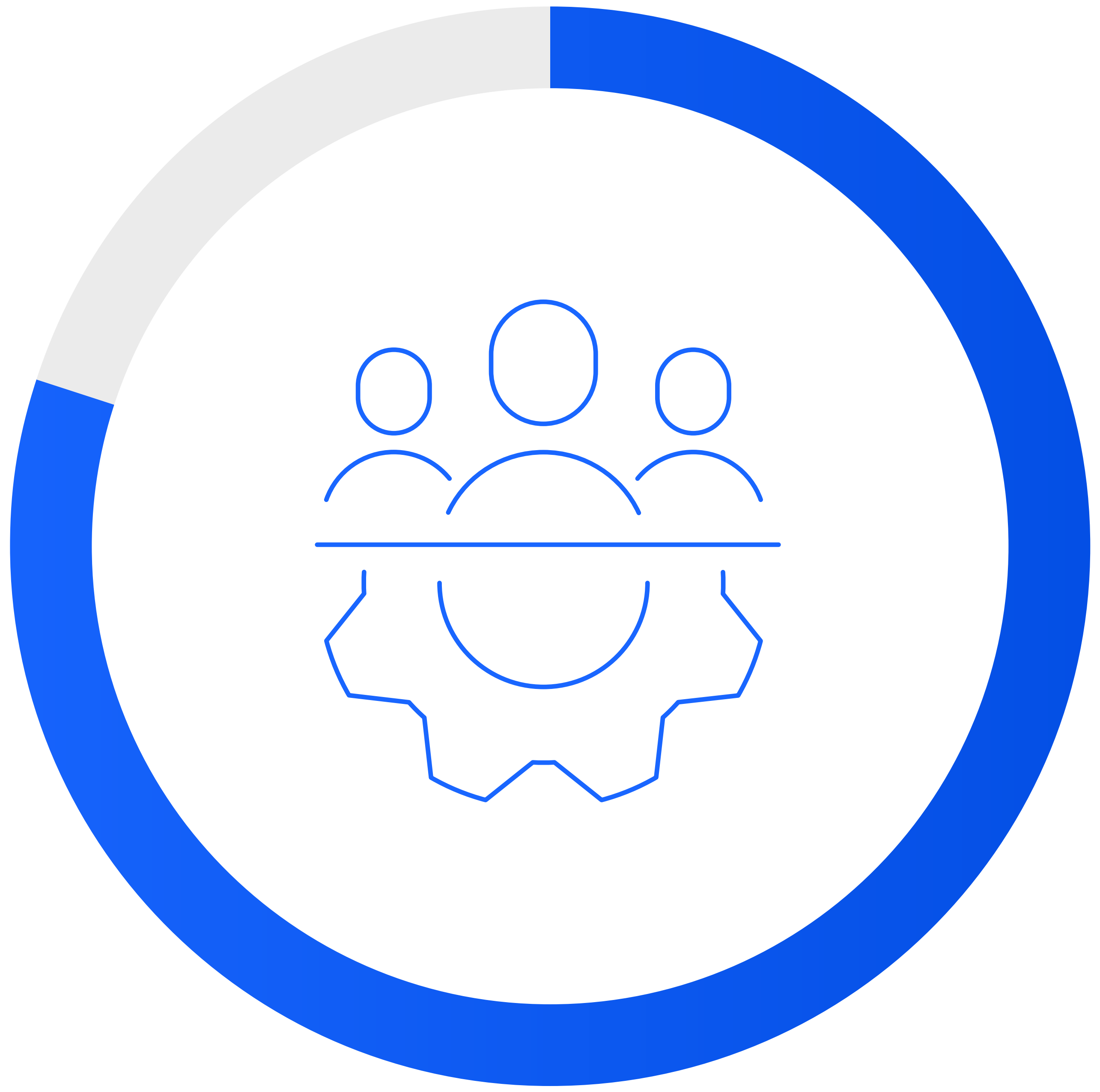
dynatrace

The observability guide to platform engineering

Use observability and security to drive analytics and automation at scale.



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

Bart Willemsen, Gary Olliffe, Arun Chandrasekaran. 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.

October 16 2023

What is platform engineering?

Platform engineering is a modern, core engineering discipline that has emerged as a way to accelerate the development and deployment of software that is resilient and effective at scale.

The goal is to operationalize DevSecOps and SRE practices by providing an internal self-service platform with development workspace templates. These templates reduce the cognitive load on engineering teams and provide fast development feedback cycles out of the box.

An internal development platform (IDP) encompasses a set of tools, services, and infrastructure that enables developers to build, test, and deploy software applications. These platforms are tailored specifically to the needs, demands, and goals of an organization.

IDPs enable platform engineering teams to deliver:

- Accelerated innovation and development
- Improved developer experience
- Higher developer productivity
- Reduced infrastructure cost

And more

IDPs empower development teams to dedicate more time and effort to delivering best-in-class products to their end customers.

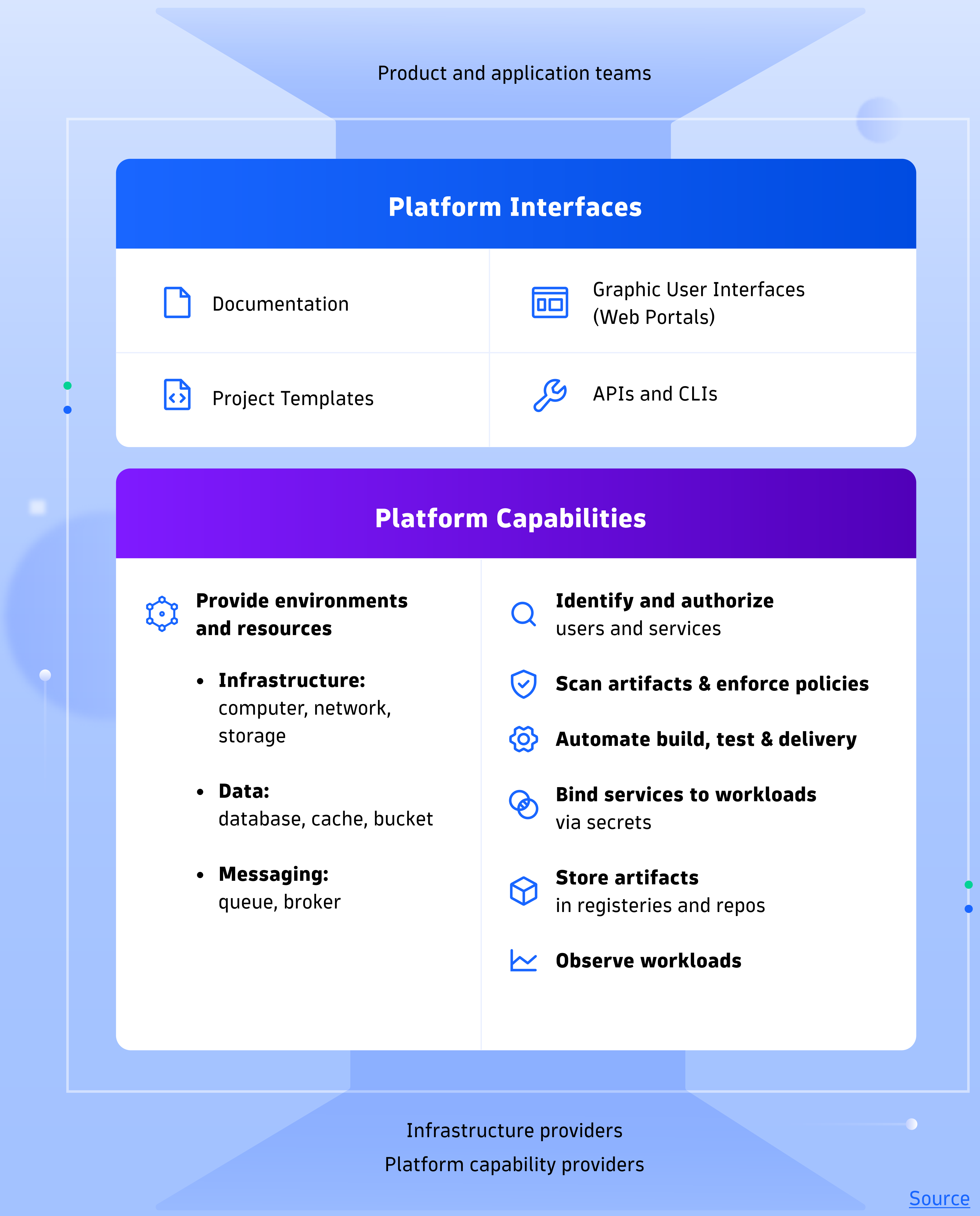
Effective Internal Development Platforms

Most IDPs are containerized and built on a Kubernetes-centric infrastructure with a core set of technologies. The diagram to the right shows what a basic IDP should include.

Core components of an IDP:

- Self-service templates*
- Application containerization
- Infrastructure as Code (IaC)
- Security
- Compliance
- Automation
- Observability

*e.g. for onboarding a new application, creating a new test environment, or automating incident remediations.


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

Source

dynatrace

Critical capabilities of an IDP are provided as self-service to development teams. Some of these self-service capabilities include:

- Platform services (service mesh, data storage, security vaults, policy agents)
- Delivery services (continuous integration, continuous delivery, Git, GitOps)
- Observability services for monitoring, automation, and security
- Templates for development workspaces including all of the above

Platform engineering also offers the ability to:

- Automate key workflows like validation
- Achieve optimal performance and security
- Ensure resiliency
- Optimize resource usage

Treat your platform as a product

A key aspect of platform engineering is approaching your platform with a product management mentality. Design and optimize every aspect for the benefit of the user — the developer.

Looking at an IDP with a product management lens:

The product:

An IDP that provides self-service for infrastructure, services, and support for the development teams as they build, test, and deploy applications at scale.

The customer:

Development teams who want to accelerate the creation of high-quality, resilient apps — with low effort for onboarding new applications and new developers.

Maturing DevOps with platform engineering

“You build it, you run it” doesn’t scale anymore.

DevOps alone doesn’t meet the demands of cloud-native software development. Platform engineering applies DevOps at a cloud-native scale.

Effective DevOps supports cross-functional cooperation and boosts efficiency. Platform engineering builds off of traditional DevOps practices and takes it a step further.

By providing a framework and tech stack to apply DevOps principles at scale with Kubernetes-centric infrastructure, platform engineering allows organizations to realize many of the benefits DevOps promised to deliver.

In practice, mature platform engineering includes centrally maintained development tooling, templates for CI/CD toolchains, processes, and application lifecycle orchestration solutions via a unified platform that development teams can use for greater efficiency. It enables a frictionless developer experience with minimum overhead, reducing cognitive load and providing fast feedback out of the box.

Platform engineering's standardized processes and technologies across development and operations teams lead to significant gains in developer productivity and improvements in developer experience.

Additionally, the platform engineering approach allows organizations to take advantage of economies of scale. Platform engineering consolidates sprawling tech stacks to:

- Manage cost
- Optimize resource usage
- Improve governance

Cloud-native platforms require a new approach to observability

To be broadly adopted by internal development teams and deliver on its promise of unlocking DevSecOps at scale, a Kubernetes-centric IDP requires other services and components to tackle challenges related to:

- Visibility
- Resource utilization
- Security
- Orchestration
- Collaboration

Top engineering teams unlock faster deployments with fewer errors by empowering developers to manage their own deployments. Enabling self-service requires developers to debug code quickly — which isn't an easy task in complex environments. Observability makes all the difference.

Observability isn't just about the user-facing application. Effective platform engineering — as DevOps at cloud-native scale — enables self-service for developers and development teams (through Golden Paths or templates), often requiring automation in the background. To provide guard rails and governance, the system has to be observable.

When teams implement it properly, platform engineering brings end-to-end observability to the full software development lifecycle — from development to release, to operate, and to retire.

Implementation

Fortunately, organizations don't have to replace every tool, vendor, and practice to implement platform engineering. Often, DevOps and cloud-native processes continue — managing systems and pipelines and working on automation and self-service.

What changes?

- How those platforms are implemented
- Who you're building those platforms for (and who's interacting)

Benefits

Accelerate development and support developer productivity

According to the State of DevOps Report, over two-thirds of all organizations (68%) implementing platform engineering have already seen increased development velocity.

Organizations with platform engineering teams enjoy various speed and efficiency benefits. These organizations report:

1. Enhanced system reliability
2. Improved efficiency and productivity of work
3. Faster delivery time

Improve developer experience

A best-in-class developer experience gives your organization a strategic advantage. Improving the developer experience:

- Reduces cognitive load of the developer,
- Reduces overall burden on the operations team,
- Improves quality of services from internal IT operations resources, processes, and practices,
- Provides a more efficient path for product delivery, and
- Provides fast feedback cycles to developers.

Optimize resources and mitigate cost

With a shared pool of resources for all projects, standardized practices, and the power of AI, platform engineering optimizes resource usage, leverages economies of scale, and mitigates cost.

Standardize tech stack to minimize risk and increase governance

Effective platform engineering enables teams to keep complexity low and standardize delivery while supporting developer autonomy.

- Security and compliance are built in by the design of the platform components and services.
- Golden Path templates standardize delivery, providing governance.
- Identifying and remediating issues or misconfigurations across templatized and standardized services becomes quick and easy.

Core platform observability and security principles

Kubernetes is a good starting point for building a platform. It allows platform engineering teams to provide self-service capabilities and features to their DevSecOps teams, but it also introduces complexity to the cloud environment.

To manage and overcome the complexity introduced by today's business needs and complex multi-cloud environments, organizations need the automatic answers and insights that only a unified observability and security solution and platform can provide.

Follow these core principles when building your Internal Developer Platform (IDP) to manage an application or service throughout its Software Development Lifecycle (SDLC).

1. Core platform observability

A critical first step to actionable platform insights is ensuring comprehensive monitoring of the artifact; the full software development lifecycle; and the underlying IDP.

Artifact: How does my product behave in various environments (staging, testing, production)?

Platform Toolstack: What is the health and utilization of the tools in use (e.g., BACK stack)?

Note: This does not yet include metrics of how efficiently the artifact moves through the pipeline.

Best practice: Treat (critical) platform services like any other business-critical service. Prioritizing observability allows for effective incident triage, issue resolution, and improved developer experience.

Aspects of core platform observability

- **Ensure resiliency, availability, and security** by applying the same practices to all platform services.
- **Optimize usage, performance, and licensing** to positively impact the application lifecycle and reduce cost.
- **Detect misconfiguration or misuse** by discovering upcoming problems and issues to educate users.

2. Release information

Attach current version and stage information to services for easy triaging and issue resolution. Teams can promote the version and stage information to Kubernetes containers as environment variables or annotations.

3. Pipeline efficiency

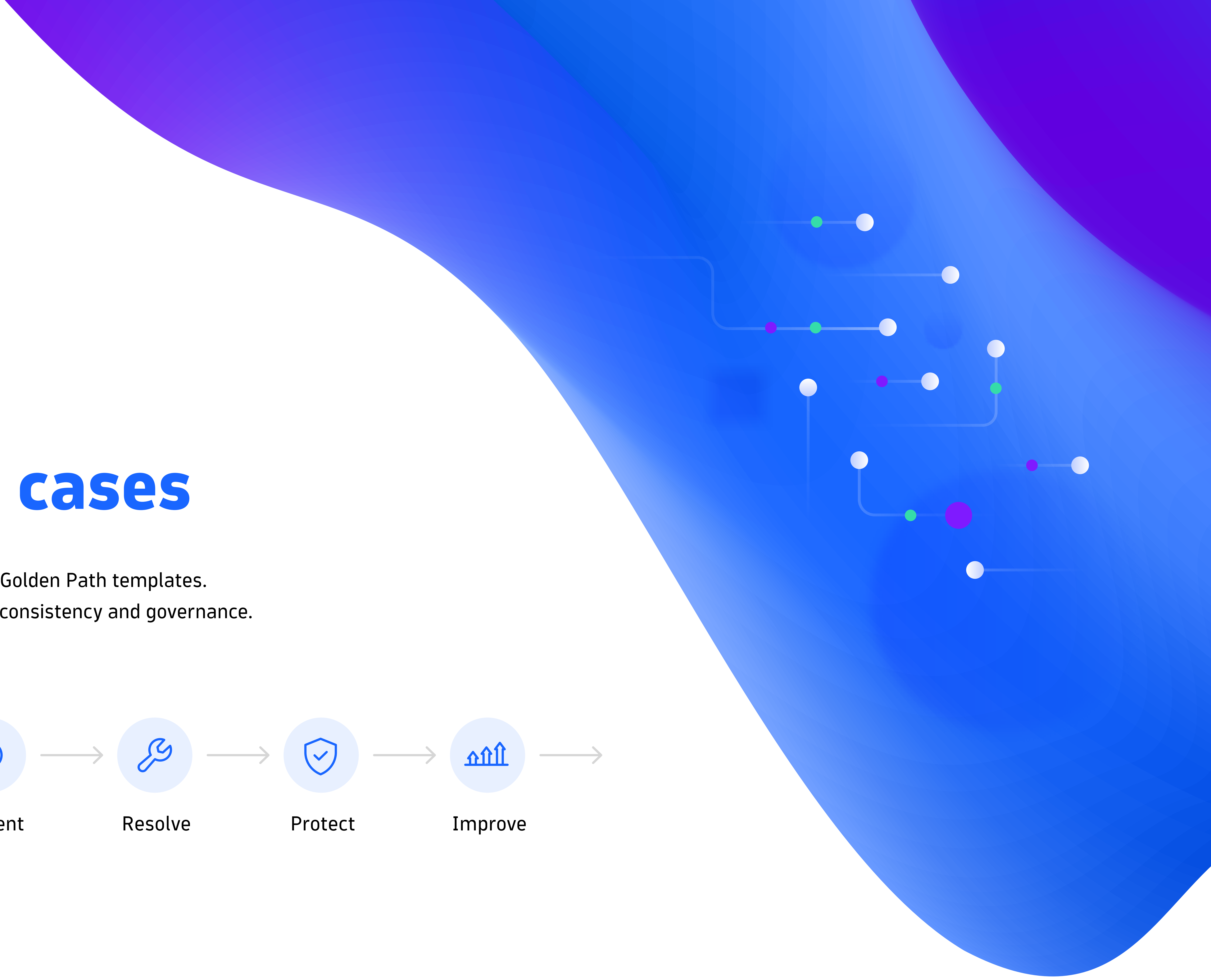
Once monitoring of the platform tool stack is in place and there is visibility into the health of the IDP, teams can set up processes for measuring pipeline efficiency. Logs, events, and telemetry data from pipeline or workflow executions are critical to this analysis.

Teams can create DORA (DevOps Research and Assessment) metrics by analyzing data from pipeline and workflow logs, events, and traces. These metrics serve as benchmarks for evaluating performance and play a crucial role in informing investment and resource allocation decisions.

Most commonly used tools in a cloud-native environment already emit events or telemetry data, yet this area (currently) lacks particular standards and semantic conventions.

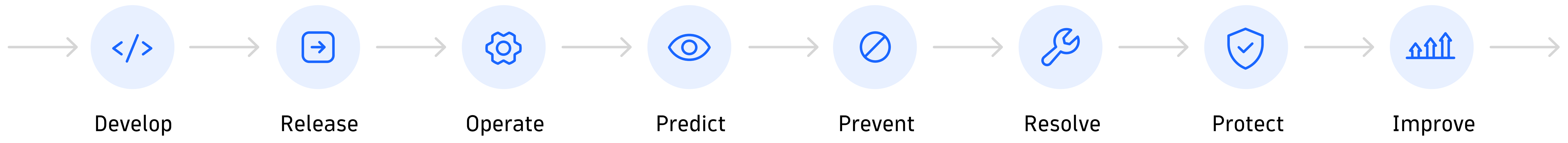
4. AI Observability

With the rise of generative AI in DevOps Portals and IDP tools, observability will be even more critical for managing costs and ensuring a best-in-class experience.



Platform engineering use cases

Effective platform engineering delivers most capabilities in a self-service manner through Golden Path templates. Templates provide autonomy for rapid, secure innovation while maintaining guard rails for consistency and governance.



The following use cases are either available to the platform engineering team to make their work more observable and easier, or they can provide automation and self-service capabilities for their users — the development teams.

Observability guide to platform engineering

Develop	Release	Operate	Predict	Prevent	Resolve	Protect	Improve
✓ Test pipeline observability	✓ Release validation	✓ Cloud cost optimization	✓ Predictive K8s operations	✓ Preventing customer-facing bugs	✓ Instant intrusion response	✓ SOC noise reduction	✓ Always-on app profiling
✓ Continuous testing validation	✓ Progressive delivery	✓ Observability for infrastructure	✓ Forecasting in data analytics	✓ Unified exposure protection	✓ Incident triaging and remediation	✓ Automate security findings at scale	✓ K8s utilization improvement
✓ Observability driven development	✓ Pipeline observability	✓ K8s monitoring	✓ Forecasting in workflow automation	✓ Continuous security posture awareness	✓ Error budget alerting	✓ Contextualized threat detection and incident response	✓ Business aligment improvement

dynatrace

Develop

This category contains all activities around planning, developing, building, and testing services and applications on an IDP provided by a platform engineering team.

Test pipeline observability

From millions of test events to a single source of truth.

Provide observability into test results automatically across different tools and ensure all test results are available in a single source of truth.

Gain insights through test observability

- Ingest test events and metadata into a single unified observability platform.
- Visualize test results and KPIs in one central view or dashboard.
- Identify long-lasting tests to easily optimize for faster test feedback.

Continuous testing validation

120x reduction in evaluation times — from days to minutes.

Testing is often highly automated already, but teams still frequently test in siloes.

To automatically and proactively ensure a positive customer experience:

- Combine, continuously evaluate, and baseline test results,
- Incorporate SLOs, security findings, and synthetic tests,
- Reduce the number of manual operations by engineers to verify the quality of a new release, and
- Integrate CI/CD delivery pipeline with tools like Dynatrace Synthetic Monitoring to ensure user journeys work as expected.

Observability-driven development

Decrease the mean time to observability from hours to seconds.

In the previous chapter, we outlined the importance of covering ownership and release information early in the lifecycle by extending the provided metadata to include observability and security rules, SLOs, and even automation and remediation steps.

Standardizing these aspects based on technology, team, or criticality of service enables the use of simple "as-code" templates as Golden Paths, ensuring consistency and governance across the lifecycle.

Standardizing across teams can:

- Bring **observability** and **transparency** to development teams out of the box
- Provide **real-time insights** into what's happening
- Conduct **reliability and security checks** at an early stage to enable faster reaction to potential issues before they reach production
- **Increase accountabilit**y of application teams and empower them to get early insights into their implementations
- **Provide autonomy** while maintaining governance for the same criticality of service

Release

This category contains all activities around release and deployment along the Software Development Lifecycle (SDLC).

Release validation

Reduce change failure rate and production deployment lead time by up to 99%.

Automatically validate new releases — taking downstream and upstream dependencies into account — based on baselining, observability data, SLOs, and security information. Use the results to drive meaningful follow-up actions. Provide release validation templates as part of every delivery process for fast feedback on the negative side effects of a new version. Faster feedback leads to a better developer experience, a higher release quality, and a higher innovation pace.

Progressive delivery

Reduce problem exposure by up to 99% with progressive delivery.

Releasing updates and new versions of an application or service in a gradual and controlled manner helps teams provide fast, secure, and high-quality services without disrupting business.

Based on the observability data, including SLO trends and synthetic checks, an automated progressive delivery release like canary or blue/green can be established. These include:

- Automated workflow execution to control the software delivery,
- Direct Kubernetes interactions to apply configuration changes,
- Governance for the delivery process by leveraging events, and
- Targeted notifications to inform the team in charge of the delivery process.

Pipeline observability

100% Real-time standardized coverage of software development lifecycle health.

People often say, “You can’t manage what you can’t measure.” The statement holds true for the software development lifecycle. To calculate critical metrics, such as lead time for change, it is important to take note of the pipeline telemetry data.

Using pipeline observability, the data (e.g. failed builds, lead time) can be calculated and acted upon automatically. With automated data, these metrics — like the DORA metrics “lead time for change,” “deployment frequency,” “change failure rate,” and “mean time to resolve” — are readily available in several dimensions. These dimensions include application/service, platform service/pipeline, and technology ownership.

In addition to metrics, logs and traces of pipeline runs can help debug erroneous pipelines or identify time-consuming hotspots for updating.

Operate

This category contains activities around monitoring and operating services and applications in production environments.

Cloud cost optimization

Reduce cross-zone network traffic by up to 55%.

By combining network data with infrastructure and billing information, teams can identify and analyze top cost contributors. With comprehensive analytics, cost optimizations can be achieved across environments.

Generative AI observability

Increase reliability while keeping costs from surging.

With the rise of generative AI in DevOps Portals and IDP tools, it is becoming increasingly important to track usage and consumption. Tracking AI use allows streamlined access and effective cost management.

Observability data further allows the proper sizing and configuration of generative AI tools by increasing experience and productivity for development teams.

Observability for infrastructure

Reduce MTTR by 30% by breaking down data silos.

For IT operations teams, the central challenge in ensuring the health of the IT infrastructure lies in the daunting task of identifying the root cause of issues.

By applying Golden Paths and as-code templates, users can quickly trace performance issues to their source by "following the red" to the problematic infrastructure entity. This approach facilitates an immediate understanding of how entities such as hosts, processes, and their associated relationships contribute to the identified issue.

Kubernetes monitoring

Manage your platform with 30+ out-of-the-box health alerts.

Establishing scalable cluster lifecycle management within a diverse multi-cluster environment with various distributions requires a centralized repository with a single source of truth.

This centralized view:

- Serves as the hub for ingesting, visualizing, and analyzing telemetry data from different layers of the Kubernetes stack.
- Enables the execution of configuration actions based on observability insights, such as resource consumption and performance management across all clusters.
- Offers extended centralized monitoring and alerting capabilities, particularly for node failure incidents.

Predict

This category contains information about Predictive Kubernetes operations, focusing on the substantial reduction of overprovisioning and storage needs. Predictive AI and forecasting can address many incidents that used to result in wake-up calls for people on standby.

Predictive Kubernetes operations

Reduce overprovisioning and storage needs by up to 75%.

Resizing disks happens frequently within cloud-native environments. Predictive AI enables automatic and timely disk resizing, avoiding system outages while keeping costs low.

Forecasting in data analytics

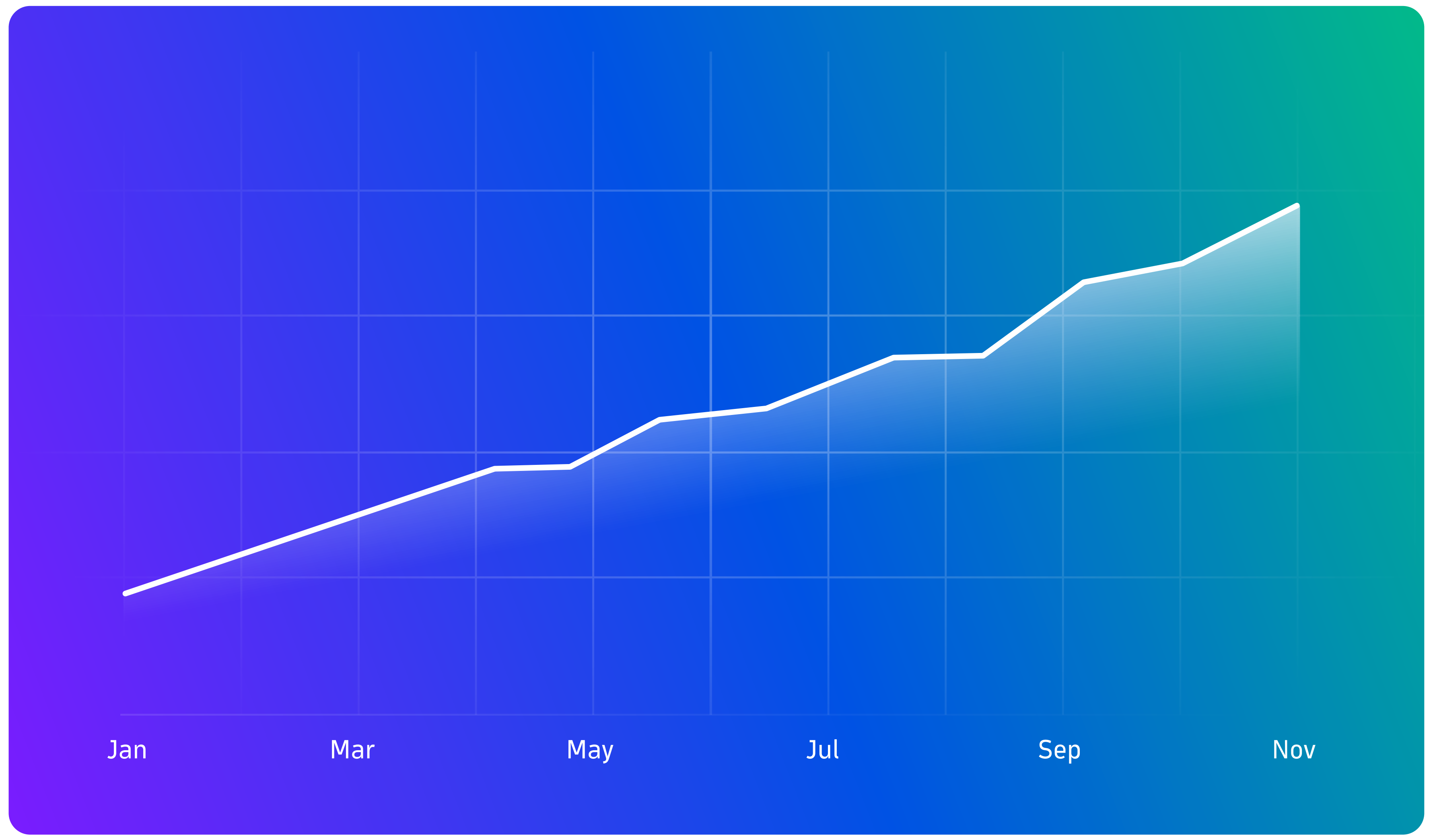
Zero disk-size-related, after-hour alerts for SREs.

Anticipatory management of cloud resources within highly dynamic IT systems is a critical success factor for modern companies. Operators must closely observe business-critical resources such as storage, CPU, and memory to avoid resource-driven outages.

Forecasting in workflow automation

100% reduction of recurring manual tasks.

Predictive AI in workflow automation makes it possible to raise tickets and take action before problems arise.



Prevent

This category contains information on preventing end-user-facing bugs and accelerating risk remediation through unified exposure protection. Users can also maintain continuous security awareness through instant reports and prioritized views of vulnerabilities across different environments.

Preventing customer-facing bugs

Reduce end-user-facing bugs by up to 36%.

Ingesting and automatically analyzing application log data from production makes it easy to issue tickets with context. Log data also empowers organizations to create and assign work to the respective development teams so that they can act and resolve bugs before end-users encounter them.

Unified exposure protection

Cut risk remediation from days to hours — up to 95% faster.

Observability and security are converging to facilitate prioritization and risk assessment of security findings — resolving the most common frustration with the existing tools, specifically during the software development stages. Combining this with automation leads to effective engagements with the teams who need to act on those security findings.

Continuous security posture awareness

Instant live security reports.

Get a unified and prioritized view of different exposures across your application tiers and production and pre-production environments.

- Prioritize vulnerabilities with a custom risk specific to your organization.
- Learn which remediation activities to prioritize for impact, then let automation handle them.
- Learn the details of a vulnerability — which entities are impacted, whether there are links to databases, etc.

Resolve

This category provides strategies for addressing incidents efficiently. Once an incident occurs, it is important to take the right steps immediately. Sometimes, the first step is to inform the right team with the required information. Other times, issues can be remediated and resolved automatically.

Instant intrusion response

Intrusion response in minutes — not days.

By scanning ingested logs for patterns that indicate attacks or intrusions — such as instant schema-less security queries — it is possible to inform the relevant development and security team automatically and immediately. Provide context-rich information, like the attacking IP address, which can then be automatically shut out of the system.

Incident triaging and remediation

Improve MTTR by up to 99%.

Using observability and security data combined with context and further metadata — such as ownership information — it becomes easy to automatically triage issues as they arise and follow through with the proper response.

The proper response could be any automated action available, including:

- Informing stakeholders through communication tools,
- Opening incident tickets,
- Triggering configuration management tools, and
- Remediating directly by executing Kubernetes jobs or creating pull requests.

Error budget alerting

Increase triaging efficiency by 90%.

If you have 99 problems, how do you prioritize?

An easy way to focus on the right problems is to look at the ones actively impacting Service-Level Objectives (SLOs). If the number of issues in an environment still calls for further prioritization, automatically triage and inform stakeholders easily by alerting them on the SLO error budget burn rate (how fast the SLO error budget is consumed).

Protect

This category addresses strategies to handle many security events, eliminate false alarms, and respond effectively to threats. All environments, infrastructures, and applications must be protected — with the right measures. Organizations can protect their assets more easily with a consistent Internal Developer Platform (IDP) that provides governance and security.

Security Operations Center (SOC) noise reduction

Reduce security event storms by 99%.

Hyperscaler tools emit a myriad of security events every day, but you can avoid fatigue with deduplication and consolidation. Teams can also create automatic ticket generation and assignment based on development and security ownership.

Automate security findings at scale

Eliminate false positives.

- Understand the threat posed by your code and third-party or open-source libraries.
- Block malicious requests attempting to exploit code weaknesses.
- Precisely detect and block attacks while applications keep serving their users.

It’s all integrated with OneAgent and available at the flip of a switch.

Contextualized threat detection and incident response

From threat hypothesis to tangible evidence in minutes.

Today, everyone in your organization plays a role in security, and all your data is relevant. That's why it needs to be easy to ingest any data and have it available for manual, detailed security analytics.

Given the pace of cybersecurity attacks today, there is no way around automation. When your valuable security analysts get in front of an incident or investigate a malicious pattern, you need to make sure it's worth their time.

With data contextualization, workflow actions, and dedicated security analytics apps, you have all the tools you need to be proactive about security. These tools reveal bad actors in your system, what they are up to, and how to prevent them from being successful. With the merger of observability and security, it's easy to understand how to elevate your analysts by leveling up the security incident first, fully contextualizing the incident, and prioritizing based on importance.

Stop attackers in their tracks.

Leverage observability and security data in context to see every detail of your environment.

Take advantage of the data consolidated at your fingertips.

Support your analysts whenever they need to look at an incident with automated actions easily combined into a workflow.

Improve

This category provides guidance on the continuous improvement of resources that drive business. These improvements can range from cost- to time-savings to ensuring best-in-class customer experiences at all times.

Always-on app profiling

5x faster performance bottleneck issue resolution.

With observability data around CPU, thread, and memory usage, it is easy to quickly identify the biggest performance bottlenecks and flag them for improvement. Such bottlenecks could manifest themselves as inefficient string operations, leading to high memory allocation and pressure on the garbage collector, which can in turn lead to scalability and performance issues.

Kubernetes utilization improvement

Automated K8s utilization improvement reduces spending by up to 25%.

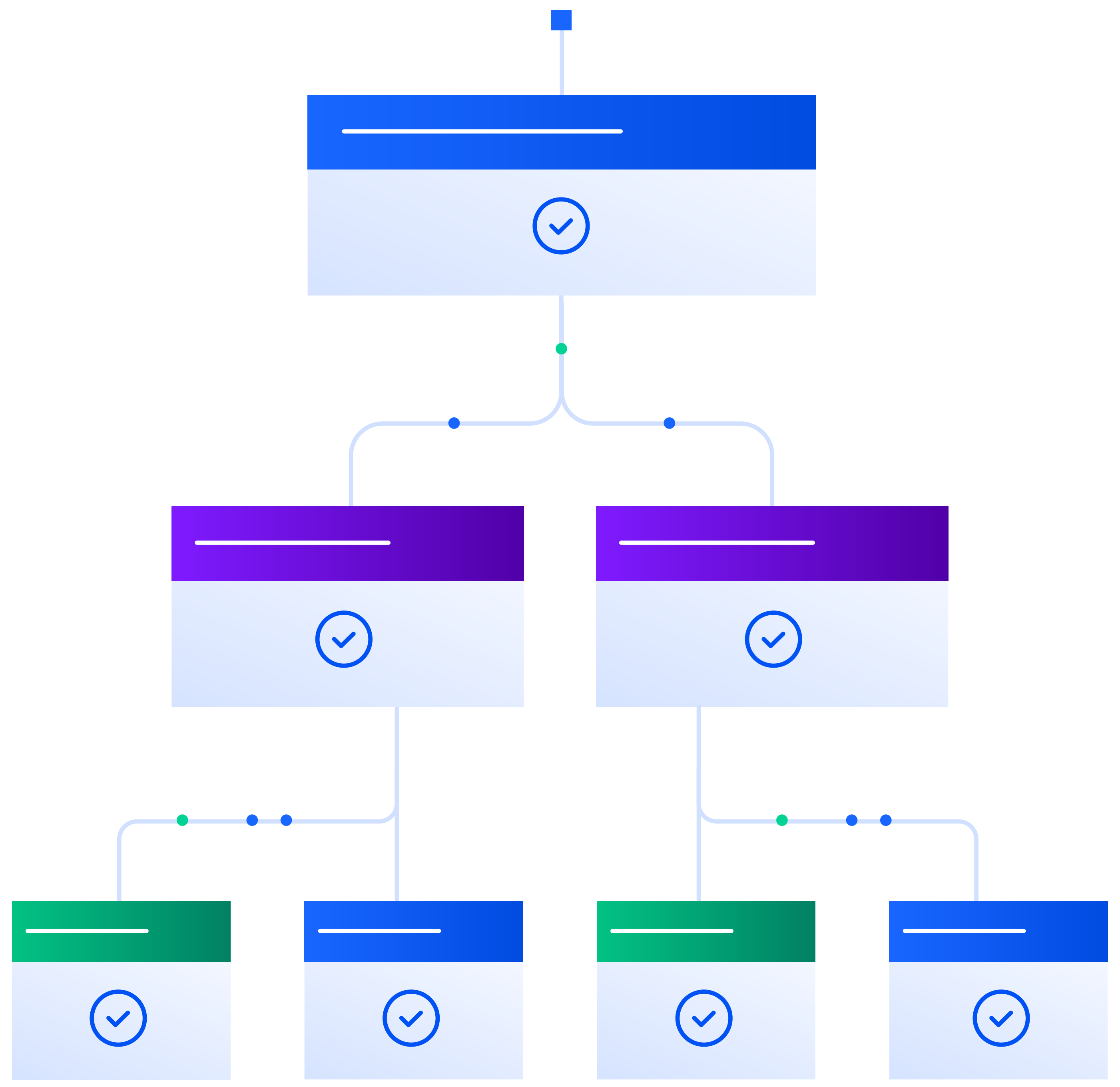
Continuously update and improve K8s requests and limits automatically, reducing cognitive load on developers and providing peace of mind for platform engineers.

By analyzing Kubernetes metrics and SLOs for resource allocation and utilization, it is possible to set environment configurations to resolve misalignments.

Business alignment improvement

Bring context to your business processes and accelerate collaboration.

It is critical to align platform engineering initiatives with the primary value stream. Aligning initiatives to assets facilitates an easier prioritization of improvement work and enhances communication with the line of business.



Measuring platform success

Governance and consistency at scale

Golden Paths guide developers, saving time and reducing risks. But they're not one-size-fits-all. Flexibility exists for individual customization and exploration.

The result is a platform that scales effortlessly, where standardized efficiency works with developer autonomy to offer governance and consistency.

By applying product management principles and observability to the IDP, it is possible to provide meaningful insights into platform infrastructure, tools, and services as well as Golden Paths adoption and efficiency regarding feedback for developers.

DORA metrics

Google's DevOps Research and Assessment (DORA) team established the DORA metrics in an effort to provide key insights into the performance of a software development team.

4 key DevOps Performance Metrics:

- **Deployment frequency:** Measures how often a team successfully releases to production.
- **Lead time for change:** Measures the time it takes for committed code to get into production.
- **Change failure rate:** Measures the percentage of deployments that result in a failure in production requiring a bug fix or roll-back.
- **Mean time to restore service (MTTR):** Measures how long it takes an organization to recover from a failure in production.

The DORA team extended these metrics by adding a fifth in 2021:

- **Reliability:** Representing availability, latency, performance, and scalability.

Benchmarking in real-time and across dimensions (technologies, teams, etc.) provides crucial insights for a platform engineering team.

Developer experience

Platform engineering is ultimately about driving developer productivity. But how do you define and measure developer productivity and satisfaction? It can't be reduced to a single metric, but the SPACE framework captures important dimensions of developer productivity:

- Satisfaction
- Performance
- Activity
- Communication and collaboration
- Efficiency and flow

The SPACE framework does not provide a ready-to-use list of metrics like the DORA metrics, rather it provides guidance on categories to consider.

More information on the SPACE framework can be found here.

Take action!

Implementing DevOps and platform engineering is now a requirement for organizations that want to deliver value in the cloud. These practices are crucial for boosting productivity and achieving success in today's tech landscape.

Dynatrace's purpose-built solution for platform engineering reduces complexity through automated workflows, including auto-scaling, deployment validation, and anomaly remediation.

By leveraging the power of the Dynatrace platform and the new Kubernetes experience, platform engineers can implement the following best practices. These strategies empower development teams to deliver best-in-class applications and services to their customers.

Centralize and standardize. The ability to effectively manage multi-cluster infrastructure is critical to consistent and scalable service delivery.

Provide self-service platform services with dedicated UI. Self-service platforms enable development teams to improve the developer experience and increase their speed of delivery.

Automation, automation, automation. Adoption of GitOps practices enables platform provisioning at scale.

Context. Get access not only to data, but also to answers.

Learn more at https://dt-url.net/PE-doc

02.21.24 BAE10551_EBK_agency