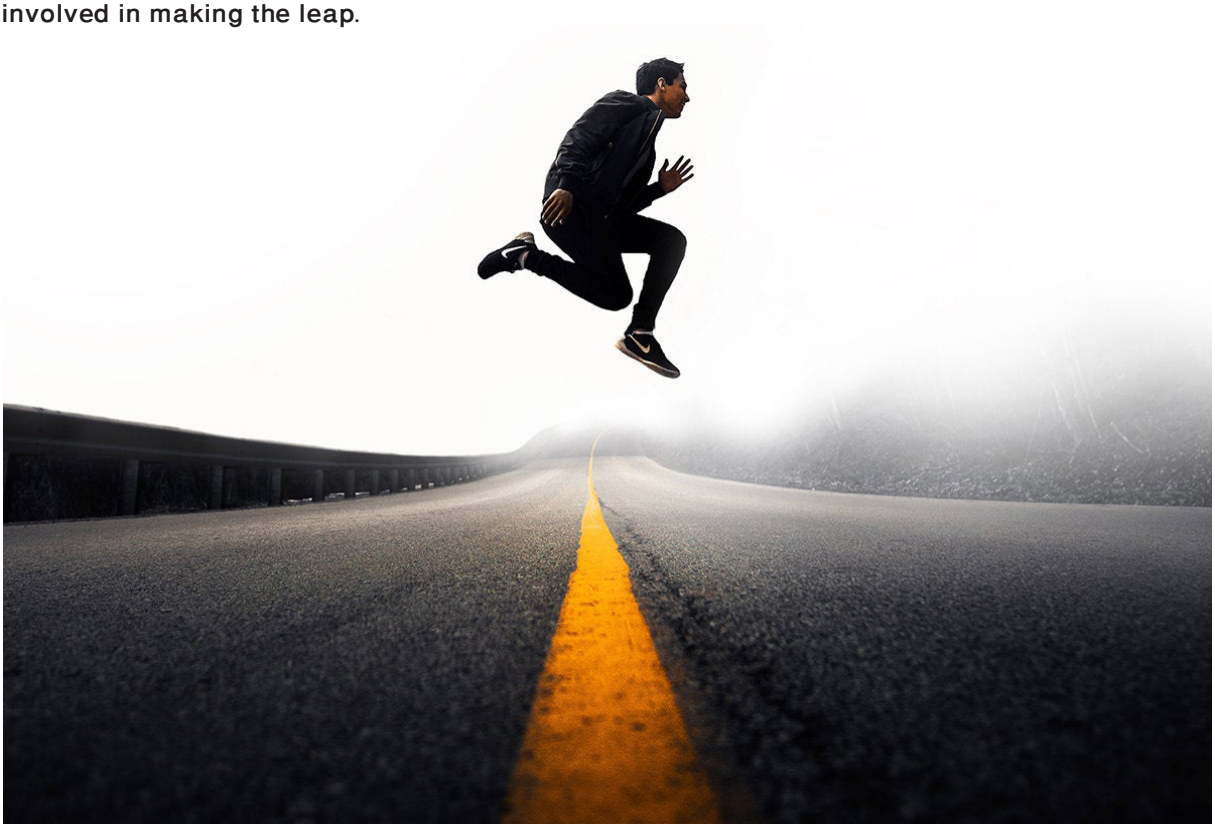
Why Platform Engineering?

The shift from devops to platform engineering could be transformational. Here›s why and what›s involved in making the leap.



It wasn't long ago when engineers configured systems manually. In those days, developers drafted procedural documents for administrators to follow when deploying applications. Devops tools and practices, including deploying with CI/CD, configuring infrastructure as code, and managing containerized systems, all enabled IT teams to improve systems reliability, security, and performance.

The result is that more devops teams could deploy application changes and scale cloud infrastructure faster. Devops teams went from quarterly release cycles to more continuous deployment practices, and cloud engineers developed automations to scale cloud infrastructure based on computing demands.

Using devops practices, agile development teams built and enhanced applications for mission-critical workflows and revenue-generating experiences. Developing microservices and deploying to multicloud architectures provided more flexibility but also increased the complexities

when responding to outages, performance issues, or other major incidents. Many organizations adopted site reliability engineering (SRE) practices and deployed Alops platforms to improve the reliability and performance of their services and applications.

TABLE OF CONTENTS

- IT struggles with broad devops adoption
- Platform engineering evolves devops
- Improving developer experience and productivity
- Creating reusable, configurable, self-service components
- Benefits of platform engineering

IT struggles with broad devops adoption

Maturing devops and SRE capabilities requires significant investment in developing skills, practices, and culture change. For CIOs and IT leaders, two fundamental problems tend to emerge.

0 seconds of 30 secondsVolume 0%

First, many organizations struggle with technical debt and skills gaps, so while their devops and SRE practices are advanced, broad adoption is more challenging. These organizations might successfully build CI/CD and IaC for their cloud-native and modernized applications, but they struggle to leverage these capabilities into standardized practices.

For more technically advanced organizations, the issue is different. These organizations are more likely to adopt self-organizing practices and empower teams to configure devops tools specific to their application architecture requirements and implementation values. They may have standardized platforms but each team uses the tools differently. Thus, they deploy customized CI/CD pipelines, IaC automations, cloud architectures, and monitoring configurations.

For these organizations and their leaders, the question is how to enable devops best practices into repurposable patterns. More specifically, how to empower agile teams to spend more time developing applications and less time on cloud configurations and automations.

Nominations are open for the 2024 Best Places to Work in IT

Platform engineering evolves devops

Platform engineering is an evolution of devops practices intended to help larger organizations develop standards, support reusable configurations, and deliver systems engineering as an internal product capability.

"Platform engineering is a step forward in devops," says Marko Anastasov, co-founder of Semaphore CI/CD. "Platform engineering enables developers to follow devops practices more easily by creating a "golden path" developers can use for rapid application development."

For large organizations, platform engineering may require a separation of duties from the developers who build applications to the platform engineers who create devops-as-a-product. "A

platform engineer focuses on creating the means for developers to self-service the tools, libraries, and infrastructure they need to write applications," says Anastasov.

Improving developer experience and productivity

While simple in concept, platform engineering isn't trivial to execute because it requires a product development mindset. Platform engineers must develop a product that agile development teams want to consume, and developers must let go of their desires for DIY (do it yourself) devops approaches.

One place to start is infrastructure and cloud provisioning, where IT can benefit significantly from standards, and developers are less likely to have application-specific architectural requirements.

SponsoredPost Sponsored by Coventry University

Coventry University is a truly global university with more than 13,000 international students from over 150 countries. And with campuses in Coventry and London, you can choose the city location...

Donnie Berkholz, senior vice president of product management at Percona, says, "Platform engineering covers how teams can deliver the right kind of developer experience using automation and self-service, so developers can get to writing code and implementing applications rather than having to wait for infrastructure to be set up based on a ticket request."

Therein lies the customer pain point. If I am a developer or data scientist who wants to code, the last thing I want to do is open a ticket for computing resources. But IT and security leaders also want to avoid having developers customize the infrastructure's configuration, which can be costly and create security vulnerabilities.

"Companies will adopt platform engineering more because they care about their internal developer experience. Anything that gets in the way of developers is literally costing money when those employees are less productive," continues Berkholz.

Creating reusable, configurable, self-service components

One way to consider platform engineering is to ask developers to fill in the following statement: "My team can't invest enough time to address <technical concerns> because we're spending time developing, maintaining, or improving <technical automations> and <infrastructure configurations>."

Technical concerns are often expressed as non-functional requirements, such as improving testability, performance, scalability, and security while reducing technical debt. These all improve end-user experience with the application, and many devops teams would like to devote more time to these areas.

Contrast that with technical automations and infrastructure configurations, in which building basic capabilities can be a prerequisite to software development, while ongoing investment improves the development team's productivity.

Unfortunately, the more time development teams devote to these areas, the less time they can spend on delivering functionality and improving the non-functional technical concerns.

In scenarios where teams invest significant time in these three areas, and where there are common requirements across multiple teams, platform engineering emerges as a solution that can yield benefits.

"Just as devops frameworks reinvented scalability, availability, and operability, platform engineering presents a conveyor belt of swappable tools for an assembly line of dev teams," says Marcus Merrell, vice president of technology strategy at Saucelabs. "This allows them to circumvent traditional bottlenecks such as testing, execute efficiently across projects, and deploy the necessary tools to meet diverse needs in real-time and get to market faster."

Benefits of platform engineering

Improving quality and delivering capabilities faster are common objectives of platform engineering, so how does this approach address them?

"Platform engineering is the practice of creating shared, internal services that solve problems for engineers in one place," says Chris Cooney, developer advocate at Coralogix. "Platform engineering is excellent at solving some of the key problems that occur at scale."

In other words, large IT departments with many development teams stand to benefit from platform engineering practices. Cooney identifies these problems platform engineering targets:

• Grow consistency between teams and reduce the single-solution mentality

• Discover and re-use shared components rather than rebuilding and customizing

• Build compliance into the platform

The path to adoption

All of this sounds promising, but skeptics will point out that this isn't the first time large IT departments have attempted to productize internal technology solutions and platforms. Before diving into platform engineering, organizational leaders and their teams may want to answer several questions

• Where can platform engineering deliver value across multiple teams through efficiencies or compliance improvements?

• How should we organize the platform engineering development work without creating new bottlenecks?

• What's the carrot and the stick to motivate more development teams to leverage the capabilities delivered by platform engineering?

• Does platform engineering shift focus so that development teams spend more time on delivering functionality and improvements in non-functional capabilities?

While platform engineering is promising, IT organizations should start small with simple ambitions. Identify areas with clear benefits, few technical barriers, and common requirements, and start there.