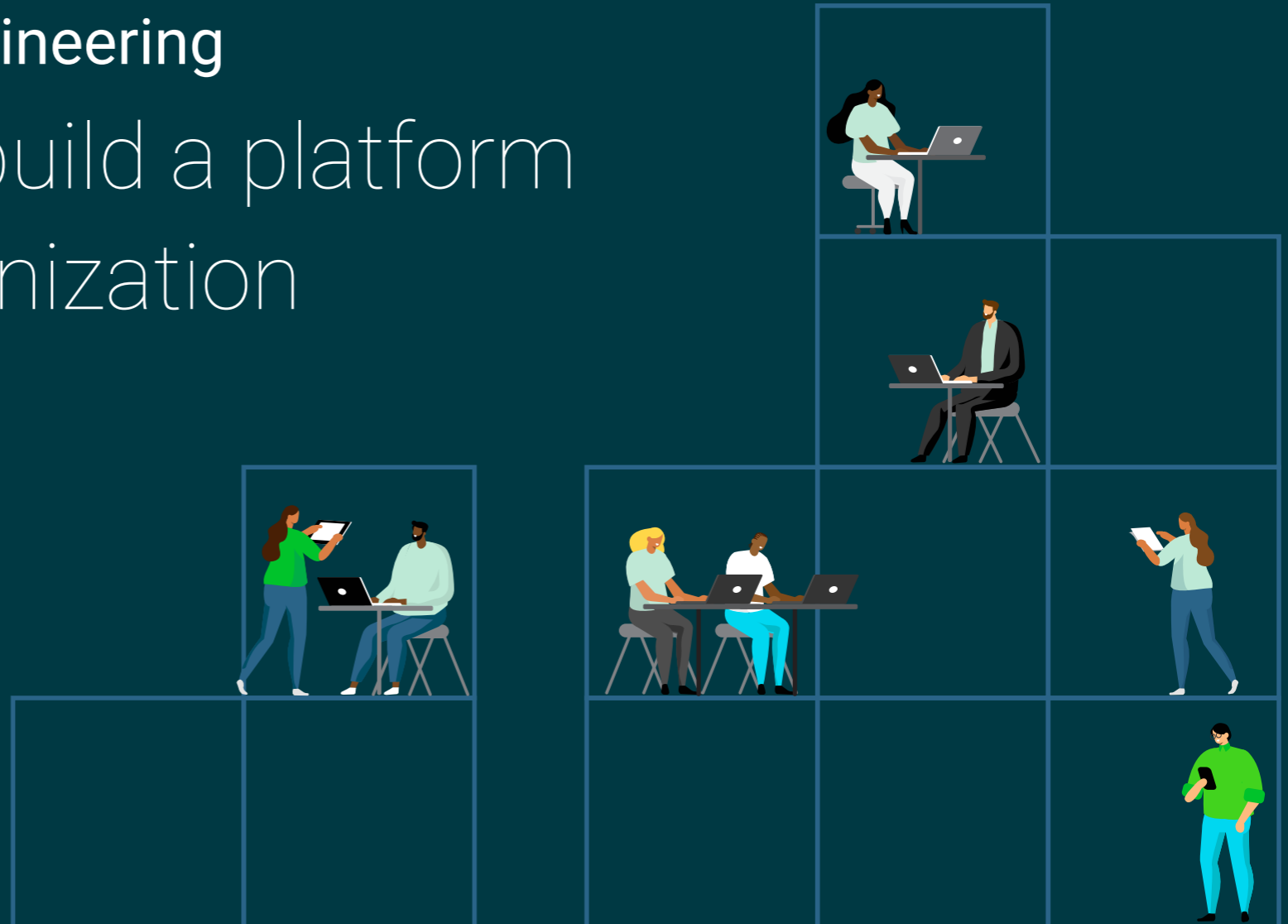



The path to platform engineering

Why and how to build a platform team in your organization





Before the growth of the DevOps methodology, teams working on an application's code were functionally separate from those responsible for deployment. This siloed operational model was notorious for causing botched releases, slow upgrades, and unhappy customers. In contrast, the DevOps approach has dramatically reduced inefficiencies and friction between software development and IT operations.

DevOps covers every phase in the application development lifecycle, from planning and building to deploying, monitoring, and iterating. It aggregates the tools and processes required for software development (dev) and IT operations (ops) and introduces new efficiencies throughout the application lifecycle. By giving development teams the autonomy to manage their own tools and applications, DevOps encourages faster development with better focus on delivering value to customers.

However, as organizations mature in their software delivery processes, applications often scale to hundreds or even thousands of interdependent services. In an environment with this much operational complexity, developer autonomy quickly comes into conflict with the business need for consistency, visibility, and organizational control. The cognitive load of managing a sprawling set of tools, services, and dependencies undermines the efficiency of the DevOps process and puts businesses at risk.

To alleviate some of this burden and better align development practices with business priorities, many organizations turn to platform engineering. In this guide, we'll look at what platform engineering is, what problems it solves, and how you can implement a platform team in your organization to effectively scale up your DevOps practices.

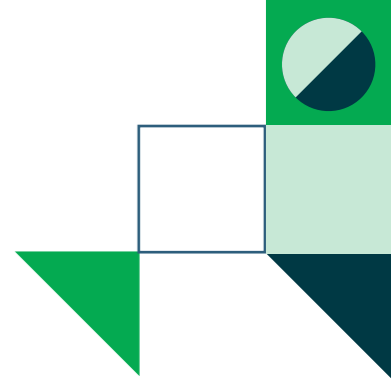
What is platform engineering?

Platform engineering is an emerging discipline focused on enhancing developer productivity by reducing the complexity and uncertainty of software delivery. It addresses some of the biggest challenges of doing DevOps at scale, including enforcing best practices and reducing the burden of managing complex tools and infrastructure across the application lifecycle.

Platform engineers build a standardized set of secure, compliant, and performant tools that developers can self-serve through a centralized interface known as an internal developer platform (IDP).

Platform teams view the internal platform as their product and developers as their customers. They work to optimize the platform to serve developer needs while fulfilling business requirements like security, regulatory compliance, and cost controls.

Organizations adopt the platform engineering approach under many different names – developer experience, developer productivity, systems and tooling, and infrastructure engineering, among others. A common misconception is that platform engineering is a replacement for site reliability engineering (SRE). In reality, these two disciplines focus on different areas. SRE improves application reliability by minimizing downtime and the possibility of cascading failures. On the other hand, platform engineering builds the infrastructure and services that streamline existing workflows and accelerate software delivery.



How platform teams optimize DevOps efficiency

Different software development teams have different needs. In a large organization, if each development team makes its own DevOps policies, the result is complexity, bottlenecks, and increased risk of security vulnerabilities or compliance violations. The solution is to shift some DevOps responsibilities to a platform engineering team.

The platform team evaluates the needs of DevOps teams across the organization to identify the tools, services, and workflows required to deliver quality software quickly. The team then works to make these tools available through reusable components and processes. This might involve building automated processes for setting up new Git repositories or provisioning test environments, creating configuration templates for continuous integration and delivery (CI/CD) pipelines, or building centralized solutions for managing secrets, authenticating users, and monitoring application performance.

Promoting the reusability of tools and configurations is a core objective of any platform engineering team. Reusability makes the application delivery process more efficient, reduces the resources spent on tools, and improves code consistency.

Developers can minimize time spent writing repetitive code and instead focus on innovation. Business leaders can rest assured that engineering teams are following approved processes that are secure, compliant, and cost effective.

Platform engineering allows teams to scale up existing DevOps processes to meet any level of demand without adding unnecessary overhead. Let's look closer at a few ways that platform teams help organizations optimize and mature their DevOps capabilities.

Building an internal developer platform

An internal developer platform (IDP) is a self-service layer that the platform team builds and maintains for developers. Developers can use the IDP to self-serve deployment pipelines, environments, logs, databases, and any component their applications need. The IDP promotes consistency, efficiency, and coordination by aligning infrastructure and tooling across the organization.

To build an IDP, platform engineers work closely with developers and business leaders to understand all the activities involved in the software development process as well as the organization's unique security, compliance, and budget requirements. They determine which tools and services will best serve their teams' needs as well as how to build efficient abstractions to reduce complexity. They then develop an optimal platform architecture to provide a scalable, reliable user experience.


The platform team integrates common tools and makes them available through the IDP, ensuring each development team has the right level of access, with just enough configurability to suit their unique requirements. Then, the platform team measures the impact of the IDP on developer performance and satisfaction to iterate and adjust as the development process evolves.

Establishing paved roads to reduce cognitive load

With an IDP, platform teams can provide a "paved road" for developers: an optimized set of tools and processes for accomplishing common tasks that can be accessed with minimal effort, often with a few clicks on a UI or a simple API call. This allows developers to eliminate repetitive tasks and avoid learning complicated technologies.

Importantly, developers are not always required to follow the paved road to deployment. If they have a legitimate need to use a technology not offered by the IDP, they are usually free to, as long as it passes all mandatory security and compliance checks. If over time the tool gains enough organic adoption in the organization, the platform team may consider incorporating it into the IDP. But until then, developers are responsible for configuring and maintaining the tool on their own, without the support of the platform team.





Simplifying, standardizing, and scaling processes

Platform engineering consider the needs of different teams so they can develop customized procedures and scale DevOps processes more efficiently. Rather than requiring every individual developer team to learn Kubernetes, CI/CD pipeline configurations, and application monitoring best practices, a platform team can bake these components into the IDP service offerings to achieve economies of scale across the organization.

An effective platform will also take into account common internal processes like onboarding a new developer. With proper documentation and automated workflows for assigning permissions and configuring development environments, platform teams can drastically shorten a new employee's time to productivity.

Monitoring metrics and KPIs

Platform teams are often responsible for tracking high level metrics across the engineering team to better understand how the organization is progressing toward its goals. These metrics may include team performance indicators like workflow durations, recovery times, throughput, or the failure rate for new changes. They may also include measurements of the availability and reliability of the internal platform itself, or user adoption and satisfaction scores to understand the impact of the platform on developer experience.

Cost optimization is always a concern for platform teams, so they will frequently measure platform costs relative to the number of users or services to make informed tooling decisions.

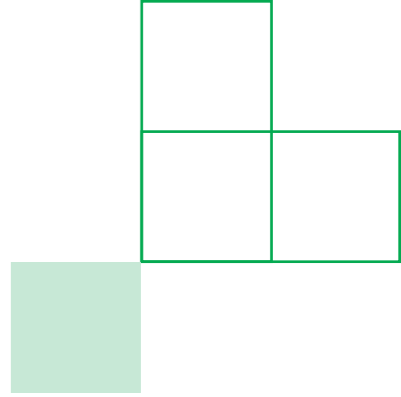
Platform teams encourage a culture of continuous learning and improvement. By tracking meaningful metrics and incorporating that data into their systems and decision making, they can not only improve developer experience and productivity but also have a noticeable impact on the organization's ability to achieve its objectives.

Optimizing team performance to deliver increased customer value

Ultimately, a platform team exists to enable development teams to deliver the maximum value to customers with the minimum expenditure of resources and effort. A successful platform will unlock team velocity and help developers ship new features faster and more reliably. It will reduce the risk of bugs, downtime, security breaches, and compliance violations that can negatively affect customer sentiment. And it will keep costs in check so that the organization can invest more in innovation.

All of these factors contribute to an increased flow of value through the organization and ultimately to customers. A faster, more agile organization will not only deliver a more pleasant user experience, but it will also be in a better position to incorporate and quickly respond to customer feedback, building improved loyalty and trust in the process.

Is your organization ready for platform engineering?



Platform engineering helps organizations overcome challenges in delivering software at scale.

While some consider the size of an engineering department to be the only reliable indicator of whether a dedicated platform team is needed, in reality, that is only one factor among many.

To determine whether your organization is ready to implement a platform team, first look closely at your business and product development practices to better understand the problems you are trying to solve and how the platform team can best address them. Here are some important points to consider.

Maturity level

To successfully introduce a platform engineering team, you need both a mature product and a mature software delivery practice.

A mature product will have an established market and a clear value proposition that your team can iterate against. When your organization has a clear understanding of how the product meets the needs of a specific user base, development teams

can be precise in delivering features that maximize user value. At this point, a platform team can play a crucial role in unlocking team productivity so that your organization can be responsive to user needs and innovative in delivering updates and feature enhancements.

Likewise, organizations should have mature software delivery processes already in place so that the platform team can build structures that reinforce positive patterns and prevent inefficient, risky, or costly behaviors. Features of a mature software delivery practice include robust testing; small, frequent changes to the main branch; loosely-coupled architectures; and use of continuous integration and delivery pipelines to automate time consuming manual processes.

Culture

Don't rush the transition from ad hoc DevOps to a centralized platform team. Take the time to understand your organization's goals, challenges, and the changes that are required. Any organization with a strong culture, which may have evolved over years or even decades, will probably be skeptical of change. Platform engineering presents a new way of doing things — developers who are comfortable with the old ways are often hesitant to change. To get the most value from a platform team, first nurture a culture that values collaboration, continuous learning, and finding new ways to improve and accelerate the development process.

Organization size

The size of your organization can have a significant impact on the benefits that a platform team can provide. A small organization with a handful of developers working on a monolithic codebase will likely not have a significant need for standardization, as there will be minimal overlap in responsibilities across the team. Teams of this size often won't have resources to invest in a centralized platform team and will instead choose to invest in building out the core functionality of the application.

Larger organizations typically have more complex software architectures and specialized teams that require significant amounts of infrastructure to support. At this level of complexity,

a platform team can be a valuable investment as it will improve coordination between teams and create a more consistent, secure, and scalable development process.

Organization type

The journey to platform engineering can be different depending on whether you are a cloud-native startup working to scale up your growth or a large legacy organization seeking to modernize your development practices.

In most startups, the software delivery system constantly evolves to meet the needs of the applications at the minimum possible cost. After finding product-market fit, an organization usually needs to scale significantly and stabilize software development efforts. In these circumstances, it can be helpful to start exploring the idea of a centralized platform to improve consistency and keep costs in check.

Many large legacy organizations have well defined products and established customer bases but rely on outdated tools and processes that make it difficult to compete with the speed and agility of smaller cloud-native organizations. A platform team can help facilitate a digital transformation by building stable, reliable infrastructure to integrate existing systems with more recent tools and services. The platform team can also ensure that any new tools meet strict security and compliance requirements so that the organization can improve its development velocity without increasing its risk exposure.

What makes a good platform team great

A platform engineer solves DevOps challenges, so they need to understand DevOps objectives and processes. Let's look at the skills needed to form a strong platform team.



Communication and collaboration

A key platform team responsibility is working closely with developers and business leaders to understand the unique processes, requirements, and challenges faced by different stakeholders in the development process. Incorporating user input and feedback in the platform design will increase usage rates and improve developer experience and productivity.



Process and workflow design

A platform team should have the skills to break down tasks into smaller processes. They should also be capable of creating independent, automated workflows for development and deployment-related activities. A platform team should be capable of creating workflows and infrastructure that can easily scale. For instance, a workflow should be able to accommodate more developers creating more deployments without sacrificing stability or increasing complexity. It should also be efficient in its resource usage and scale down when feature development is completed or priorities change.



Analysis and optimization

DevOps operations generate large amounts of data that can provide substantial insight into optimization opportunities. A platform team should be able to analyze this data effectively. For instance, they should have easy access to automated test logs to find ways the configurations could be adjusted to accelerate these tests.



Infrastructure as code (IaC)

Platform engineering is a layer between software developers and the IT infrastructure. It provides abstractions and workflow templates so developers can build, deploy, and maintain software without worrying about how to manage the underlying infrastructure. That's why a platform engineering team must have practical experience with different IaC tools, such as Terraform, Kubernetes, Pulumi, Ansible, Chef, and Puppet.



DevOps tools and methodologies

The platform engineering team brings together a set of tools and workflows to increase developer productivity. Therefore, they must have a solid understanding of the different underlying technologies embedded in DevOps processes. For example, they need knowledge of the technologies involved in containerization, automation, security, testing, and observability.

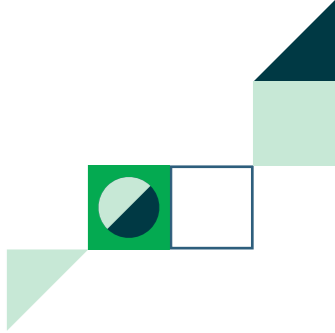
Each team within a DevOps setup has different needs, depending on the tasks for which they are responsible. A platform team meets these needs by identifying tools and creating separate workflows that incentivize DevOps best practices. To do this, the platform team also needs a deep understanding of DevOps culture and methodologies, as well as how to monitor and measure DevOps team success.



Budget management

Platform teams are often tasked with making financial decisions, such as which tools to purchase to optimize engineering costs. A platform engineering team that understands the cost implications of their decisions can make the most effective use of the organization's budget and keep tooling and infrastructure costs at a minimum.

Attracting the right skills to a platform engineering team



Here are some things your organization can do to attract the right engineers to your platform team.

Set clear objectives

Talented engineers relish a challenge, but to get them to rally behind the team's mission, you need to set clear objectives. Are you focused on improving team velocity, increasing application security, scaling your user base, or controlling costs? This also ensures you understand the value that the platform team will bring to the organization and the types of skills you should seek out.

Set realistic timelines

Creating an efficient platform can take many months – or even years. Setting a realistic timeline will make a platform engineering team more attractive to talented engineers. Breaking down the objectives into milestones gives the team a sense of accomplishment, which is crucial to retaining and attracting talent.

Embrace diversity

A platform team performs coding, data analysis, and developer surveys. The diversity in skills will make the team attractive to engineers who enjoy learning from other team members. If a team does not have any engineers who enjoy data analysis or talking to people, for instance, the platform they create will not be very effective. A lack of diversity in your team also makes it unattractive to other talented people with different skills.



Get the right tools

Provide a platform engineering team with all the tools they need to make it an attractive opportunity for skilled engineers. Platform engineers will need access to advanced tooling to manage automated testing, cloud based infrastructure, container orchestration, networking, security scanning, monitoring and alerting, and more.

When the team has the right tools, they can focus their efforts on building reliable components and workflows that meet the needs of software developers.

Empower your platform team with best in class CI/CD

To empower platform engineers, you need to provide them with the right tools: easy to use, easy to integrate with the latest technologies, and powerful enough to handle complex tasks. At the same time, you need the additional security, control, and visibility that will allow you to scale responsibly and protect your customers, employees, and investors from risk.

A robust continuous integration and delivery pipeline is a critical part of any platform team's toolchain. It provides an automation engine that can trigger templated workflows to build, test, and deploy code in a frictionless, scalable way. At an even higher level, a mature CI/CD pipeline can serve as a control plane for codifying, enforcing, and measuring business priorities in development.

At CircleCI, we provide a number of features that help platform teams automate and streamline critical development tasks.

CircleCI orbs are portable, reusable snippets of configuration code that you can use to easily integrate popular tools and services in your workflows or share common processes across projects. With **config policies**, platform teams can also set global rules to ensure every pipeline in the organization includes the required tools, processes, and security checks.

The **Insights dashboard** gives platform engineers a graphical view of performance so they can quickly identify areas for improvement and optimize pipelines for maximum efficiency. It provides time-series data on long-running and frequently failed workflows, resource usage, and credit consumption so that platform teams can ensure their pipelines are delivering the speed developers demand and the cost effectiveness the business requires.

CI/CD saves developers time and improves the development process. It forms the core of a platform engineering toolkit that enables self-service, reusability, and other key principles of platform engineering methodologies. To learn more about how CircleCI can provide the foundation for your internal developer platform, [sign up for a free account](#) or [contact us](#) to design a plan that meets the specific needs of your team.

Conclusion

DevOps has transformed how we approach software development, increasing agility and easing collaboration. However, managing all that complexity has become a major challenge for many organizations. This has led to the rise of platform engineering, a discipline focused on accelerating software development by empowering developers with self-service workflows that are simple, scalable, and secure.

Platform teams simplify and bring increased efficiency to tasks previously assigned to DevOps teams. To get the most out of a platform team, an organization needs a clear product vision, mature DevOps practices, and a culture of continuous learning and improvement.

Once in place, an effective platform team needs a good understanding of the application, development teams, and business requirements to build out a platform that will enable the organization to deliver better software, faster.

To maximize the efficiency of a platform team, you need to equip them with the right tools. With industry-leading features, security, and support, CircleCI can significantly increase the effectiveness of your platform team. Learn more today at circleci.com.