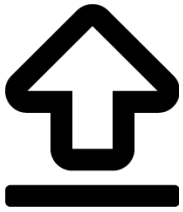


  Ensuring performance is extremely complex, when you combine every target device/OS, multiple API service interactions, and the ever-shifting array of frameworks and libraries today's builders must deal with. Modern backend technologies like Node.js, Laraval and Django could benefit greatly from code-based observability, but they are poorly covered by legacy tools.

  There's still a dozen different ways to write Javascript alone (React? Next.js? SwiftUI? Jetpack?) and talk to back-end APIs and systems, but devs are expected to test and debug each platform option, even for obscure languages and custom elements that exist in these environments.

  Builders want to spend more time coding new features, and less time configuring telemetry and looking at APM dashboards. **Sentry** provides a unique solution for this challenge, embedding instrumentation into code that provides performance insights within the flow of the developer's IDE, regardless of the language or framework.

- Finally, the buy-in of **business leaders and marketers** can make all the difference in modernizing APM, since customer desires and complaints drive product design requirements and budgetary priorities.



  In the old days of APM, lagging performance reports would drive management teams to allocate more IT budget on additional hardware or cloud capacity–but no company can spend their way out from under suboptimal code.

In the DevOps spirit of empathy, leaders should understand that if developers are accountable for the health of their apps in production, they should have their own code-level tools to find errors and performance issues. Managers can measure the ROI of shifting much of that assurance work left, rather than leaving it up to morale-crushing break-fix exercises for the most constrained ops resources.

Developer-first monitoring to manage media magic

Large media companies have some of the best case studies of modern APM in action, as they face a perfect storm delivering programming and recommendations to millions of customers, who are using many combinations of computers, mobile devices, gaming, and smart TVs with unique control features.

Disney+, a leading global entertainment streaming service, was doing regular code reviews and release retrospectives for performance, but they needed to pick up the pace of agile releases to adjust to fast-changing requirements from viewers. Slogging through so many error logs in each run became cost prohibitive, and simply allowing errors to flow into Slack or Github would create an untenable level of noise.

The firm instrumented error monitoring directly into their group dev environments using an open source solution from Sentry. With one line of code in the IDE, they started seeing how a particular click-over or video preview delay experienced on the customer's screen could correlate directly to a particular line of code, and even the committer of that code in the repository.

Based on that initial success, they moved to a managed service version of Sentry, which provides working developers with real-time insights into potential performance issues and root causes from thousands of real-world error logs and customer journeys.

The Intellyx Take

Developers have enough work on their plate coding and debugging apps without the burnout of digging through system logs within legacy APM interfaces that only an operational engineer could love.

In reality, developers are humans who simply want to make applications that work smoothly in front of customers. Modern developer-first APM solutions can give them superpowers to not only find the root causes of performance problems within their applications, but to trace errors down to the line of code, and the committer of that code.

A better developer experience means devs can focus on areas that make them happy, like building innovative new products and features at higher velocity, or perhaps even taking some time off without worrying about their apps crashing.

When developers can drill down and resolve customer issues within the flow of their everyday work, they gain superpowers that add incredible value to their organizations– with no X-ray vision or alien spider bites required.

About the Author

Jason "JE" English (@bluefug) is Principal Analyst and CMO at Intellyx, a boutique analyst firm covering digital transformation. His writing is focused on how agile collaboration between customers, partners and employees can accelerate innovation.

In addition to several leadership roles in supply chain, interactive, gaming and cloud computing companies, Jason led marketing efforts for the development, testing and virtualization software company ITKO, from its bootstrap startup days, through a successful acquisition by CA in 2011. JE co-authored the book Service Virtualization: Reality is Overrated to capture the then-novel practice of test environment simulation for Agile development.

About Sentry

For software teams, **Sentry** is essential for monitoring code health. From Error tracking to Performance monitoring, developers can see clearer, solve quicker, and learn continuously about their applications — from frontend to backend. Loved by more than 3.5 million developers and 85,000 organizations worldwide, Sentry provides code-level observability to many of the world's best-known companies like Disney, Peloton, Cloudflare, Eventbrite, Slack, Supercell, and Rockstar Games. Learn more at sentry.io or follow Sentry on GitHub, Twitter, Dribbble or LinkedIn.