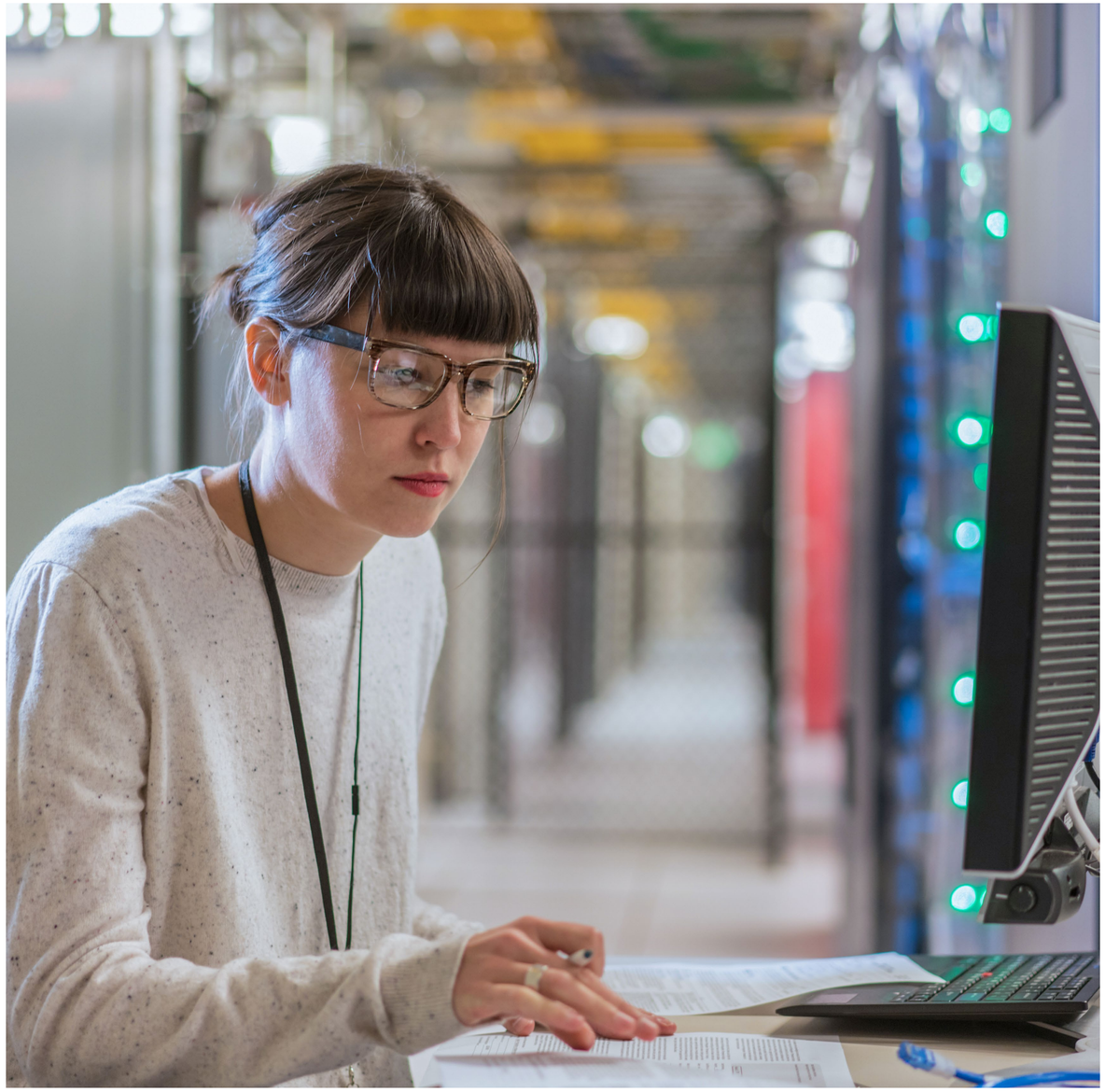
IBM Instana

Boost observability to streamline application modernization



Contents

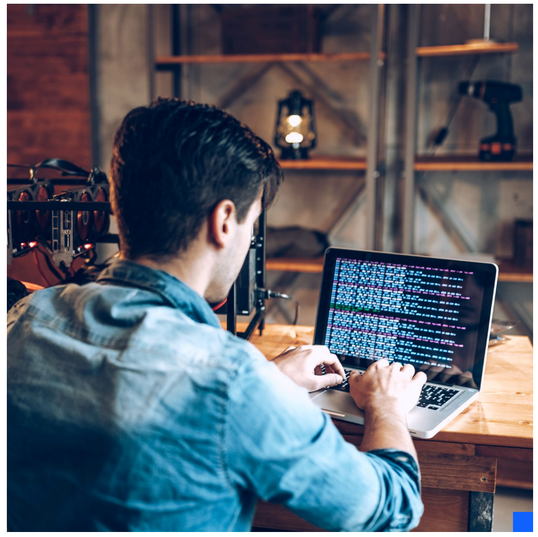
01

Overview

Everyone agrees application modernization is a hot topic. And one that most enterprises are discussing or will be soon. Application modernization is an involved process that includes an assessment of your current app portfolios, determining which ones are most critical to be modernized, then formulating a cost-effective plan to take action. Unfortunately, the reality of application modernization is usually more complex than what developer advocates convey on social media.

While container-based microservice applications have gained traction, many legacy applications still require migration. On-premises infrastructure and monolithic applications are often the starting point, with cloud-native applications being the end goal. However, each organization needs to chart its unique course based on its current state, requirements and feasibility. Significant migration is already taking place and is expected to increase in the coming years.

Elements of application modernization



Code
Writing and updating code for an application is an arduous task. Just getting it out the door is challenging. However, Murphy's Law tell us that eventually, the application will need to be updated while the code owner is unavailable. The owner of the application may also leave the company and no longer be available.

The new person could be in for a rude surprise. Custom code, personal comments, odd ordering, no comments, not using accepted syntax and poor formatting can all turn simple tasks into detective work. Outdated Java language could disable critical integrations on your website. And that could damage the business.

Application components
A monolithic application on bare metal servers can't become cloud native by magic. The application needs to be broken into microservices, its components placed into containers and those containers orchestrated with a tool like Kubernetes.

Just that process is a lot of work but it does provide a start for application components. The more complex the components, the more time that's required of developers and engineers, which often creates urgency to move to a cloud hosting service. Let them manage the infrastructure so your team can manage the applications.

Infrastructure
Remember when we stated that code, component and infrastructure changes can happen in parallel? If you're migrating an application to a cloud environment, you have several infrastructure application component options.

- **Rehost.** Lift and shift the application and rehost it in the cloud.
- **Replatform.** Host the application in the cloud and make minor infrastructure changes.
- **Repurchase.** Purchase software as a service (SaaS) versions of your applications.
- **Refactor.** Rewrite some or all of an application and possibly deploy a new application architecture, such as converting a service-oriented architecture (SOA) application to containers.

03

Beginning the modernization journey

Where do you start? It depends on where you are. Generally, you want to move to the next step on the sliding scale toward cloud native, but each step has its own set of challenges.

You are here	Your next step
Bare metal, on premises	Virtualized, on premises
Virtualized, on premises	Data centers
Data centers	Private cloud
Private cloud	Public cloud
Public cloud	Hybrid cloud
Hybrid cloud	Cloud native

Determining the success of each step in the process requires both a benchmark measurement of application performance before modernization and a new measurement for comparison after. Maintaining visibility to collect the same set of metrics at each stage can be challenging for some application performance monitoring (APM) tools.

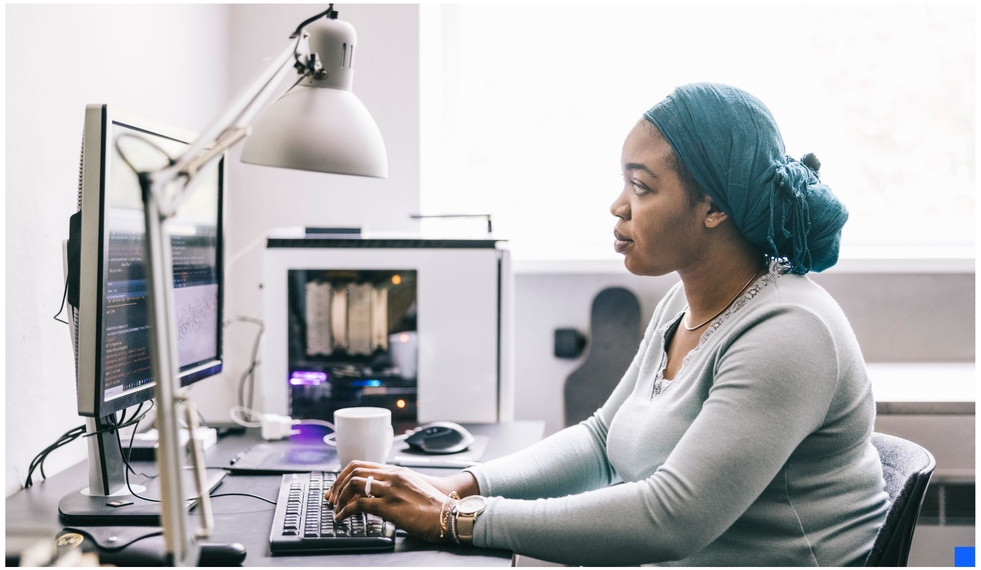
Defining legacy for this ebook

The phrase legacy application could have several different definitions. For this ebook, we're providing a definition of legacy applications that makes it easier for us to use legacy and legacy application than the longer, fully qualified name.

In this discussion, legacy application will mean some form of a Java Enterprise Edition (Java EE) or .NET (dot net) application, either deployed as a monolith or in an SOA environment. For that purpose, this ebook might also refer to Generation 1 and 2 APM tools as legacy monitoring or legacy APM. Any discussions of other legacy tools will be called out specifically.

04

Benefits of modernizing applications



Migrating code, application components and infrastructure involves complex decisions and requires a great deal of planning. Considering the difficulty, is this something you're prepared to take on? Here's why you should.

Modernizing applications is a key part of the digital transformation journey which helps automate manual processes. Automation can have a positive impact on both service performance and maximizing human resources. Processes that require frequent manual intervention can be slower, subject to more errors and more expensive to operate. Here are some reasons for modernizing applications.

- Optimize developer time.
- Accelerate innovation and the continuous integration, continuous delivery (CICD) pipeline.
- Reduce costs and redundancy.
- Become more responsive to user needs.
- Build more uniformity into the organization and processes.
- Deprecate older environments.
- Resolve issues quicker.

Optimize developer time

Many legacy applications are based in Java or .NET, but some lack the sophistication of newer applications. Updates and routine maintenance can be difficult for development teams. The lack of historical knowledge can make one missing developer a bottleneck for all work on an application.

As one developer takes over for another, the new developer can piece together an application with fixes and patches in that developer's own style. As a result, the inconsistencies are overwhelming, such as links to JavaScript function files in some places and JavaScript directly in the code in others. Simple fixes turn into hours spent poring over code to find the needle in the haystack.

When developers apply rigor and user-experience (UX) principles to their code, it's easier for anyone to work with the code or, better yet, let you automate routine work. Comments define what each section references in the application. Scripts and style sheets indicate what those elements do. And whether the originating developer uses tabs or spaces, it's consistent.

The result is less time figuring out code and more time doing work, less stress and more accomplishment, and better team performance.

Accelerate innovation and the CICD pipeline

In most cases, older applications are less resilient. As a result, developers often spend too much time dealing with emergencies to keep the application running instead of innovating for customers.

When developers complete more work early in a development project, they start a virtuous cycle of resource allocation that helps keep projects on time and reduce errors. By contrast, when schedules slip, they start a vicious cycle of scrambling to address emergencies. That means devs run tests at each stage manually instead of running continuous automated tests. They manually instrument code instead of automating it. They painstakingly document their infrastructure instead of using a tool that can automatically discover it. This process can cause misleading test results. When that happens, code can hit production before it's ready.

Organizations are applying architectures designed for running applications—such as containers, microservices, serverless computing and Kubernetes orchestration—to application development and delivery. Automation and orchestration tools like Jenkins, Red Hat® Ansible®, Chef and Puppet can help keep teams on the same page. Each major public cloud platform—Amazon Web Services (AWS), Microsoft Azure and Google Cloud—offers Kubernetes-managed container processing as a service.

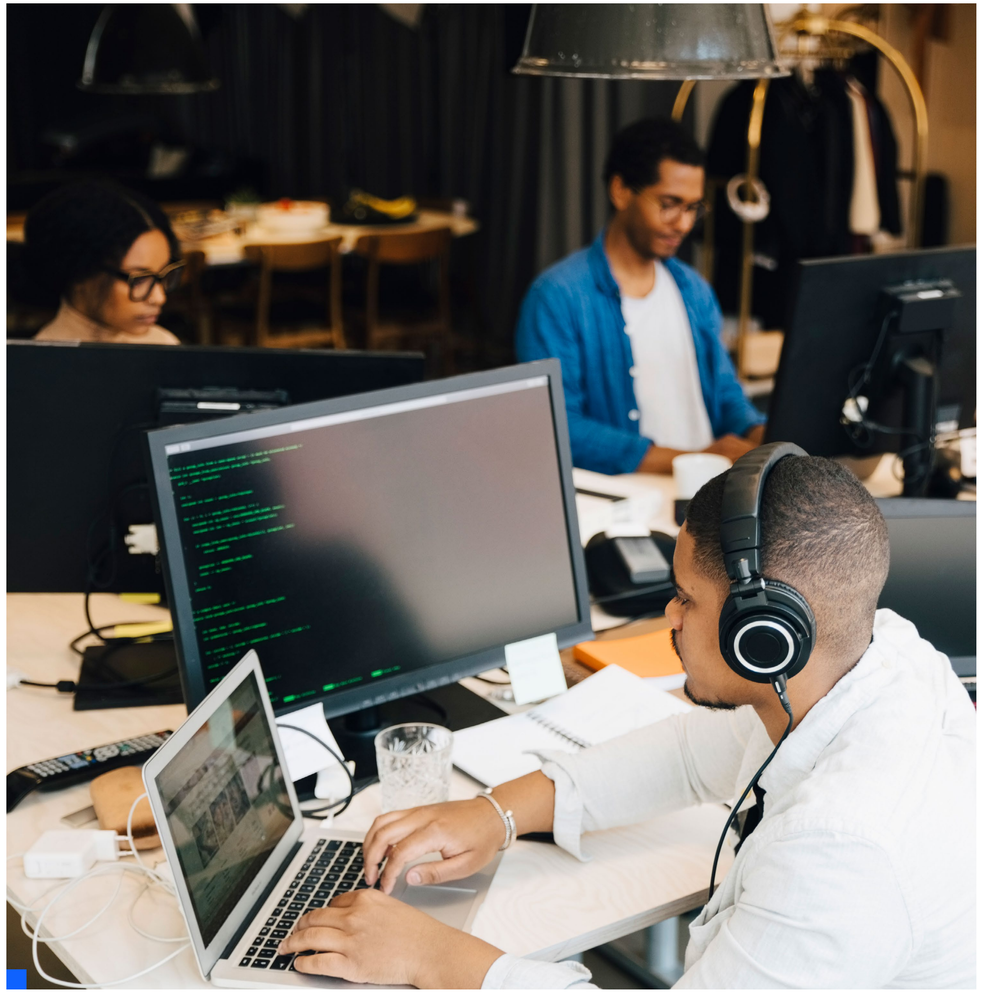
Reduce costs and redundancy
One of the advanced steps organizations take before moving to cloud hosting is to run data centers. Global data centers are great for load balancing and redundancy, but they can be very expensive.

Data centers also often come with long-term contracts. Migrating to the cloud is a time-consuming process. To allow yourself adequate time, you should start at least one year before your data center contract expires.

And if you thought data centers offered redundancy, try dozens of containers with microservices in the cloud. That's redundancy. Some large corporations still use on-premises architectures and they make them work. But the pattern toward cloud adoption is unlikely to reverse.

Become more responsive to user needs
The requirements and wish lists for customers and employees should be your guiding light. Customers have little patience anymore for delays or interruptions—both online and in apps.

For example, if your web page takes too long to load, chances are good a customer will simply click somewhere else. For online retailers, this can result in significant revenue reductions from lost sales. For optimum results, aim for about 2–4 seconds for a page to load or an app process to complete.



Build more uniformity into organization and processes
Remember how developers create their own fixes and patches to cobble together code? There are real-world consequences. Different applications are coded differently, making it difficult to hold them to minimum standards. How can you apply a standard to an application that doesn't use any of the components you're trying to standardize? Organizations modernize applications to bring them into alignment, so they can be brought into compliance.

Deprecate older environments
Here's a scenario. An application is built on Java 11. The current version is Java 20. If the team is going to upgrade the version and improve code accordingly, why not take this opportunity to modernize the application in other ways? Break it into microservices. Move it to the cloud. Make it a cloud-native application.

Resolve issues
Distributed components enable incident responders to disable some containers or other pieces without turning off all of them. In other words, distributed, cloud-based microservice application components remove the single point of failure that plagues monolithic applications. Containers also enable developers to isolate and test pieces of an application without affecting the whole thing.

Orchestrators such as Kubernetes also make it easier to apply fixes to applications without customer downtime, which is a huge win for the provider and user alike.

05

Why companies make the mistake of not modernizing

Companies most often delay application modernization for two main reasons: short-term pain and the difficulty of measuring progress. Balancing short-term discomfort with long-term benefits can be challenging, as the former is more certain while the latter can be a gamble. Measuring progress along the entire modernization journey can also be a significant obstacle.

Short-term pain
Here's a quick list of reasons organizations delay or cancel application modernization projects.

- **Client hesitation.** The modernization process usually requires some disruption, and for industries such as financial services and healthcare, privacy concerns as well as disruption are causes for alarm.
- **Time.** Remember the migration? Each of those steps could easily take up to a year. Just breaking a monolith into microservices requires understanding what capabilities you want to deploy separately and breaking the process into steps so you can measure progress.
- **Money.** In the short term, both the code upgrades and the architectural changes can be expensive. When you total the costs of developer time spent on handling emergencies, adding new hires, turnover, training and planned code freezes, the hidden expenses can add up quickly.

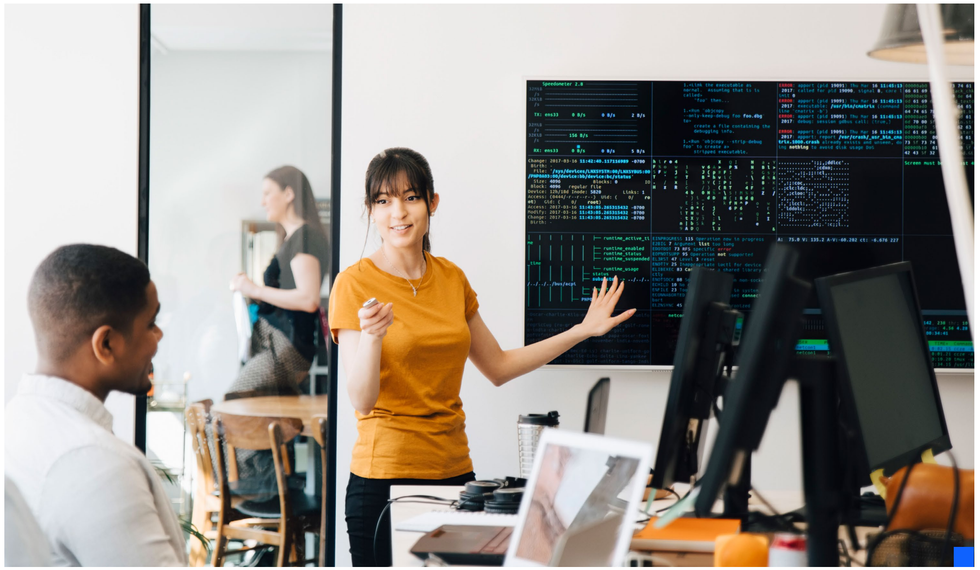
Companies hoping to stop spending millions on data center operations often end up paying additional costs for hosting service.

Inability to benchmark and compare performance
As management consultant Peter Drucker famously wrote, "If you can't measure it, you can't improve it."

As you'll see, modernization isn't simple. Organizations running monolithic applications can use legacy APM tools to measure the appropriate metrics. However, once the application components are broken up and running in containers, some legacy APM tools can't see inside containers. Plus with cloud hosting, even fewer legacy APM tools are up to that observability task.

06

The role of observability in application modernization



It’s critical to add observability into your portfolio as you plan and execute application modernization and digital transformation. Application quality control is a requirement to do modernization.

- Benchmark and compare with the same measurements.
- It’s easier for a containerized Docker solution to measure on premises.
- A legacy APM tool can’t measure cloud native apps.

Benchmarking before you plan to modernize an application is critical so you can determine where your application performance is weak, where it’s strong and where it’s on track. There are some basic best practices to keep in mind.

Benchmarking for application quality control

When you benchmark and compare performance at each stage, it’s critical that you record the same measurements so the comparisons are valid. Benchmarking is a critical feature that deserves careful attention. It’s the only justifiable way to track application quality control.

To begin modernizing a monolith application, the first step is to decouple its capabilities into components and microservices. However, operating microservices effectively can also pose risks. Start by decoupling simple edge services to help mitigate these risks and provide the most comprehensive metrics and request traces.

The next step involves placing microservices in containers within a virtualized environment, followed by moving the components to data centers and ultimately, to the cloud. This requires new authentication services, sensors for cloud architectures and integrations with APIs. An old APM solution may not be sufficient for these tasks.

Observability: More than just metrics, traces and logs

Observability tools, defined by metrics, traces and logs, may not provide sufficient details about legacy applications. This is one reason why you should consider an observability platform capable of meeting the unique needs of enterprises—especially those running legacy apps across both legacy and hybrid environments.

Legacy APM tools can monitor monolithic applications, but they're not equipped to handle microservices, containers and cloud architectures. When applied to containerized application infrastructures, they function as "black-box" monitoring and cannot inspect the application's internal workings. The same problem is exacerbated with cloud-based or cloud-native applications. To address these issues, an enterprise observability tool is the clear solution. Here are some advantages of starting with such a tool.

Discovery. See every application and infrastructure component to miss nothing and bring context to everything.

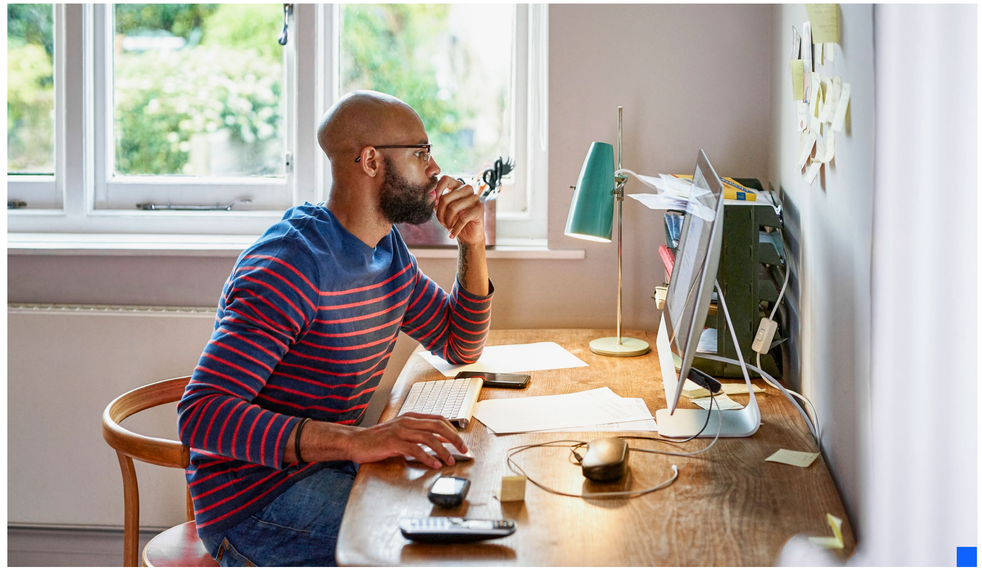
High granularity. Leave no more than one second between monitoring beats.

Trace requests. Trace requests in legacy applications and trace every request regardless of environment, even in cloud-native applications.

Instant notifications. Turn information into intelligent action.

07

IBM Instana enterprise observability

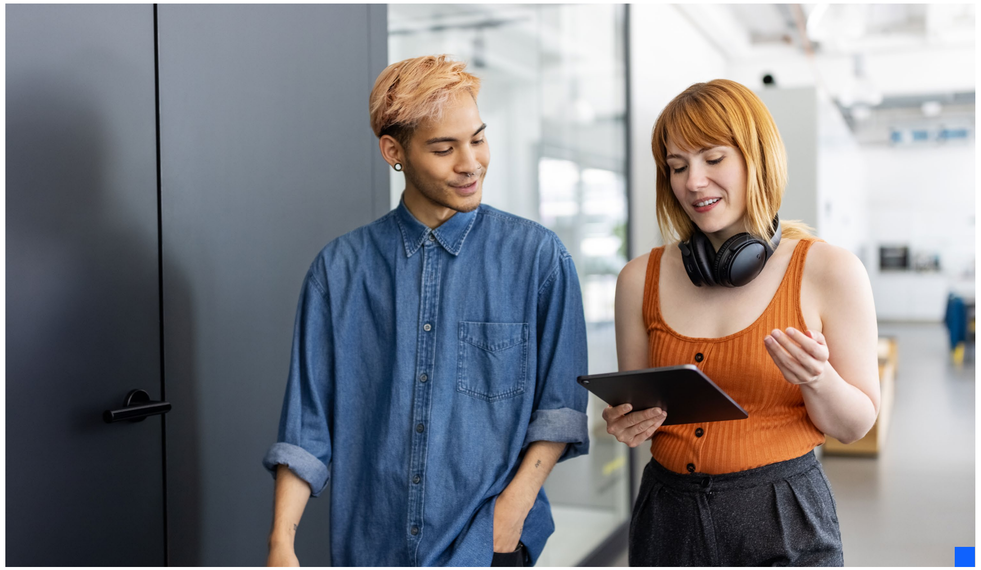


IBM Instana™ is an industry leader in enterprise observability. Here are a few ways IBM Instana adds functionality to make application modernization work more efficiently.

- **Automated discovery.** IBM Instana automatically discovers every application and infrastructure component the moment it's installed.
- **No sampling.** Legacy applications spot-check transactions and some elements of traces. Instana never samples, so it delivers an enhanced version of metrics for legacy applications along with every transaction and event from microservice-based applications.
- **Request tracing.** IBM Instana traces every request across all the systems it moves through. This tracing is automated, saving the developers' time.
- **One-second granularity.** A new infrastructure snapshot every second helps ensure up-to-date measurements.
- **Three-second notifications.** When incidents happen—and they will—notify the right people and kick off a remediation process immediately.

08

Is IBM Instana right for you?



IBM Instana provides an industry-leading real-time automated enterprise observability platform. Its application performance monitoring capabilities are ideal for organizations operating complex, modern, cloud-native applications. IBM Instana is ready to go to work anywhere your workloads run—in public clouds, private clouds, hybrid clouds, on mobile devices, on premises or in an IBM® z Systems™ environment.

IBM Instana gives you expanded control over modern hybrid applications, thanks to its precise metrics, full end-to-end traces for all transactions and AI-powered contextual dependencies discovery inside hybrid applications. For systems reliability engineers, IBM Instana helps improve the reliability and resilience of cloud-native applications by preventing issues from turning into incidents. And by providing blazing fast remediation times when incidents do occur.

See the power of IBM Instana for yourself. Sign up today for a free 14-day trial of the full version of the product. No credit card required.

IBM Instana free trial →

Explore IBM Instana →

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
May 2023