INSTANA
an IBM Company

Microservice Observability

Table of Contents

03 The Brave New World of Software Design

05 The Technologies of the Microservices Age

08 The Challenges of Microservices

12 Using Intelligent Analysis to Thrive in the Brave New World of Microservices

18 A New Generation of Observability

23 Conclusion

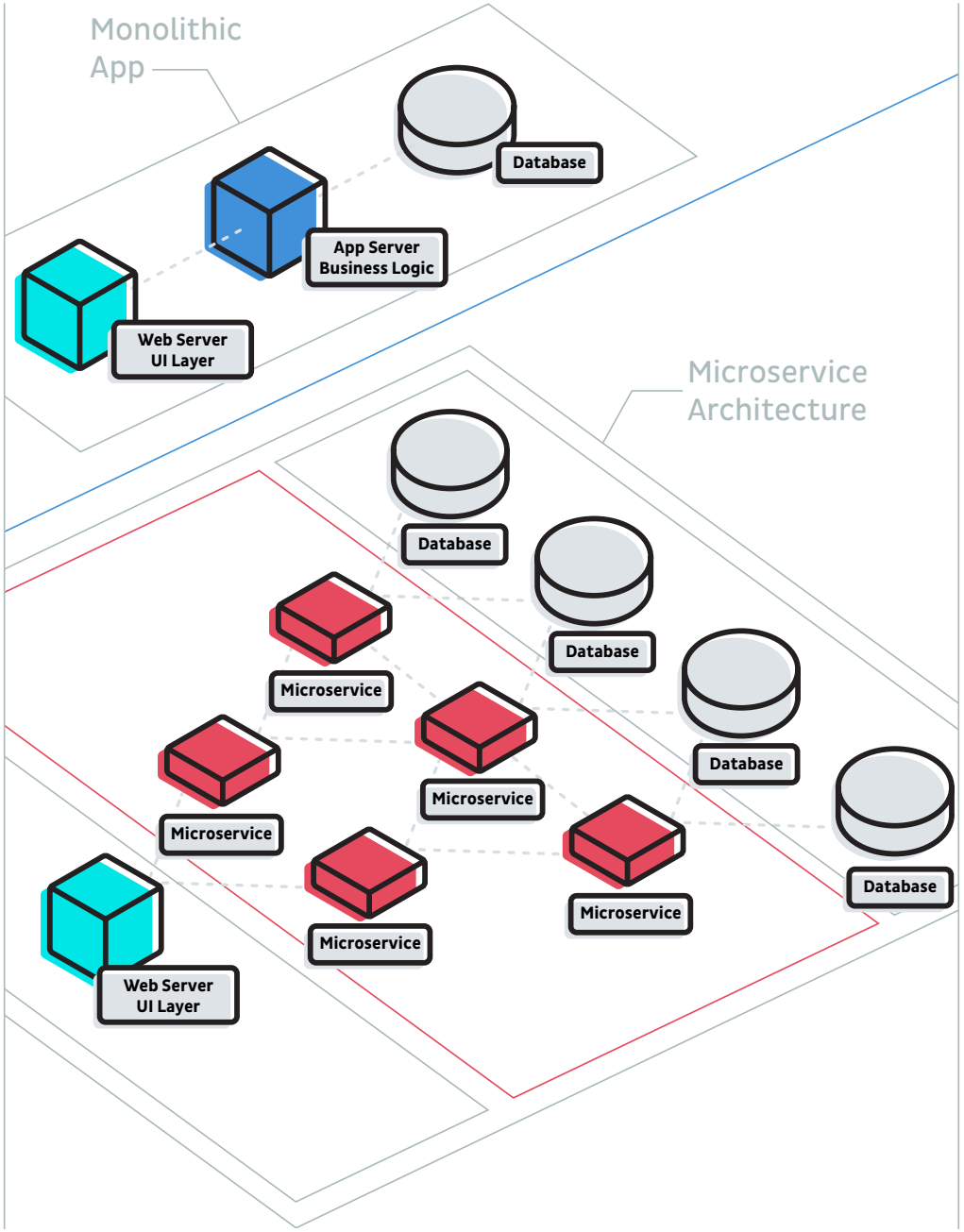
24 About Instana

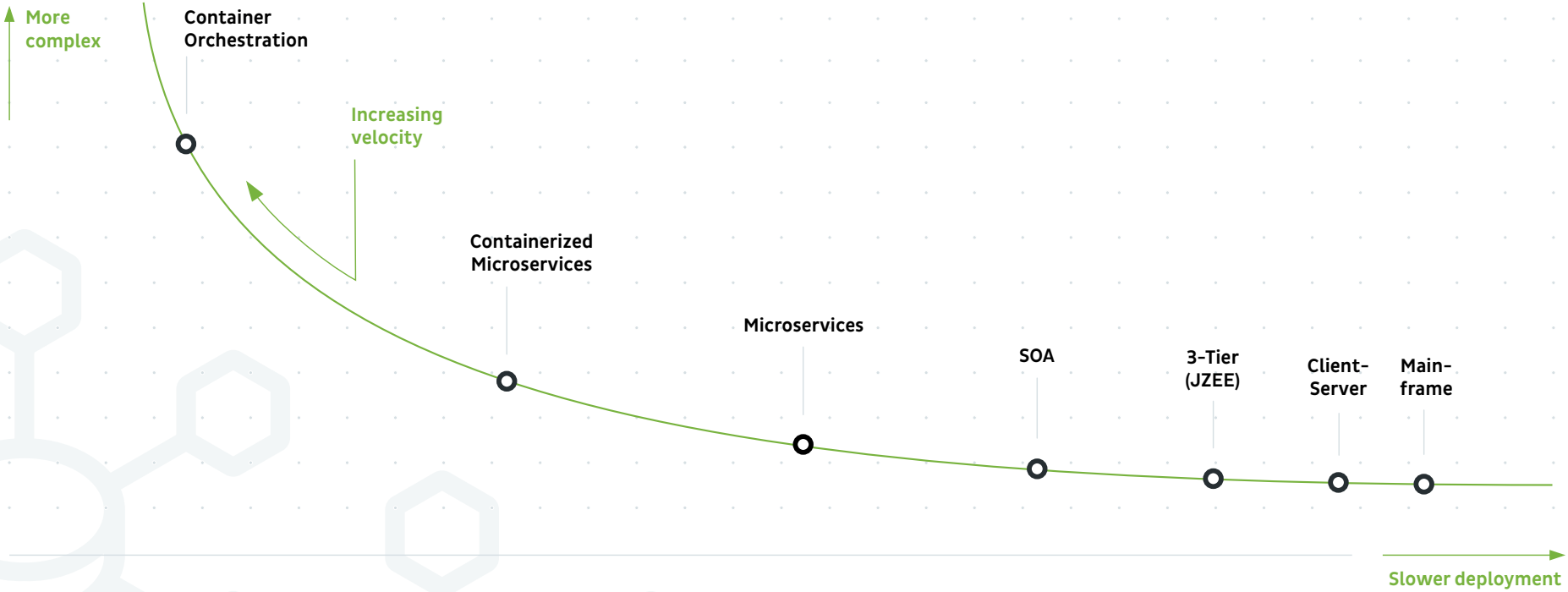
The BraveNew World of Software Design

There is no dispute that microservices have completely changed the world of software development. Thanks to the microservice architecture model, applications can be developed and updated at a much higher velocity. Applications – and organizations – are more nimble, more scalable, more robust and more resilient against disruptions.

Microservices can be updated quickly and easily. They enable fully continuous delivery and allow DevOps teams to make the most of their skills and resources. But while microservices break through many hurdles associated with traditional software delivery, they also create new types of challenges.

From unique architectures including containers, orchestration, hybrid and multi-cloud, and even serverless infrastructure to the ephemeral and agile nature of service-based applications, microservices create great opportunities to optimize application delivery. But dis-integrating applications, services and infrastructure, a new level of complexity in application environments creates all kinds of new management challenges.





As application delivery frequency has increased, so has application complexity.

Complicating matters further is the fact that microservices applications are often distributed across a cluster of servers, tend to be hosted across multiple layers of infrastructure (such as containers) and are ephemeral / fluid regarding locations and application pathways.

In essence, when an organization migrates to microservices to achieve greater velocity and agility, the tradeoff is higher operational complexity. As a result, while microservices deliver a clear business benefit by making software faster and more agile, they also significantly complicate management of overall service quality and performance. This does not mean that the service quality of a microservices environment cannot be effectively monitored and managed. It certainly can. But doing so requires a new approach to Observability and monitoring. Traditional monitoring strategies will come up short in a microservices world.

The Technologies of the Microservices Age

Let's begin by defining the new technologies and design philosophies that continue to shape the way software is conceived and delivered in the age of microservices. Today, the following concepts and technologies define the production of software:

- Continuous delivery
- Polyglot pipelines
- Containers
- Highly distributed environments
- Software-defined everything
- Everything-as-a-Service
- DevOps

Continuous delivery

Crafting code at a slow, staccato pace based on plans made far in advance no longer works. Today's software engineering teams strive to release code on a rapid, continuous basis by reacting to user feedback as quickly as it arrives. Continuous delivery helps to make software delivery agile and nimble. It also makes it easy to adjust or add a new microservice because changes to a single microservice do not disrupt the larger application.

Polyglot pipelines

Software engineering teams also place a premium on the ability to switch easily between different development frameworks as their needs and the tools available to them change. The ability to maintain a polyglot pipeline is another crucial factor in achieving agility.

Containers

The introduction of production-ready container platforms (especially Docker) over the past several years affords software delivery teams a leaner, faster way to write, stage and deploy code. Containers lend themselves well to microservices apps because the various microservices that comprise an app can be rapidly deployed across different groups of containers. Apps also scale easily within a containerized environment, especially when service discovery is automated and orchestrators help to manage configurations automatically.

Highly Distributed Environments

It wasn't that long ago that applications and services were rarely distributed across multiple servers. That began to change with the introduction of Service-Oriented Architectures, or SOAs, in the mid-2000s, along with the move to cloud-based environments. With microservices, the transition is complete. Production environments are commonly distributed across multiple servers.

This distributed design adds a great deal of scalability and resiliency because it allows hosting infrastructure to be added or subtracted from an environment without downtime. Distributed environments also ensure that the failure of an individual server will not disrupt the continuity of a running application that is distributed across many servers.

Software-defined everything

One of the contributing factors to the difficulty mapping modern applications is software-defined environments. Now widespread, software-defined entities (infrastructure, servers, etc.) create layers of abstraction between applications / services and any underlying infrastructure. While this makes microservices easier to deploy since individual services no longer have to be tied to particular hosts, it creates all kinds of Observability issues.

Everything-as-a-Service

As more workloads move to the cloud, the consumption of on-demand compute and storage resources has increased, usually as a pay-as-a-service methodology. While this makes their infrastructure more scalable and suited to host microservices applications, service-based workloads have their own observability challenges.

DevOps

The embrace of agile development, coupled with the DevOps movement that followed it, has reshaped the way organizations think about and approach software design and delivery, especially observability and monitoring. Today, constant collaboration and communication between everyone involved in software production are both priorities. So is the ability to be flexible in defining the roles that individual team members are expected to assume. Microservices help to enable DevOps by facilitating software that is as agile as the team that creates it.

All of the above makes software delivery today radically different than it was prior to the container / orchestration revolution — before Docker became ready for production use, before DevOps became common across all types of organizations, and before software-defined storage, networking and everything else inaugurated a new type of development and deployment strategy.

Challenges of Microservice Observability

The changes described above have exerted a decidedly positive impact on software delivery in most respects. Software production and deployment today is much faster, more reliable and more agile than it was just a few years ago. Organizations that embrace modern software delivery techniques are also better positioned to support new business services and cater in a more nuanced way to precise business needs.

The challenges of microservice observability:

- Complex dependencies
- Continuous delivery
- Containerized environments
- Microservices architectures
- Everything-as-a-Service
- DevOps and fluid roles

However, as noted in the introduction, the explosion of new technologies in recent years also comes with some caveats. The innovations that make agile, microservices based software delivery possible also make observability, monitoring and service quality management more difficult, increasing the overall difficulty of delivering high-performance, highly scalable applications around the clock. Consider the following ways in which the software delivery practices of today make observability, monitoring and performance management more difficult.

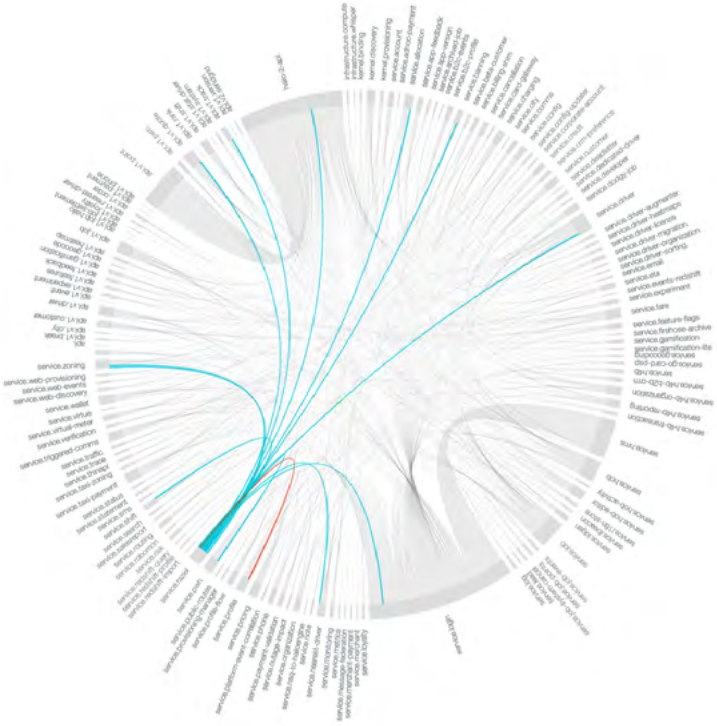
Complex Dependencies

Mapping dependencies in a microservices environment is both extremely important and extremely complex. In legacy environments, especially monolithic applications, inter-service dependencies are relatively few in number and easy to identify. It's easy to tell which applications are running on which servers, or which storage service is linked to which app.

In a microservices architecture, however, the dependencies and performance patterns that link services and infrastructure together are much more complicated. Microservices are often deployed using containers that are distributed across a cluster of servers. Network routes are adjusted constantly by load balancers in response to shifts in demand.

The ports that individual microservices use to communicate with each other can be changed without notice by orchestrators. For these reasons and more, dynamically mapping the ever-changing dependencies within a microservices environment is extremely complex and difficult (if not impossible) to do manually.

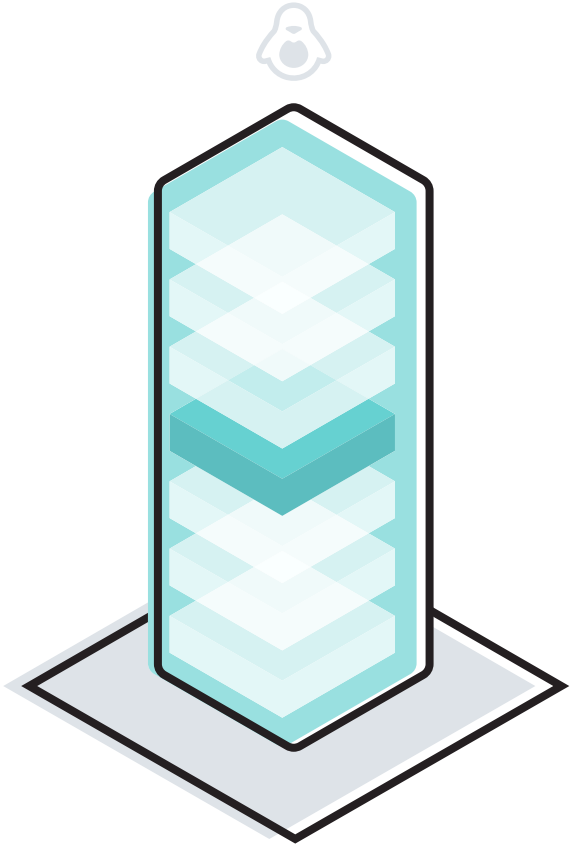
This makes the task of incident resolution / root cause analysis difficult as well. For example, a storage service problem could be the result of a failed server, a coding mistake, a disk failure or a slowdown on the virtual network that connects your storage array to the rest of your application. In an event like this, mapping the intricate dependencies of the storage service is the only way to identify the exact cause of the incident to resolve it.



A Journey into Microservices:
Dealing with Complexity
Halo Tech Blog

Continuous Delivery

Agile code development features services and architectures developed in small increments. While that makes software delivery faster and more flexible, it also means changes are introduced at a faster pace. Effective Observability and Monitoring requires the ability to see changes in real time, and more importantly to redeploy any monitoring based on updates to the service.



One host quite commonly has many containers

Containerized Environments

In containerized environments, there are significantly more components within an application to monitor. Instead of a few dozen servers and applications (the number of monitoring objects one would have in a large traditional environment), a containerized environment could be made up of thousands of individual ephemeral containers. As a result, the number of objects that require observability and monitoring, as well weas the rate of change within the environment as containers spin up and down increases dramatically. In fact, the number of objects to observe and monitor grows exponentially in microservice environments.

Microservice Architectures

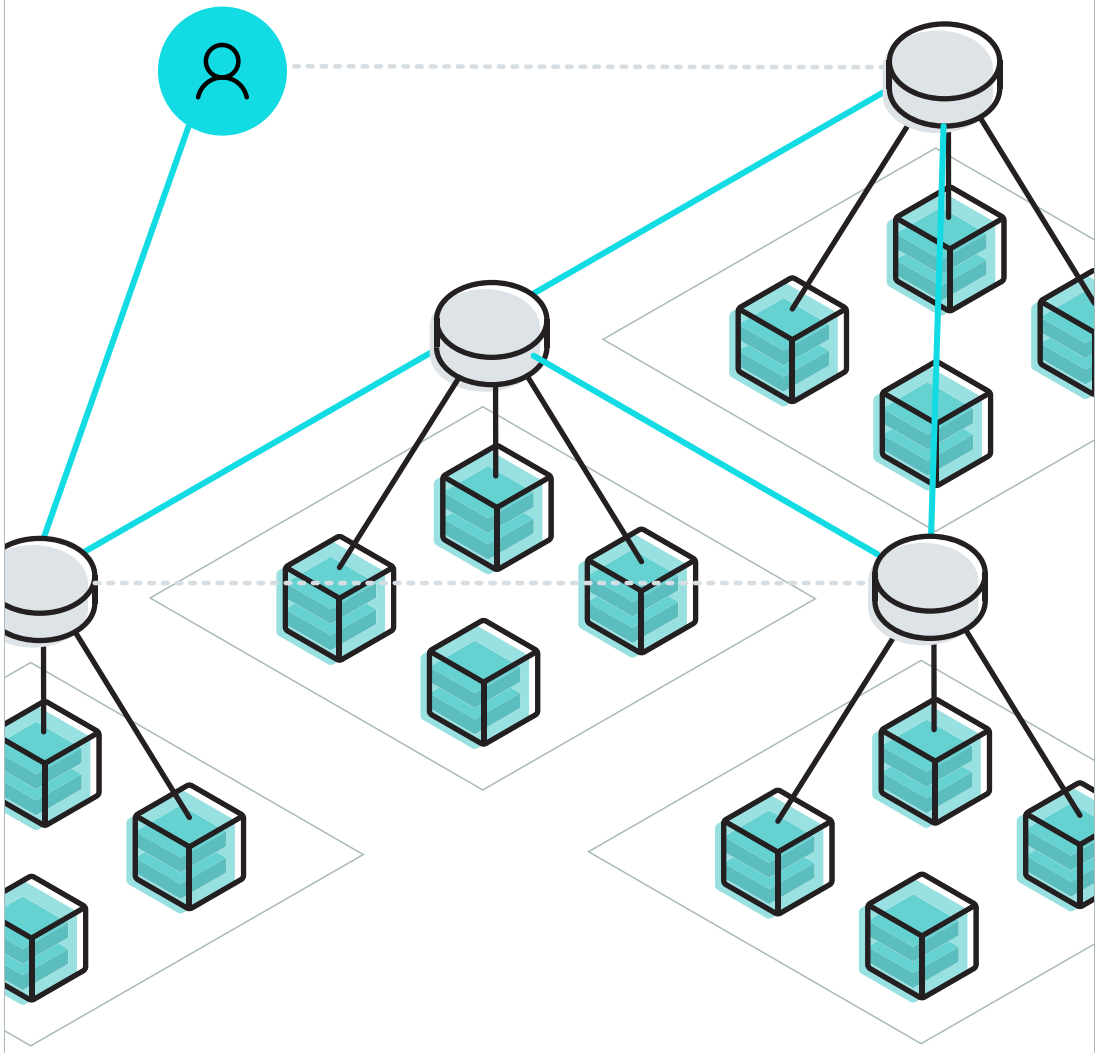
It's easy to see that in a microservices environment, there are more services to monitor. But it's not enough simply to know whether your Application is up or down. You must also keep track of the many microservices that make up the applications, monitoring them not just for uptime, but also for quality of service, so that you can identify and address reliability issues within a specific microservice before they degrade the performance of the entire application.

Everything-as-a-Service

The embrace of the Everything-as-a-Service model means many organizations no longer own the underlying infrastructure on which their applications, services, microservices and containers live. But it also means there's no easy way to map any dependencies between microservices. When traditional monitoring and touching the servers that host your software is no longer an option, you will need an observability platform tools that can provide visibility into the infrastructure no matter where it is, even if it is managed by third parties.

DevOps and Fluid roles

Under the DevOps model, individual team members' roles are fluid and shifting. This means everyone now has a hand to play in application delivery and service quality management, including engineers and admins who were not specifically trained in application technologies or in the architecture of the apps. For this reason, DevOps teams require quality management solutions that are sufficiently intuitive and automated for non-specialists to use effectively.



Using Intelligent Analysis to Thrive in the Brave New World of Microservices

Now that we understand the challenges associated with monitoring and ensuring quality in the microservices age, let's examine the strategies that software delivery teams can adopt to meet the challenges.

Using Intelligent Analysis to Thrive in the Brave New World of Microservices

- Separate incidents from noise
- Make monitoring information actionable
- Understand incidents and dependencies precisely
- Share information easily across your team
- Maintain both real-time visibility and historical visibility
- Minimize manual configuration
- Achieve continuous understanding

As you'll see, all of the strategies and abilities for effective service quality management in a microservices environment (which are outlined to the left) center on automating workflows and using tools to achieve results on a scale that humans alone cannot feasibly produce. Intelligent analysis, machine-assisted learning and automation are at the root of successful service delivery and quality management for microservices.

Separate Incidents from Noise

Because a microservices environment involves so many moving parts and a high volume of activity, it's essential to be able to separate incidents that require handling from operational noise quickly, using intelligent analysis that is built into your tools. An incident is any event or issue that negatively impacts the quality of a service or microservice. Incidents happen when a service or microservice fails to deliver within a period of time that is acceptable to the organization. In contrast, noise is data generated by normal activities. Noise is not associated with a quality-of-service problem.

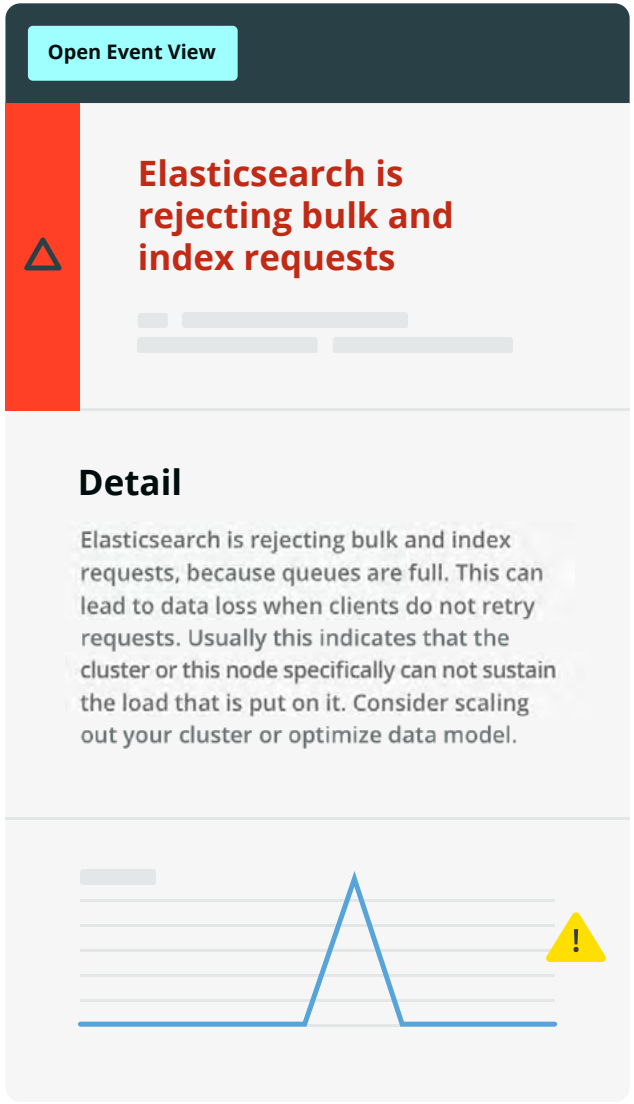
Separating incidents from noise means being able to recognize (by glancing at your application and service dashboards) which types of data are associated with a potential problem (like a microservice running short of compute resources), and which are just natural, unremarkable events (such as containers spinning down as demand for a service decreases).

Make Monitoring Information Actionable

For similar reasons, the monitoring data your team collects should be immediately translatable into action. In other words, your team and the software it uses should be able to leverage monitoring information quickly in order to understand and fix a problem. To achieve this, look for tools that go beyond collecting data. Your tools should also be able to leverage intelligent analysis to turn data into actionable information by taking advantage of machine learning and automation. They should also be capable of detecting anomalies and mapping service dependencies automatically so that you can focus on managing the incidents that your tools identify, rather than having to invest time in interpreting data manually. There are 3 key steps to efficient service quality management: eliminate noise, make monitoring data actionable, and precisely understand incident dependencies.

Precisely Understand Incidents and Dependencies

Knowing that a problem exists with one of your services is only the beginning of the battle. You also need to be able to identify the root cause of the issue quickly and immediately understand how it impacts other parts of your environment. Tools that can automatically map dependencies enable this level of insight. Understanding how services interact and how a problem with one component can affect others is essential, because in a fast-moving microservices environment composed of many layers of infrastructure, the source of an incident could lie in many different places—in a host server, in middleware, in application code, in a container, on the network, and so on.



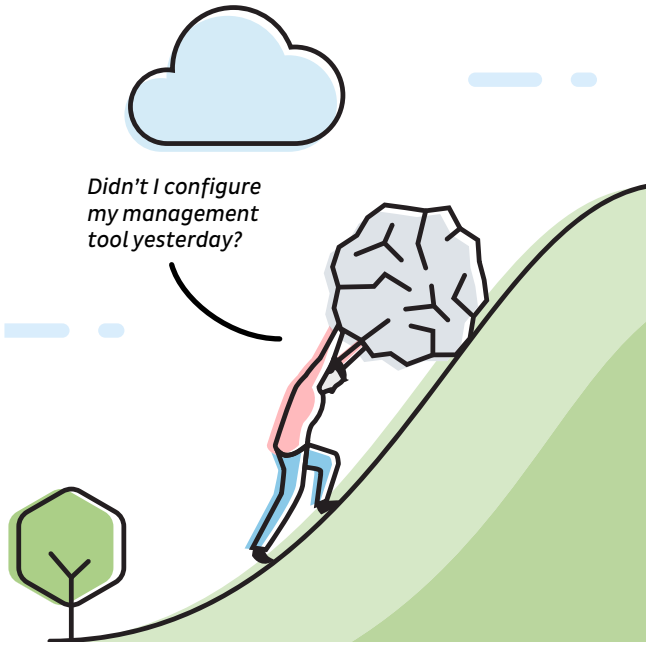
A curated knowledgebase and machine learning allows Instana to precisely identify root cause and assist with problem resolution

Easily Share Information Across Your Team

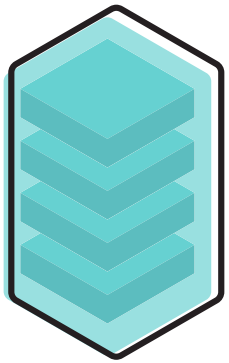
In a DevOps world, it's not enough for only part of the software delivery team to have access to service quality data. You need a way to share information seamlessly across the entire organization. For example, if your tools detect a quality-of-service issue and suggest that it can be traced to application code, it's crucial for the operations team to communicate that information quickly to the developers who can fix the problem within the application. Likewise, service quality management tools should provide visibility into the full application stack for all team members in order to enable a continuous feedback loop. When it comes to information sharing, the central goal of your operations workflow should be the implementation of continuous understanding of events—for all members of the team.

Maintain Both Real-Time Visibility and Historical Visibility

It's important to be able to detect and understand problems in real time as they occur. Real-time reaction requires an ability to separate incidents instantaneously from noise, identify the root of a service problem, and use service quality and performance data to develop an immediate response. It's equally crucial to be able to time-shift—that is, to step back in time by viewing historical data and dependencies at a specified point in the past. Time-shifting enables you to understand how an incident may have evolved over time, and which initial conditions triggered it, by examining information such as the initial layout and structure of an application, or previous allocations of containers to hosts. Your tools should support both types of scenarios. This is the only way to achieve continuous understanding of both present and past events.



In Microservices environments, the only constant is change. Manual configuration of monitoring is a never-ending cycle.



With so many objects and so much change, the only way to accurately understand service quality is by continuously discovering any and all updates

Minimize Manual Configuration

In a hyper-scaled environment, it's critical to have tools that automate configuration as much as possible.

That's because in a microservices environment, updating configuration information or adding services manually takes a great deal of time. This is true both because of the large number of components in a microservices environment, and the complex dependencies that arise from a microservices architecture. Attempting to configure tools for the environment manually undercuts your ability to be agile and scalable, and therefore defeats much of the purpose of implementing a microservices architecture in the first place.

For that reason, your service quality management strategy should take advantage of tools that automate configuration and service discovery as much as possible. The deployment of new code, new apps, new servers, and so on within your environment should be automatically discovered along with their associated dependencies. This is another essential ingredient for achieving continuous discovery and understanding.

Along the same lines, monitoring agent configuration should be automated. Your team should be able to focus its time doing what only humans can do (interpreting and acting upon complex health data), rather than the tedium of manually installing and setting up agents.

As noted above, automated anomaly detection (delivered via machine learning) and the mapping of service dependencies also help you to minimize the amount of manual configuration that your monitoring workflow requires. When your monitoring tools can detect anomalies and dependencies automatically, your team does not need to spend time configuring its tools to focus on these points of interest.

Achieve Continuous Understanding

A highly dynamic microservices environment is always changing. Performance and Service Quality information or trends may cease to be relevant just minutes after you identify them. For that reason, it's essential to implement a service quality management workflow that provides continuous understanding.

Continuous understanding is facilitated by ongoing, automatic rediscovery of environment data, dependencies and configurations. Automation and intelligent analysis are required components of continuous understanding, because the only feasible way to gain continuous insight into your microservices is to automate the processes that provide visibility.

INSTANA
an IBM Company

A New Generation of Observability

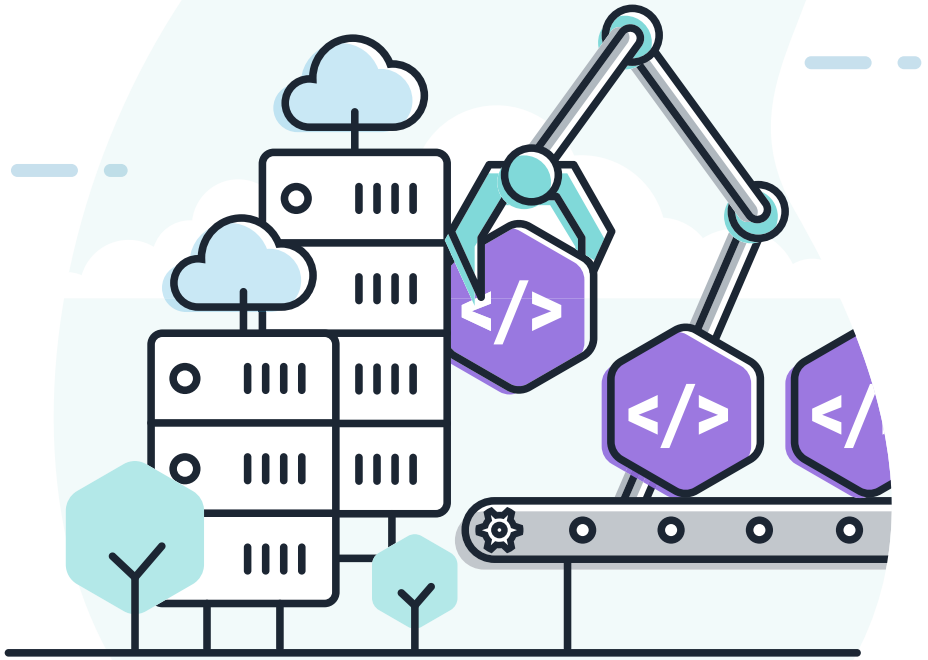
Achieving the complex, flexible, ultra-scalable development and deployment strategy in the most agile development environments requires a completely different approach to observability, monitoring and performance management.

At Instana, while we've always prided ourselves on our automated approaches to monitoring, the next generation of management platforms is Enterprise Observability. Built on the backbone of a completely automated application monitoring solution, Instana's Enterprise Observability Platform is just what agile organizations need to deal with the dynamic complexity inherent in microservice applications.

Traditional monitoring can't handle the ephemeral nature of microservices, containers, orchestration or other critical Cloud-native infrastructure and application environments. Meanwhile, simple Observability tooling can help individual stakeholders, but not the broad set of Dev, Ops and IT Decision Makers involved in modern application operations. That's where Enterprise Observability is required.

Automation

Automation is a natural desire in any monitoring solution – but for modern application teams using agile development methodology and running an always-full CI/CD pipeline, automation is their life. But if Observability and Monitoring are manual, the entire process can get stalled or derailed at the monitoring step. Enterprise Observability automates the entire monitoring lifecycle – discovery, mapping, monitoring configuration, alerting, even root cause analysis.



Context

In monolithic applications, each transaction runs through the same stack and code base as every other transaction, allowing certain functional shortcuts like sampling or reverse engineering when deploying monitoring and tracing; but in dynamic microservice applications, the only constant is change.

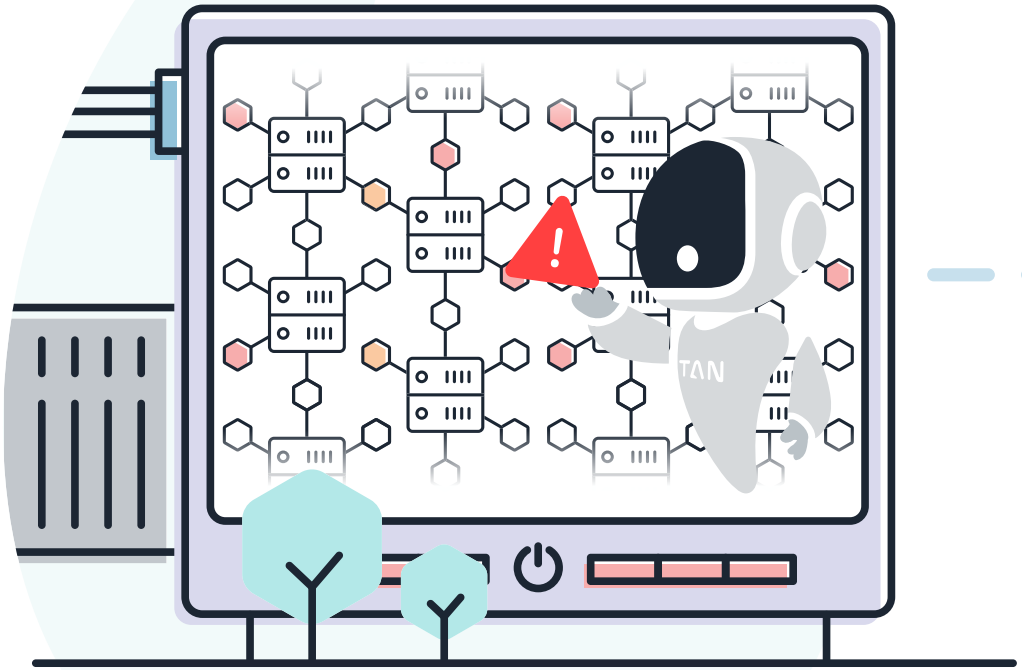
Understanding how every component (or entity) interacts with others is critical to understanding where to start solving problems. Further, sampling of application response metrics, infrastructure configuration and request tracing is no longer an effective proxy for reality.

Enterprise Observability ties everything together – service maps, infrastructure configuration, request traces, open source protocols, even profiles – delivering understanding, both programmatically and to end users (Dev+Ops and decision makers).

Intelligent Actions

Collecting data without the desire or ability to do anything with it is simply a waste of time and money. The purpose of Observability in container-based applications remains the same as application monitoring or APM solutions – to validate application performance; and to find and fix problems when they occur.

Since container-based applications are more complex, finding the root cause of problems requires assistance. Enterprise Observability uses its knowledge of application and infrastructure architecture, its immense collection of events and understanding of requests (traces) to identify triggering events and inter-component relationships to highlight where attention is required by DevOps to fix problems and optimize application performance.



Ease of Use

As organizations roll out agile development processes, increase the frequency of their application updates and fill their CI/CD pipelines, more stakeholders require the actionable information we've been discussing. That's why ease of use, which was never a stalwart of APM solutions, is critical to Enterprsei Observability. The more people that can leverage an Enterprise Observability platform, the better overall application performance will be, and the more efficient the entire DevOps organization will be.

Conclusion

While organizations tend to think of Observability issues as a new set of problems unique to microservice environments, the truth is that Enterprise Observability simply expands modern APM (or APM 3.0) to support Cloud-Native applications.

Organizations that do not embrace new solutions built on the premise of automated monitoring stand to lose money in operational inefficiencies and additional application downtimes. Regular server monitoring and APM monitoring neglect to reflect the reality of microservice environments. In contrast, while manual instrumentation of microservice code can provide some details, it's not scalable as update frequencies increase.

Today, Dev+Ops teams require a totally new generation of enterprise observability and performance management platforms — platforms that automate setup, discovery, observability, monitoring and tracing. Optimally, these solutions would operate without much, if any, human configuration either up front or after any application updates.

Instana's Enterprise Observability Platform automates application discovery, agent deployment and monitoring configuration across the full microservice technology stack. Instana reduces troubleshooting with automatic root cause analysis and mass correlation of requests, metrics, traces and profiles. The platform identifies the triggering event and most likely cause of each incident. Essentially, Instana does what a human operations team cannot – deliver ful observability and performance monitoring across modern microservice environments.

Learn more by reading the eBook:

Foundations of Enterprise Observability

About Instana, an IBM Company

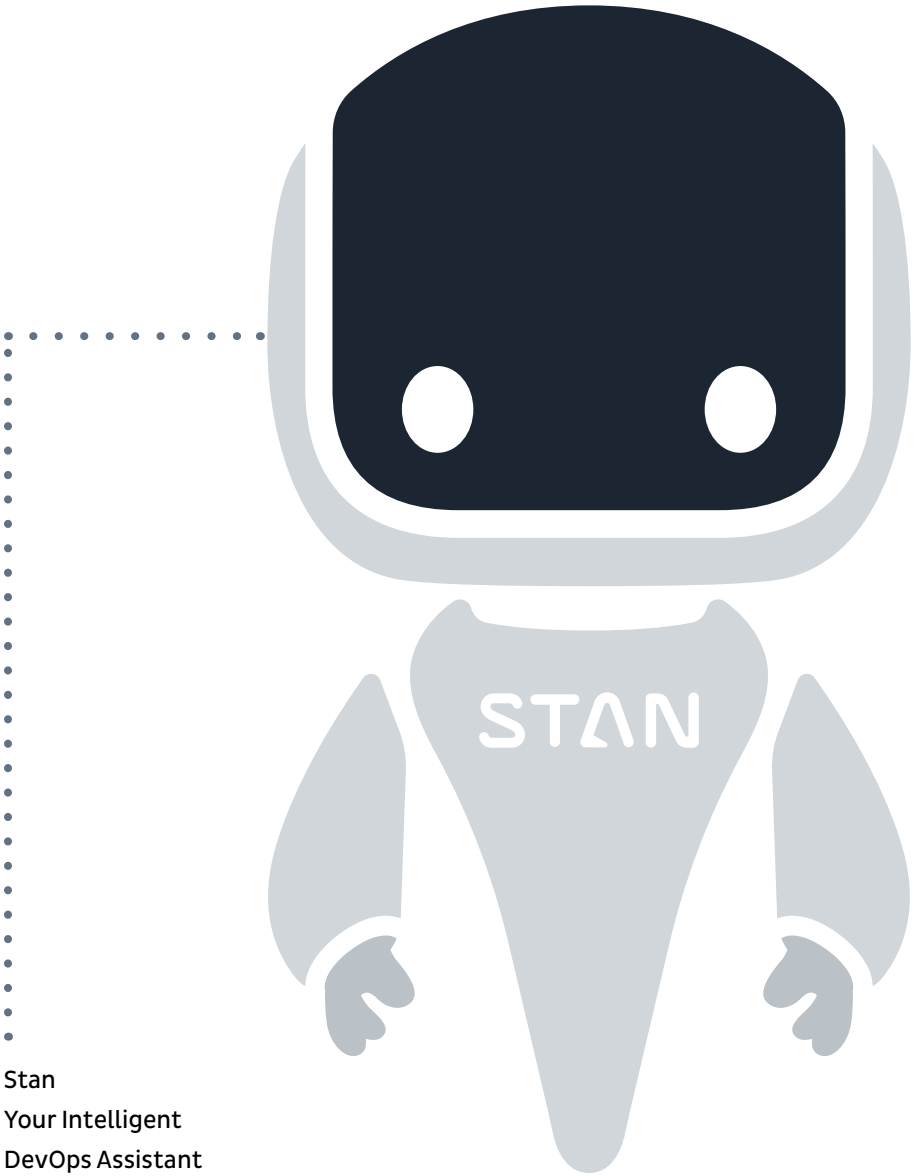
Instana is the leading provider of Observability and Application Performance Management solutions for Cloud-Native Microservice applications. The company's Enterprise Observability Platform, powered by automated APM, ingests all observability metrics, traces every request, and profiles all processes continuously and automatically, delivering the actionable information with full context to Dev+Ops to help them optimize their application performance and pipelines.

Learn more at https://instana.com

We Invite you

... to try Instana's Automatic Monitoring for yourself. You can be monitoring your application in just a few minutes.

Start Your Trial Today



Stan
Your Intelligent
DevOps Assistant

INSTANA

an IBM Company

IBM, the IBM logo and [ANY OTHER IBM MARKS USED] are trademarks of IBM Corporation in the United States, other countries or both. Instana® and its respective logo are trademarks of Instana, Inc. in the United States, other countries or both. All other company or product names are registered trademarks or trademarks of their respective companies.

©Copyright 2021 Instana®, an IBM Company