E-book series

Microsoft Azure

Platform engineering

An introduction to start right, stay right, and get right

Introduction

When tools, services, and systems support efficient ways of working at scale—without compromising privacy, security, or compliance—developers are freed up to apply their ingenuity to the more creative aspects of collaborating and product-making. Organizations get the impact they've invested in. End customers get products they love and trust. And the developers who deliver that impact and build those products feel more purposeful and satisfied than ever before.

That's the promise of *platform engineering*, a practice built on DevOps principles that we adopted across Microsoft.

We're always looking for ways to drive engineering excellence and support high-performing teams. DevOps practices continue, but our customers tell us that inconsistent compliance and manual processes slow progress and increase costs.

Even as organizations take advantage of the latest cloud-native systems, Forrester Research reports that they "have yet to realize fundamental expectations of collaboration, cost optimization, and process efficiencies."[1]

Our own experience is a case in point. In the past several years, our development teams have embraced distributed computing, cloud-native services, open source, AI, and more. Their day-to-day work has grown more complex, and their learning curve has spiked as new tools—and more of them—are adopted. Developers use an average of 16 tools per day,[2] and it takes 23 minutes[3] to regain focus after switching from one context to another. Their role requires them to outwit hackers while meeting aggressive deadlines and speeding innovations to market. And they're burning out.

We needed a human remedy for the modern developer experience—something so common that researchers just call it DevEx.[4] As engineers in a world where all companies are becoming software companies, we realized that our own productivity and agility starts with DevEx. We approached platform engineering with the goal of resolving specific pain points and blockers, but along the way we saw improvements in compliance, security, costs, and time to value—all the factors that impact productivity, efficiency, and innovation.

This e-book shares our best learnings so your organization can build the foundation of a platform engineering practice and you can start seeing the benefits for yourself.

[1] A Forrester study on unleashing cloud-native for new levels of value delivery. MIT Technology Review. March 14, 2022.
[2] Meyer, André N. et al. The work life of developers: Activities, switches and perceived productivity. IEEE Transactions on Software Engineering. December 1, 2017.
[3] The cost of interrupted work: More speed and stress. 2008.
[4] Noda, Abi, et al. DevEx: What actually drives productivity. Association for Computing Machinery. May 3, 2023.

Start right, stay right, and get right

Platform engineering is a practice and a cultural shift. When DevEx is the focus, two clear goals emerge:

- Developers must have fast access to what they need within a secure, governed framework. We call this *self-service with guardrails*.
- Developers are your customers, and your engineering systems and application platforms are the products they need.

We recommend setting up a product team dedicated to creating an internal developer platform (IDP), as we did. This team helps drive DevEx and can identify and ease the challenges of supporting teams. Maybe Operations is drowning in tickets, or other delivery systems are out of compliance. The dedicated platform engineering team uses its central position to drive three initiatives for your organization.

Start right by equipping developers with self-service tools.

Self-service gives developers the autonomy they need to deploy resources on demand and stay productive while remaining within the bounds of your organization's guardrails—security, compliance, operations, standards, and costs. Templates are the building blocks of platform engineering. They help developers start right with fewer errors. Using infrastructure as code (IaC) through continuous delivery (CD) pipelines is another important part of enabling self-service and scaling compliance. You can even apply an "everything as code" pattern to policies and people. If it's code, it can be tracked, versioned, checked, automated, and repeated consistently.

Stay right by enforcing guidance, governance, and policy.

Keeping your initiative on track requires visibility into engineering systems and applications, monitoring and incident management, and continuous improvement practices. Combining start-right templates with stay-right guidance and automation helps ensure that developers stay right.

Get right by starting with the easiest platforms and projects.

DevEx is a mindset shift, and it takes time. You can start by making the most of existing investments and bringing them into compliance. The best practices and building blocks in this guide can help you find the right place to start.

DevEx is at the center of platform engineering, a movement that improves the experience of developers and supporting teams. To start right, developers need easy access to tools and systems that comply with your requirements. They stay right through automated governance and policy. To get right, your organization needs to think of developers as an internal customer and look for ways to improve their experience.

RECOMMENDED READING

- What is platform engineering?
- Platform engineering principles
- Empower Developers through self-service with guardrails

Build a paved path

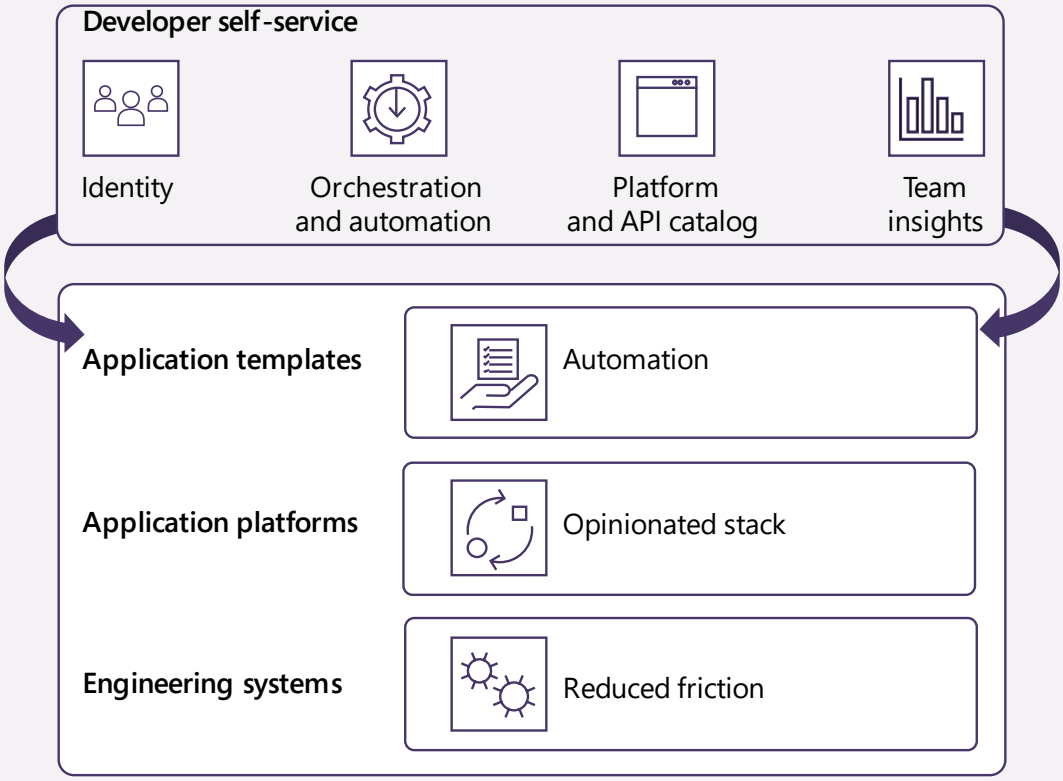
Without any organization-level guidance, development teams can quickly start duplicating work and creating tool sprawl. It's not just inefficient—it also increases your risks around security and compliance. A paved path keeps developers speeding along in the right direction. Paved paths are happy paths—they reduce bureaucratic hurdles for both development and operations teams.

The simplest way to build a paved path for your teams is to reuse existing investments. The goal is to give them an IDP that curates the best available services, tools, and processes for deploying, running, and operating applications and their runtime environments.

An IDP is uniquely suited to your organization and goals. To build it, you need input from key stakeholders—your security, architecture, and operations teams. Your platform engineering team needs to collect input about the IDP, set the standards, and codify the best practices into reusable templates and system capabilities. The team builds and manages your IDP like a product and treats your developers as its end customer. If designed and socialized well, your IDP goes viral and DevEx gets a boost.

Our own paved path for secure, governed development with team-level flexibility is 1ES (for One Engineering System). Used across Microsoft, 1ES includes development environments, collaboration tools, and continuous integration (CI)/CD pipelines that set the baselines for security, integration, and management. With standardized engineering systems as the foundation, our application developers can more easily extend and customize as needed. We lean toward opinionated stacks that support standardization through predefined conventions, best practices, and design patterns.

Developers also need access to APIs, orchestrators, templates, and other tools. We automate as much as possible, using IaC templates to deploy consistent, compliant environments that are governed by policy. We provide a shared Azure Kubernetes Service (AKS) cluster and provide self-service access to individual developers so they can deploy their applications to a dedicated namespace, integrate with other services, and move their work forward.



An IDP builds a paved path by defining a range of engineering systems, application platforms, and application templates for your developers. It gives them self-service access to the APIs and tools they need for identity, orchestration, and insights. The input of a designated platform engineering team is key to driving standards and selecting the right tools and processes that help your teams start right and stay right.

RECOMMENDED READING

Each customer is important

Adopt a product mindset

Build your team

Make it easy

Adoption efforts can falter if developers feel pressured into narrow choices. One team might like Jenkins, Docker Registry, and Terraform; another could prefer Azure Pipelines, Helm charts, and Azure Container Registry. The idea is to provide great tooling and supported experiences that smoothly guide your teams through critical requirements and standards.

To make it easy for your internal customers to find what they need, you can begin centralizing resources. A common first step is to create a catalog of the command line interfaces (CLIs), APIs, and other tools that developers need.

Centralized access to resources reduces tool sprawl so developers can start right and operations teams can govern and monitor.

Many organizations build a self-service portal that enables developers to freely create and destroy cloud resources. For example, consider the requirements of a team creating a stateful service application. Team members can select suitable, preconfigured templates from a catalog and generate a pull request that automatically creates a shared Kubernetes cluster along with Azure Key Vault for storing secrets and keys, Azure Cosmos DB for data, and other resources that comply with our policies. The CI/CD pipeline has built-in checkpoints that keep workstreams in compliance.

However, centralization does not equal portal. The goal is to give developers access to the platform with the tools they prefer, such as CLI and REST, and promote discoverability and reuse. You can expand your efforts as needs change.

RECOMMENDED READING

- Improve discovery and eliminate waste
- Design a developer self-service foundation

Start where you are

Platform engineering is a cultural shift that takes time, and it isn't a sequential process. You can start small and meet developers where they are—in the workflows and tools they're already using. A great place to start is by automating the high toil areas, where your efforts have a direct and immediate impact on developers. What are their key challenges and pain points? What are your business priorities? Is it security, compliance, time to value? Depending on the answers, you can set targets that deliver maximum business value, whether that's an investment in common deployment patterns, development environment and tools, or other resources.

Before investing in specific resources, it helps to conduct an inventory. With better visibility of the resources across your ecosystem, you can choose where to invest and what to deprecate. For example, several services that help developers discover resources also help your operations teams manage, track, and clean up those resources, including Azure Deployment Environments, Azure API Center, and package registries like GitHub Packages or Azure Artifacts.

As you scale your efforts, your inventories help you audit and uncover technical sprawl. You can begin centralizing control of your resources. When you find and fill gaps, you can better protect identity and secrets, enforce policies and role-based access control, and verify compliance.

To stay right, it helps to build security, compliance, and cost controls into the resources used by developers. As a first step, we recommend defining permissions and policies for individual templates and enforcing least privilege and least access policies according to role, project, and stage of development.

RECOMMENDED READING

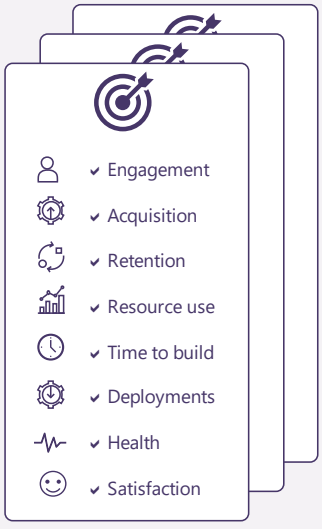
- Meet users where they are
- Use the everything as code pattern

Measure success

Measuring success with key performance indicators informs product decisions and helps you refine and invest in the right areas. Ours evolved over time. Today we use a variety of metrics to help us see how customers use 1ES. We look at the acquisition, retention, and engagement of specific features and measure user satisfaction. We assess pipeline throughput to see how efficient the DevOps process is, measuring the time to build, test, deploy, and improve. We use live-health metrics to monitor our IDP like any other product platform, and track how quickly issues are detected and fixed. We also survey employees to collect their concerns and monitor for burnout. Even vacation time can be used as a metric of our own productivity and the success of our DevEx efforts.

Metrics can help you make the case for your next investment, but the best metric is the one that matters most to your organization. Simply start with that one, work toward improvement, and repeat.

With DevEx as a key component of platform engineering, measuring customer satisfaction is always important. The platform engineering team needs to listen to its internal customers and build a feedback loop. A focus on customer-driven requirements can help you retain key talent as you move along in your platform engineering journey.



Success metrics help you track satisfaction with your IDP and direct platform engineering efforts.

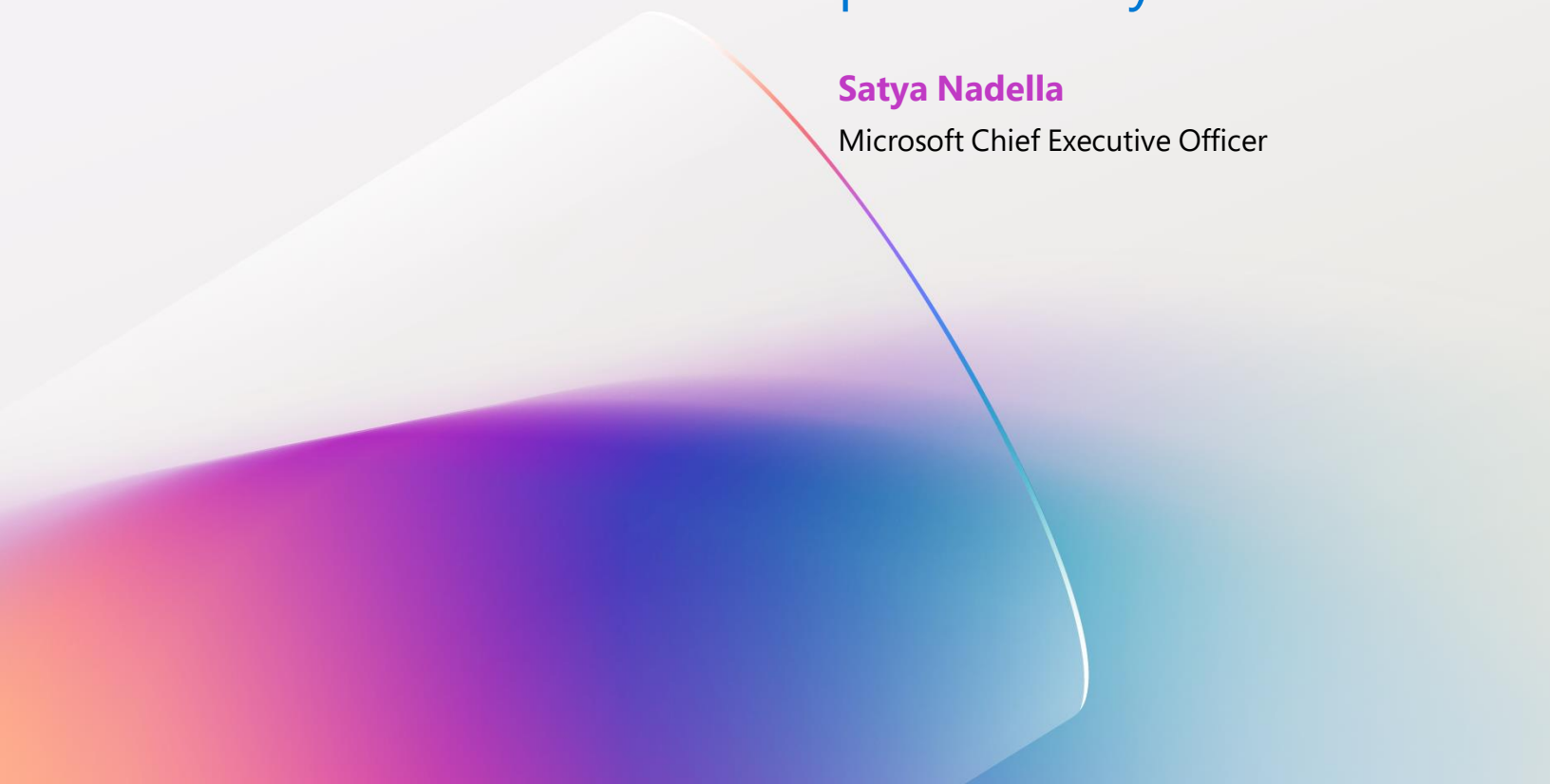
RECOMMENDED READING

- Plan and prioritize
- Start your platform engineering journey
- Apply software engineering systems
- Refine your application platform

"There cannot be a more important thing for an engineer, for a product team, than to work on the systems that drive productivity. So I would, any day of the week, trade off features for our own productivity."

Satya Nadella

Microsoft Chief Executive Officer



Principles of platform engineering

- Make each customer important.
- Adopt a product mindset.
- Empower developers through self-service with guardrails.
- Improve discovery.
- Eliminate waste through inventories and relationship-tracking.

Next steps

To help you set up the right practices for your organization, we've made our learnings available on the Platform engineering website.