Innovation Insight for Internal Developer Portals

Published 1 February 2022 - ID G00761445 - 12 min read

By Manjunath Bhat, Mark O'Neill, **and 1 more**

Product teams often struggle due to disparate tools and disjointed workflows as they accelerate digital transformation. Software engineering leaders leading platform teams must establish internal, self-service developer portals to enable consistency and scale cloud, agile and DevOps initiatives.

Overview

Key Findings

- Improved developer experience is a key factor in attracting and retaining software engineering talent, as well as more effectively delivering software.
- Platform teams enabling cloud adoption must enable rapid innovation while implementing governance, security and compliance controls. These two goals often conflict because traditional approaches to cloud governance controls stifle speed and agility.
- The adoption of developer portals will suffer unless they continuously adapt to changing development needs, architecture patterns and deployment models.

Recommendations

Software engineering leaders responsible for building products and platforms must:

- Improve developer experience and effectiveness by establishing internal developer portals to streamline the software delivery life cycle and support reuse, sharing and collaboration.
- Enable governance without sacrificing agility by using developer portals that provide self-service cloud access through built-in guardrails while still enabling rapid delivery and innovation.
- Continuously innovate portal capabilities by appointing a platform owner for the developer portal to manage its roadmap, gather feedback and market its capabilities.

Strategic Planning Assumption

By 2025, 75% of organizations with platform teams will provide self-service developer portals to improve developer experience and accelerate product innovation.

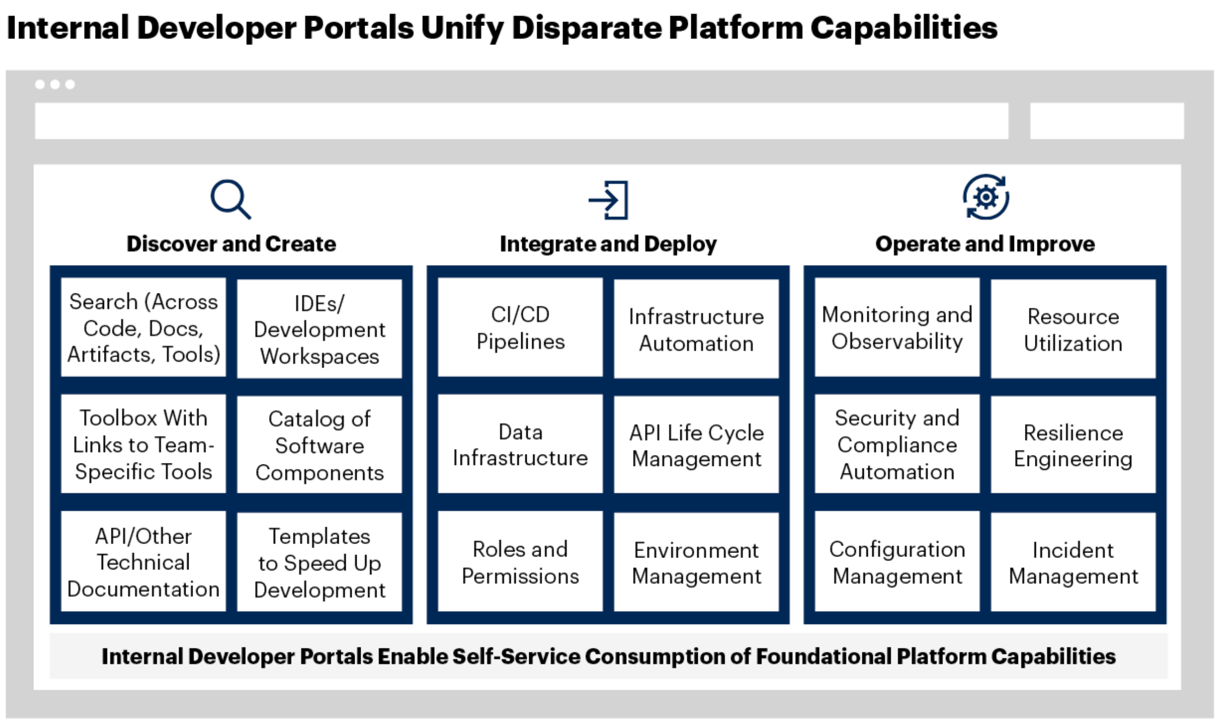
Introduction

Software engineering leaders increasingly focus on improving developer experience and removing friction from the software delivery process. Improving developer experience enables software engineers to focus on delivering innovative software and delighting customers. Developer experience is important because it improves flow in the complete software delivery life cycle.

Most organizations use a complex collection of platforms, tools and frameworks across different layers of the technology stack. This internal maze of technologies creates unnecessary overhead, duplicates effort and hurts developer productivity. As a result, developers end up doing nonessential work to manage the overhead. Developer productivity and happiness requires removing roadblocks, and tool complexity is one of those roadblocks.

How can software engineering leaders help to minimize friction and maximize flow in the design, development and delivery of software? Internal developer portals provide an extensible, configurable and customizable framework that can be used to abstract away complexity in design, development, deployment and postdeployment (operations) workflows. Figure 1 depicts foundational capabilities for such a portal.

Figure 1: Internal Developer Portals Unify Disparate Platform Capabilities


Internal Developer Portals Unify Disparate Platform Capabilities
Discover and Create
Search (Across Code, Docs, Artifacts, Tools)
IDEs/ Development Workspaces
Toolbox With Links to Team-Specific Tools
Catalog of Software Components
API/Other Technical Documentation
Templates to Speed Up Development
Integrate and Deploy
CI/CD Pipelines
Infrastructure Automation
Data Infrastructure
API Life Cycle Management
Roles and Permissions
Environment Management
Operate and Improve
Monitoring and Observability
Resource Utilization
Security and Compliance Automation
Resilience Engineering
Configuration Management
Incident Management
Internal Developer Portals Enable Self-Service Consumption of Foundational Platform Capabilities

Source: Gartner

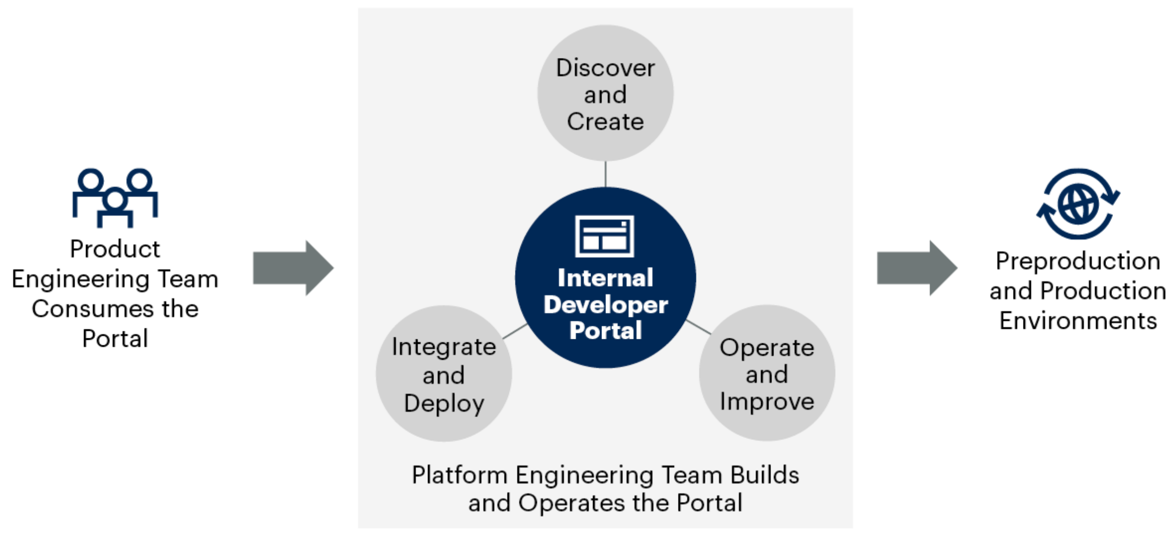
Software engineering leaders are asking, **"How can we build an internal development platform that enables us to publish and discover shared artifacts, including APIs, vendor-procured and open-source software, collaboration and DevOps automation tools and I&O resources?"**

Internal developer portals provide an answer. *These portals enable software engineering leaders to create a versatile "app store" that increases software reuse, improves the developer onboarding experience, streamlines software delivery and facilitates knowledge sharing.*

Platform teams build, own and operate the developer portal. The portal improves developer experience by providing self-service access to platforms and tools to manage the software development life cycle. Figure 2 depicts the positioning of internal developer portals and their value to product engineering teams.

Figure 2: Internal Developer Portals Help Streamline the Software Delivery Process

Internal Developer Portals Help Streamline the Software Delivery Process


Product Engineering Team Consumes the Portal
Discover and Create
Internal Developer Portal
Integrate and Deploy
Operate and Improve
Platform Engineering Team Builds and Operates the Portal
Preproduction and Production Environments

Source: Gartner
761445_C

Description

Internal developer portals enhance developer experience by reducing friction throughout the software delivery life cycle. Developer portals streamline the development, deployment and life cycle management of software artifacts. They integrate with software delivery platforms to enable continuous delivery and infrastructure automation tools for abstracting away the underlying infrastructure complexity.

Integration with documentation tools, source code and artifact repositories enables organizations to adopt and scale innersource practices. Finally, it enables continuous operations by providing access to a toolbox via plugins for automation, monitoring and incident management. Product and platform teams can use developer portals as a one-stop shop for tracking visibility and ownership of services and components.

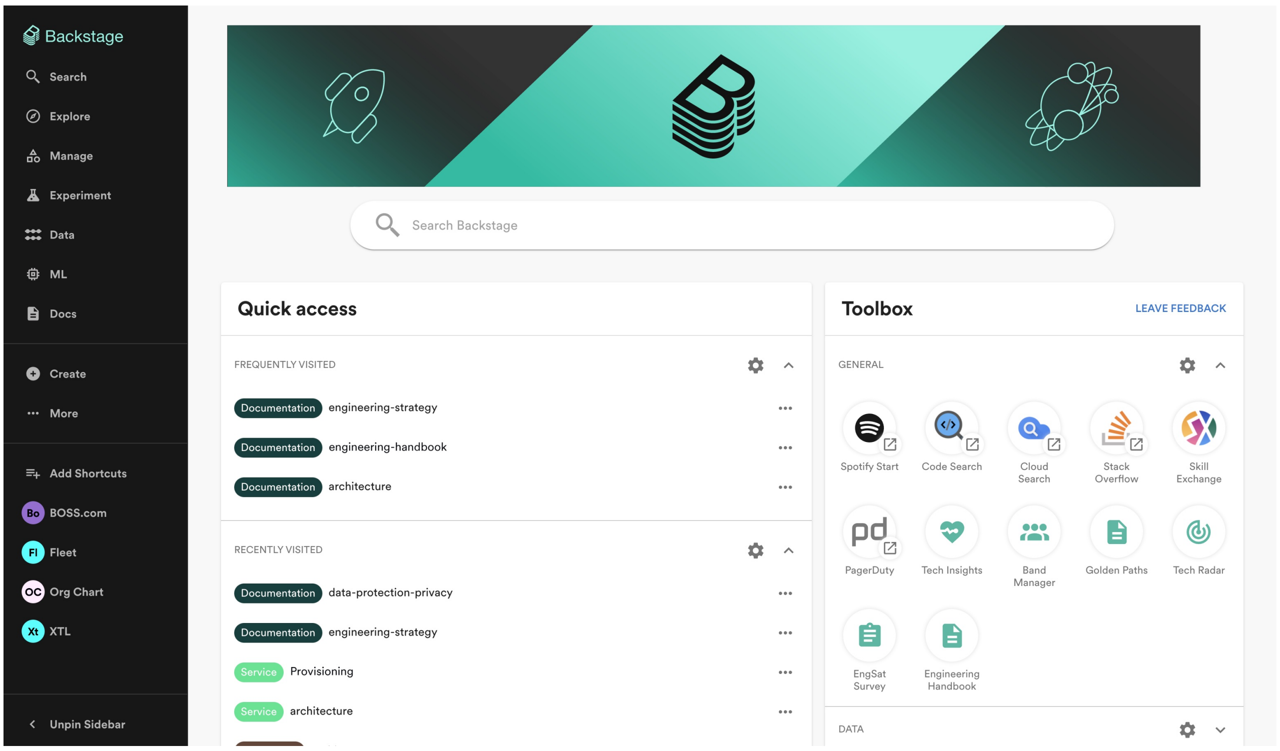
Internal developer portals are to developers what trails are to hikers in a jungle. They provide a well-trodden path from concept to customer value amid a chaotic mix of tools and practices.

Internal developer portals have three primary characteristics:

1. **Abstraction:** Abstract away the underlying complexity across multiple technology layers — data, applications, open-source libraries, programming language and scaffolding frameworks, infrastructure, and APIs.
2. **Developer-centric view:** Provide developers with visibility of the complete software development life cycle — from ideation to operations.
3. **Pluggable framework:** Enable extension of the portal, either via plugins or APIs, and empower developers to customize the portal for their own needs.

Backstage, created and open-sourced by Spotify, enables the building of internal software catalogs and developer portals. The platform supports software templates (scaffolding for new building software components) and technical documentation (writing documentation in markdown). It also helps manage operational needs (monitor health of services). The plugin architecture makes the platform extensible to integrate with the rest of the IT infrastructure (e.g., access management, container management and API management). See Figure 3.

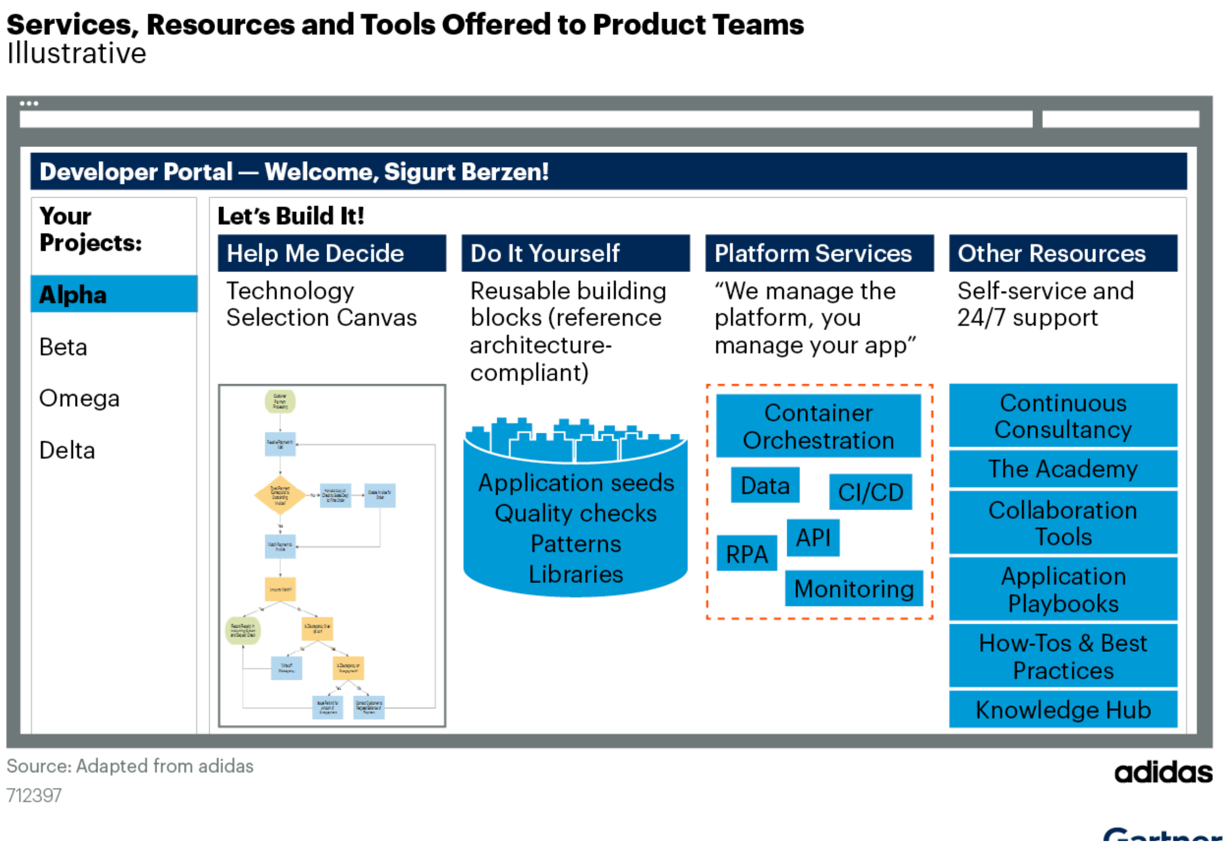
Figure 3: Backstage Developer Portal Homepage



Source: Spotify

In some organizations, platform teams choose to build their own internal developer portals. Figure 4 illustrates an example of such an internal, self-service developer portal built by the adidas platform team.

Figure 4: Sample Self-Service Portal (adidas)



Internal developer portals should make it easy for developers to perform Day 0, Day 1 and Day 2 activities throughout all phases of the software delivery life cycle (SDLC).

Day 0: Discover and Create: Provide ready-to-use templates to create new software applications, services and components with embedded patterns, procedures and policies. This includes a software catalog, technical documentation and integration with preferred developer tooling such as code repositories and build automation pipelines.

Software Catalogs enable tracking assets and dependencies, and they provide "situational awareness." The catalogs aggregate metadata about services/components/assets such as service owner information, deployment history, compliance policies, support model and service level objectives.

Day 1: Integrate and Deploy: Provide a single dashboard to manage distributed infrastructure across cloud and on-premises environments. The portal also supports multiple DevOps automation tools and the ability to self-manage development and test environments. Portals that are integrated with API management platforms make it easy for developers to create, manage and explore API definitions and policies for any service.

Day 2: Operate and Improve: Provide access to service dependency graphs, operational metrics, logs and traces. For containerized deployments, the portal can provide an application-centric view into Kubernetes clusters (including pod visibility and performance).

The team that owns, manages and runs the service will benefit from understanding the interdependencies, how the services are performing, when the last change was deployed, who owns it, who is on call and which alerting/incident response tool they should use. Maintaining the metadata as part of the software catalog is crucial to deriving the desired benefits from the developer portal.

Although developer portals can be tailored to different roles in the product team (such as product owners, site reliability engineers or IT operations), the primary target role is the developer. As shown in Table 1, internal developer portals improve developer experience and simplify development workflows by unifying a disparate set of platform services and tools into one cohesive portal.

Table 1: Developer Portals Should Simplify Access to Platform Services

Platform Services and Technologies	Markets and Tool Categories

Development environments	▪ Browser-based integrated development environments (IDEs) ▪ AI-augmented development and testing tools ▪ Tools for design ▪ Collaboration and technical documentation (how-to guides for development workflows and product information)
Data services	▪ Cloud AI developer services (CAIDS) ▪ AI orchestration platforms ▪ SQL/NoSQL data stores ▪ Data integration tools
APIs	▪ API gateways ▪ API management ▪ API developer portals ▪ API documentation ▪ API security
Security, governance and compliance tools	▪ Container security ▪ Application security testing (AST) ▪ Software composition analysis (SCA) ▪ Identity and access management (IAM) ▪ Secrets management

Enterprise integration, DevOps and RPA tools	▪ Value stream delivery platforms (VSDPs) ▪ Value stream management platforms (VSMPs) ▪ Low-code application platforms (LCAPs) ▪ Robotic process automation (RPA) ▪ Enterprise integration platform as a service (iPaaS) ▪ Data integration tools
Infrastructure and operations management	▪ Container management ▪ Infrastructure automation (IaC) ▪ Policy management ▪ Incident management
Reliability and resilience	▪ Application performance monitoring (APM) and observability tools ▪ Performance and resilience

Source: Gartner

Benefits and Uses

Software engineering leaders who use developer portals are seeking to achieve four primary benefits:

- Reduce friction in development activities and shorten lead time
- Improve collaboration, visibility and ownership using innersource practices
- Minimize time to resolve incidents through efficient discovery and visualization of services
- Accelerate cloud adoption and drive business innovation with adaptive governance

Reduce Friction in Development Activities and Shorten Lead Time

Internal developer portals reduce friction by providing a unified experience depending on your role. For developers, the portals enable self-service throughout the SDLC. This includes self-service access to preapproved software packages, CI/CD tools, development and test environments and end-to-end visibility. Product owners can get continuous feedback from production systems as they launch new features.

Improve Collaboration, Visibility and Ownership Using Innersource Practices

Innersource portals enable developers to more easily learn about — and contribute to — software projects within their organization that might interest them. They also provide product owners with a forum to share their team's work and encourage participation and contribution from developers outside of their teams. The innersource portal serves as a hub where developers can share and reuse knowledge; source code; and other existing assets, services and APIs.

One of the biggest pivots that we've made with InnerSource is to switch from a governance model to a self-service model. During the second year of our journey, we created our internal InnerSource Marketplace and moved to a self-service model where engineers were able to innersource their code by simply adding an "innersource" topic to their GitHub repo.

— Melinda Malmgren, American Airlines Developer Experience Group

---

Minimize Time to Resolve Incidents Through Efficient Discovery and Visualization of Services

Developer portals can minimize incident resolution times by providing consistent visibility into services and their interdependencies, including service owners. This enables a scalable mechanism for SREs, operations and product teams to support a continuously growing digital footprint without relying on tacit knowledge. Integration with the required automation, monitoring and incident response tools enables product teams to take end-to-end ownership, including reliability and resilience of production environments.

Accelerate Cloud Adoption and Drive Business Innovation With Adaptive Governance

Development and operations teams have long engaged in a tug of war between the need for autonomy and agility and the need for governance and control. Platform teams codify the necessary security, cost and compliance policies for managing cloud infrastructure. Product teams consume cloud services through standardized templates and blueprints in the portal. This approach enables an adaptive governance model.

Risks

Organizations adopting developer portals must avoid three primary pitfalls:

- Trying to Shoehorn Development Workflows Into Organizationwide Blueprints
- Building Developer Portals Without Involving Developers
- Assuming That Internal Developer Portals Are Turnkey Solutions

Trying to Shoehorn Development Workflows Into Organizationwide Blueprints

Developer portals serve as a “one-stop shop” where developers can find all tools, playbooks, sample projects, templates and documentation. However, this does not mean that all teams must have the same portal experience or the same standardized toolsets. Rather, platform teams responsible for standing up the portal must provide ways to tailor the portal based on the specific needs of the product teams. Otherwise, there is a risk of forcing developers to use tools and workflows they are not comfortable using, which hurts developer satisfaction and effectiveness.

> **Platform engineering teams must treat developer portals as a product that continually evolves to meet agility, collaboration and automation goals.**

Building Developer Portals Without Involving Developers

Building developer portals without involving developers — the actual users — is a recipe for failure.

This is because developers will try to work around systems that impede their workflows. Worse, the portal’s capabilities can remain unused, rendering it ineffective. Platform teams must treat developer portals as a product and continually evolve them based on product teams’ needs. Thus, software engineering leaders must appoint a platform owner to ensure it remains aligned with developer needs and market its capabilities within the organization.

Assuming That Internal Developer Portals Are Turnkey Solutions

Developer portals are not turnkey solutions. They must be configured and integrated with existing tools and systems to be useful and effective. The representative providers mentioned in this research offer platforms that simplify building internal developer portals by way of preintegrated capabilities, plugins and add-ons — so you don’t have to build out the integrations from scratch.

Internal developer portals must also not be confused with API developer portals (which are often part of API management platforms). Instead, they integrate with API management platforms (e.g., Backstage integration with Apigee). Internal developer portals have a broader remit and span the complete delivery life cycle, integrating different aspects of building, delivering and operating software.

Adoption Rate

The adoption of developer portals is closely tied to improving developer experience while addressing the need for adaptive governance. Therefore, the greater the maturity of an organization's DevOps and platform engineering practices, the higher their likelihood of using a developer portal. Gartner frequently sees self-service ranked as one of the most desirable attributes when organizations create DevOps maturity models. Organizations adopting developer portals tend to have dedicated teams — often called "engineering excellence," "engineering productivity," "developer experience" or "platform engineering" teams.

Emerging open-source tools like Backstage.io and commercial providers such as Humanitec, Calibo and Mia-Platform provide platforms to build self-service developer portals. These platforms intend to obviate the need for organizations to build a portal from scratch by providing a customizable framework. The platform enables easy access to shared components, API documentation and templated workflows for software developers.

Backstage.io maintains an open-source listing of adopters in its GitHub repository. [1] Backstage, a Cloud Native Computing Foundation (CNCF) sandbox project, offers a subset of developer portal capabilities. Given its open-source roots, it has gained a lot of interest and public visibility. For example, American Airlines was one of the earliest adopters and built its internal developer portal (Runway) based on Backstage. [2]

However, Gartner inquiries indicate that Backstage implementations may require substantial effort in standing up the service. The lack of enterprise support for open-source software can be a barrier to adoption in some organizations. SaaS providers of Backstage, such as Roadie, are emerging to address this concern by eliminating the need for hosting, upgrades and ongoing platform management.

Recommendations

Software engineering leaders responsible for building products and platforms must:

- Improve developer experience and effectiveness by establishing self-service developer portals to streamline software development, deployment and operations.
- Enable adaptive cloud governance by using self-service developer portals to establish governance and gain economies of scale while enabling rapid delivery and innovation.

- Continuously enhance the portal by appointing a platform owner to define and manage a product roadmap and market its capabilities to developers.
- Establish KPIs for:
  - Agility, such as developer onboarding time and lead time
  - Reliability, such as change failure rate and mean time to repair
  - Business outcomes, such as customer retention and increased revenue
  - Automation, such as self-service provisioning of development and test environments

Representative Providers

- Atlassian (Compass)
- Backstage (open-sourced by Spotify)
- Calibo
- Humanitec
- LeanIX
- Mia-Platform
- OMNI Arsenal
- Opsera
- Roadie (Backstage as a service)
- Wipro (devNXT)

Evidence

[1] backstage/ADOPTERS.md, GitHub.

[2] Adopter Spotlight: American Airlines Demos Runway, Spotify Backstage.