Platform Engineering explained to decision-makers

Luc Legardeur, CEO of WeScale

wescale

Table of contents

Introduction

At the dawn of 2025, global economic and geopolitical uncertainties are forcing CIOs to implement a drastic policy of cost rationalization. Every investment must be decided based on expected ROI.
After the boom years of free post-COVID money, it's never been more important to do more with less because ,despite everything, innovation, even in times of crisis, is a constant necessity for all companies in all sectors.

As a decision-maker in French Tech, I will try to explain in this eBook why WeScale believes that Platform Engineering is one solution for making the most of your investments in AI, Data, and Security, to name but a few.

And we're not the only ones in the world promoting this approach.

Platform Engineering is becoming increasingly popular, and offers several advantages for organizations.

Here are a few figures to illustrate that this is not a fad but an underlying trend.

Gartner forecasts that by 2026, 80% of large organizations will have set up Platform Engineering teams, compared with 45% in 2022.

According to a 2023 survey of practitioners in the Platform Engineering, DevOps, Cloud, and cybersecurity communities, 83% of organizations are in the process of adopting Platform Engineering.

Motivations for adopting Platform Engineering vary from one organization to another.

Below is a brief, non-exhaustive list:

- Increase the productivity of software engineering teams.
- Improve software quality.
- Reduce deployment times.
- Make applications more stable.
- Reduce the number of bugs.
- Reduce the overall TCO (Total Cost of Ownership) of developed applications.
- Reduce the TTM (Time To Market) of digital products.
- Reduce the risk of security breaches.
- Enable developers to be multi-skilled.
- Improve internal and external customer satisfaction.
- Reduce time spent on software maintenance.
- Reduce the cognitive load on development teams.
- Reduce the time spent implementing security policies.
- Improve application portability.

If this introduction has aroused your curiosity about the benefits of Platform Engineering, I encourage you to read on.

As you read through the various chapters of this eBook, which I had a great deal of pleasure writing, you'll discover the convictions that the men and women of WeScale have forged by supporting our customers, as well as feedback from Platform Engineering practitioners who have implemented this new paradigm around the world.

If you would like to find out more, please do not hesitate to contact us by following this link:
www.wescale.fr/contactez-nous

I hope you enjoy your reading.


Adoption of Platform Engineering
17%
Not envisaged
20%
Fully adopted
19%
Plan for adoption
44%
In the process of adoption

What is Platform Engineering?

Platform Engineering is the discipline of designing, building,, and maintaining a self-service platform called an IDP (Internal Developer Platform) and other bricks designed to serve an organization's developers.

This platform provides them with standardized tools, workflows, and environments to simplify and accelerate the software development and deployment cycle.

The objectives of Platform Engineering are as follows:

Standardize processes and tools

By centralizing best practices, configurations, and workflows, the initiative ensures greater consistency across the organization by providing a single framework that reduces duplication of effort.

Increase productivity

The productivity of developers is one of the goals pursued by digital product clients.

Platform Engineering enables:

- Automation of repetitive tasks linked to integration, deployment, and observability.
- Reduce application development times by reducing the number of repetitive tasks.
- Encapsulation of technical infrastructure details such as containers, CI/CD (Continuous Integration/Continuous Deployment) chains and network configurations behind a simple interface for developers.

Improve DevEx (the developer experience)

A consistent, intuitive interface gives developers more straightforward access to the services, tools, and environments they need. These services are set out in a ready-to-use Service Catalogue.

This approach reduces developers' cognitive load, freeing them from complex infrastructure-related tasks they don't necessarily control.

Internal Developer Platform

Platform Engineering is based on a fundamental building block, an IDP (Internal Developer Platform), whose characteristics are as follows:

Self-service: developers can operate resources such as creating a cluster, deploying an application, or configuring a CI/CD pipeline without disturbing the infrastructure team.

Complexity abstraction: infrastructure tools such as Kubernetes or databases are configured and ready to use via simplified interfaces.

Built-in observability: Developers can easily monitor performance and detect potential problems by analyzing metrics without necessarily having to dive deep into the developed code.

Customizable and modular: An IDP adapts to the organization's specific needs yet is flexible enough to evolve.

An example of an IDP use case

One organization uses Kubernetes to orchestrate its containers, and Jenkins is used for CI/CD implementation. Each developer must learn how to configure these tools without a platform.

IDP then provides these developers with :

- Pre-configured pipelines.
- An abstraction of Kubernetes configurations.
- A GUI to deploy an application at the click of a button.

This allows developers to focus on code rather than infrastructure management.

TAKE AWAY

Platform Engineering is important for the following reasons:

Reducing complexity

With multi-cloud environments, more and more microservices, and complex CI/CD pipelines, developers can't master everything or simply don't have the time to address everything.

Reduced time-to-market

Standardizing tools and workflows reduces errors and speeds up deployment. Platform Engineering reduces TTM (Time to Market).

Cost reduction

A well-designed, scalable platform reduces the time spent on manual or repetitive tasks, optimizing human and technical resources.

What are the benefits of Platform Engineering?

As I briefly mentioned in the introduction to this eBook, Platform Engineering offers several key benefits for company decision-makers.

They are linked to improving operational efficiency, reducing costs, and accelerating innovation.

Here are the main advantages of this approach for decision-makers.

Lower infrastructure costs

By centralizing and standardizing development and operating environments, your company can reduce the costs of infrastructure, resource management,, and human error, which have a far from negligible financial impact.

Optimizing resources

Your teams can better manage hardware and software resources by sharing common platforms, avoiding duplication and improving operational efficiency.

Improved productivity for development teams

Development teams can concentrate on creating value rather than wasting time managing infrastructure or tools. This means that digital products can be put into production more quickly.

Process automation

By implementing standardized, automated tools and processes for deployment and configuration management, teams gain in productivity, agility and scalability.

Standardization and governance

A unified platform enables defining technical and governance standards, reducing the risks associated with security, compliance, and version management.

Improved safety

Decision-makers can apply consistent security practices across all environments, reducing vulnerabilities.

Agility

Thanks to a flexible, modular architecture, Platform Engineering enables companies to adapt quickly to new market demands or technological developments.

Scaling

A well-built platform can easily evolve to meet user, data, or service growth without major overhauls. By standardizing the approach, it can support more complex projects without adding technical debt.

Innovation

Development teams can experiment and innovate faster, and spend more time solving complex business problems rather than managing technical aspects.

Inter-team collaboration

Platform Engineering creates a common base between development, operations, and security teams, which significantly facilitates collaboration and coordination between these often dispersed teams. Companies that have implemented Platform Engineering report greater fluidity, better teamwork, and improved overall performance (KPIs - Key Performance Indicators - to be read further).

TAKE AWAY

Summary of profits

Category	Key benefits
Developer productivity	Less repetitive tasks, more time for coding
Standardisation	Fewer errors, faster adoption of technologies
Time-to-Market	Accélération des cycles de développement et déploiement
Operating costs	Acceleration of development and deployment cycles
Reliability	Safer deployments, increased observability
Collaboration	Reduced dependency between teams, easier communication
Innovation	Environments for rapid experimentation and adoption of new technologies
Developer Experience	Reduced cognitive load, greater developer satisfaction
Scalability	Large-scale team and infrastructure management

What is the ROI elements of Platform Engineering ?

As with any new initiative, decision-makers need tangible evidence to back up the claims made by promoters like WeScale. Below, is a simple, quantified methodology, the fruit of our experience, and elements gleaned from various sites, books, blogs, and feedback. They are far from exhaustive but provide a good basis for transposing them to your organization. They are presented by category.

Methodology for assessing ROI in your organization

The figures below are to be collected over a given period, for a given digital product, for a given department, or for your company as a whole.

Analysis of current costs before implementation

- **Developer and Ops costs (DevCosts):** Costs of developers' loaded salaries for the entire scope observed.
- **Cost of incidents (IncidentCosts):** Frequency in number multiplied by the average cost per incident, excluding costs related to loss of image and, in general, anything not immediately quantifiable (see case studies). This figure can vary considerably depending on the criticality of the application.
- **Infrastructure costs (InfraCosts)**: Run's infrastructure costs for the scope observed, plus the cost of non-production infrastructure - development, acceptance, and pre-production (see case studies).

Investment analysis for setting up the platform (Investment)

- **DesignCosts**: Internal and external time and costs devoted to design and implementation.
- **License costs: (AddLicenceCosts)**: Additional costs that may be required.
- **Training costs (TrainingCosts)**: hours or days spent upgrading team skills.

Assessing the potential impact of Platform Engineering (Impact)

- **ImprovedProductivity%**: Measuring the % of time freed up for high value-added tasks, i.e., development time and deployment time saved (see case studies).
- **Incident reduction (ImprovedIncidentsCosts%)**: Observe the % reduction in incident costs after implementation.
- **ImprovedRessources%**: Estimate infrastructure cost savings in % (see case studies).
- **Time to Market reduction (TTMReductionGain)** : Additional revenue generated by its reduction.

Calculation of ROI (Return on Investment) in %

This formula is provided as a guide and is as follows.

ROI in %

$$\frac{(\textbf{ImprovedProductivity\%} \times \textbf{DevCosts} + \textbf{ImprovedIncidentsCosts\%} \times \textbf{IncidentsCosts} + \textbf{InfraCosts}) \times \textbf{ImprovedRessources\%} + \textbf{TTMReductionGain} - (\textbf{DesignCosts} + \textbf{AddLicenceCosts} + \textbf{TrainingCosts})}{(\textbf{DesignCosts} + \textbf{AddLicenceCosts} + \textbf{TrainingCosts}) \times 100}$$

You can also use our online calculator.

Factors influencing ROI

Size of technical team: the larger the team, the greater the productivity gains.
Infrastructure complexity: Complex environments generally benefit more from standardization.
Current level of automation: If DevOps practices are already advanced, savings may be lower.
Technology maturity level: Companies already adopting the Cloud or Kubernetes often see accelerated gains.

Real-life case studies and statistics

Humanitec

Companies implementing an Internal Developer Platform (IDP) have increased productivity by between 20% and 40%.
The Platform paid for itself in 6 to 12 months.

Spotify

Spotify has seen a 30% increase in developer productivity thanks to a centralized developer portal (the open-source Backstage product) with less time spent on non-developmental tasks.

Coûts des incidents en Europe

Yves Grandmontagne, a journalist with IT Social, reports that 65% of European organizations estimate the total cost of an incident to be € 115,000.

Gartner

In 2014, Gartner reported that 33% of companies lost between €1 and €5 million for one hour of platform downtime.

Microsoft

Microsoft reported in 2023 that the reduction in resource provisioning time was around 40%, thanks to the implementation of Platform Engineering.

Twillio

This American company, specializing in unified communications, saw a 30% reduction in infrastructure costs by implementing Platform Engineering while increasing its traffic by 25%.

Netflix

Netflix, with its Chaos Engineering practices inherited from the creation of Chaos Monkeys, estimates that it has reduced incident-related costs by 40%.

Average annual development and production infrastructure costs

These data have been collected from software publishers and customers facing these issues. Of course, these metrics are subject to adjustment.

Average annual development infrastructure costs for a team of 10 developers.

- Compute: €18,500
- Storage: €3,600
- Databases: €9,600
- Networks: €6,000 (depends on data exchange between environments)
- CI/CD: €8,520
- Test automation tools: €6,000/year
- Load Testing Tools: €2,400
- Monitoring tools: €3,600
- Safety Tools: €3,000

Over 12 months, the total annual cost of development infrastructure for a medium-sized software program is €55,000.

Run's average annual infrastructure costs for medium-sized software.

- A medium-sized software package has the following characteristics for our calculation.
- Active users: between 5,000 and 50,000 per month
- Competing users between 500 and 1,000
- Network calls: between 100,000 and 1 million per day
- Web or mobile application using authentication, storag,e and APIS for customer interactions.
- Uptime SLA: from 99.5% to 99.9% availability.
- Latency: less than 1 second.
- 10-20 virtual machines
- Containerized environments: Kubernetes or Docker.
- Capacity: between 1 TB and 10 TB.
- Managed database from 100GB to 500 GB.
- Network bandwidth is used from 1TB to 10 TB per month.

The associated costs are:

- Compute (Virtual Machines or Containers): € 120,000.
- Storage: € 12,000.
- Database: € 18,000.
- Network: € 12,000.
- Monitoring and Observability: € 6,000

Over 12 months, the total annual cost of Run's infrastructure for medium-sized software is €168,000.

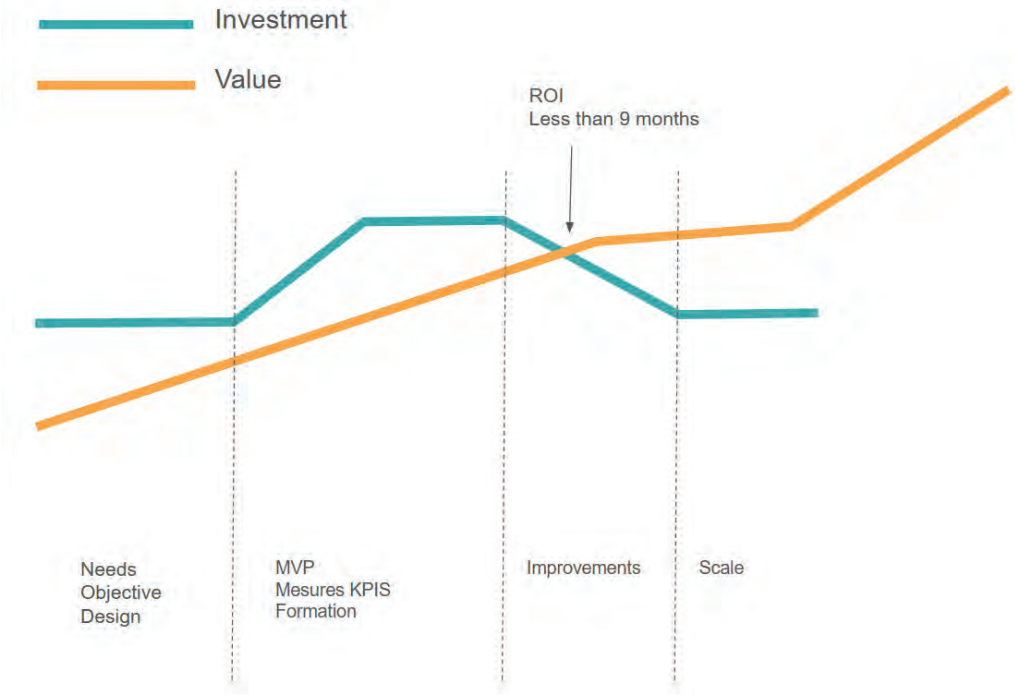
You can use the websites of the 3 main Cloud Providers to adjust some of the figures.

- AWS Pricing Calculator
- Google Cloud Pricing Calculator
- Azure Pricing Calculator

TAKE AWAY

The elements of ROI depend strongly on the organization, its sector of activity, the complexity of its infrastructures, and the level of DevOps practices already in place.
They may be questionable.

As the diagram below shows, at WeScale we see that the ROI of a Platform Engineering approach is achieved in less than 9 months.



Who decides to implement Platform Engineering in organizations?

Implementing a Platform Engineering approach involves several players in the organization. Like any new initiative, its sponsorship must be present at several levels. It is generally the fruit of strategic and collaborative reflection between the various technical and operational leaders.

Below is a summary of the stakeholders involved, their responsibilities in the decision-making process and their respective roles.

Technical leadership (CTO, CIO)

Liability

Decide on major technological orientations and strategic priorities, aligning them with corporate objectives.

Role

The CTO (Chief Technology Officer) and/or CIO (Chief Information Officer) identify the need to improve the efficiency of development teams and reduce operational complexity. Both ensure that this initiative supports digital transformation and innovation.

DevOps / Infrastructure team

Liability

This team integrates the challenges posed by the growing complexity of infrastructure management. It identifies current workflow limitations and their impact on developer productivity.

Role

This team and its leader propose solutions to centralize and standardize tools via an Internal Developer Platform (IDP), plan its implementation, and define the MVP (Minimum Viable Platform - not Product in this case) and subsequent steps for scaling it up.

(Technical) Product Managers / Product Owners

Liability

They play an essential role in identifying the needs of developers, who are the platform's end-users.

Role

Defines functional priorities and use cases for the platform.
Liaise between developers and the infrastructure team to design a platform adapted to everyone's needs.

Development teams

Liability

Developers describe the problems they encounter daily, such as the cognitive load associated with infrastructure management and the lack of standardization. They provide feedback on current tools and workflows.

Role

They participate in the MVP development to obtain a simplified, self-service platform as quickly as possible to address the most critical problems first.
Of course, development teams also take part in pilot phases to test proposed solutions.

Executive management (CEO, COO)

Liability

This leadership level validates the financial and human investment required to implement Platform Engineering.

Role

Le CEO/COO s'assure que cette initiative s'aligne sur les objectifs globaux de l'entreprise, comme la réduction des délais de mise sur le marché ou l'amélioration de la qualité des logiciels. Le budget d'investissement sur une Platform Engineering est validé à ce niveau.

Security team (CISO)

Liability

The CISO and his/her teams ensure that the platform complies with security and compliance best practices.

Role

The security team validates the chosen solutions for their robustness in terms of identity and access management, as well as system monitoring.

Typical decision-making process

Need identification

The need may be raised by DevOps teams, developers or technical leadership (CTO).

Analysis and Business Case

An in-depth analysis is carried out to assess the benefits, costs and resources required. The business case is then presented to senior management.

Strategic validation

The CTO or CIO validates the initiative as a strategic priority.

Planning and implementation

A dedicated team of developers, DevOps/Infrastructure team members, and (Technical) Product Managers have been set up to design and implement the platform.

TAKE AWAY

The decision to implement a Platform Engineering strategy is based on interdisciplinary collaboration, requiring strong technical leadership and alignment with the company's digital objectives.

Summary of profits

Role	Decision or contribution
CTO / CIO	Strategic decision and global validation
Équipe DevOps	Technical proposal and validation
Security team (CISO)	Implementation of security policies
Technical Product Managers	Identifying and prioritising needs
Dev Teams	User feedback and adoption
Board level	Budget validation and alignment with business objectives

At what scale is Platform Engineering implemented?

Platform Engineering is often implemented on a company-wide scale, but its implementation is incremental. Its scope and application can vary according to the needs, size, and complexity of the company.

Organizational and global scale

Triggering event

For large companies or organizations with multiple teams and departments (e.g., tech, product, QA, etc.), the goal of having a global approach is tempting, but it doesn't happen overnight. In general, the need arises when organizational standards are required to avoid duplication of effort and reduce complexity.

Features

The platform serves all the organization's development teams. It is designed to be modular and extensible, adapting to the specific needs of each team or project. Standardization and consistency are strongly emphasized throughout the organization.

For example

An organization with hundreds of microservices, where each team has to manage its own CI/CD pipelines and Kubernetes configurations. Centralized tools enable faster onboarding of recruits.

Domain, team scale

Triggering event

In medium-sized companies or for specific departments within a large organization. The needs of a team or domain (data engineering, DevOps, front-end applications) are different or specialized. Still, the need is identified for some of the reasons mentioned in the previous chapters.

Features

The platform is customized for the needs of a particular domain. Teams can have autonomy to choose specific tools and workflows while respecting global standards. Often used as a pilot starting point before an organization-wide roll-out.

For example

A data engineering team using specific pipelines for data processing.

Project scale or for a specific digital product

Triggering event

In smaller companies or startups, or when the platform is implemented for a specific project or product because resources are limited and it is not yet possible to deploy a platform at scale or in the context of a first experiment before scaling up.

Features

The platform is restricted to a project or product, focusing on immediate needs. The aim is often to solve problems of complexity or productivity within a reduced framework.

For example

A startup or a team is setting up a simple platform to automate the deployment of its first application, or a specific need for scalability triggers the approach on a limited perimeter.

TAKE AWAY

- Large companies will benefit enormously from implementing an organization-wide platform, even if we recommend an agile, incremental approach.
- Startups, teams, or small businesses can begin on a smaller scale, targeting specific objectives.
- An organization-wide platform makes sense if development and deployment processes are already standardized.
- For highly specialized needs, a domain-specific platform may suffice.
- A dedicated team of Platform Engineers can manage an organizational platform, making the initiative cross-functional.
- Mature organizations with complex infrastructure (multi-cloud, Kubernetes, CI/CD) often need an organizational approach.

What is WeScale's approach?

Implementing Platform Engineering in a company is a structured, iterative process that requires good planning, collaboration between different teams, and modern development and DevOps practices.

Platform Engineering should be built according to the rules of the art, adopting a Product approach. The key to success is a product owner or manager.

Here are the key steps that WeScale suggests for implementing an effective Platform Engineering approach.

Assess the organization's needs

Objective: Identify the main challenges, opportunities, and priorities to justify setting up a platform.

This phase requires:

- An audit of existing workflows to analyze current processes (development, deployment, environment management).
- Identify friction points:
  - Time wasted by developers on operational tasks.
  - Production lead times.
  - Complexity of configurations and pipelines.
- Involve stakeholders:
  - Discuss with development, DevOps, security and management teams to understand their needs.

Define vision and objectives

Objective: Establish a clear strategy and measurable objectives.

Vision: Describe the role of the platform (e.g. "Enable developers to provision environments with a single click").

Measurable objectives

- Reduce average time to production (e.g. from 10 days to 2 days).
- Improved developer productivity (e.g. 20%).
- Reduction in configuration-related incidents (e.g. 50%).

Building the Platform Engineering team

Objective: Create a dedicated team responsible for the design, development and maintenance of the platform.

Team composition

- Product Owner / Product Manager: To set up a product approach, the ceremonies and artifacts needed to implement Platform Engineering are considered as a product in its own right.
- Platform Engineers: To provide expertise in DevOps infrastructure and tools.
- Full-stack developers: To work on user interfaces or portals.
- DevOps managers: to align the platform with existing practices.
- Safety experts: To integrate safety practices right from the start.

Cross-team collaboration: The team must work closely with developers and Ops to ensure platform adoption. The PO/PM is responsible for orchestrating this collaboration.

Designing the platform

Objective: Create a clear, scalable architecture to meet identified needs.

Define the main functionalities

- Automatic environment provisioning.
- Ready-to-use CI/CD pipelines.
- Configuration and secret management.
- Observability (monitoring, logs, metrics).
- Self-service portal or API for developers.

Choosing tools and technologies

We select the right tools for your ecosystem, integrating existing ones.

Here is a non-exhaustive list

- Orchestration: Kubernetes, Nomad.
- Infrastructure as Code: Terraform, Pulumi, Crossplane
- Developer portal: Backstage, Humanitec.
- Observability: Prometheus, Grafana, Datadog.
- Automation: ArgoCD, Tekton.

Design the platform architecture

There are several BluePrints commonly used in our industry, and we design/adapt the architecture that best suits your needs. The principles are as follows:

- Centralized infrastructure for all tools.
- Interfaces are accessible via API or portal.
- Integration with existing systems (e.g. identity management).

Build a minimum viable version (MVP)

Objective: We will work with you to design an initial version to validate assumptions and obtain rapid feedback.

Prioritize key functionalities: Deploy the most critical functionalities to resolve key friction points (e.g., environment provisioning or CI/CD pipelines).

Test with a pilot team

- Select one or two teams to use the platform.
- Gather feedback to adjust functionality and user experience.

Iterating and improving the platform

Objective: With or without us, the aim is to refine and extend the platform in line with user feedback and evolving needs.

Collecting metrics

- Time saved by developers.
- Number of errors or incidents avoided.
- Platform adoption rate.

Add functionality

Once the MVP has been validated, we can integrate additional functionalities (advanced observability, permissions management, etc.) if you wish.

Improving the user experience: You and us will work on the interface and workflows to make it even easier to use.

Train and engage teams

Our consultants can provide ad hoc training.

Objective: Ensure that all teams in the organization adopt the platform.

- **Training sessions:** show developers how to use the platform's features.
- **Documentation:** Provide clear, detailed guides.
- **Regular communication:** share the platform's successes (reduced lead times, improved productivity).

Maintaining and upgrading the platform

Objective: We will support you, to ensure the sustainability and relevance of the platform with punctual and planned missions whose objectives are:

- **Implement continuous monitoring:** Use observability tools to monitor platform performance.
- **Update tools:** regularly integrate new technologies and functionalities.
- **Constantly collect feedback:** Stay in touch with users to adapt the platform to their changing needs.

Time required to implement Platform Engineering

We have observed the following set-up times:

- Analysis and planning (1 to 3 months)
- Platform design (1 to 2 months)
- Development and prototyping (3 to 6 months)
- Ongoing optimization (6 months or more)

Factors influencing lead times

- **Size of organization**: A small company can deploy a platform more quickly (6 to 9 months), while a large organization will require more time (12 to 18 months).
- **Complexity of existing infrastructure**: The more complex the existing infrastructure (e.g. multi-cloud, microservices), the greater the effort to integrate and standardize will be.
- **Available resources**: A dedicated and experienced Platform Engineering team speeds up the process. It will take longer if teams have to juggle their day-to-day responsibilities with building the platform.
- **Team adoption**: Implementation will be faster if developers quickly adopt the new tools and workflows. Otherwise, additional training and support will be required.

General estimates of lead times by type of organization:

- **Small organization**: 6 to 9 months.
- **Medium to large organization**: 12 to 18 months.
- **Complex organizations**: Up to 24 months or more

TAKE AWAY

Start small: We focus on a key problem to demonstrate value quickly.

Working closely with end-users: Developers need to be involved at every stage.

Measure results: Track productivity gains, incident reductions, and cost savings.

Adapt to change: The platform must evolve with the organization's needs.

Adopt a product approach: The Platform must be considered and managed like any other digital product and built in an iterative, agile way while sticking to the needs of the primary users: the development teams.

Step	Objective
Assess needs	Identify challenges and priorities
Defining objectives	establish a clear vision and KPIs
Building the team	Forming a multidisciplinary team
Designing the platform	Defining the architecture, choosing the tools
Building an MVP	Launch an initial version and test with a pilot team
Iterating and improving	Refining functionality and user experience
Training teams	Ensuring successful adoption through training
Maintaining and developing	Monitor, update and extend the platform

By following WeScale's approach, you can gradually build a robust platform, tailored to your organization's needs, and maximize the benefits of Platform Engineering.

What are the main Platform Engineering technologies?

Below is a non-exhaustive list of the main tools and technologies we encounter and use in Platform Engineering, categorized according to their role in the design, management and operation of Internal Developer Platforms (IDPs). There are many more, and the list continues to grow.

Orchestration and container management

These tools enable you to manage containers and orchestrate their deployment in a distributed environment.

- **Kubernetes**: the essential container orchestration platform for managing large-scale deployments.
- **Docker**: the standard solution for creating, sharing, and running containers.
- **Nomad (HashiCorp)**: A lightweight alternative to Kubernetes for orchestrating containers and workloads.

Infrastructure as Code (IaC)

These tools automate the management and provisioning of cloud and on-premise resources.

- **Terraform (HashiCorp)**: Infrastructure management via declarative files for multiple cloud providers.
- **Pulumi**: Alternative to Terraform, using programming languages such as Python or JavaScript to define the infrastructure.
- **Crossplane**: Extensions that leverage Kubernetes to manage infrastructure as code.
- **AWS CloudFormation**: AWS-specific solution for managing resources via JSON or YAML templates.

CI/CD (Continuous Integration/Continuous Delivery) pipelines

These tools automate application build, testing and deployment processes.

- **GitHub Actions**: The workflow offering for the leading code hosting solution.
- **Jenkins**: One of the most popular tools for creating CI/CD pipelines.
- **GitLab CI/CD**: Integrated into GitLab to manage the complete application lifecycle.
- **CircleCI**: Flexible CI/CD platform for automated deployment.
- **ArgoCD**: GitOps tool for deploying and managing Kubernetes applications.
- **Tekton**: native Kubernetes framework for building CI/CD pipelines.

Developer Portals

These tools provide a centralized interface for accessing the organization's workflows, tools and resources.

- **Backstage (Spotify)**: Open-source platform for centralizing documentation, tools, and services.
- **Humanitec**: Platform for building IDPs and enabling self-servicing workflows.
- **Port**: Alternative to Backstage with similar features for DevOps teams.

Monitoring and observability

These tools can be used to monitor systems, collect metrics, and diagnose problems.

- **Prometheus**: Open-source monitoring solution for cloud-native environments.
- **Grafana**: Data visualization platform for metrics from Prometheus and other sources.
- **Datadog**: SaaS solution for application, infrastructure, and log monitoring.
- **Open Telemetry**:an open-source framework for collecting traces, metrics and logs.
- **New Relic**: Complete observability platform for modern applications.

Secret and identity management

These tools ensure application security by managing identities and secrets (tokens, API keys).

- **Vault (HashiCorp)**: Secure storage and secret management for applications.
- **AWS Secrets Manager**: Secrets management for AWS environments.
- **Keycloak**: Identity and Access Management (IAM).
- **Okta**: SaaS identity and access management.

System automation and configuration

These tools facilitate task automation and system configuration.

- **Ansible**: YAML-based configuration management for infrastructure automation.
- **Chef**: infrastructure automation using Ruby to define configurations.
- **Puppet**: Automated system administration and configuration.

GitOps

GitOps is an approach to managing infrastructures and applications using Git as the source of truth.

- **ArgoCD**: Kubernetes-native GitOps deployments.
- **FluxCD**: GitOps framework for automated deployment.
- **Weave GitOps**: GitOps tool for Kubernetes environments.

Databases and storage

These tools manage data storage in cloud and on-premises environments.

- **Rook**: Cloud-native storage orchestrator for Kubernetes.
- **PostgreSQL** : Open-source relational database, commonly used in platforms.
- **Amazon S3**: scalable object storage for unstructured data.

Collaboration and workflow management

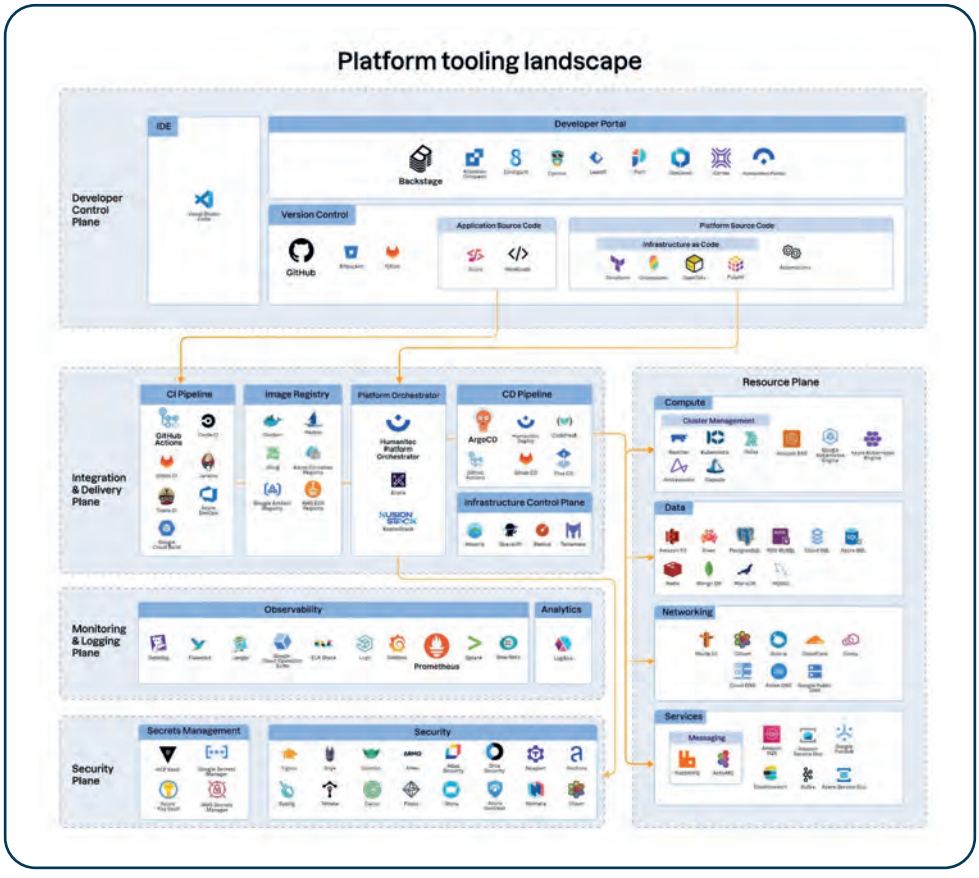
These tools improve collaboration between teams by centralizing information and tasks.

- **Jira (Atlassian)**: Project management for technical teams.
- **Notion**: Flexible tool for centralizing documentation and workflows.
- **Slack**: real-time communication, often integrated into DevOps pipelines and tools.

Safety and compliance

These tools ensure that platforms comply with safety and compliance standards.

- **Kube-bench**: Analysis of Kubernetes cluster compliance with best practices.
- **Falco**: Monitoring the security of Kubernetes workloads.
- **Trivy**: Vulnerability scanner for containers, Kubernetes, and IaC.



TAKE AWAY

This list brings together the essential tools for implementing Platform Engineering but is not intended to be exhaustive, as technologies are evolving very rapidly in this field.

Category	Main tools
Orchestration	Kubernetes, Docker, Nomad
Infrastructure as Code	Terraform, Pulumi, Crossplane
CI/CD	Jenkins, ArgoCD, Tekton
Developer portals	Backstage, Humanitec, Port
Monitoring	Prometheus, Grafana, Datadog
Secret management	Vault, AWS Secrets Manager, Doppler
Automation	Ansible, Chef, Puppet
GitOps	ArgoCD, FluxCD, Weave GitOps
Database	Rook, PostgreSQL, Amazon S3
Collaboration	Jira, Notion, Slack
Security	Kube-bench, Falco, Trivy

What notable companies have implemented Platform Engineering?

Netflix

Platform Engineering practices: Netflix is one of the pioneers in adopting Platform Engineering practices. The company has built a cloud-based platform infrastructure with AWS, enabling high agility, scalability, and resilience. They have developed in-house tools such as Spinnaker (continuous deployment tool) and Chaos Monkey (resilience testing tool) to automate processes and guarantee rapid, secure deployments.

Objective: Improve development speed, reliability and scalability while fostering a DevOps culture.

Spotify

Platform Engineering practices: Spotify uses a robust cloud platform to manage its streaming music applications. They have implemented a platform dedicated to deployment automation, continuous integration, and microservices management. Their Platform Engineering team focuses on providing a reliable, scalable infrastructure that enables development teams to concentrate on product innovation rather than environment management.

Objective: To provide an infrastructure that enables development teams to concentrate on writing code and reduce the obstacles associated with infrastructure management.

Google

Platform Engineering practices: Google has an extremely advanced cloud infrastructure platform with Google Cloud Platform (GCP). Their Platform Engineering team contributed to developing Kubernetes, a container orchestration system that enables flexible, automated resource management. They also use practices such as Site Reliability Engineering (SRE) to ensure system stability while enabling high development speed.

Objective: Promote scalability, flexibility, and automation to improve the efficiency of engineering teams and reduce costs.

Amazon (AWS)

Platform Engineering practices: Amazon uses the concept of Platform Engineering at the heart of its business, with tools such as AWS Lambda for serverless service deployment and Amazon EC2 for on-demand infrastructure management. Their engineering platform integrates governance, security, infrastructure automation and cloud resource management practices to ensure rapid service delivery and scalability.

Objective: Provide scalable, automated infrastructure for their own needs and for their customers via AWS.

Airbnb

Platform Engineering practices: Airbnb has built a robust platform to support its thousands of developers. Their platform includes continuous integration tools, automated deployment practices, and microservices management. They have also invested in configuration and infrastructure management tools to ensure their development teams can rapidly deploy features while maintaining high availability.

Objective: Ensure an agile, scalable, and reliable infrastructure that supports rapid growth and facilitates team collaboration.

X (anciennement Twitter)

Platform Engineering practices: Twitter has created teams dedicated to managing its cloud infrastructure and development tools to reduce silos and simplify the integration of new services. They use systems like Kubernetes and in-house tools to manage microservices and containers. They aim to provide a scalable infrastructure that supports millions of requests per second.

Objective: Improve development speed, system resilience, and resource management to support a large-scale platform.

Uber

Platform Engineering practices: Uber has created a centralized technical platform that enables its development teams to manage infrastructure and deployment more automatedly. Their Platform Engineering team ensures that applications can be rapidly scaled and that teams can work with standard tools while maintaining security and stability.

Objective: Offer a reliable, flexible platform that enables the company to expand rapidly on a global scale while guaranteeing high performance.

Microsoft

Platform Engineering practices: With the adoption of Azure, Microsoft has invested heavily in Platform Engineering practices to support its cloud services. This includes the use of Azure DevOps, containerization services such as Azure Kubernetes Service (AKS), and tools for deployment automation and infrastructure management. Microsoft also uses automation and microservices management practices to deliver high scalability for its services.

Objective: Accelerate development processes while ensuring robust security and high performance on a global scale.

Slack

Platform Engineering practices: Slack uses a cloud platform that enables agile and automated management of its infrastructure. To support the rapid growth of its product, they focus on microservices management and continuous integration. Their Platform Engineering team is responsible for optimizing deployment tools and infrastructure, reducing complexity for development teams.

Objective: Enable rapid deployment and efficient service management while guaranteeing high availability.

Shopify

Platform Engineering practices: Shopify uses Platform Engineering principles to manage its infrastructure tools at scale. They have built a cloud platform that supports millions of merchants and enables their development teams to quickly and efficiently deploy new features while ensuring reliability and security.

Objective: Create a modern, flexible, automated infrastructure to support the platform's rapid growth and improve user experience.

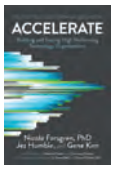
What inspired me to write this eBook?

First and foremost, of course, the women and men who work at WeScale are my greatest sources of inspiration. Every day, they all help me to understand where our market is heading. Ultimately, I'm just a conductor of their knowledge; my role is to spread the word about what they do so well.

Books

"**Team Topologies: Organizing Business and Technology Teams for Fast Flow**" - Matthew Skelton & Manuel Pais.

This book explains how to structure teams to optimize collaboration and introduces the concept of Internal Developer Platforms (IDPs).



"**Accelerate: The Science of Lean Software and DevOps**" - Nicole Forsgren, Jez Humble, Gene Kim

Although it focuses on DevOps, this book provides a solid grounding in the performance of technical teams, ewhich is ssential for Platform Engineering.

"**Kubernetes Patterns**" - Bilgin Ibryam & Roland Huß

Ideal for understanding how to build platforms on Kubernetes and exploit templates to design robust workflows.

Blogs and articles

Platform Engineering.org

This site is dedicated to Platform Engineering practices and concepts, with articles, guides, and case studies. WeScale is a partner and co-organizer of PlatformCon2025 in Paris, to be held in October.

Humanitec Blog

Humanitec is a major player in Platform Engineering and offers in-depth resources on Internal Developer Platforms. Luca Galante is our friend, and he too is helping us to organize PlatformCon2025 in Paris.

Thoughtworks Technology Radar

Thoughtworks regularly publishes analyses of technology trends, including those related to Platform Engineering.

Conferences and videos

KubeCon + CloudNativeCon

An annual conference dedicated to Kubernetes and Cloud Native technologies, with many sessions covering Platform Engineering.

DevOps Days

Although focused on DevOps, this conference series often explores related topics such as platform engineering.

PlatformCon

It's undeniably the world's largest conference dedicated to Platform Engineering. And as mentioned above, we're proud to be organizing PlatformCon2025. You can register your interest and be informed when Early Bird tickets go on sale by clicking here.

YouTube Channels

Humanitec Channel : Content focused on IDPs and practical demonstrations.

Cloud Native Computing Foundation (CNCF) : Technical videos on Kubernetes and its use cases in Platform Engineering.

Open-source tools for hands-on learning

Backstage (bySpotify)

Open-source platform for building an Internal Developer Platform. (GitHub link)

Crossplane

Open-source tool for managing infrastructure as code, often used in platforms. (GitHub link)

ArgoCD

Used to manage continuous deployments on Kubernetes, a key component of an IDP. (GitHub link)

Terraform

One of the most widely used infrastructure management tools for Platform Engineering. (Documentation)

Communities and forums

Platform Engineering Slack Community

An active community of experts and beginners sharing ideas and solutions (Join).

CNCF Community

The Cloud Native Computing Foundation offers forums and discussion groups on Kubernetes and cloud-native practices (Official site)

Reddit

Subreddits like r/devops or r/kubernetes often contain discussions on Platform Engineering.

Gartner **Gartner studies**

About

Luc Legardeur
CEO of WeScale

Biography

Luc Legardeur is a well-known French entrepreneur in the information technology sector. After graduating from Sup'Info in 1991, he began his career at Oracle, before holding management positions in France and Europe for North American and European software vendors such as SmartDB, Selectica and Access Commerce.

In 2004, he founded Xebia France, a consultancy specialising in agile software development and innovative technologies, where he holds the position of CEO. Under his leadership, Xebia distinguished itself and became recognised in the sector for its expertise in agile methodologies and its commitment to technical excellence.

In 2014, anticipating the rise of cloud computing, Luc Legardeur helped create WeScale, a company dedicated to helping organisations make the transition to the cloud. Since January 2025, he has been its Chairman.

In 2018, Luc Legardeur co-founded Zeenea, a company specialising in data governance and metadata management. As CEO, he led Zeenea until its acquisition by HCLSoftware in September 2024.

At the same time, Luc Legardeur is a co-founder of several other companies, including XebiaLabs, a publisher specialising in DevOps; Cellenza, specialising in Microsoft technologies; UX Republic, focusing on user experience; Texei, a consultancy specialising in SalesForce; Thiga, a digital innovation consultancy specialising in product businesses; and Hymaïa, a company specialising in Data and AI consulting and strategy.

Passionate about agile methodologies, he co-authored the book 'Scrum in Action' and is the founder of the French Scrum User Group, actively contributing to the spread of agile practices in France.

Book a meeting →

Paris

34, Boulevard Haussmann
75009 Paris
—
contact@wescale.fr
www.wescale.fr
blog.wescale.fr
01 83 75 05 26

Nantes

1, Allée Cassard
44000 Nantes

Follow us on our networks

- Wescale
- @YesWescale
- WescaleTV
- WeSpeakCloud