dynatrace

Kubernetes platform observability best practices guide

How to streamline Kubernetes monitoring, security, and optimization at scale for platform engineering

Introduction

As organizations continue to modernize their tech stacks, Kubernetes has emerged as the de facto standard for managing containerized environments at scale. As a result, it is also becoming the platform on which many organizations are creating internal development platforms (IDPs).

Two-thirds (66%) of all organizations have adopted Kubernetes in production as a successful solution in helping companies modernize their operations.

– CNCF 2023 Annual Survey

However, the complex, dynamic, and ephemeral nature of Kubernetes renders observability of applications and multicloud infrastructures particularly challenging. End-to-end observability is key to aligning cross-functional teams and garnering actionable insights, as monitoring gaps in complex Kubernetes environments can significantly hinder teams' productivity and collaboration.

Taking a holistic inventory of an entire technology stack and its numerous components is difficult. However, determining whether those components are healthy, secure, and running optimally produces an entirely new set of challenges.

76% of technology leaders say the dynamic nature of Kubernetes makes it more difficult to maintain visibility into this architecture compared with traditional technology stacks.

– Dynatrace 2024 State of Observability report

Kubernetes challenges

Challenges that platform engineering and development teams face when attempting to implement Kubernetes at scale include:

1. Complexity and cluster instability

Given the complexity of the Kubernetes platform, cluster instability and wider application issues can arise for many reasons. The platform's complicated architecture and numerous components can make configurations and monitoring particularly difficult.

2. Resource and cost management

When not running efficiently, Kubernetes environments can quickly become expensive and resource-intensive. Proper resource allocation and scaling capabilities are necessary to avoid performance degradation and extraneous costs associated with Kubernetes environments.

3. Security

Security is a major concern when dealing with Kubernetes infrastructure. The increased number of variables in highly dynamic environments, manual configurations, and separate cluster ownership introduces more opportunities for vulnerabilities to emerge. By design, Kubernetes lacks built-in security, potentially allowing malicious actors to exploit unrestricted communication between pods and other Kubernetes-specific components.

4. Limited observability

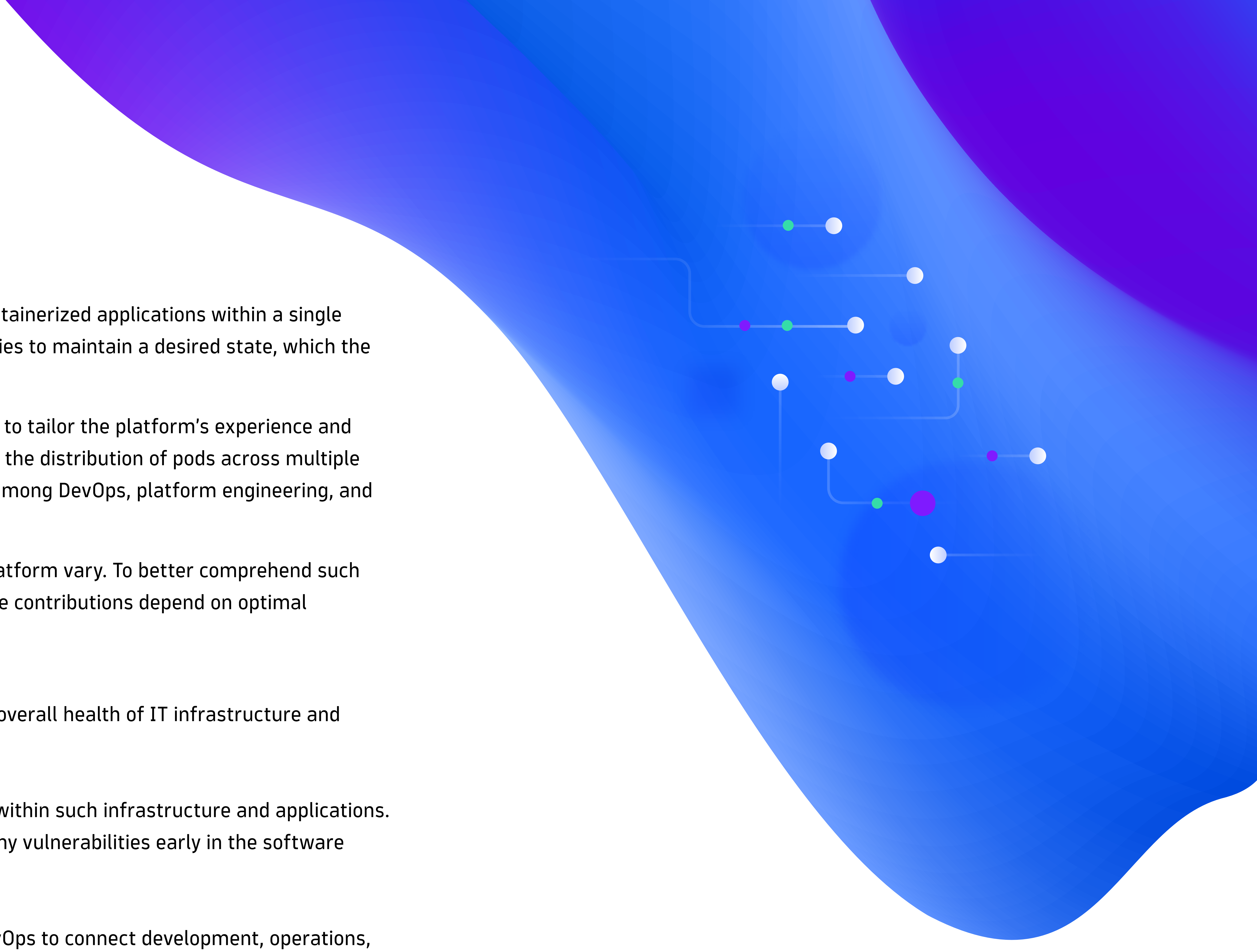
Traditional approaches to monitoring and observability can create tool sprawl and lead to an inefficient use of Kubernetes at scale. These approaches can leave significant monitoring gaps in highly dynamic environments, resulting in a lack of visibility into Kubernetes' health and performance.

5. Overwhelmed engineering teams

Moving beyond simpler monolithic applications into dynamic application containerization increases the cognitive load on engineering teams, which can temporarily result in decreased productivity. Platform engineering has emerged to address this complexity, but teams often lack the Kubernetes fluency needed to design and manage large environments at scale.

Still, the benefits of a well-implemented Kubernetes environment far outweigh any of its complications.

Why Kubernetes?



Kubernetes offers a highly flexible and scalable way to run, manage, and orchestrate containerized applications within a single underlying platform. A major reason Kubernetes is so popular is its self-healing capabilities to maintain a desired state, which the user establishes using declarative configurations.

Kubernetes also offers ample extensibility and open-source compatibility, allowing users to tailor the platform's experience and functionality to their unique needs and preferences. By ensuring high availability through the distribution of pods across multiple nodes, Kubernetes has become the go-to platform for operational efficiency and agility among DevOps, platform engineering, and security practices alike.

While each of these practices can benefit from Kubernetes, their dependencies on the platform vary. To better comprehend such differences, it helps to first understand how these practices interact and how their unique contributions depend on optimal Kubernetes functionality.

DevOps practices address the planning, developing, testing, deployment, and overall health of IT infrastructure and applications.

Security practices manage and mitigate vulnerabilities, threats, and attacks within such infrastructure and applications. Security is often built into DevOps processes ("DevSecOps"), helping identify any vulnerabilities early in the software development lifecycle (SDLC).

Platform engineering is a foundational practice that has emerged out of DevOps to connect development, operations, and security teams with greater operational efficiency by provisioning and maintaining standardized tools and practices through internal development platforms (IDPs). This self-service-based approach reduces the cognitive load on developers, which improves the developer experience (DevEx).

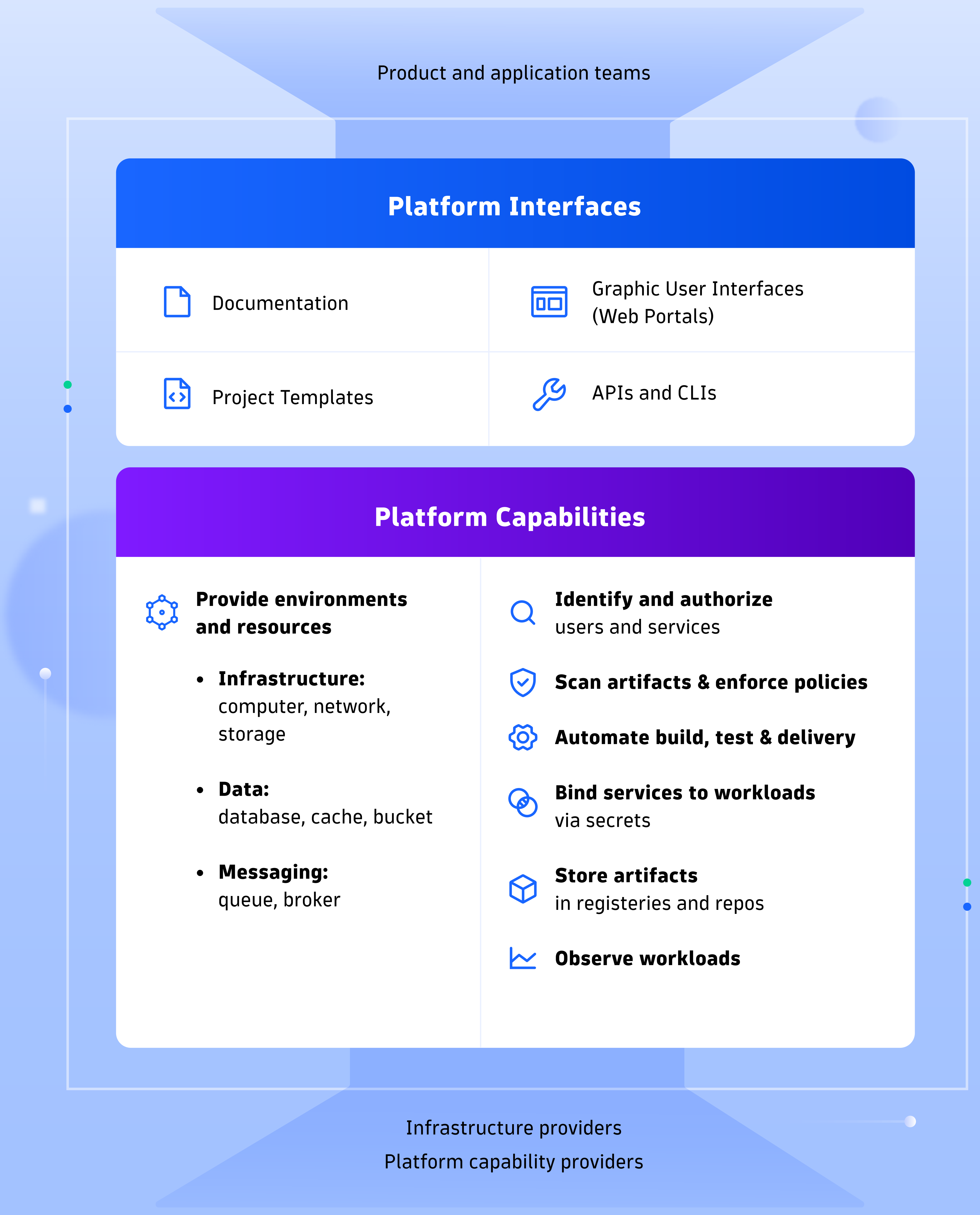
Self-service deployment models

As the "platform for building platforms," Kubernetes often serves as a foundational component of IDPs by enabling development efforts at scale using self-service deployment models. Self-service models consist of golden path templates, workflow automation to reduce manual toil, and digital guardrails to ensure governance and compliance. Teams can implement these practices using a well-equipped IDP, which should include key features such as self-service templates, application containerization, configuration as code (CaC), and other capabilities that streamline and empower the development process. The diagram to the right provides an overview of what an IDP should include at the most basic level.

By implementing an IDP with these characteristics and following a strong self-service model, teams can foster a positive and satisfying developer experience while optimizing productivity. With Kubernetes as a critical component of the IDP stack, these containerized environments must be healthy, secure, and running optimally to avoid performance degradation. Given the underlying role of Kubernetes in IDP functionality, problematic or unhealthy clusters can have a domino effect on multiple other components and subsequently compromise every stage of the SDLC.

Considering the importance of a high-functioning Kubernetes environment, it is crucial that observability and security are built into the platform by design. Instead of being an afterthought or mere addition to a system, observability must be embedded into Kubernetes design and maintenance, as observability can offer comprehensive visibility and guidance to ensure Kubernetes health, optimization, and security.

> Gartner® predicts that "By 2026, 80% of large software engineering organizations will establish platform engineering teams as internal providers of reusable services, components, and tools for application delivery, up from 45% in 2022."
>
> — Gartner, Top Strategic Technology Trends for 2024: Platform Engineering, October 2023. GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used herein with permission. All rights reserved.



Product and application teams
Platform Interfaces
Documentation
Graphic User Interfaces (Web Portals)
Project Templates
APIs and CLIs
Platform Capabilities
Provide environments and resources
Infrastructure: computer, network, storage
Data: database, cache, bucket
Messaging: queue, broker
Identify and authorize users and services
Scan artifacts & enforce policies
Automate build, test & delivery
Bind services to workloads via secrets
Store artifacts in registeries and repos
Observe workloads
Infrastructure providers
Platform capability providers

Adapted from CNCF 2023 Annual Survey

Unified observability and security for addressing cloud-native complexity

There are countless tools available to development teams for addressing their numerous cloud challenges. However, a holistic approach to observability can help organizations realize the significant value presented by cloud-native technologies like Kubernetes.

A unified observability platform empowers teams by delivering:

Comprehensive, end-to-end observability

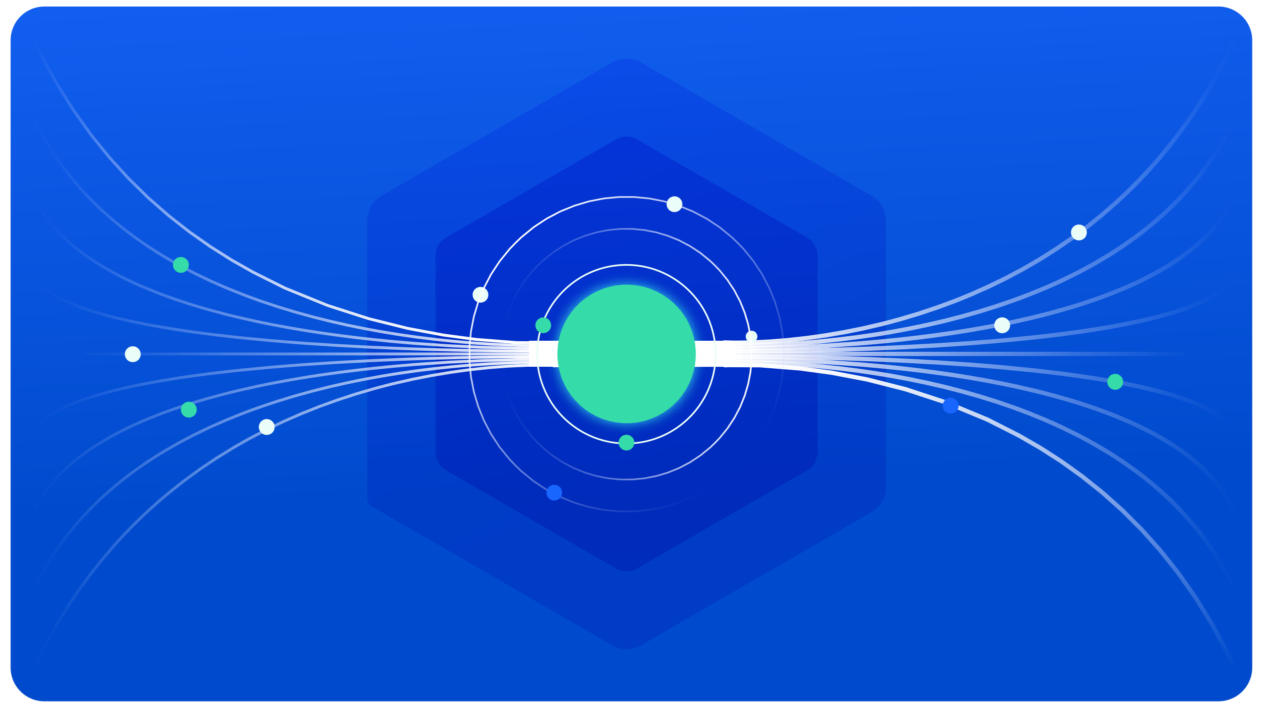
To conquer the complexity associated with the distributed nature of Kubernetes and its myriad microservices, it is imperative to have comprehensive end-to-end observability of the Kubernetes environment, the IDPs built on top, and the applications deployed to the platforms. Comprehensive observability enables teams to gather data and insights from every corner of their Kubernetes environment, ensuring that no system event or component goes unmonitored. It also enables teams to track their workloads and helps them quickly pinpoint the root cause of an issue.

Platform health management

An optimal Kubernetes practice requires that all components are consistently healthy. A Kubernetes platform health management solution should provide an accessible interface that readily displays and communicates the health status of all critical Kubernetes objects, including clusters, nodes, namespaces, workloads, and other relevant components. A solution should identify and display performance degradation in real time and send automated alerts to the appropriate people. A platform health management solution should also deploy consistent, automatic testing to rapidly and proactively identify any issues. Additionally, an easy-to-understand platform health solution helps reduce Kubernetes knowledge gaps, presenting opinionated health states in detail without excessive complexity.

Centralized data in context

Kubernetes platform operations and application teams must be well-aligned to ensure cluster optimization. To facilitate alignment, a centralized source of data is crucial. By storing all data in a singular location, application and platform operations teams are always working from the exact same sets of information. With a centralized view of CPU, memory usage, and other Kubernetes system demands, teams can better align on how to allocate and scale resources. Using centralized datasets, teams can directly compare the quantity of resources that different components or services are consuming, helping identify areas with the highest potential for optimization.



Kubernetes security

The lack of built-in security, prevalence of manual configurations, and constant updates to the Kubernetes platform itself make it challenging to keep track of the newest vulnerabilities that may arise while running outdated Kubernetes versions. A unified, end-to-end observability and security solution equipped with the ability to identify vulnerabilities in production is required to ensure that the multitudes of clusters and the services running on them remain secure. By identifying vulnerabilities at runtime, a solution should detect and block attacks in real time and help teams prioritize and facilitate remediation workflows.

Superior developer experience

To combat the Kubernetes complexity that threatens to hinder developer satisfaction and productivity, a solution should streamline and automate the developer experience (DevEx) for Kubernetes and the full CI/CD tool stack. Automation as a self-service through an IDP is especially helpful in a Kubernetes context, where maintaining a large quantity of workloads and pods would prove virtually impossible if attempted through manual effort alone. AI that provides intelligent code suggestions and workflows uniquely optimized for an organization's Kubernetes environments reduces the cognitive load associated with Kubernetes complexity.

By adopting these pillars of Kubernetes management, platform teams can gain a holistic understanding of all Kubernetes objects and their performance. With a comprehensive view, teams can confirm these objects are healthy and secure, ensuring the environment is running optimally at scale. Furthermore, by providing the capabilities mentioned above, a unified observability platform serves as a centralized administration console for Kubernetes. With a centralized administration console, cross-functional teams can gain a singular, aligned view of all components across the entire Kubernetes stack.

The adoption of a centralized administration console alone does not guarantee Kubernetes operational excellence, but there are certain best practices that all organizations can implement to achieve success within Kubernetes environments.

Best practices for managing and optimizing Kubernetes

When getting started, addressing the complexity of Kubernetes environments and the actions required to optimize them can quickly become overwhelming to platform builders and developers alike. To help navigate this complexity and find the right solutions, it can be helpful to think about core use cases in two categories:

Kubernetes platform health and performance

A Kubernetes management solution should allow you to answer the following questions:

- **Comprehensive Kubernetes inventory:** Do I know what I have, its location, and who is responsible?
- **Kubernetes platform health:** Is my platform healthy, stable, and secure?
- **Resource utilization:** Is my platform running optimally and efficiently?

Platform delivery and observability to enable developers

Once you're confident your platform is healthy, optimized, and secure, your solution should make it easy for development teams to work more efficiently.

- **Observability-driven development (ODD).** Ensure visibility across every step of the SDLC to get immediate feedback on the health and performance of artifacts, reducing mean time to observability.
- **Level up DevEx.** Provide best-in-class developer experience and enable agility and flexibility by designing platforms specifically suited to developers' needs. In certain scenarios, this can be achieved by integrating with open-source frameworks for building developer portals, such as Backstage. However, designing the platform with developers in mind is the best way to ensure a positive DevEx.

The following best practices describe how platform builders and developers can achieve these Kubernetes goals using an observability-based approach.

Best Practice 1: Gain end-to-end observability of your entire Kubernetes estate

Comprehensive observability of the Kubernetes estate is essential for navigating and monitoring Kubernetes complexity. The simplest and most efficient way to achieve end-to-end observability is by introducing a holistic and fully equipped platform solution. This solution should offer observability of the entire environment across all clusters and provide relevant data and intelligent insights for every node, pod, service, and other Kubernetes objects. This observability solution should also offer visibility into any of the developer toolsets or frameworks provisioned by an IDP. The key is to gather data across the entire estate for a holistic understanding of how the Kubernetes environment is functioning, as opposed to collecting and analyzing isolated data points.

Automate observability deployment

Observability should be automatically rolled out into every Kubernetes cluster to streamline developer tasks and reduce the complexity associated with manual configuration. Teams can use configuration-as-code templates that allow development teams to set up monitoring, observability, and security policies efficiently, minimizing the need to create custom solutions.

Expand observability benefits

Platform teams can greatly benefit from end-to-end observability, which provides key insights into DevOps pipelines and includes capabilities such as application performance monitoring (APM) and tracing.

APM offers a comprehensive view of application health and functioning within the Kubernetes context and beyond, identifying potential bottlenecks and optimization methods for Kubernetes components and the services they interact with.

Distributed tracing is another essential asset, enabling end-to-end visibility of requests as they flow across the entire environment.

Simplify observability toolchains

Adopting a unified solution provides a single source of truth that teams can refer to, facilitating much-needed communication, efficiency, and collaboration when dealing with complex Kubernetes environments that produce massive volumes of data.

To learn more about how to implement Dynatrace on Kubernetes for comprehensive observability, visit the Dynatrace documentation topic set up Dynatrace on Kubernetes.

Best practice 2: Ensure cluster health and security

Any IDP needs to be as resilient, secure, and available as the end-user applications it supports. Development teams must be able to trust the platform on which they are developing customer applications. This trust is built on the confidence that all Kubernetes components and the services they rely on are consistently in a healthy state.

Centralize multi-cluster and single-cluster analysis

A broad range of issues can arise when working with Kubernetes that may impact its health and stability. Issues can result from problems with workloads and nodes, networking, or Kubernetes core objects. Common problematic events include nodes running out of resources, containers frequently restarting, and pods getting stuck in a pending state due to misconfigurations or missing resources. Effective troubleshooting requires teams to centralize multi- and single-cluster analysis to easily view the health and status of Kubernetes components and core objects.

Analyze utilization details, logs, and traces

Observability backends that collect data exposed from the Kubernetes API provide an inventory of objects and resource utilization details, such as phases and conditions.

Observability solutions should provide details about logs and events on workloads. Access to events is critical for troubleshooting issues, while logs are important for understanding if issues are related to an application. In addition to logs and events, inventory and usage data are required to clearly understand a given situation without the trial and error of running a command line in Kubectl.

Tracing is another key capability of an observability solution. If a workload experiences issues due to compromised dependencies, it is vital to understand the impact of those issues, their prevalence and path across the environment, and their effect on specific users.

Amplify accuracy and efficiency with AI and automation

To gain actionable and automatable insights from observability data, a solution needs AI-driven analytics capabilities to help identify the root cause of issues and prioritize them based on severity. The solution should trigger notifications for responsible teams and initiate automated workflows where appropriate.

These AI-powered operations significantly simplify Kubernetes management. They enable non-Kubernetes experts across platform and development teams to quickly identify and resolve issues and vulnerabilities, while pointing out root causes to prevent future occurrences.

To learn more about how to ensure cluster health and security for Kubernetes, visit the Dynatrace documentation topic Assess and troubleshoot cluster health.

Best practice 3: Ensure Kubernetes security with runtime vulnerability detection, prevention, and remediation

Despite the need for secure containerized environments across organizations of all verticals, Kubernetes offers no security protections out of the box. Because modern applications are made up of highly distributed microservices, they often contain hundreds or even thousands of vulnerabilities. This leaves systems susceptible to cyberattacks, such as denial of service (DoS), cross-site scripting (XSS), remote code execution (RCE), and more. To build and deploy a secure cluster, teams must perform a wide range of actions across the SDLC.

Automatically detect vulnerabilities at runtime

To quickly evaluate the high volume of vulnerabilities in modern applications, teams need the ability to automatically detect and prioritize vulnerabilities and attacks in real-time. This automatic real-time detection should account for multiple types of vulnerabilities, such as those that stem from flaws in third-party libraries and application code.

A unified observability solution should be able to perform this detection at runtime, ensuring teams can identify, prioritize, and remediate vulnerabilities on the affected Kubernetes object before they proliferate and expose the system to further exploitation.

Prioritization and auto-remediation

Not all vulnerabilities pose the same level of risk. For example, a severe vulnerability on a service that's not actively in use may be a lower priority for remediation than a moderate vulnerability that's exposed on a high-traffic service.

For this reason, an observability solution should be able to use AI to automatically detect how vulnerabilities are exposed. An observability solution should also tag vulnerabilities with identification and remediation information, such as specific risk level, issue description, affected component, and fix recommendations. With this data, teams can concentrate their efforts on the most business-critical issues and automate responses and remediation workflows.

To learn more about how to implement these specific Kubernetes security best practices, as well as many others, watch this *Is It Observable* episode on How to Secure your K8s Cluster. You can also visit the Dynatrace documentation topics Prioritize vulnerabilities and Manage remediation for more information.

Best practice 4: Optimize resource utilization

Teams need a way to automatically right-size and optimize Kubernetes resources to control costs and effectively utilize resources.

Right-size Kubernetes resources automatically

One way to reduce costs and increase efficiency is to identify workloads that underutilize their requested resources. You can also implement workflow automation to scale resources up or down based on projected consumption needs with no manual involvement required. This involves identifying clusters with the potential for optimization, analyzing and prioritizing them based on historical data, then optimizing them to make better use of resources.

Optimize workload resources

With the knowledge of how your workload usage behaves over time and which workloads should be prioritized for optimization, you can begin optimizing the requests and limits of the workload by reducing requests to a lower value that still guarantees consistent performance even when usage spikes occur. To avoid CPU throttling or out-of-memory kills, do not immediately reduce the limit quantity to the lowest viable amount. Rather, take note of how the workload behaves with the slightly reduced number of limits, then adjust accordingly.

Optimize disk space

Optimizing disk space is another best practice for managing the numerous services across your Kubernetes environment. Inadequate allocation of resources can push capacity limits, leading to service slowdowns and other issues. When attempted manually, disk resizing is a cumbersome and tedious process. However, workflow automation can streamline disk resizing processes and free up ample time for development and platform teams.

To learn more about how to optimize resource utilization for Kubernetes both manually and automatically, visit the Dynatrace documentation topics Optimize workload resource usage and Predictive Kubernetes operations.

Best Practice 5: Enable best-in-class DevEx

Given their pivotal role in delivering high-performing software and driving innovation, it is crucial that organizations provide a superior developer experience (DevEx). With a positive DevEx, developers are empowered with the resources and practices that foster greater productivity, efficiency, and satisfaction. Consider the following best practices to enhance DevEx.

Implement observability-driven development

Mitigating issues before they proliferate is critical when working in dynamic, containerized cloud-native environments like Kubernetes. Observability-driven development (ODD) provides holistic visibility into developing issues at every phase of the SDLC, enabling consistency and governance.

ODD also makes developers' lives easier by providing them with simple, fast, and more secure ways of deploying applications. By requiring only minimal information upfront, developers gain immediate access to essential observability data to ensure performant applications.

Standardize technologies and processes

Standardizing technologies and processes across teams can significantly reduce the mean time to observability. To achieve this standardization, teams can enable out-of-the-box observability and transparency through the following best practices.

- Establish “golden paths” for developing and deploying new services and applications that give development and SRE teams necessary insights into their workloads for effective troubleshooting and identification of performance bottlenecks.
- Set guardrails to ensure that monitoring, security, and SLO rules are met based on the organization’s unique mix of technology, maturity, and criticality requirements.
- Define the minimal set of information and metadata required from development teams at each stage of the SDLC based on stakeholders' requirements from key teams, such as DevOps, developers, security, and SREs. This ensures that observability and security standards are applied automatically and consistently.
- Use configuration as code tools like Monaco or Terraform to prepare dedicated templates with pre-defined rules. This practice enables observability to be built in from the start, reducing developers' manual toil in the onboarding process and improving DevEx.

Day zero observability is achieved by including important observability data in extensible and flexible golden path templates built for developers, reducing their cognitive load and ensuring secure and compliant application development.

To learn more about standardizing technologies and processes with practices such as configuration as code, visit the Dynatrace documentation topic Dynatrace Configuration as Code.

Integrating with developer portals

The best IDPs are managed like products, and products must cater to customer needs. Developers are the platform team's primary customers, so a best-in-class developer experience should be provided. A key aspect of improving developer experience includes adding tailored services based on specific needs and requests. Integration with a developer portal can greatly enhance and ease the developer experience when implemented in a manner suited to the organization's needs.

Developer portals such as Backstage provide developer-oriented capabilities that extend IDPs, offering flexibility and autonomy across the entire SDLC.

Be sure to install plugins that provide easy access to observability and security insights to make the most of your developer portal. To learn more about integrating developer portals with your observability solution, visit the Dynatrace documentation topic Integrate Dynatrace into Backstage to level up Developer Experience.

Adopting a centralized administration console equipped for Kubernetes complexity

Managing multi-cluster environments can be challenging, but with a well-equipped, centralized administration console, overcoming Kubernetes complexity becomes much easier for experts and non-experts alike. With this single source of truth, teams can gain comprehensive visibility of their entire Kubernetes estate while improving collaboration and communication.

Offering this level of comprehensive visibility, the Dynatrace platform is a unified observability and security solution that acts as a centralized administration console. Removing the need for disparate tooling, Dynatrace provides all the necessary components for monitoring, optimizing, and securing your cloud-native environments. Through the Kubernetes Platform Monitoring solution, teams can centrally collect, visualize, and analyze telemetry data across multi-cluster environments at the Kubernetes object level. All observability data—metrics, logs, traces, and events—are stored in a centralized location with context relating to system dependencies and functionality. This centralized view provides clarity into Kubernetes workloads, empowering platform, development, and security teams to gain alignment, preserve resources, and embrace innovation.

AI-driven insights to automate Kubernetes optimization

Going beyond simple dashboarding, Dynatrace provides insights into Kubernetes objects and offers intelligent suggestions driven by causal AI for remediation and optimization processes across the Kubernetes estate. Additionally, predictive AI from Dynatrace Davis delivers proactive insights to address forecasted problematic events, such as issues with resource usage and consumption to help prevent issues before they become outages.

Using Dynatrace as a centralized administration console and single source of truth significantly enhances developer experience and cross-team collaboration by reducing the need to switch between disparate tools to fully understand the Kubernetes stack. The ability to toggle Dynatrace features on and off also enables a customizable experience for different teams.

No matter where you are on your Kubernetes and platform journey, Dynatrace can meet you there.

Working with complicated Kubernetes infrastructure doesn't have to mean sacrificing productivity or overwhelming developers. With Dynatrace, solving Kubernetes complexity is just one deployment away. Try the Dynatrace Kubernetes Platform Monitoring solution today.

Set up Dynatrace on Kubernetes

Platform Engineering Documentation

Dynatrace (NYSE: DT) exists to make the world's software work perfectly. Our unified platform combines broad and deep observability and continuous runtime application security with the most advanced AIOps to provide answers and intelligent automation from data at enormous scale. This enables innovators to modernize and automate cloud operations, deliver software faster and more securely, and ensure flawless digital experiences. That's why the world's largest organizations trust Dynatrace® to accelerate digital transformation.

Curious to see how you can simplify your cloud and maximize the impact of your digital teams? Let us show you. Sign up for a free 15-day Dynatrace trial.

dynatrace.com/blog @dynatrace

dynatrace