Service Quality Management

in the Microservices Age

INSTANA

Table of Contents

In this eBook

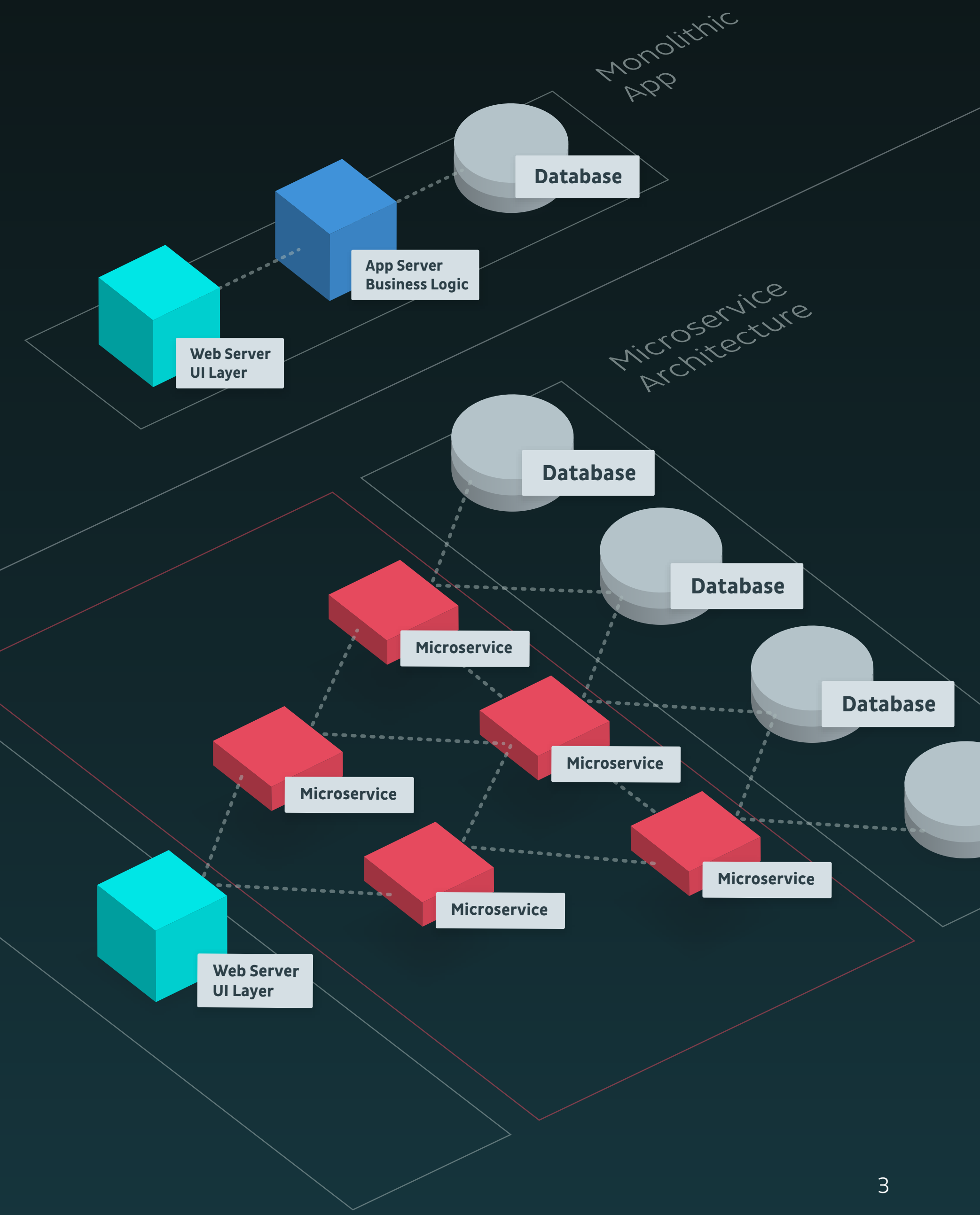
... we examine what makes microservices different from traditional app architectures, the special management challenges microservices present, and how teams can effectively address those challenges by adopting a new approach to service quality management, tailored for the brave new world of microservices.

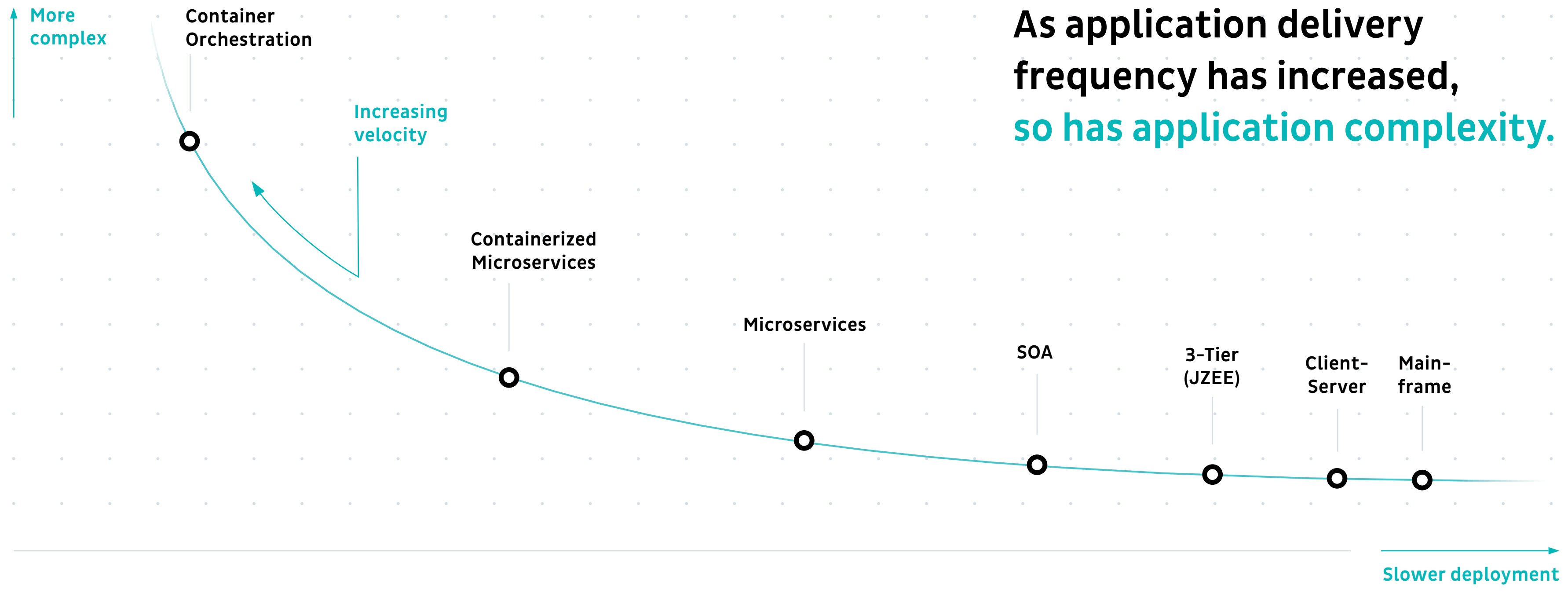
The Brave New World of Software Design

Microservices are changing the world of software. There is no disputing that.

Thanks to the microservice architecture model, applications can be developed and updated at a much higher velocity. They are also more nimble and scalable, and more robust and resilient against disruptions. They can be updated easily and quickly. They enable fully continuous delivery and allow DevOps teams to make the very most of their skills and resources.

Yet, while microservices solve many problems associated with traditional software delivery, they also present a new type of challenge. Because microservices architectures divide apps into multiple pieces—essentially dis-integrating them – they introduce a new level of complexity to application environments.





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, and are hosted using multiple layers of infrastructure, such as a container environment running on top of virtual machines that in turn are hosted on physical servers. For this reason, the exact location of any particular microservice is fluid and ambiguous.

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. Traditional monitoring strategies will come up short in a microservices world.

The Technologies of the Microservices Age

We begin by defining the new technologies and design philosophies that are shaping 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, 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

Just ten years ago, apps 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. Today, 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

Today, it is difficult to map an application, service or piece of data to a particular server. That's because software-defined environments, which are now widespread, abstract applications and services away from the underlying infrastructure. This design makes microservices easier to implement because individual services no longer have to be tied to particular hosts.

Everything-as-a-Service

Organizations today have moved many of their workloads to the cloud. They consume compute and storage resources as a service. This makes their infrastructure more scalable and suited to host microservices applications.

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. 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 just a few years ago — 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.

The Challenges of Microservices

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 Microservices:

- **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 service quality management more difficult, increasing the difficulty of delivering high-performance, highly scalable applications around the clock.

Consider the following ways in which the software delivery practices of today make service delivery and quality assurance more difficult.

Complex dependencies

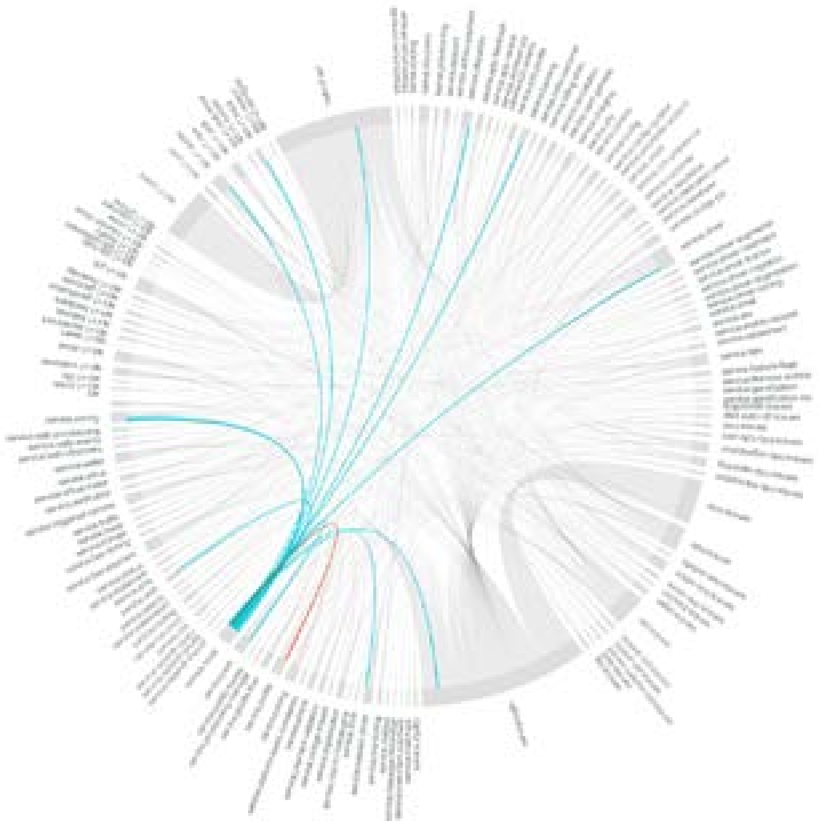
Mapping dependencies in a microservices environment is extremely important, but also extremely complex.

In a legacy environment, dependencies are relatively few in number and easy to identify. It's not hard to know which apps are running on which servers, or which storage service is linked to which app.

Under 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.

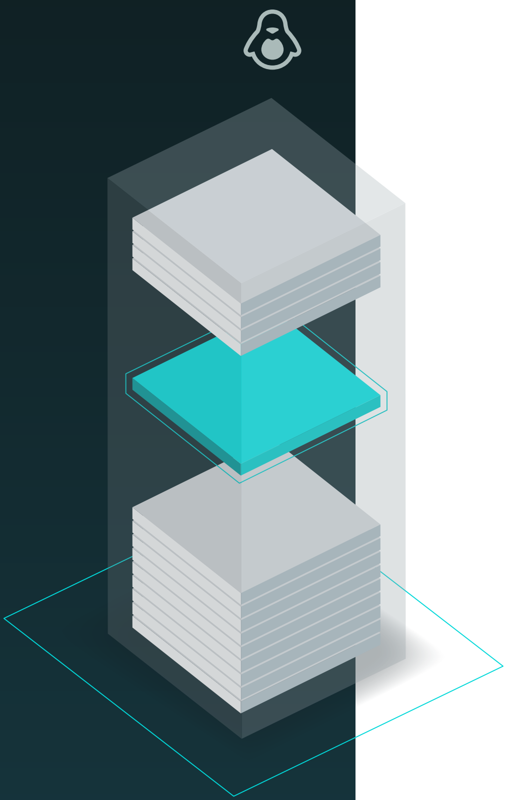
For these reasons and more, mapping the dependencies within a microservices environment is highly complex and difficult to do manually. So is the task of tracing an incident back to the root cause.

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 in order to resolve it.



A Journey into Microservices: Dealing with Complexity
Halo Tech Blog

One host quite commonly has many containers



Continuous delivery

Agile code is developed in small increments. While that makes software delivery faster and more flexible, it also means changes are introduced at a faster pace. Effective Service Quality Management requires DevOps teams to see changes when they occur and understand immediately the impact on scale, performance, error rates and more.

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 need to be monitored and the rate of change within the environment as containers spin up and down increases dramatically when an organization migrates to microservices.

The number of objects to monitor grows exponentially in microservice environments.

Microservices architectures

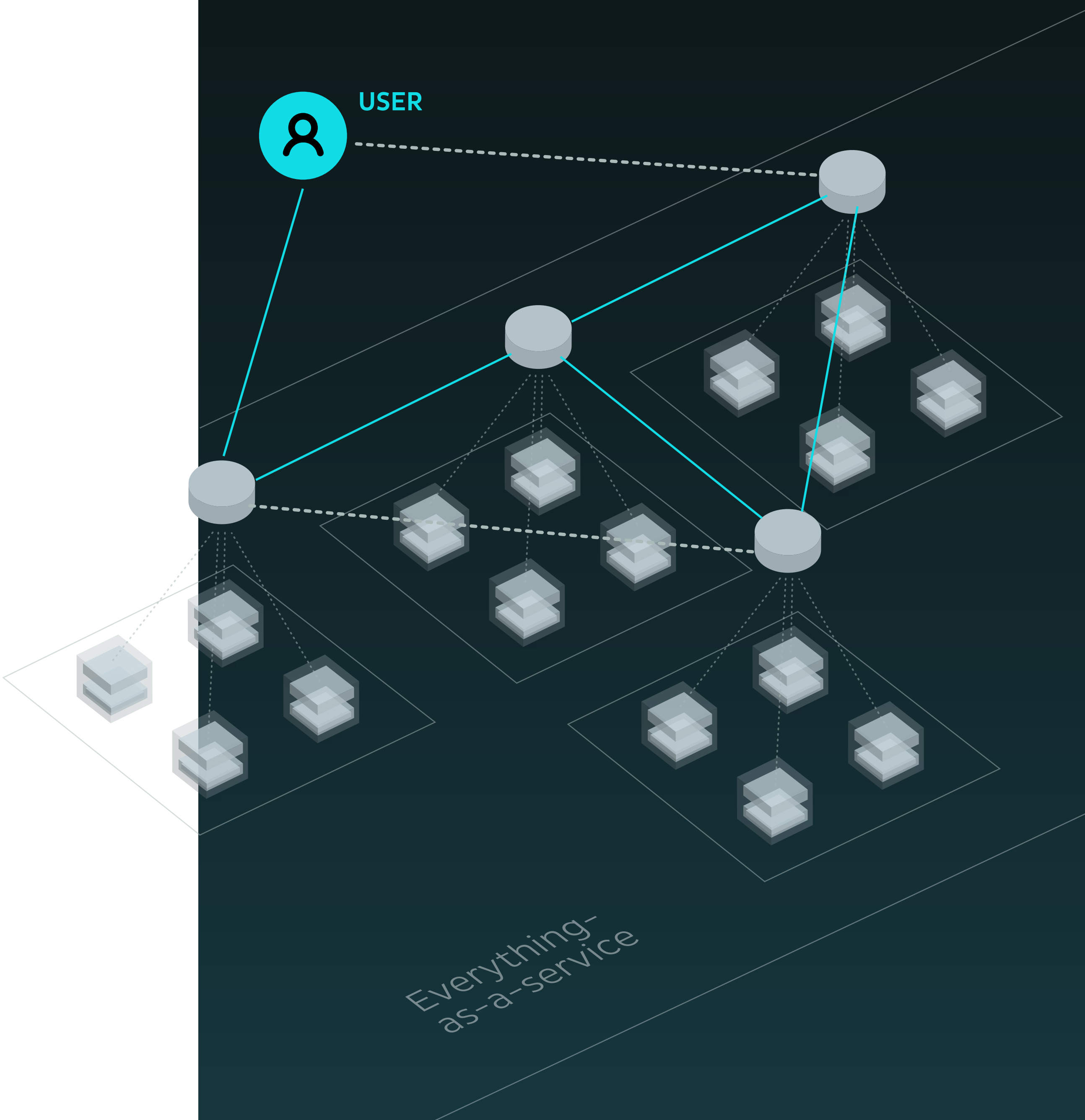
In a microservices environment, there are more services to monitor. It's not enough simply to know whether your app is up or down. You need to keep track of the many microservices that comprise your app—and you need to monitor them not just for uptime, but also for quality of service, so that you can identify and address reliability issues within a specific microservice before they degrade the performance of the entire app.

Everything-as-a-Service

The embrace of the Everything-as-a-Service model means many organizations no longer own or are able to map the underlying infrastructure on which their applications, services, microservices and containers live. When seeing and touching the servers that host your software is no longer an option, you need 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.

Understand incidents and dependencies precisely

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.

Share information easily 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

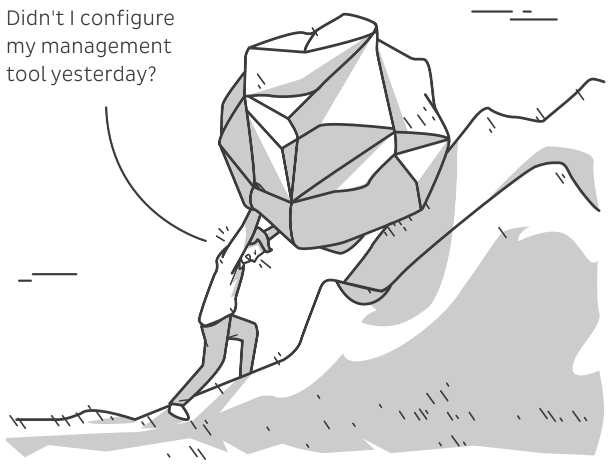
It's important to be able to detect and understand problems in real time as they occur. Real-time reaction requires an ability to separate incidents instantaneously from noise, identify the root of a service problem, and use service quality and performance data to develop an immediate response.

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.

INSTANA

Minimize manual configuration

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

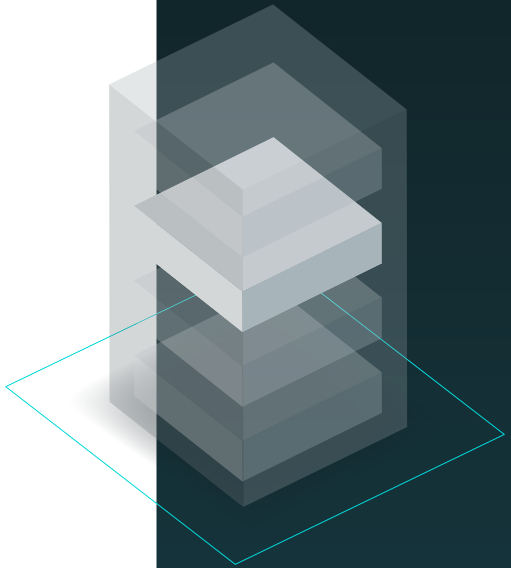


In Microservices environments, the only constant is change. Manual configuration of monitoring is a never-ending cycle.

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.

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



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.

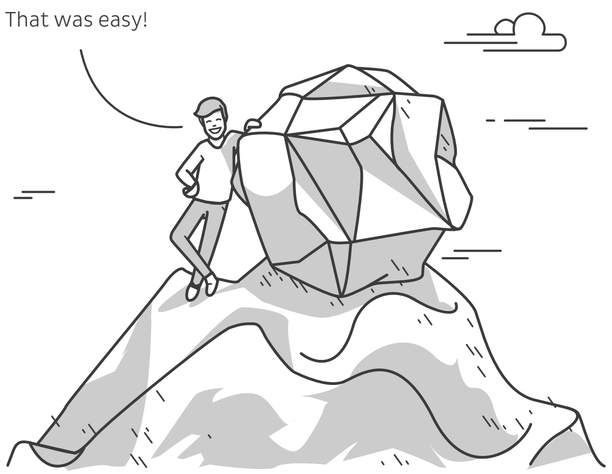
As noted above, automated anomaly detection (delivered via machine learning) and the mapping of service dependencies also help you to minimize the amount of manual configuration that your monitoring workflow requires. When your monitoring tools can detect anomalies and dependencies automatically, your 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.

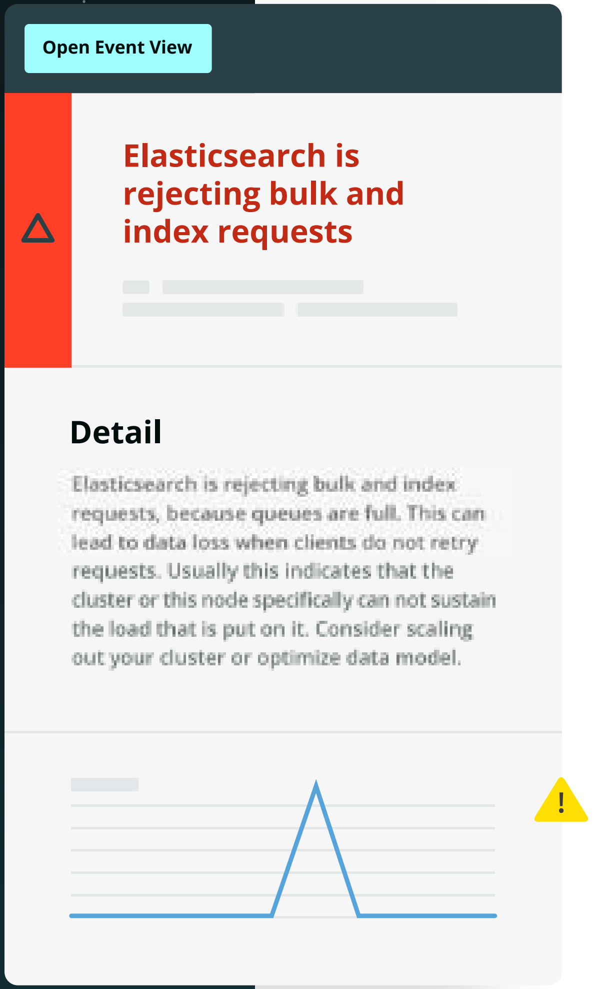
Continuous understanding is facilitated by ongoing, automatic rediscovery of environment data, dependencies and configurations.



Intelligent automated discovery and service health monitoring allows Operations teams to manage their systems and applications, not their management tools.

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.

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



Elasticsearch is rejecting bulk and index requests, because queues are full. This can lead to data loss when clients do not retry requests. Usually this indicates that the cluster or this node specifically can not sustain the load that is put on it. Consider scaling out your cluster or optimize data model.

A New Generation of Service Quality Management Tools

Achieving the complex, flexible, ultra-scalable monitoring strategy discussed above requires a fundamentally new approach to managing service quality. At Instana, we like to think of quality management solutions for microservices as an entirely new generation of systems management tools.

Organizations can no longer make do with older generations of tools that were conceived primarily to track the uptime or downtime of servers and measure application performance, or the somewhat more sophisticated set of solutions designed for the cloud.

Today, teams need a totally new generation of management tools—tools that automate setup, discovery and service quality monitoring decisions on their own, without requiring human expertise or configuration.

Instana is a leading provider of this new generation of Service Quality Management tools. Instana's Dynamic APM solution automates application discovery, agent deployment and monitoring configuration across the full microservice technology stack. Instana reduces troubleshooting effort by eliminating the noise and exactly identifying the triggering event and most likely cause of each incident. Essentially, Instana does what a human operations team cannot - manage the service quality across the entire microservice application environment.

About Instana

Instana provides the only APM solution that automatically discovers services, deploys agents and monitors component health for microservice and containerized applications.

Built to handle the demands of agile organizations, Instana continuously aligns application maps with any changes, detects behavioral anomalies and automatically provides a health score for each technology component.Organizations benefit from real-time impact analysis, improved quality of service, and optimized workflows that keep applications healthy.

The solution notifies DevOps teams when service quality is at risk, providing precise information identifying the triggering event and potential root cause.

Organizations benefit from real-time impact analysis and actionable recommendations to deliver higher service quality on all their business applications.

We invite you

Try Instana today

Instana Documentation

… to check out **Instana's documentation** to get a sense of just how much our solution can do. And if you're ready, give Instana a **try today**.

Stan
Your Intelligent DevOps Assistant

INSTANA

www.instana.com