

AHEAD



A COMPREHENSIVE LOOK AT THE

Challenges, Opportunities & Best Practices for Platform Engineering Adoption

Foreword

The challenges with infrastructure management and modern software development only continue to grow, and organizations need to find innovative ways to overcome them. Platform engineering has emerged as a critical foundation for organizations as they strive to enhance agility, scalability, innovation, and consistency across development teams.

The opportunities for platform engineering are substantial, but not all organizations are prepared for the shifts in people, processes, and technologies required. This report will cover all aspects of platform engineering to guide IT leaders who want to adopt forward-thinking strategies within their organizations.

Table of Contents

Introduction	3
1. Enhancing Development Efficiency with Platform Engineering	4
Why Organizations Need Platform Engineering	4
What Are the Key Elements of an Internal Developer Platform?	5
2. Growth & Adoption for Platform Engineering at Scale	7
Best Practices for Scaling Platform Engineering	7
Measuring the Success of Internal Platform Implementations	8
3. How to Develop Platform Engineering Teams	10
The Essential Roles & Skills within Platform Engineering Teams	10
Strategies for Creating and Expanding Resources for a Platform Engineering Team	12
4. How Observability & SRE Support Platform Engineering	13
The Benefits of Observability in Platform Engineering	14
The Role of SRE in Platform Engineering	15
5. Platform Engineering Security & Compliance	16
Tips for Combining Security with Platform Engineering	16
The Role of Security Automation in Platform Engineering	17
Conclusion	19

Introduction

Innovation is crucial for business success, but today's modern infrastructure is becoming increasingly challenging to operate. Distributed systems with microservices architectures are complicated to deploy, and the dynamic nature of containers makes maintaining cloud infrastructure difficult to manage. As a result, many development teams are forced to deal with complex infrastructure challenges instead of focusing on developing high-quality software.

Platform engineering is a newer discipline that focuses on building in-house software acceleration platforms that streamline and automate the development process to boost efficiency and creativity. By removing many of the roadblocks software developers face on a daily basis, an effective platform engineering strategy allows organizations to innovate faster and stay ahead of competitors.

Many leading organizations are already rolling out platform engineering, and adoption is only expected to continue. In fact, [Gartner](#) predicts that 80% of large software engineering organizations will establish platform engineering teams as internal providers of reusable services, components, and tools for application delivery by 2026.



Platform engineering and internal portals aim to equip developers with the tools and frameworks necessary to execute on expected timelines and deliver desired outcomes. According to Gartner, internal developer portals allow software engineering executives to “establish a flexible ‘app store’ that enhances software reuse, elevates the experience of onboarding developers, simplifies software delivery, and promotes knowledge sharing.”

In this chapter, let’s take a closer look at how internal portals and investing in platform engineering can streamline and accelerate the delivery of high-quality software. We’ll also cover the key elements of an effective internal developer portal.

WHY ORGANIZATIONS NEED PLATFORM ENGINEERING

In an ideal world, developers wouldn’t have to worry about the intricacies of infrastructure operations or the integration of numerous resources and applications, and would instead be able to focus on creating code and adding features. But the reality is that many development teams are held back by manual and repetitive tasks related to managing configurations and infrastructure.

The problem is that software developers are some of the most expensive assets that a technology department has, and the cost of environment delays and incomplete deployments add up. The mitigation of these challenges and costs is the ultimate goal of internal developer platforms.

Platform engineering and internal development platforms offer many benefits:

Increasing the Pace of Development: Instead of dealing with infrastructure and deployment issues, developers can spend more time writing code. This means platform engineering can accelerate development velocity and reduce the time to market for new features.

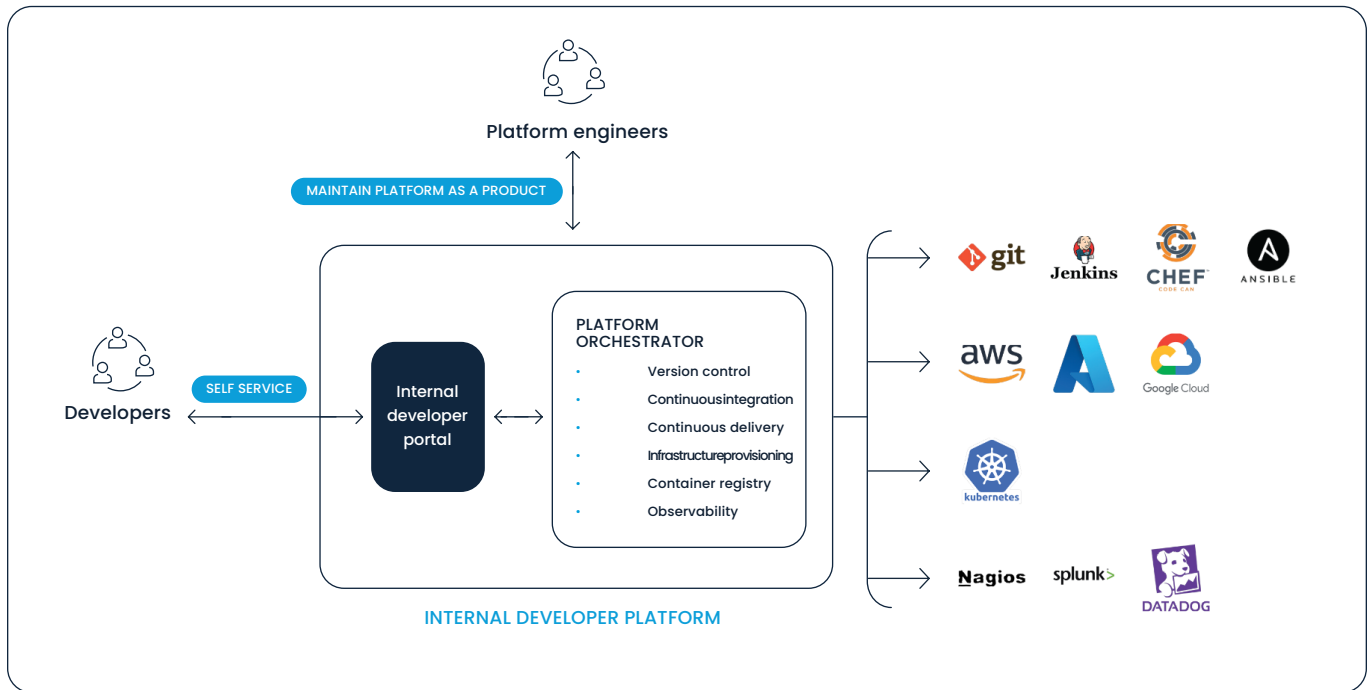
Boosting Trustworthiness: Many internal platforms aim to establish “paved roads (also known as golden paths) to success” that are easy to use and save time. Developers tend to trust and naturally adopt these compelling pathways and platforms, leading to more reliable software and reduced risk.

Enabling Scalability: When designing internal portals, the platform engineering team can choose the best infrastructure technologies that optimize performance and maximize uptime. This enables developers to build more scalable applications that better meet user demand.

Attracting & Retaining Talent: By eliminating many of the infrastructure headaches involved with deploying software, and in turn improving the developer experience, internal self-service platforms can help organizations attract and retain engineering talent. This is crucial for overcoming the longstanding skills gaps in the tech industry.

WHAT ARE THE KEY ELEMENTS OF AN INTERNAL DEVELOPER PORTAL?

The primary goal of the platform engineering team is to create and maintain internal developer portals that simplify and improve the software delivery process. This requires developing consistent tools, procedures, and guidelines that are developer-centric and aligned with overall business goals.



Real-World Use Case

AHEAD assisted a global financial services organization in building an approach to developing and maintaining “golden paths.” For the first two paths, the client – with the help of AHEAD architects – tackled provisioning data factories from traditional and structured file sources, as well as easily deploying applications to the cloud.

Leveraging proven methods of infrastructure as code (IaC), docs as code, pipeline automation, and policy as code, our engineers were able to create golden paths that enable outcomes for developers without encumbering them with implementation details or myriad architectural decisions. The golden paths are offered to developers as part of an internal developer portal for consistent self-service consumption.

Here are some key things to consider when building an internal developer portal:

Self-Service

Allowing developers and other teams to request the cloud services they need on demand improves the developer experience and reduces dependency on IT and operations teams. Many organizations use [ServiceNow](#) or [Backstage](#) to implement self-service portals.

Reusable Components

Providing developers with established standards and patterns through a library of reusable elements and templates can streamline development and enhance productivity. It's also a great way to improve software quality, security, and reliability by choosing reusable components and shared services that are already proven.

Automated CI/CD Pipelines

Automating deployment pipelines allows developers to push high-quality code into production faster. CI/CD tools like [Jenkins](#) and [Harness](#) enable continuous integration, testing, delivery, and deployment to the cloud without manual processes or complicated scripts.

Everything as Code

Developers can specify their infrastructure as code (IaC) with platforms like [Terraform](#) and [AWS CloudFormation](#) and configurations as code (CaC) with tools like [Ansible](#). These files can then be maintained within [Git](#) or another version control system. Similarly, Open Policy Agent (OPA) or AWS Config allow software teams to declare and enforce governance policies.

Monitoring and Observability

Platform teams should provide monitoring and observability tools to aid developers in finding and fixing bugs in their software. Popular tools like [Datadog](#) and [Splunk](#) offer development teams deep visibility into application and infrastructure performance. [More on observability for platform engineering in Chapter 4.](#)

Security

Integrating security measures into internal platforms, and in turn, the software development process, is crucial for safely delivering new applications. Security-driven platform engineering includes a zero-trust framework, DevSecOps, automated threat detection, and more. [We cover security for platform engineering more in Chapter 5.](#)

Although many IT leaders initially focus on the technical aspects of platform engineering, it's only one piece of the puzzle. Successfully scaling platform engineering requires rethinking the role of people, processes, and technologies within the organization.

In this chapter, we'll cover best practices for scaling platform engineering, and take a look at some of the most relevant metrics for measuring a successful internal platform implementation.

BEST PRACTICES FOR SCALING PLATFORM ENGINEERING

1. Understand Developer Needs

The most successful platform engineering teams take a [product-centric approach](#) when building out new internal platforms. This involves conducting surveys and workshops to understand developer needs, and then launching a minimum viable product that aligns with existing development workflows.

2. Foster a Culture of Collaboration

Platform engineering needs to collaborate closely with development, operations, infrastructure, and security teams to deliver useful, reliable, and safe internal platforms. Sharing insights across these various functions also helps each team work towards unified goals to improve software delivery processes and application quality.

3. Prioritize Automation

Automation should be the standard for most aspects of software delivery, including deployment pipelines, infrastructure provisioning, and security monitoring. This minimizes errors and accelerates development velocity, leading to greater value for users and customers.

4. Invest in Training

Platform engineering teams should aim to give developers more than just tools and shared services by educating them on best practices. Internal portals can include training programs and documentation that help developers improve their technical skills and further encourage them to integrate platforms into their existing processes.

5. Continuously Improve

The platform engineering team should strive to continuously improve software development processes. This means incorporating feedback from developers and insights from platform usage data into future iterations of internal platforms to increase their usefulness and minimize waste.

MEASURING THE SUCCESS OF INTERNAL PLATFORM IMPLEMENTATIONS

Assessing the impact of internal developer platforms is crucial for the growth and adoption of platform engineering. The specific metrics you choose to measure may vary depending on your organization's goals and the nature of your software development projects, but the common themes are developer productivity and software quality.

Here are some key measurements related to platform engineering:

- **Time to market:** The time it takes to deliver software features or applications. A shorter time to market for software changes should be a primary goal and outcome of platform engineering.
- **Feature deployment frequency:** How often new features are deployed to production. A higher deployment frequency indicates development teams are using internal platforms to more efficiently deliver value to users.
- **Deployment success rate:** The percentage of deployments without any errors or downtime. Internal platforms should lead to greater reliability and higher deployment success rates.
- **Lead time for changes:** The time it takes for a code change to go from idea to production. A more efficient development process — as a result of platform engineering — should lead to faster code changes and lower lead times.
- **Error and incident rates:** The frequency and severity of errors, incidents, and outages in production. Platform engineering should help reduce the number of issues and their impact.
- **Developer productivity:** Metrics like lines of code written, bugs fixed, and tasks completed can help measure individual developer productivity. An effective internal platform that improves the developer experience should also noticeably improve developer productivity.

Real-World Use Case

AHEAD architected, designed, and deployed platform engineering for a Fortune 250 energy company, working with them to build the business use case strategy and providing the architectural guidance needed to re-imagine self-service in their environment. Following an 'everything as code' mentality, we improved governance, security, compliance, and time to market for their AWS infrastructure by creating patterns and practices needed for wide-scale modernization and migration. AHEAD also worked with the customer on organizational change management to upskill existing staff, train end-business consumers, and educate business leaders for their applicable roles. The patterns created by the platform engineering team enabled multiple energy verticals to deliver specific services for AI Virtual Assistance, Edge/OT compute, and modernization of app workloads to containerization and cloud-native architectural patterns.

- **Code quality:** Metrics like code coverage and code completion rate can indicate whether code quality is being improved or maintained. Platform engineering teams should give developers the tools they need to reduce the introduction of new bugs and accelerate the delivery of high-quality code.
- **Security and compliance:** Metrics like the number of vulnerabilities discovered, their risk level, and remediation times can give organizations insights into security and compliance. By integrating security measures into internal platforms and adopting policy as code, the platform engineering team can reduce the amount and severity of vulnerabilities and compliance violations.

3 How to Develop Platform Engineering Teams

The skills and roles of traditional infrastructure and operations teams do not enable efficient platform delivery, forcing IT leaders to rethink their organizational and staffing strategy to develop new platform engineering teams.

In this chapter, we'll look at the essential roles and skills required for effective platform engineering teams, and strategies for creating and expanding resources to support them.

THE ESSENTIAL ROLES & SKILLS WITHIN PLATFORM ENGINEERING TEAMS

Traditional infrastructure teams are organized into independent technology siloes, making it difficult to build and deliver unified platforms. IT leaders should consider investing in dedicated roles and resources that can work cross-functionally with different infrastructure, operations, and software teams.

The core skills of a successful platform engineering team include:

Product Management Skills

Internal platforms need to be viewed as products to ensure they meet developer requirements and align with overall business goals. The platform owner, and to some extent platform architects, should bring product management skills to the platform engineering team.

Software Engineering Skills

Many infrastructure and operations teams lack the software engineering expertise required to work with modern technologies like APIs, infrastructure as code, cloud-native infrastructure, and more. Platform engineers should be largely responsible for bringing this software engineering expertise to the platform engineering team.

In order to build out the skills necessary for platform engineering, IT leaders will need to consider new roles to fill the gap:

Platform Owner

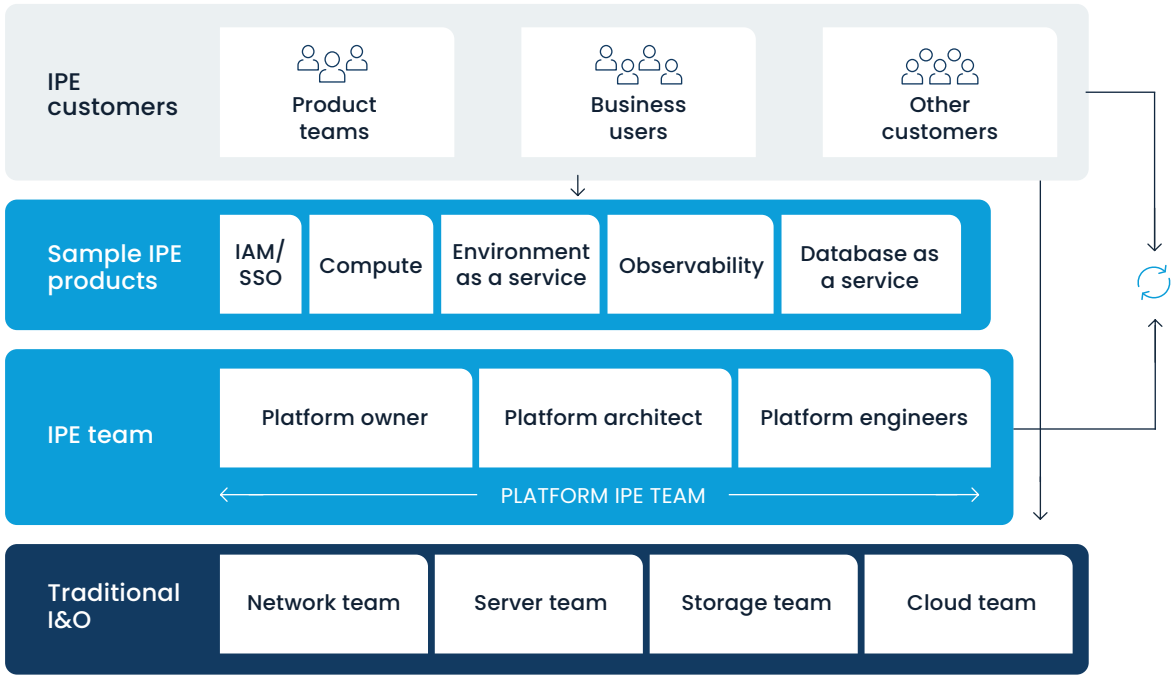
Similar to a product owner, the platform owner is responsible for an individual platform and its end users. This role requires close collaboration with developers and other platform users as well as the ability to lead the architects and engineers to execute on a platform strategy.

Platform Architect

The platform architect is responsible for developing and managing platform roadmaps based on user feedback and other stakeholders. This role requires a mix of technical and non-technical skills to translate user feedback into future infrastructure projects.

Platform Engineer

Infrastructure platform engineers are responsible for the delivery and operation of internal platforms. This role requires leveraging strong software engineering skills to build, integrate, and support new internal infrastructure platform capabilities.



STRATEGIES FOR CREATING & EXPANDING RESOURCES FOR A PLATFORM ENGINEERING TEAM

The competition for technology expertise is fierce, which means IT leaders need to consider new ways to attract and retain talent, develop new skill sets from within, and leverage external service providers.

Here are some strategies for creating and expanding infrastructure and operations teams to build out a platform engineering function:

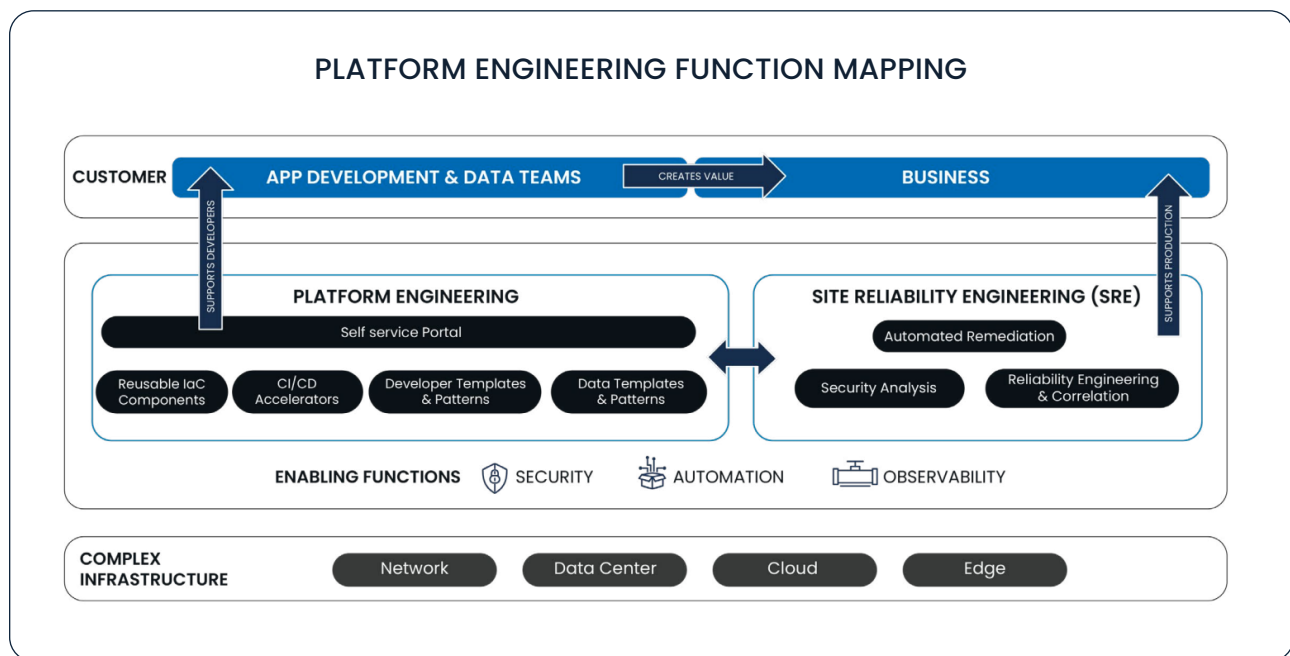
- 1 Hire software engineers with a passion for platform engineering** to bring crucial software engineering skills to platform engineering teams. Recruiting software engineers with an interest in platform engineering helps ensure they will be motivated to adopt the unique skills required to build and operate infrastructure platforms.
- 2 Partner with external providers** to accelerate the rollout of new platform engineering initiatives. There are external consultants with expertise in platform engineering that can bring their experience from other projects and client engagements, and provide access to niche skills otherwise difficult to obtain.
- 3 Collaborate with application development teams** to better understand technical skills gaps and reduce duplication of skills when investing in new engineers. A collaborative approach with software teams can also help the platform team leverage existing expertise to augment their efforts.
- 4 Augment with part-time or freelance resources** to overcome budget constraints while still delivering platform capabilities. This can be helpful to build out the specific functionality of an internal platform that does not require a dedicated full-time employee.
- 5 Invest in training programs** to keep teams continuously learning and engaged long term. By providing compelling incentives and avenues to develop new platform engineering skills, IT leaders can adapt their team to changes in the technology landscape and evolving user requirements.
- 6 Develop outreach programs with higher education** to build a pipeline of technology professionals with up-to-date skills and knowledge. This includes recruiting recent graduates from these institutions as well as implementing internship programs to connect with upcoming talent.

4 How to Develop Platform Engineering Teams

Although cloud computing and microservices have delivered enormous benefits, they've also led to complex, distributed systems. It's much harder to ensure the scalability, security, and high availability of modern applications which rely on underlying infrastructure services spanning many different cloud servers, virtual machines, and containers. Two key aspects of dealing with these obstacles are observability and site reliability engineering (SRE).

Observability provides a comprehensive view of system behavior and application health by analyzing metrics, logs, and traces. These insights enable proactive issue resolution and performance optimization for both applications and infrastructure.

SRE focuses on creating reliable, scalable systems and helps balance innovation with system stability. As part of this effort, SRE teams leverage observability tools and practices to better understand system and infrastructure performance.



In this chapter, we'll look at the benefits of observability and SRE in platform engineering, and how integrating these methodologies can lead to the delivery of applications with better performance and reliability.

THE BENEFITS OF OBSERVABILITY IN PLATFORM ENGINEERING

Observability involves monitoring and analyzing the performance of the platform itself and the applications it hosts to drive development and operations decisions. By providing valuable insights about a platform's functionality, observability can greatly improve reliability, performance, and overall system health.

- **Proactive Issue Detection:** Observability equips platform engineering teams with tools and methodologies to actively monitor and analyze system data in real time. This helps teams detect anomalies or deviations that may indicate potential issues or system degradations.
- **Faster Troubleshooting & Root Cause Analysis:** Observability data can be analyzed to detect patterns and automate system repairs, saving significant time and resources. AI and machine learning can also sift through more data and uncover more insights than a limited human engineering team.
- **Improved Performance Optimization:** Observability offers detailed insights into the performance of different system components. By continuously monitoring metrics, logs, and traces, teams can identify performance bottlenecks, inefficient resource usage, and opportunities for optimization.
- **Enhanced End-User Experience:** Performance monitoring enables the organization to optimize systems for better experiences that drive user satisfaction. Proactive issue detection and troubleshooting also prevents outages that could impact user retention.

THE ROLE OF SRE IN PLATFORM ENGINEERING

Site reliability engineering (SRE) blends software development with operations, with a focus on automating tasks and ensuring system reliability. Observability also plays an integral role in SRE, serving as the foundation for understanding, maintaining, and enhancing system performance.

SRE aims to balance the need for innovation and stability, influencing how companies manage and operate their IT systems globally. Every new software change comes with risks, so it's important to have a structured approach for taking calculated risks when releasing new features.

Here are some key principles SRE has evolved to include:

- **Service level objectives (SLOs)** define specific, measurable service attributes like availability, latency, or throughput. SLOs set clear standards for the service users can expect, and serve as a baseline for SRE teams to uphold.
- **Error budgets** quantify the acceptable risk or failure rate that a service can handle while still meeting its SLOs. This helps teams balance new feature releases with maintaining service reliability.
- **Release engineering** supports developers and DevOps engineers in reliably deploying new services and features. This discipline uses specialized tools and practices like feature flagging and artifact repositories for consistent, repeatable release processes.
- **Distributed system monitoring** focuses on identifying and resolving system issues by tracking latency, traffic, errors, and saturation metrics. Monitoring is crucial for maintaining system health and performance.
- **Automation and tooling** help reduce or eliminate manual tasks like deployments and fixing failures. This makes processes more efficient and reliable, freeing up engineers to concentrate on higher impact tasks.

Integrating observability and SRE into platform engineering can help address the complexity and demands of modern software systems. These practices offer a framework for developing, managing, and optimizing scalable, reliable platforms that support continuous innovation and deliver exceptional user experiences.

Observability helps engineering teams manage system health and how well things are running while SRE ensures services work well and remain stable. Combining observability and SRE helps organizations guarantee their internal platforms are always ready to handle whatever comes next with confidence.

As internal platforms scale, challenges related to security vulnerabilities can become more pronounced. The alarming rise in cyber threats—with a staggering 104% increase in reported incidents in 2023—emphasizes the critical need for a unified and secure approach to platform engineering.

In this chapter, we'll provide tips for combining security with platform engineering, and cover the role security automation can play in delivering safer internal platforms.

TIPS FOR COMBINING SECURITY WITH PLATFORM ENGINEERING

Incorporating security into platform engineering processes can help organizations strengthen their infrastructure against evolving cyber threats. By collaborating with security teams during the development of new internal platforms, the platform engineering team can lay the foundation for a secure software lifecycle.

Here are some best practices for designing, developing, and operating internal platforms with security in mind:

Adopt a [zero trust](#) approach that assumes a breach until proven otherwise. This means verifying every access request by default and continuously validating trust for every user. A zero trust architecture can be built into the design of internal platforms to ensure that every infrastructure change request is evaluated.

Leverage cloud native technologies that provide built-in security features, such as encryption, identity management, access control, and more. Properly configuring these features ensures security is maintained across all cloud services and infrastructure components.

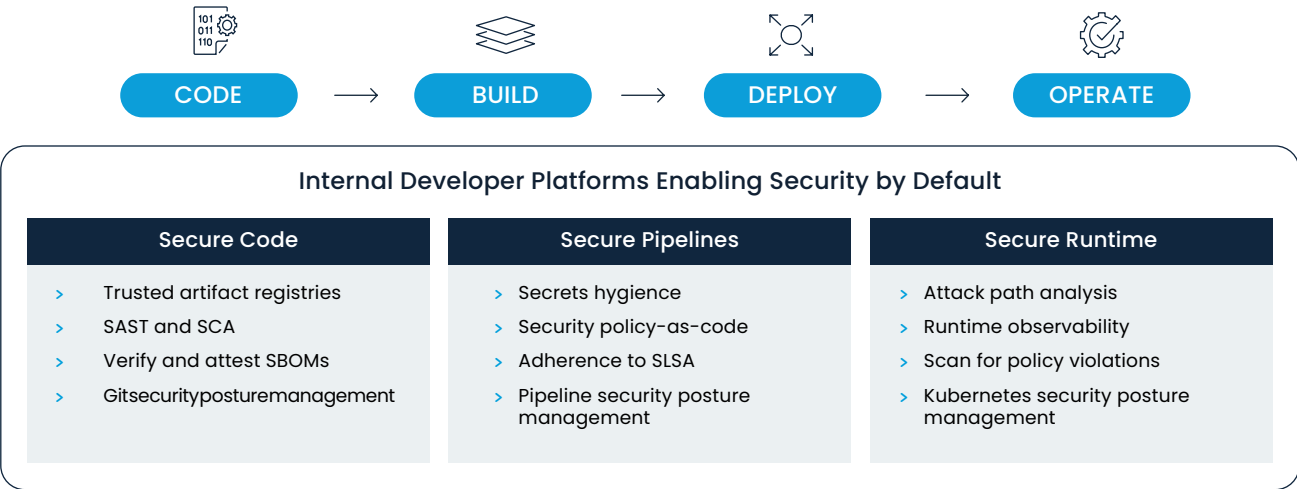
Implement DevSecOps principles that integrate security into every stage of the development process. This includes adopting static application security testing (SAST), dynamic application security testing (DAST), and other automated security tools. A DevSecOps approach also promotes a culture of shared responsibility among developers, security teams, and operations.

Employ a code-signing policy to verify the authenticity and integrity of programs, files, and software. This helps prevent unauthorized or malicious code from being deployed in production, and secures the software supply chain against tampering attempts.

Include a software bill of materials (SBOM) to provide a detailed inventory of software building blocks and dependencies within the software supply chain. Automated software composition analysis (SCA) tools can help identify third-party or open source components and the potential vulnerabilities they may contain.

Follow industry standards or frameworks, such as NIST, ISO, OWASP, and CIS Center for Internet Security. The cybersecurity industry is constantly evolving as new threats emerge, so it's important to regularly review and update security practices based on these standards.

REFERENCE MODEL FOR SECURITY CAPABILITIES IN INTERNAL DEVELOPER PLATFORMS



THE ROLE OF SECURITY AUTOMATION IN PLATFORM ENGINEERING

Automated security measures can be integrated directly into internal platforms to prevent developers from inadvertently violating security protocols or compliance standards. These guardrails can mitigate security risks without impacting developer workloads.

Here are some best practices for integrating security automation into internal platforms:

- **Automated user account provisioning and de-provisioning** is crucial for reducing configuration errors and minimizing the risks associated with outdated access rights and resources. Efficiently managing user accounts is a fundamental first step, and can have a substantial impact on security when combined with other measures like RBAC and IAM.
- **Role-based access control (RBAC)** policies are a way to implement granular control over user permissions based on certain roles and responsibilities, and to automatically apply consistent access management across the organization. This helps reduce the risk of unauthorized activities by ensuring that users only have access to the resources they need for their roles.

- **Permission auditing and monitoring** can help proactively identify overly permissive access rights or permissions that no longer align with certain user roles. Alerts about suspicious or unauthorized access can also allow security teams to respond quickly to malicious actions.
- **Automated Identity and Access Management (IAM) policy generation and enforcement** can streamline the process of defining and applying access rules across the organization. Automating the creation of IAM policies will guarantee accuracy and consistency in access control, reducing the risk of unauthorized access to certain resources.
- **Policy violation remediation** can be automated to quickly address and correct any deviations from established security policies. By automatically detecting policy violations like excessive permissions and quickly initiating corrective actions, organizations can further enhance their security postures.

Conclusion

Platform engineering is the key to overcoming the challenges of modern infrastructure and delivering high-quality software to customers faster than ever. Today's enterprises need to accelerate innovation to remain competitive, and this requires streamlined and automated development processes to boost efficiency and creativity.

An experienced enterprise solutions partner like AHEAD helps companies rapidly modernize their software delivery processes and infrastructure. AHEAD Platform Engineering as a Service provides the tools and expertise necessary to adopt platform engineering at scale. This includes templates and runbooks related to software deployments, infrastructure management, and more.

[Reach out](#) to us to learn more about our platform engineering solutions and services to better manage your modern cloud infrastructure and software delivery processes.

About the Author



Stephen D. King, Chief Architect, Digital Solutions

With 20+ years of experience working on digital transformation, software modernization, platform engineering and cloud enablement, Stephen has a track record of building and leading highly skilled technical teams by fostering a culture of innovation, collaboration, and excellence. Based in Houston, TX, he works to drive innovative technology solutions and architectures to create positive business impacts for customers across a wide array of industries.

AHEAD

Combining cloud-native capabilities in software and data engineering with an unparalleled track record of modernizing infrastructure, we're uniquely positioned to help accelerate the promise of digital transformation.

Contact Us

National Hubs

CHICAGO

444 W. Lake Street
Suite 3000
Chicago, IL 60606

NEW YORK

500 Fifth Avenue
Suite 1500
New York, NY 10110

ATLANTA

1117 Perimeter Center
W406
Atlanta, GA 30338

SAN FRANCISCO

2000 Crow Canyon Place
Suite 250
San Ramon, CA 94583