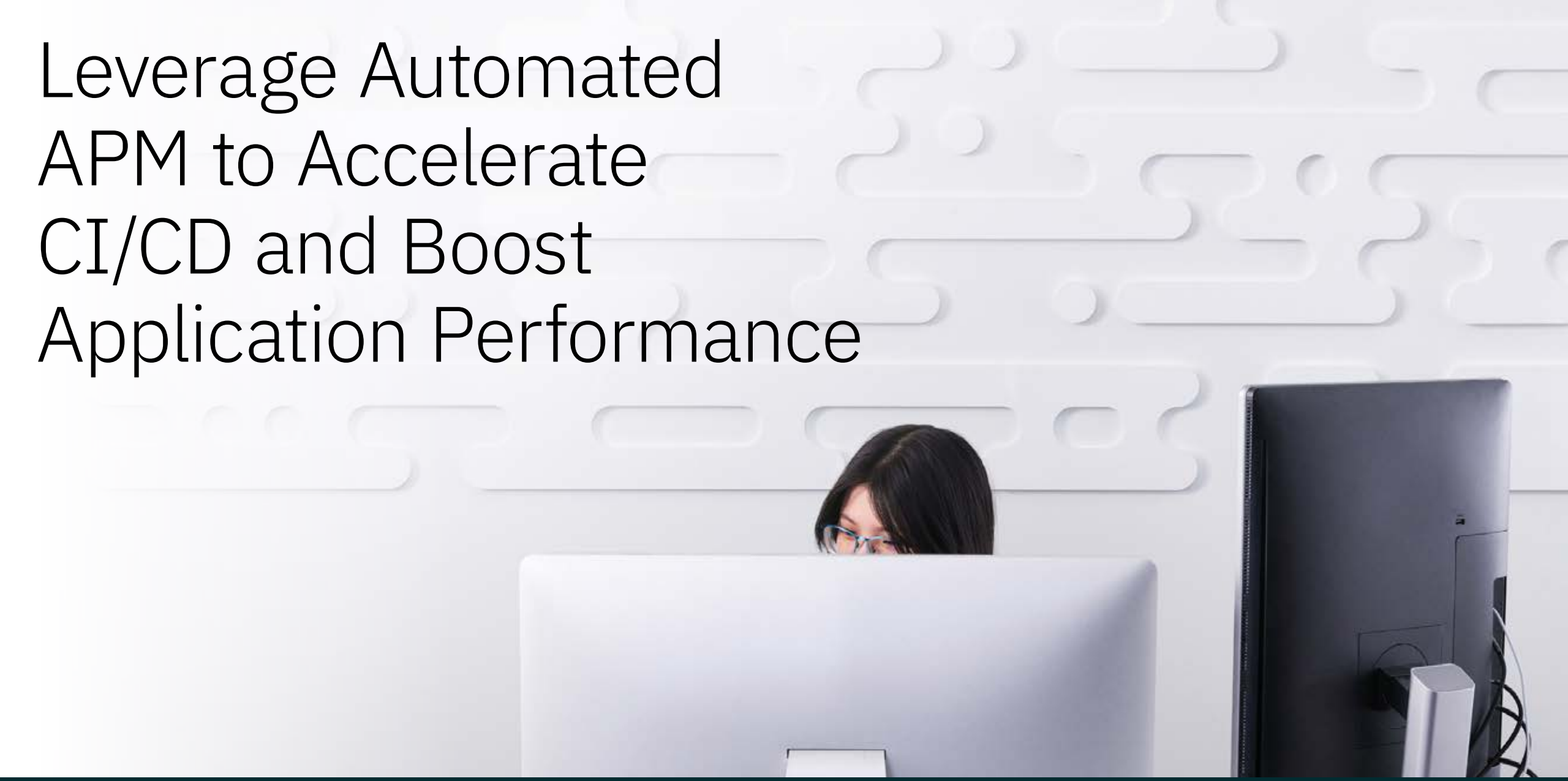
Leverage Automated APM to Accelerate CI/CD and Boost Application Performance



INSTANA
an IBM Company

Start here

Contents

01 Executive summary

Continuous integration and continuous delivery (CI/CD) has become the goal for a majority of organizations. Meanwhile, modern technologies such as Docker and Kubernetes have grown widespread introduction application environments. The result of these trends is that applications and their infrastructure are becoming increasingly dynamic, constantly changing to meet higher scalability requirements and fast-evolving application functionality.

Yet the typical organization still relies on performance monitoring technology designed before CI/CD was the application delivery model. These solutions require significant manual effort from IT staff, hampering the ability of organizations to use staff time effectively, control IT spending and gain meaningful visibility into their applications and infrastructure.

Adopting an automated performance management strategy is the only way organizations can break through application challenges and fully realize the opportunity presented by CI/CD-powered application environments. Embracing automation at every layer of performance management enables organizations to reduce costs while improving outcomes.

02
Change is the new normal

It may sound like a cliché to observe that we live in a fast-changing world. From the perspective of IT organizations, however, the fast-changing nature of today's application environments is extreme.

As we engage with our customers, their mandate is clearly a bias for speed. The faster a company can deliver custom business applications, the more value IT is delivering. Effectively, this makes the ROI of IT go up!

This simple reality is driving the adoption of new techniques and technologies, particularly CI/CD, an automation exercise that enables faster delivery, better business capability and ultimately improved quality for custom applications.

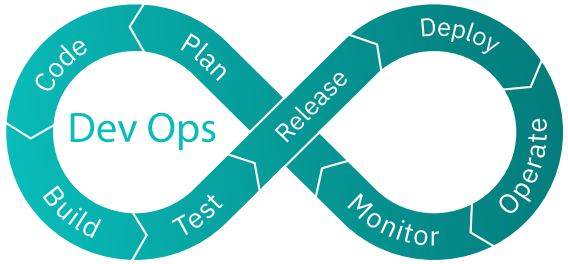


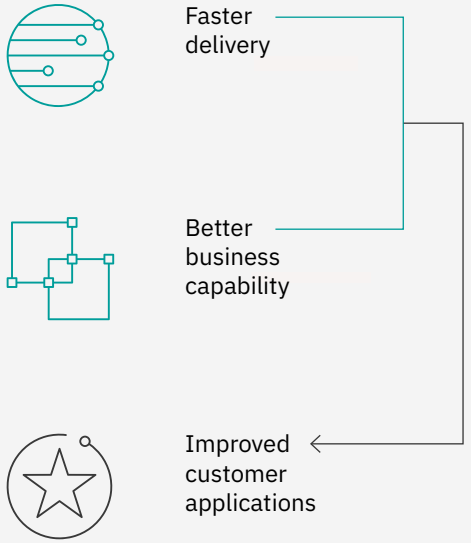
Figure 1. The continuous integration and delivery pipeline

The last stage in the CI/CD cycle is monitoring. It follows, therefore, that the more automated and seamless the monitoring of a business application, the easier it is to complete the CI/CD cycle and restart the loop of improving the application.

Change is the only constant in dynamic application environments. The structure of the application changes continuously at every layer. New hosts come online and disappear all the time, with containers making the provisioning even more dynamic. Developers continuously build and provision completely new APIs and services without checking with operations. Even the application code can change because of improvements or bug fixes at any moment. The closer a team gets to CI/CD, the more frequently changes occur.

As a result, performance management configuration, monitoring dashboards, dependency mappings and alerting rules must be able to evolve automatically to keep pace with the environment they are monitoring. Otherwise, IT teams lack the necessary accurate visibility into the environments they manage, which leaves the organization at great risk of failure that could impact users.

CI/CD automation enables:



Complex dependencies
These constant changes impact the actual dependencies among different components. Any specific service depends on a unique vertical stack of software along with data or processing from other services.

Why is it important to always understand dependencies? Troubleshooting! Getting to the root cause of an issue in complex environments is an exercise in dependency analysis. What's causing slow or erroneous requests? Requests traverse many services across many frameworks and infrastructures, so an understanding of structural dependencies of every request is invaluable to answer that question. But as we detailed in the last paragraph, dependencies constantly change!

Attempting to interpret such dependencies manually is simply not feasible—especially when dependencies change quickly due to code deployments or infrastructure scaling. Even if human operators succeed in mapping dependencies manually at one moment in time, their mappings will quickly become outdated. What's more, manual dependency interpretation is a huge resource drain and takes your best engineers to accomplish.

Why is it important to always understand dependencies? Troubleshooting!

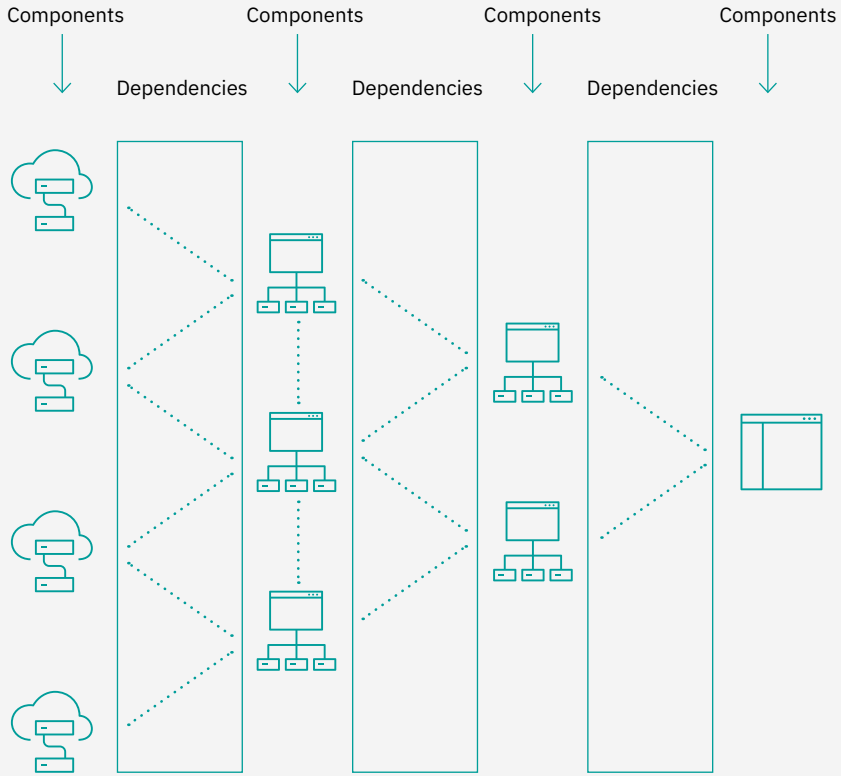


Figure 2. Any specific service, such as hardware, software or code, depends on a unique vertical stack of software along with data or processing from other services. Change cascades and may compound issues throughout the environment.

The rise of dynamic applications
For application development, that need for speed has driven broad adoption of technology that enables rapid construction and delivery of new services.

In particular, the continuous integration and delivery pipeline is being recognized as a primary process for enabling speed and quality. In addition, to augment that process, new architectures such as containers, microservices, serverless computing and Kubernetes orchestration are being investigated and adopted.

With more workloads moving to dynamic technologies—eight billion pulls on the Docker Hub in a month and 130 billion total pulls since the hub was launched in 2014[1]—there are new realities for the pace and scale of change within application environments.

The IT industry is consolidating around CI/CD terminology and methodologies. Meanwhile, public cloud platforms now offer Kubernetes managed container processing as a service that is easily integrated into CI/CD delivery pipelines. Clearly, there's a major change in the world of application construction and delivery.

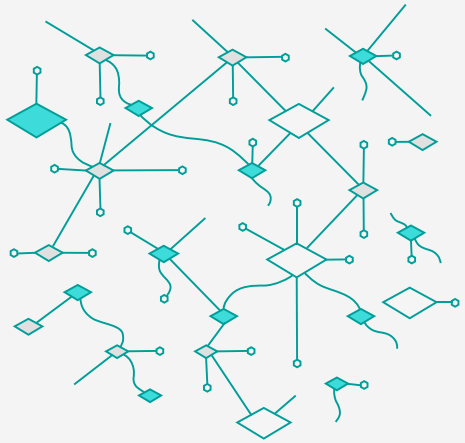


Figure 1. Changes impact actual dependencies among different components and can bring a system down in seconds.

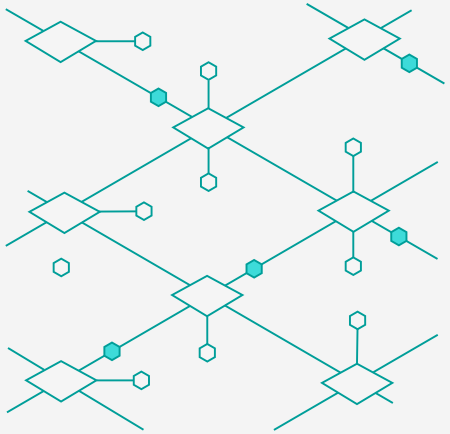


Figure 2. Instana ingests observability metrics, traces every request and profiles every process automatically. It breaks through your environment complexity to show you how everything fits together in context.

03 Why manual performance management fails

The challenges of manual performance monitoring

Because traditional application performance monitoring (APM) and monitoring tools weren't designed for dynamic applications, a new set of open-source technologies emerged to help development teams manually set up their own monitoring through manual coding. Whether providing performance metrics, tracing paths of an application or exposing other details of code, these open-source monitoring tools create their own set of challenges when it comes to production performance monitoring.

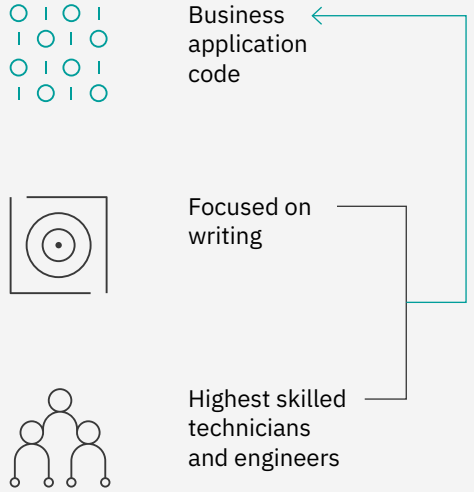
To manually monitor the performance of an application, engineers must complete the following tasks:

- Write data collectors
- Perform manual code tracing to track distributed requests
- Configure a data repository
- Identify and designate dependencies, usually through reverse engineering
- Select data to correlate
- Build dashboards to visualize correlation
- Configure alerting rules and thresholds

The biggest problem is that the people performing these tasks are usually the highest skilled—and highest paid—technicians and engineers in the company.

In short, although open source implies simplified performance management, too many manual tasks are required, resulting in deployment slowdowns, cost increases and additional labor requirements. And that's before you look at the opportunity cost of your precious developers writing monitoring code instead of business application code.

The aim is to have the highest skilled—and highest paid—technicians and engineers in the company writing business application code, not monitoring code.

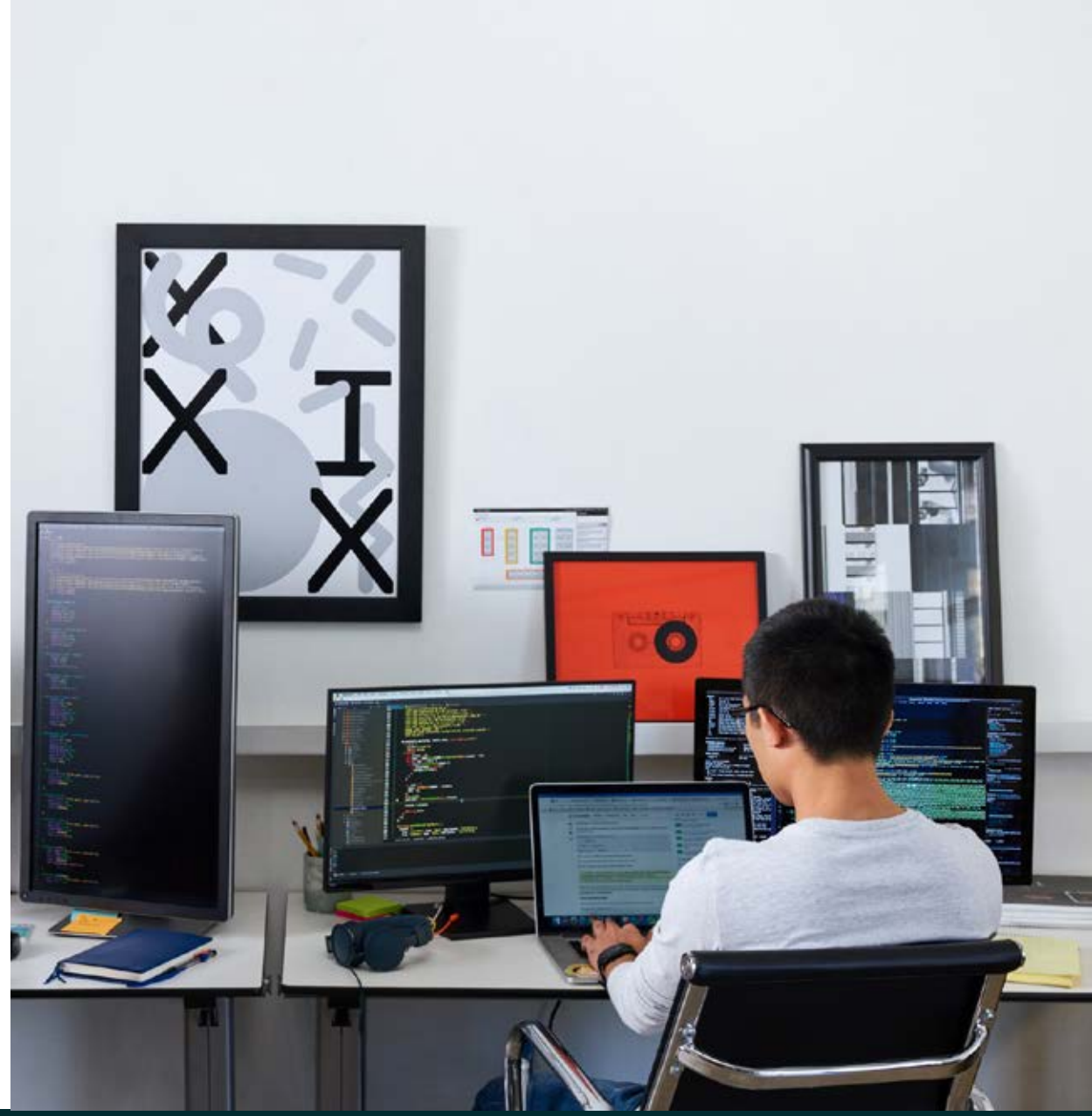


The true effort of building your own

Historically, manually setting up a monitoring system didn't present a problem because neither the application code nor the application infrastructure—middleware, app servers and so on—changed very often. IT would provision a box, set its IP address, load some software, set up the monitoring and then never touch it again for years.

Nor was an application environment that was built on traditional technologies likely to fluctuate rapidly in scale; the number of application instances and host servers running at any given moment typically didn't change. As a result, manual configuration of the tech stack in static environments didn't impact the organization's ability to monitor and manage performance.

But when workloads move into dynamic environments based on technologies such as containers and on methodologies such as CI/CD, manual performance management strategies and build-it-yourself solutions break down. This is true for several reasons.



Rule deterioration
When your environment changes constantly, the rules that your monitoring tools use to determine whether applications and services are healthy need to change continuously as well. If they don't—which is likely to happen if you depend on manual intervention to update rules—the rules will quickly deteriorate. For example, when a new service is deployed or an orchestrator moves workloads, health check rules will cease to accurately interpret environment dependencies until the rules are updated.

If rules are not updated manually, monitoring alerts and insights will be based on outdated configurations. This deterioration undercuts visibility and increases the risk of an infrastructure or service failure.

Manual monitoring impedes speed
Tasks such as writing tracing and monitoring code are too time consuming. So are interpreting monitoring information by hand and manually updating performance management rules whenever application code or environment architectures change.

Simply put, humans can't support rapidly changing environments without automation. Attempting to manage performance manually will significantly slow down application release cycles. And for the business, it means poor use of the expensive expertise that IT staff represent.

Accelerating software delivery is paramount for organizations seeking to keep pace with fast-changing user demands. Business needs will only increase this demand, so looking forward, applications will certainly become more dynamic.

04 Implementing automated APM

If manual monitoring is so laborious, then it begs the question: why hasn't every organization automated monitoring yet? There are actually three reasons for this:

– Until recently, fully automatic monitoring technology did not exist.

– Software engineers tend to code their own solutions to problems, leading to the monitoring-as-code approach, which is functional in the short term but not maintainable or scalable.

– Teams try to use previous investments in existing monitoring tools, hoping they'll work in their new high-speed environments, but unfortunately, they don't.

With these reasons in mind, let's take a look at the capabilities that should be part of a fully automated APM solution:

– Automatic discovery and monitoring of the full infrastructure and application stack. This is a key element of the CI/CD process that is missing today and slowing down the release process.

– Real-time complex dependency mapping, with automatic updates for any change. This is critical for root cause analysis to address performance issues quickly.

– Automatic, rapid identification of performance problems. Quickly identifying performance issues based on automatic rule configuration and monitoring is the only way to minimize false positives and avoid application delivery delays.

– Rule configuration and alerting that use machine learning and AI to establish dynamic baselines for healthy application behavior, then identify anomalies based on those baselines. This eliminates the need for tedious configuration, monitoring and data interpretation by humans while minimizing false-positive alerts.

– Automatic monitoring setup—agent deployment, code instrumentation, infrastructure discovery and more. Maximize the data collected while minimizing the effort required from human administrators to collect it.

Both existing performance management tools and modern open source monitoring tools lack this automation functionality. Though the exact work is different, both sets of tools require too much manual configuration, programming, setup and administration to optimize monitoring.

05
Maximizing CI/CD agility

Is your CI/CD as agile as it could be? Successful management of dynamic application environments doesn't end with automated monitoring.

Your IT teams should continually assess the extent to which they have successfully automated all performance management tasks by asking themselves the following questions:

- How long does it take to achieve sufficient visibility into application performance after we push out a new release?
- How long does it take to update monitoring rules when a new application or service deployment occurs?
- How much time and effort do our developers expend writing tracing code?
- How many performance or availability incidents are we missing per month or quarter?
- How are we handling alert storms—meaning a rapid stream of alerts in a short period? Are we able to respond to each alert effectively without suffering from alert fatigue? Can we trace alerts quickly to root causes so that we know when multiple alerts are stemming from the same underlying issue?
- Is our monitoring and performance management process as automated as the rest of the application delivery pipeline? If not, how can we automate it further?

Regardless of your IT team size, automation is critical for an effective performance management strategy so that you can achieve high-speed delivery of new business services.

IBM Observability by Instana, the automated performance management solution born in the age of microservices, cloud computing and containers, can help you truly deliver on the promise of CI/CD.

Designed specifically to manage the performance of dynamic CI/CD-driven application environments, IBM Observability by Instana uses AI and automation to deliver comprehensive, actionable insight with no manual effort.

06
About Instana, an IBM Company

Instana, an IBM Company, provides an **Enterprise Observability Platform** with **automated application performance monitoring** capabilities to businesses operating complex, modern, cloud-native applications no matter where they reside—on premises or in public and private clouds, including mobile devices or IBM Z® mainframe computers.

Control modern hybrid applications with Instana's AI-powered discovery of deep contextual dependencies inside hybrid applications. Instana also provides visibility into development pipelines to help enable closed-loop DevOps automation.

These capabilities provide actionable feedback needed for customers as they optimize application performance, enable innovation and mitigate risk, helping DevOps increase efficiency and add value to software delivery pipelines while meeting their service and business level objectives.

Learn more →

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
April 2021

1 Docker knits together Hub stats, says Pulls over 8 billion, DevClass, 5 February 2020.