THENEWSTACK

Platform Engineering

What You Need to Know Now

The New Stack

Platform Engineering: What You Need to Know Now

Alex Williams, Founder and Publisher

Ebook Team:

Andrew Tillery, Digital Marketing Manager

Ben Kubany, Project Manager

Celeste Malia, Marketing Consultant

Diana Gonçalves Osterfeld, Designer

Heather Joslyn, Ebook Editor

Jennifer Riggins, Author

Judy Williams, Copy Editor

Supporting Team:

Benjamin Ball, Director of Sales and Account Management

Joab Jackson, Editor-in-Chief

Michelle Maher, Assistant Editor

Vinay Shastry, Director of DevOps

20230822.1

Table of Contents

Sponsor

We are grateful for the support of our ebook sponsor:

Any app, every cloud, one modular platform

Innovate on your own terms. VMware Tanzu empowers developers to focus on building great apps by providing faster, more secure paths to production and automating platform operations at scale.

Introduction

It's been 15 years since the DevOps movement arose, promising to help teams become faster at responding to customer needs. In theory, DevOps helps software teams "move fast and break things" and do more with less.

In reality, most organizations are stuck at the middle of their transformations and the tool choice has grown overwhelming. Meanwhile, developer productivity hasn't grown significantly, according to 2023 data from the CD Foundation. This leaves both individuals and organizations paying the price.

Developers are burning out at astronomical rates. Cybersecurity attacks are growing exponentially. Too much autonomy puts your team and organizational sustainability at risk. In response to overly complex systems and processes, teams engineer their way around problems. This isn't scalable and often distracts them from delivering business value.

What's the answer? For an increasing number of organizations, it's platform engineering.

Platform engineering is a sociotechnical set of practices and tools that can make the lives of both devs and ops easier. By using a curated internal developer platform, portal or workflow to standardize how developers build, deliver and deploy their code, platform engineering allows them to provision their own infrastructure, with complexity abstracted away.

Platform engineering emphasizes discoverability, extensibility and self-service, aiming to provide developers with everything they need to get their code safely to end users — and to roll back quickly when needed.

From observability, monitoring and troubleshooting to compliance and security, a developer platform looks to automate any hurdles that keep developers from delivering value to end customers faster.

A platform team — sometimes called a developer enablement team — takes a "Platform as a Product" approach. It views the organization's developers as its "customers," and concentrates specifically on improving the cross-company, internal developer experience, always looking for ways to reduce friction, frustration and repetitive work.

In short: In platform engineering, developers don't need a deep knowledge of Kubernetes; platform engineers create easily replicable solutions and free themselves from unnecessary toil.

And it's being embraced widely. By 2026, 80% of software engineering organizations will have created a platform team, according to Gartner forecasts.

A big incentive for adopting platform engineering is the sheer breadth and complexity of the cloud native ecosystem. Rather than letting devs sift through and choose their own "best of breed" tooling from an ever-expanding universe, this newer approach guides developers to follow "golden paths" toward release. Platform engineers must work to entice developers to use the platform by building and then internally marketing what devs actually want.

Platform engineering is also a way to align cross-functional stakeholders, connecting developers to business value and giving C-levels insight into the significant engineering cost center. When properly deployed, a platform strategy also results in more secure and reliable software. And platform engineering notably increases developer productivity while decreasing cognitive load.

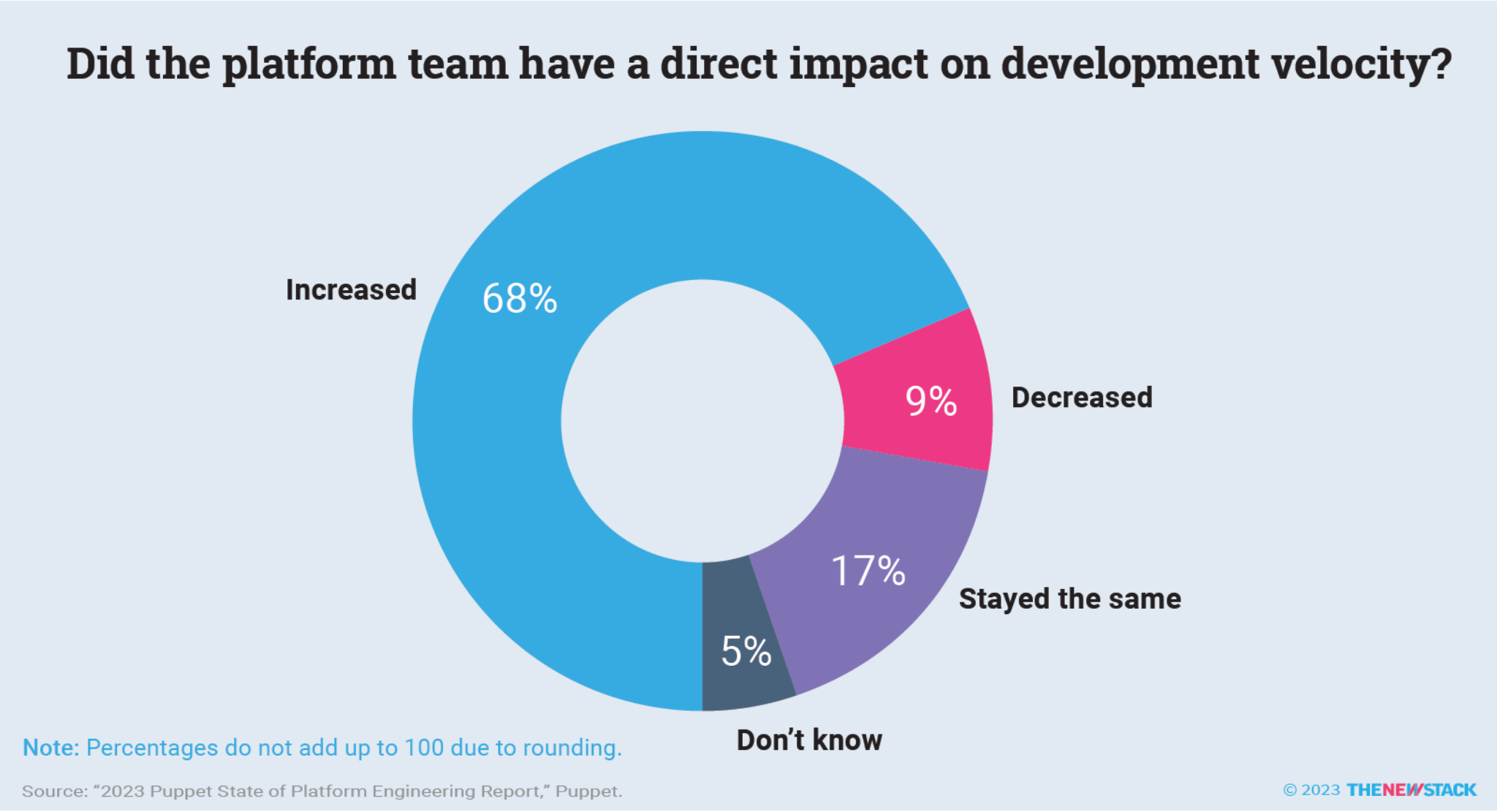


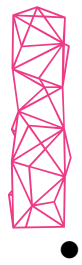
Figure 0.1 *Forty-two percent of survey participants who saw an increase in developer speed with platform engineering said velocity grew "a great deal."*

In fact, 93% of enterprises surveyed for the "2023 Puppet State of Platform Engineering Report" felt that adopting a platform engineering mindset would be a step in the right direction for their whole organization, despite only about half having taken that step.

As more and more organizations adopt some form of platform engineering, you can learn from their best practices around removing hurdles and enabling developers to reach their creative, problem-solving potential. And by improving this developer experience, you ultimately improve your end-user experience.

CHAPTER 01

Why Platform Engineering Is On the Rise



Isn't platform engineering just DevOps rebranded? Fair question. After all, at its core, the goals of platform engineering seem to echo the three ways of DevOps:

- Optimize flow to deliver end-to-end business value.
- Shorten feedback loops.
- Continuously experiment, learn and improve.

A good platform team will reflect this DevOps mindset when dealing with its own organization's developers — thinking of them as customers — and look to remove obstacles that dev teams face in achieving these three ways. Both practices' success hinges on automation, cross-functional collaboration and empathy to help teams scale complex, distributed systems.

Indeed, much of the technical side of platform engineering follows that of DevOps, with an emphasis on standardization around limited choices, automation of infrastructure delivery, and being as self-service as possible. No more formal provisioning processes for more CPU or another container, only to wait hours, if not days, for manual approval.

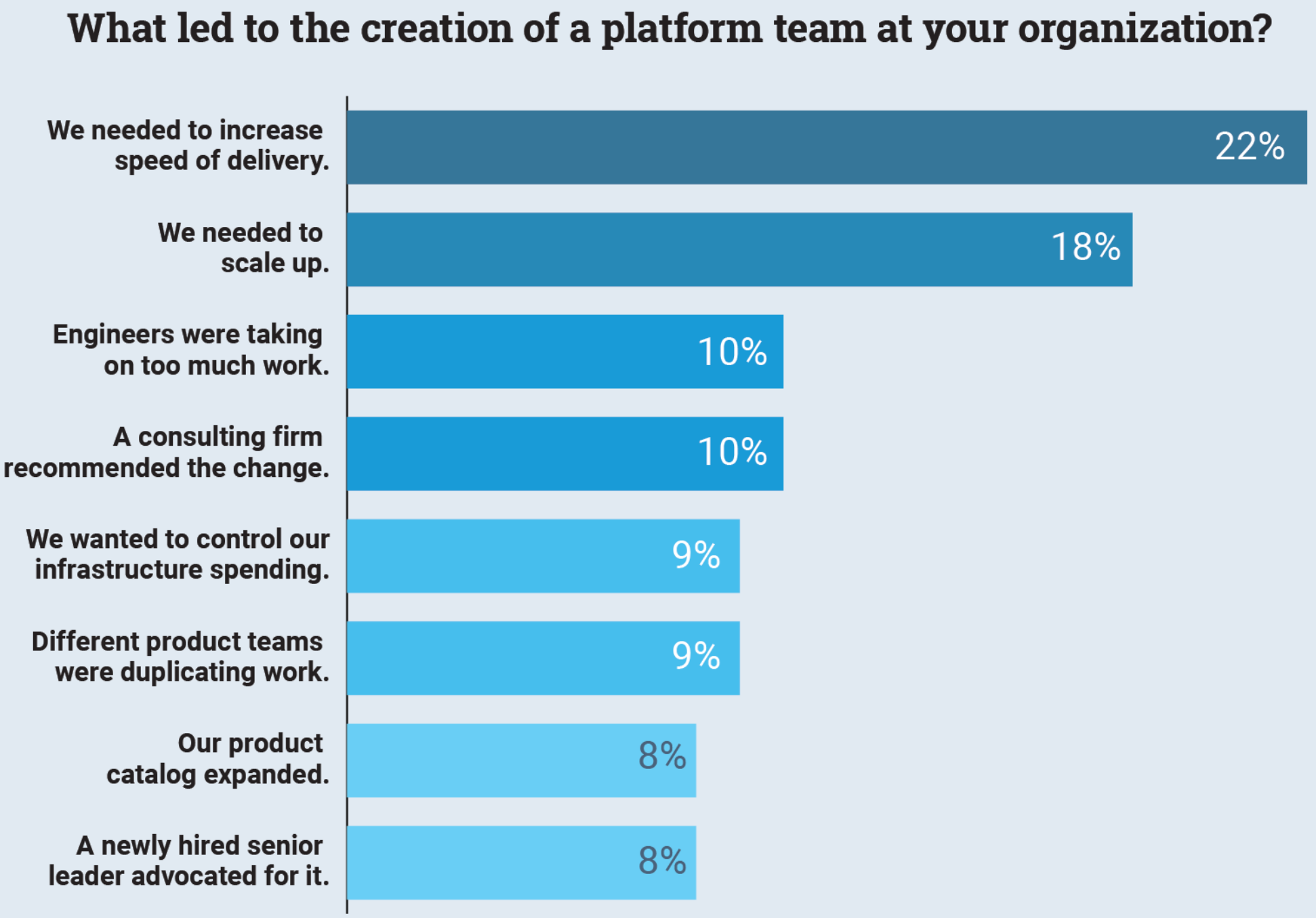
It's just that, while DevOps looks to reduce friction and increase velocity across operations, platform engineering looks to enhance the developer experience.

This all makes DevOps absolutely part of the solution — it's just part of the problem, too.

DevOps and continuous delivery have resulted in longer pipelines and toolchains all the way to the cloud, which gives development teams responsibility for up to seven layers of complexity and a need to understand each. Add to this the "shift left" trend, which assigns developers even more responsibility for securing their applications. The resulting cognitive load is hefty.

"The movement to 'shift left' has forced developers to have an end-to-end understanding of an ever-increasing amount of complex tools and workflows," said Luca Galante, creator of the Platform Engineering community. "Oftentimes, these tools are infrastructure-centric, meaning that developers have to be concerned with the platform and tooling their workloads run on."

Figure 1.1 *Developer speed and scaling needs are the most common reasons why organizations create platform teams..*



Source: "2023 Puppet State of Platform Engineering Report," Puppet.

What Slows Developers Down

Platform engineering makes a new argument: Infrastructure is simply not something developers should have to worry about. Not every developer wants to have to learn how to wrangle Kubernetes or the cloud.

Developers must constantly switch context as they interact with an increasingly cloud native tech stack. Application teams become distracted and stressed trying to perform DevOps, taking significant time away from delivering business value to the end user. And then operations teams' flow is interrupted by never-ending tasks that usually can and should be automated.

DevOps, despite its name, was intended to focus mainly on operational experience. But in reality, there's an awful lot of dev in DevOps. It has been implemented in a way that, while increasing the speed and reliability of ops, has distracted many devs from their core objectives, with a staggering 84% of developers saying they're involved in DevOps activities, according to a 2023 CD Foundation survey.

As a result, the implementation of DevOps has fallen short of its promise: cross-functional, streamlined teams that together deliver value faster. And silos persist.

"If you speak to any engineering team, and you ask them: What's the number one thing that slows you down? Most teams will say everything works well until we need to engage with another team," Andrew Boyagi, Atlassian's senior technical evangelist for Agile and DevOps, told The New Stack.

"They need to find out who the other team is. And they have to find out how to engage with them," he said. "And usually the beginning parts of that 'collaboration' are really low value because it's like: What is this microservice? Is it healthy? Where's the runbook? Where's the architecture diagram?"

Another common barrier to platform adoption is a lack of adequate documentation, which we know dramatically supports developer onboarding and self-service, yet is often an afterthought in internal platform strategies. (See more about this in Chapter 2.)

Even 15 years into the DevOps revolution, 80% of organizations feel they are only about halfway through their DevOps journey, according to Puppet's data.

Part of that is because DevOps is also a sociotechnical process, and many organizations have focused only on adopting automation practices and Infrastructure as Code. The developer experience is wholly missing from most so-called DevOps transformations.

Platform engineering, on the other hand, looks to align Dev with Ops, along with security, compliance, and senior IT and business stakeholders. While focusing on the developer experience, it also helps reduce operations' toil and helps create quicker, more secure paths to production.

The platform engineering team can help deliver on the promise of DevOps at an enterprise scale. And by starting with the developer and working down to the infrastructure, the team can create a cohesive, self-service developer experience that reduces friction and increases speed to value, which accelerates DevOps-driven flow, feedback and experimentation.

> "If you speak to any engineering team, and you ask them: What's the number one thing that slows you down? Most teams will say everything works well until we need to engage with another team."—Andrew Boyagi, Atlassian

When done right, platform engineering reduces the cognitive load and subsequent burnout more than a third of developers currently have. And as we've long known, happier workers are more productive workers who can creatively solve customer problems faster.

In fact, a 2022 Executive Pulse survey by VMware found that half of the enterprise technology executives polled said that an improved developer experience would have the greatest potential to increase revenue for their organization.

A platform engineering mindset and self-service strategy, which focuses on the developer first, can help to finally unlock DevOps maturity. And, by improving the developer experience, you are more likely to recruit and retain exceptional talent.

VMware Tanzu®

SUMMARY

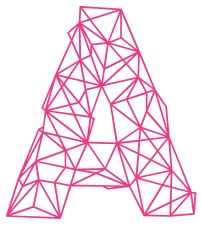
VMware Tanzu is a modular, cloud native application platform that accelerates development, delivery and operations across multiple clouds.

KEY FEATURES

VMware Tanzu Application Platform is a single, end to end integrated platform solution that enables companies to build and deploy more software, more quickly and securely, through a rich set of developer tooling and a pre-paved path to production.

CHAPTER 02

Getting Started: Organizational Culture



"Platform as a Product mindset," as famously coined by Manuel Pais and Matthew Skelton, co-authors of the 2019 book "Team Topologies," means treating your internal developer platform project like a product and your internal developers as your customers.

This can mean a big cultural change in your organization. But such a shift is necessary for any platform engineering initiative to be successful.

"I've found people get fixated on the new innovative technologies but they lose track of the people they are building it for," Nicki Watt, CEO and CTO of technology consulting company OpenCredo, said at the State of Open Con.

Jürgen Sußner, lead cloud platform engineer at DATEV, an IT service provider for German tax consultants, which employs more than 1,700 developers, echoed that notion.

"The technology alone isn't sufficient, you have to care about the people," he told The New Stack. "My role is about getting out of developers' way, to empower them to deliver software at their speed without any obstacles along the way."

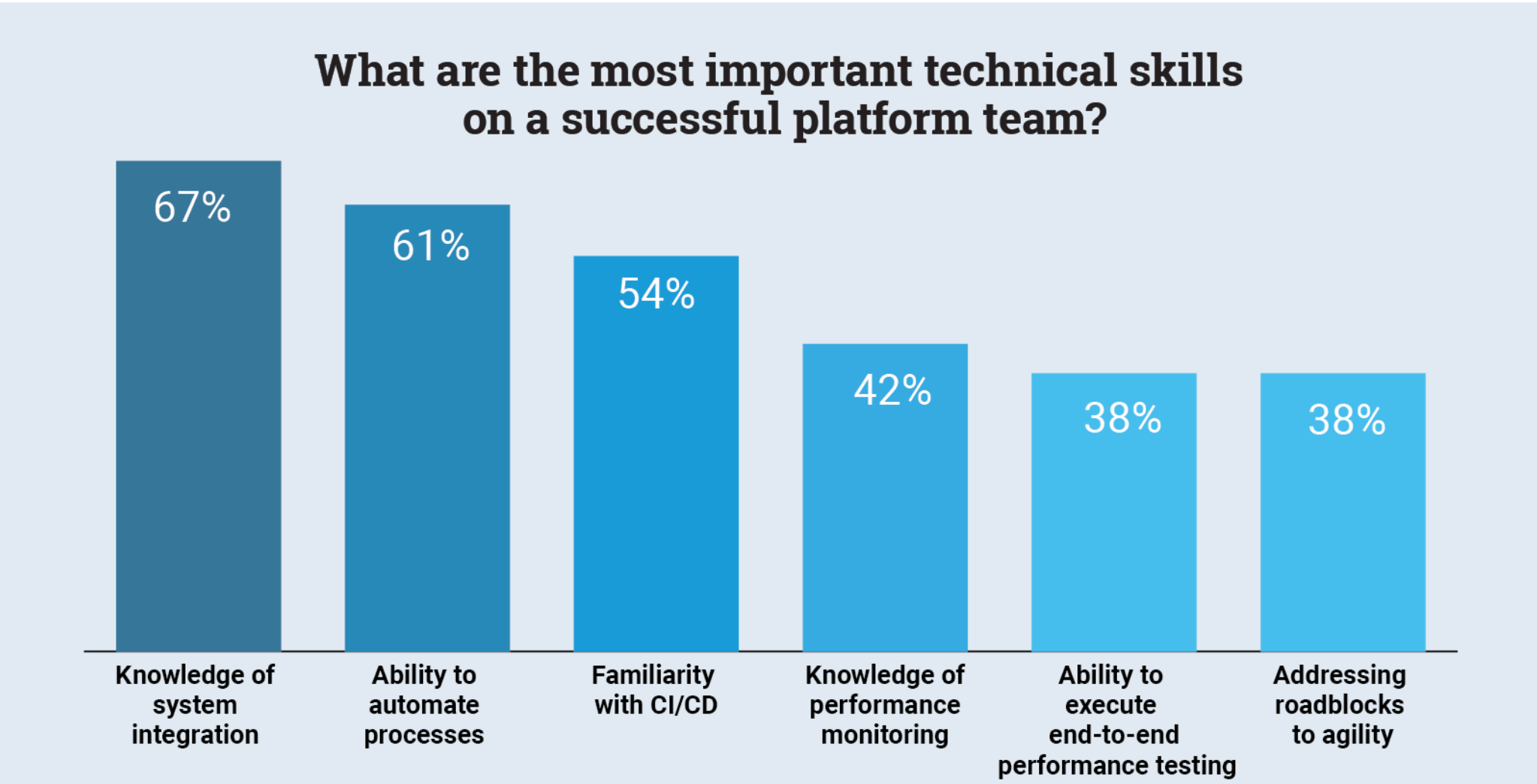
In order to modernize its data center operations and improve developers' ability to build and deliver software more quickly and autonomously, DATEV deployed VMware Cloud Foundation on VxRail to create a scalable, automated, hybrid cloud platform, enabled by VMware Tanzu.

Many platform engineering experts recommend that your internal platform project should have:

- A product owner.
- User research.
- Customer personas.
- A product roadmap and backlog.
- Tight feedback loops.
- Marketing or evangelism.
- Service-level objectives (SLOs).

By treating your platform as a product, you are consistently prioritizing your internal consumers (the developers), while aligning everything with business goals.

Figure 2.1 *A platform team especially needs members who understand system integration, automation and continuous integration/ continuous development (CI/CD).*



Gathering Feedback, Gaining Empathy

Platform engineers are, after all, engineers, and can fall into the bad habit of thinking they know best. But this urge must be resisted. Each organization needs to figure out how to engage their internal customers (aka developers) in a tight feedback loop.

Some platform teams embed their own engineers on application teams or do pair programming to gain empathy firsthand for the developer experience. This also helps you map out the user journey — and identify bottlenecks impeding it — early on.

When the platform team of identity security company CyberArk steered the organization's move from on-premises to the cloud, a subset of platform engineers were tasked with building an app on top of the platform, playing its first guinea pigs. Dogfooding like this is a great way to build empathy for your customers.

CyberArk has also adopted a culture of inner sourcing. If there is a new service to be built, the requesting app team has the option to loan a developer or two out to the platform team for a few months to help build the feature. Then, the platform team takes care of rolling it out across the company and maintaining it. This has the added possibility of attracting a platform champion who can encourage adoption when that champion returns to application development.

But remember: Developers aren't the only stakeholders to consider in platform creation. Increasingly, data scientists and machine learning engineers need to be able to leverage the platform as the pathway to the cloud, and should be among your beta testers, as their needs and baseline knowledge differs.

Always consult with security, privacy and governance stakeholders early on, as the evolving platform should support their goals. And, in order to ensure this internal product keeps receiving funding and that your developers understand how they are driving business value, make sure your roadmap is tied to overall business objectives.

> “Half of the enterprise technology executives polled in a 2022 Executive Pulse survey by VMware indicated that an improved developer experience would have the greatest potential to increase revenue for their organization.

A platform is not a one and done project — teams are changing and adapting all the time, Watt pointed out, and governance and processes cannot be one size fits all. Requirements and pathways to speed will constantly evolve.

“If you want to build a really great platform engineering developer experience, look at it as a holistic sociotechnical challenge,” she said.

Adopting a Platform as a Product mindset is a good first step.

Communication Provides Context

A Platform as a Product is never just thrown over the wall. Any changes have to be well communicated and tailored to different stakeholder audiences. Early on, develop a formal strategy for communicating your ever-evolving platform strategy, including a plan to internally market your platform.

Don’t worry, you’ll never run out of ideas. As soon as you announce you’re aiming to

build a better platform experience, you will be inundated with feature requests.

Your internal customers will want to see the product roadmap — not just the what but the how and the why behind any updates. Provide them with different pathways to offer feedback, which in turn helps you reprioritize your backlog.

As you build — and share — your platform product roadmap, be sure to factor in maintenance. Continued budgeting and a long-term vision about how it benefits the organization is what distinguishes just another platform from a sustainable platform strategy.

> “My role is about getting out of developers’ way, to empower them to deliver software at their speed without any obstacles along the way.” — Jürgen Sußner, DATEV

A platform team often starts out as a few teammates who are supporting hundreds or thousands of engineers. It’s important to overcommunicate your priorities and reasoning behind them because you simply won’t be able to meet every demand. And your job is to ease developer burnout, not feed your own.

One of the challenges you may find is that platform teams often only consist of technical members, who aren’t always well-suited for internal marketing.

“Our PlatformOps team is really good at making change. But is the development side of our organization going to be receptive to that change with the same sense of urgency?” asked Jim Kohl, senior application architect at Great American Insurance Group (GAIG).

Especially in regulated industries like insurance, developers don’t always have a choice with the technology they use, but a good platform engineering mindset aims

to persuade instead of push. That's why, Kohl told The New Stack, his enablement team looks to "productize change" by packaging changes with well thought out communications and tooling that creates campaigns to detect which projects are affected, adopting a "Change as a Product" mindset.

"Our operators are fantastic people and really helpful, but somehow something is lost in communicating between operations and development," he said.

Before his team was created to help improve the developer experience, he said, the operations team would go on Slack and post messages like, "These 100 apps have to make this change by tomorrow." Sometimes these requests leave many questions to be answered.

This is why his team's first priority was to improve cross-functional communication, which included creating a center of excellence, a knowledge base for each major change, plus optional but well-attended monthly developer experience (DevEx) forums.

> "The whole golden path is about improving developer experience. It's freeing up headspace and it's productizing change, so as developers and as teams we can focus on delivering quality products as quickly as possible and as maintainable as possible. And, as a byproduct, loving what we do." — Jim Kohl, Great American Insurance Group

Kohl was an early founder of what GAIG refers to as the foundation team, "because a lot of what we do [is] core to everyone," he said. Seven people take care of developer

experience, helping to build cross-functional pipelines and self-service app development tools.

This foundational work supports 350 applications across 20 different business organizations, all housed on the VMware Tanzu developer platform.

While the PlatformOps team keeps everything running smoothly and the security team enforces compliance, the foundation team focuses on minimizing the friction between the two teams, so that development teams can deliver products.

Documentation Drives Self-Service

Want to make your internal developer customers feel valued? Documentation is key to both proving your long-term investment in the platform and teaching your developers to fish.

Keep your documentation simple, skimmable and searchable, with the goal of reducing developer friction with your platform. Docs enable self service and discoverability, so your platform team can focus on solving problems, not opening an internal support line.

Never assume your user's base knowledge. Avoid language like "easy" or "simple," and embrace glossaries and internal get-started guides. As developers are likely interacting with your platform via APIs, don't forget to include examples of each call, parameters and responses, as well as common error codes. This will help you onboard more teams to your platform and, in turn, help teams onboard new developers. Clear and easy-to-use documentation can also help your IT and business departments communicate with each other better.

Docs are also ripe for inner sourcing. Make sure you not only document your platform, but also make it easy for developers to ask questions and help you answer them. After all, they're the core users. This also helps answer "How did you do X with Y?" questions, and provide useful real-world workflows.

No matter how you approach docs, ensure that documentation becomes a habit for your platform team.

Success Stories Equal Greater Adoption

Another valuable communication tool for platform adoption is the success story. Realtor.com, the online home buyers' marketplace, faced a reluctance to adopt platform engineering because developers feared it could add to already existing complexity caused by the company's growth.

To gain devs' support, that platform team focused on nailing early wins, delivering the first version of its Platform as a Product intentionally designed to create positive examples quickly. Then, the team adapted those first success stories to different developer experiences as a way to compel internal users to trust and use the platform.

For the Realtor.com team, a success story won't just emphasize technical feats, but it highlights collaborative and cross-functional decision making, too.

Be sure to collect developers' feedback early and often, so you can build something they want, and find those internal advocates who can lead the rest of your engineering organization to come onboard.

CHAPTER 03

Getting Started: the Technology

Let's start with two things an internal developer platform should not be.

No. 1: An internal developer platform should not be a catchall.

"That's an anti-pattern I've seen very often, where the platform becomes a bucket for everything, and just becomes a huge mess with lack of ownership, lack of focus and a lot of waste, where teams are overloaded and working on a lot of stuff that's not really a priority," Pais, the "Team Topologies" co-author, told The New Stack.

An internal developer platform is the user interface for your internal developers, so it should be focused on what they need to get the work done and nothing more.

And No. 2: A platform isn't just a Kubernetes abstraction — although many organizations' first attempts are just that, an attempt to simply relieve developers of the need to deeply understand Kubernetes.

A platform is a living thing, constantly evolving in response to internal customer needs. That's why it's better to start off small than to adopt an out-of-the-box solution before you've opened up your customer communication channels and proven your work with early wins.

Start Small: the 'Thinnest Viable Platform'

In Agile product management, you quickly launch a minimum viable product (MVP) to your target customers, with smaller release cycles and tighter feedback loops. That way, you know early on if you are building something they actually want to use — and, if not, you can quickly pivot to try something different.

Internally, the team topologies methodology suggests you should kick off your platform journey with the "thinnest viable platform" (TVP). This first move should aim to reduce friction or cognitive load — ideally both.

> "An internal developer platform should not be a catchall, "where the platform becomes a bucket for everything, and just becomes a huge mess with lack of ownership, lack of focus and a lot of waste," said Manuel Pais, co-author of "Team Topologies."

Your TVP may not be a platform at all, but rather the creation of better documentation or a searchable knowledge base. Your TVP is any first step that emphasizes self-service and simplifies the developer release cycle. And it drives those early success stories that attract more people on board.

Then, as you scale, Pais said, you strategically build onto that platform to promote internal cohesion and provide a consistent interface and internal user experience.

Inventory Your Tools

No two platforms are going to be the same, even with the same out-of-the-box solution. After all, they should be designed with your own organization's developers in mind.

Everything you build should be driven by requests that either unblock bottlenecks shared across several product teams, or come from other essential stakeholders, like the security team.

These are the common pillars of a platform strategy:

- Increased standardization across architecture.
- A self-service developer experience.
- Automated CI/CD pipeline and release cycle.
- Increasingly higher-level abstractions.
- Maintaining some sense of developer autonomy or choice.

As you start out on your platform journey, you'll likely find that a disparate set of tools are in use by developers across your organization. You're not alone. In fact, Puppet's current "State of Platform Engineering Report" found that the majority of enterprises surveyed have three to six self-service platforms running in parallel — or often, in conflict.

Your first aim is to understand what your organization already has, reaching out to teams to understand their toolsets and what they use them for. Automate as much of the tool-discovery process as possible, but also open those lines of communication

> “Platform engineering is not for your innovation sandbox and 1% developer influencers. Look to build a platform that helps the most developers achieve more with less effort.

Platform teams focus on reducing toil by taking care of what Abigail Bangser, principal engineer at Syntasso, calls “non-differentiating but not unimportant work” that repeats across teams — like security, compliance and cloud deployment.

Automating this important work allows developers the time and headspace to focus on delivering their differential value to end customers faster. This is also why Paula Kennedy, Syntasso’s COO, told The New Stack she recommends kicking off your platform journey by looking at Jira tickets to spot patterns of repetitive, cross-organizational work.

Areas ripest for cross-organizational standardization — and often sources of complaint — are usually around release cycles or provisioning, whether it’s more database capacity or CPUs in the cloud.

Streamlining Is the Name of the Game

In a proprietary report, Gartner advocated that internal developer portals — the singular interface that gives developers access to the tool chain, curated by the platform engineering team — should unify disparate platform capabilities. It breaks the platform engineering process across three aspects of the developer experience:

- **Discover and create:** search functionalities, integrated developer environments, team-specific toolboxes, software component catalogs, templates and documentation.
- **Integrate and display:** CI/CD pipelines, infrastructure automation, data infrastructure, API life cycle management, roles and permissions, and environment management.
- **Operate and improve:** monitoring and observability, resource usage, security and compliance automation, resilience engineering, configuration management and incident management.

The platform should aim to streamline the whole software delivery process, making it as simple, self-service and automated as possible for developers to safely release to pre-production and production environments.

While the focus of the platform team is its developers, be sure to talk to your ops teams. By smoothing the developer experience, you should reduce operations engineers' toil, too.

As the cloud native tool choice climbs well into the thousands, the pendulum swings back from a time of extreme developer autonomy to one where the platform offers devs a few choices. The purpose of platforms is to provide, as Netflix puts it, guardrails not gates. Your platform should not impede innovation.

Just remember that the platform isn't for everyone. It's for what Jean Yang, head of product, observability, at Postman, refers to as the 99% developers — not those at the cutting-edge companies like Facebook, Apple and other massive tech organizations, but rather the majority who deploy mostly on-premises, with a mix of new and legacy architecture.

Platform engineering is not for your innovation sandbox and 1% developer influencers. You still want them experimenting securely with new technology and processes. Who knows, their experiment may become a best practice that you one day implement across the company. But, for now, look to build a platform that helps the most developers achieve more with less effort.

API Gateways and Service Catalogs

As previously mentioned, it isn't unusual that your organization ends up running multiple platforms. A platform can start out as a basic abstraction or Confluence page all the way up to comprehensive bespoke, open source and/or out-of-the-box adoption of Platform as a Service and other platform engineering tools.

A platform engineering initiative often kicks off with an API gateway or a service catalog to help facilitate discovery and reusability. An overview of some of the most prominent platform tooling includes:

- Developer portals, like Configure8 and Port, are the user interface above the platform that enables the self-service. Backstage, a developer catalog created by the streaming media company Spotify, is a newcomer open source product that offers models to describe and visualize software systems.
- Code repositories like GitHub increasingly encompass more and more of the developer life cycle behind a single user interface. And because most developers already commit their work to such code repos, they don't have to necessarily learn new tooling to use it.

- Cloud Foundry, also open source and originally developed by VMware back in 2011, is the basis for the VMware Tanzu Application Service. VMware Tanzu Application Service is an integrated Platform as a Service that includes a developer portal, backing services, app deployment tooling and a cloud native runtime.

From the get-go, it's important not to get locked into one brand or tool that is too prescriptive. You need flexibility and agility as customer feedback pours in.

Besides taking care of that important but not differentiating work that cuts across your organization, the crucial job of a platform is to curate the user experience.

It's not uncommon to go a multiplatform route — in fact, many industry-specific regulations require isolated services and air-gapped environments. Just make sure to abstract out that complexity behind a unified developer experience that enables self-service.

No matter which tool kit you choose or build, it's important to treat your internal developers as your customers.

CHAPTER 04

Why Product Management Matters

A platform team should embrace full product management. That means the team has a product owner who keeps the other members true to their focus on internal customers, while also tying engineering to business goals.

Puppet's platform engineering survey found that the top priorities for platform teams' crossover with product management responsibilities:

- Increasing awareness of platform capabilities (47%).
- Setting realistic expectations for the platform team's role across the organization (44%).
- More closely following industry trends and keeping up with feature requirements (37%).

Even when the platform team is small, someone must be explicitly assigned to internally market and evangelize the product, announcing when new features are coming out and, especially, why.

It also helps for your consumers to have just one point of contact within the team, in addition to the usual communication pathways like Slack, demo days and lunch-and-learns. Make the most of self-service to keep improving the product. Use internal customer communication to develop feedback channels from customers to the platform team.

The Platform as a Product mindset isn't just about the product, but the management of it, too. While platform engineering centers on creating an opt-in developer experience with a catalog of service offerings, it's important to have centralized management via the platform team.

ING learned this lesson the hard way. While the Dutch financial services company's platform team was able to significantly improve time to market via hundreds of reusable components in its developer catalog, the complexity of integrating those services together and the amount of time spent on manual, repetitive configuration only served to slow the developer experience right back down.

"Engineers didn't know where to start, what to do next, who to talk to, [and] when they were integrating a service, which team was providing which part of the platform," said product lead Amir Abadir, a product manager lead at ING, at 2023's PlatformCon. "It resulted in an incoherent experience."

Platform-led configurations and controls created new steps and obstacles that meant it still took six months to deploy into production. His team realized it needed to refocus the platform strategy toward delivering a more uniform user experience, which then finally acted as the accelerator to market.

"The whole experience of our developers is what is most important, rather than our view as the platform provider looking at it as a bunch of collective services," Abadir said. "Our success is subject then to the overall experience of the platform, rather than the products that were being developed in isolation."

CHAPTER 05

Platform Engineering and Security

The goal of a platform is to cultivate an experience that makes "the right thing to do, the intuitive thing to do," according to Manjunath Bhat, an analyst with Gartner.

While much of the work of platform engineering is dedicated to development teams, it's undeniable that "right thing to do" could be extended to making the entire software supply chain more secure and compliant — hopefully automatically.

But it can be a balancing act, Bhat wrote in a Gartner report, for a developer portal to enforce governance without sacrificing agility, and facilitate self-service cloud access through built-in guardrails while still enabling rapid delivery and innovation.

Best Practices for Securing the Pipeline

So what goes into making a pipeline more secure under platform engineering?

"A secure software supply chain is the idea that you have an automated build pipeline, and you also have a way of automating how you configure and package your applications to be deployed and run in production," said Michael Coté, a senior member of the technical staff at VMware.

To start creating a more secure developer release life cycle, he recommends checking all of the third-party software components already integrated into the pipeline, to ensure "that they're secure enough and we can catalog them [with] whatever patch level they are."

Then these decisions need to be logged to create an audit trail, which provides proof of meeting regulations and standards.

As your platform team looks to secure your software chain, look for all the components — including libraries, services you depend on, your configuration — and "check them as they are going through the build pipeline to make sure you're building software that is secure," Coté said.

What that means, he continued, is "you're not deploying third-party libraries that have known problems in them, and then your audit log lets you verify that people are following the policy you've set out, that they're following the good security behaviors."

> "Securing your platform's automated build pipeline includes making an inventory of all third-party software your organization is using, checking its patch status and logging all decisions to create an audit trail, according to Michael Coté of VMware.

Then, when you've automated this secure value stream, it should become a lot easier and quicker to patch something, he said.

Therefore, communication about security should, at least in part, come via the platform team, too.

At GAIG, "we collaborate with our security team," Kohl said. "We work with them to put tooling in place and take the findings of that tooling and raise awareness of security threats that [developers] used to take for granted — we want them to worry about things."

Security awareness is a way to reach developers who don't want to be responsible for security. A self-service visualization built into the platform helps developers not only know that a vulnerability exists, but should help them find it and show how to fix it.

This becomes even more important as organizations like GAIG attempt to move from on-premises toward the public cloud, which, Kohl said, "really raises the bar. You have to change your threat modeling and approach to security."

While GAIG's foundation team members partner with security, they remain stewards of the dev team. This means they'll need to evaluate if the move to cloud — and the necessary restrictions their industry must follow — will make developers' jobs faster, slower or stay the same.

CHAPTER 06

Quantifying Developer Experience

Science has long taught us that you can't improve what you can't measure. Tracking the maturity of your platform engineering program is critical to its success.

The best way to measure the success of any technology is to ask its target audience for feedback. Fortunately, your internal developer customers are highly accessible — some even working on the same floor and definitely in the same Slack channels as you.

But there are more structured ways to measure DevEx.

For instance, a 2023 research paper by the Association for Computing Machinery, "DevEx: What Actually Drives Productivity," applies a developer-centric approach to measuring and improving productivity. It examines the lived experience of developers and the day-to-day friction they encounter, measuring three dimensions:

- **Feedback loops:** the wait time for developers to get things done.
- **Cognitive load:** how much a developer needs to know to deliver the value they've created to customers.
- **Flow state:** when developers "get in the zone," with limited distractions and a sense of purpose in their work.

These drivers are grounded in 25 sociotechnical factors, including things like unrealistic deadlines and needless interruptions (which negatively effect DevEx) and well-structured code, clear tasks and a developer's belief they are good at their job (which all positively affect it).

Here, developer experience is measured by a combination of quantitative data sources — like issue trackers, CI/CD pipelines and code repositories — and qualitative via semi-regular, anonymous surveys.

Another key measurement to all forms of customer success is the Net Promoter Score. In business, the NPS measures how likely a customer is to recommend a product or service. The engineering NPS asks: How likely are you to recommend another engineer to work here? This simple question brings a lot of insight into overall company sentiment and developer experience.

Candid group feedback sessions can also be useful. Atlassian's Boyagi and his team put together focus groups of engineering leads and engineers to regularly ask: Is the platform slowing you down or enabling you? Where can we improve?

You may also ask app teams to incorporate rapid feedback on the platform into their retrospectives. Then you can focus on the positive or negative effect of the platform on a specific value stream.

> "Developer joy comes from a combination of work that is easy to do and a great culture to support it. The success of platform engineering is not only measured in the tool that's built but by the people it serves.

But remember, it's human nature to vent and complain when given a chance. Open, undirected feedback can often be negative, Boyagi warned, so take everything as an impetus to improve, understanding that the platform team is not necessarily doing anything wrong.

For platform teams, finding bugs and rolling back changes comes in addition to increasing release velocity and providing ways to decrease the mean time to recovery (MTTR), an essential metric for platform teams to track.

The time to onboard a new developer serves as another key DevEx metric. At Spotify, a successful onboarding is measured by when a new developer executes their 10th pull request, because Spotify has determined that is when a developer starts to feel a sense of productivity. Over two years, Spotify decreased its onboarding time from 110 to 20 days, using the time to onboard method.

Generating 'Developer Joy'

"Developer joy" comes from a combination of work that is easy to do and a great culture to support it. The success of platform engineering is not only measured in the tool that's built, but by the people it serves.

Giving programmers some options within their platform is also crucial for generating developer joy — enabling coders to relish the experience of building the applications their organization needs.

Fiserv, a fintech provider that supports nearly 10,000 financial institutions around the globe, created the Fiserv Developer Studio as a self-service portal for its

customers' developers. The studio was moved from proprietary servers to the cloud with the cloud-agnostic VMware Tanzu Application Platform.

The fintech company wanted to maintain governance over how applications were built by its developers, to enable those apps to be launched over multiple cloud environments. But it also wanted to make it easier for its developers who used its product to provision the cloud infrastructure they needed 24/7.

In addition, Fiserv sought to create secure sandbox environments to allow its developers to experiment. The Cartographer feature in VMware Tanzu Application Platform allowed Fiserv to build blueprints for secure build and release pipelines that can be used for all engineering teams in its organization, speeding up product development and scaling.

"Cartographer is a game changer," Ganesh Venkataraman, Fiserv's CTO for digital technology transformation, told a writer for VMware's blog. "It gives us a consistent supply chain we can use across all Fiserv teams without locking them into using one tool. We still have the foundation of developer freedom, and the DevOps experience is phenomenal."

Observability is also incorporated into Fiserv Developer Studio's operations, through the use of VMware Aria Operations for Applications. It offers high-level visibility of analytics and insights, helping to locate any microservices issues that may require remediation.

Regardless of what platform you adopt, remember to measure for developer joy, advised Boyagi. He described this joy as a mix of not only the developer experience, but also engineering culture. After all, honest feedback — and really the success of

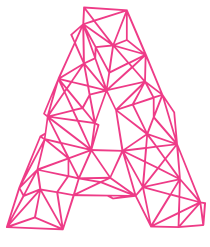
your whole platform engineering program — hinges on psychological safety, an environment where ideas can be expressed freely without fear of retribution.

"Engineering culture is the values and practices and the norms within an engineering organization. And the things that influence those are leadership, decision-making process, team structures," Boyagi said. "A healthy engineering culture is represented by engineering teams who look at promoting collaboration, innovation and continuous learning."

Developer joy then comes from a combination of an easy workflow and a great culture to support it. Remember, your developers — your internal customers — will be the ones who determine your platform engineering strategy's success.

CHAPTER 07

Measuring the Success of Your Platform



platform team still needs its own key performance indicators to monitor its own progress. One such measurement is how easy the platform is to operate, checking for things like:

- Total number of incidents.
- Time spent on incidents in a given month.
- Service-level objectives (SLOs) or service-level indicators (SLIs).
- Mean time to resolution (MTTR).
- Number of open critical security vulnerabilities.

With such a nascent trend like platform engineering, it's also natural to look for ways to measure your progression. While DevOps leans heavily into DORA metrics — deployment frequency, lead time for changes, change-failure rate, and time to restore service — all of which do affect app team velocity, there's a need for a more developer-focused way to measure the outcome of platform engineering.

The Platform Maturity Model

		1	2	3	4
Funding	How does the company value (and therefore fund) platform efforts?	One-off	Annual platform budget	Platform team budget	Profit and loss
Adoption	What compels users to start, and be successful, using your platform?	Mandatory	Build it and they will come	Internal champions	Platform advocacy
UX	How do users interact with and consume offerings from your platform?	Manual request queue	Off-the-shelf offerings	Curated entry point	Paved paths
Backlog	How are requests and requirements identified and prioritized for your platform?	Reactive	Scheduled	Evolutionary	Platform as a Product
Organizational structure	How does product engineering manage non-differentiating (and often internally common) tooling and infrastructure?	Dev and Ops	Full stack developers	Developer enablement	Platform team(s)
Cross-functional representation	How does each business requirement (e.g., compliance or performance) get enabled by platform offerings?	Off platform	Tools provided	Automated by platform team(s)	Specialists driven

Source: "Platform Maturity Model", Syntasso.

Figure 7.1 *Follow the model's four-point grading system as a way to measure the progression of your Platform as a Product.*

The Syntasso team open sourced its Platform Maturity Model, which guides teams to answer the following questions:

- How does the company value (and therefore fund) platform efforts?
- What compels users to start, and be successful, using your platform?
- How do users interact with and consume offerings on your platform?
- How are requests and requirements identified and prioritized on your platform?
- How does product engineering manage non-differentiating (and often internally common) tooling and infrastructure?
- How does each business requirement (e.g., compliance or performance) get enabled by platform offerings?

For each of these six questions, there are four stages of answers to help gauge your own organization's platform progression.

CONCLUSION

Maintaining Platform Engineering Success

In the end, platform engineering is not a quick fix or a panacea. It's an area of continuous improvement that needs continuous investment — no matter what the economy. Adopting a cross-organizational Platform as a Product mindset enables teams to do more with less.

The longer an organization has implemented platform engineering, the higher confidence they have in the continued success of their platform teams, according to the Puppet survey. It takes on average three years to see a significant increase in developer productivity, according to the report — which compares reasonably with how long it can take organizations to implement a successful cloud transformation.

With 71% of respondents in that survey reporting that their organization plans to hire people with platform engineering experience over the next year or sooner, organizations seem confident in the return on investment of a successful Platform as a Product strategy.

Are you?

About the Author

Jennifer Riggins is a culture-side-of-tech storyteller, journalist, writer, and event and podcast host, helping to share the stories where culture and technology collide and to translate the impact of the tech we are building. She has been a working writer since 2003, and is currently based in London.

Disclosure

The following companies mentioned in this ebook are sponsors of The New Stack: Port, Puppet by Perforce and VMware.

TNS owner Insight Partners is an investor in Ambassador Labs, Apollo GraphQL, Devtron, Fermyon, Jit, Kasten, Kubeshop, LibLab, Roadie, SingleStore, Slim.AI, SonarSource, StormForge and Tigera, which are also sponsors of The New Stack.