Gartner Research

Leverage Platform Engineering to Scale DevOps Platforms Into Hybrid Cloud

Carolin Zhou, Daniel Betts, George Spafford

14 March 2023

Gartner®

Leverage Platform Engineering to Scale DevOps Platforms Into Hybrid Cloud

Published 14 March 2023 - ID G00774800 - 14 min read

By Analyst(s): Carolin Zhou, Daniel Betts, George Spafford

Initiatives: I&O Platforms; Digital Technology Leadership for CIOs in China

Organizations often struggle to scale DevOps toolchains into hybrid cloud environments. I&O leaders scaling DevOps toolchains in hybrid cloud must establish platform engineering teams. Collaboration with software engineering teams focused on DevOps platform as products is essential for work needs.

Additional Perspectives

- Summary Translation: Leverage Platform Engineering to Scale DevOps Platforms Into Hybrid Cloud
(13 April 2023)

Overview

Key Findings

- Development teams face considerable cognitive load when managing siloed complex tools impacting their agility, especially when deploying to hybrid cloud environments.
- Some organizations that have implemented DevOps toolchain on-premises struggle to determine whether to use a cloud-based tool to replace the existing one.
- Organizations operating a hybrid cloud delivery model struggle to support/provide a scalable and consistent way to manage CI/CD for applications and infrastructure.

Recommendations

Infrastructure and Operations (I&O) leaders responsible for I&O platforms should closely partner with users of platform to:

- Build and evolve the DevOps toolchain by establishing a platform engineering team to manage the requirements across the different environments and identify needs shared across development teams.

- Assess if on-premises tools meet the needs of hybrid cloud deployment by working with software engineering and security teams to define future platform landscape of cloud workloads' continuous delivery needs and application architectures.

- Establish scalable and consistent ways to implement and manage the DevOps tasks by optimizing and simplifying standardized pipelines for the needs of platform users across environments.

Strategic Planning Assumption

By 2027, 80% of large organizations must embrace platform engineering to successfully scale DevOps initiatives in hybrid cloud environments, up from less than 30% in 2023.

Introduction

Agile, DevOps and related development approaches are the preferred methods for the rapid creation and continuous delivery of software products and services that fuel digital delivery. DevOps toolchains are the technology platforms that support automation of continuous delivery. Organizations are looking to adopt/extend DevOps platform that provide flexibility and improve speed, consistency, reduce costs, enhance performance and efficiency in deployment across complex hybrid cloud environments. This is especially relevant for cloud migration phase and enables them to choose the right cloud platform to run their workload. [1]

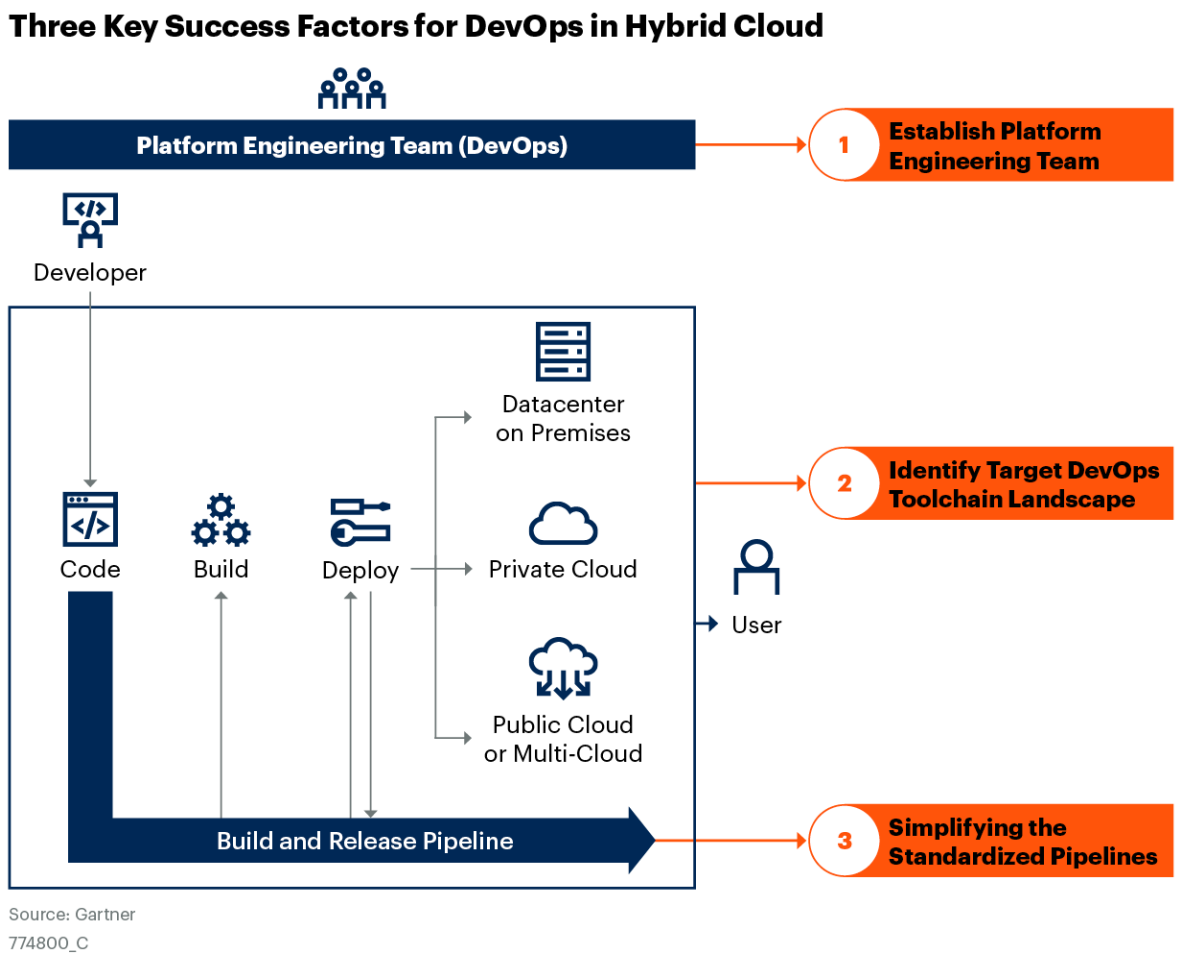
However, enterprises struggle to manage the overall requirements for, and technical debt within, the DevOps toolchain across complicated hybrid cloud environments. Creating hybrid cloud DevOps pipelines is complex and would create cognitive load for development teams to deliver themselves. Hybrid cloud environments contain new regulatory compliance and security requirements, while governance can also be disruptive and costly to implement and automate across disparate solutions. Platform engineering can be leveraged to provide the platform as a product to improve the consistency, efficiency, agility and cost effectively.

They also struggle to plan a DevOps platform architecture for hybrid cloud environments and streamline the consistency pipelines that can support the automatic and consistent deployment of different workloads to the right environment. Teams responsible for the DevOps platform are facing challenges, including:

- Traditional I&O talent and competencies find it hard to manage and govern a full stack of DevOps tools.
- Lack of collaboration culture to break department silos between I&O and software engineering teams.
- Delivering consistency and scale for devops workflows across both on-premises and cloud environments.
- Updating existing pipelines to accommodate containers and/or cloud-native applications across hybrid cloud environments.
- Continuously testing workloads and ensuring that compliance and security requirements are met across hybrid cloud environments.
- Meeting the needs of developers and software engineering teams that will not adopt a platform that is less functional, less complete, less efficient or less-well-supported than their existing preferences or options.

How should I&O leaders work with product teams or software development teams to build and evolve DevOps platform for hybrid cloud to enable agility and speed to market? This research helps I&O leaders support the consistent DevOps capabilities in hybrid cloud environments (see Figure 1).

Figure 1: Three Key Success Factors for DevOps in Hybrid Cloud



Analysis

Establish a Platform Engineering Team

Establish a platform engineering team to deliver and manage shared DevOps platforms across complex hybrid cloud environments for developers or product teams. Work closely with platform users based on their needs and/or pain points to enable self-service, platform consolidation and coherent, composable tools and/or templates. Meanwhile, tightly collaborating with the teams that provided software execution environments like K8s is also very important. This is akin to how platform engineering teams manage the life cycle of multiple kubernetes clusters across different environments and different K8s distributions (see How to Scale DevOps Workflows in Multicluster Kubernetes Environments).

Platform engineering is the discipline of building and operating self-service internal developer platforms (IDPs) for software delivery and product life cycle management. Each platform is a layer created and maintained by a dedicated product team, designed to support the needs of software developers by interfacing with tools and processes. Platform engineering aims to optimize the developer experience and accelerate product teams' delivery of customer value.

By design, the platform engineering team should be responsible for maximizing the value of the DevOps platform and tackling problems such as governance, knowledge sharing and compliance. The platform engineering team should focus on the product teams, software engineering teams or development team and security team's needs and biggest pain points (see 3 Steps to Kickstart Platform Engineering in Your Organization).

Platform Engineering Teams Must Have Platform Owners

Platforms are internal products built, maintained and improved by a dedicated team. The way that the platform engineering team manages the life cycle of the DevOps platform is via product management methodology. Thus, product owner is the owner for the whole platform engineering team and the key responsibilities are as examples below (see Why DevOps Success Requires Platform Teams).

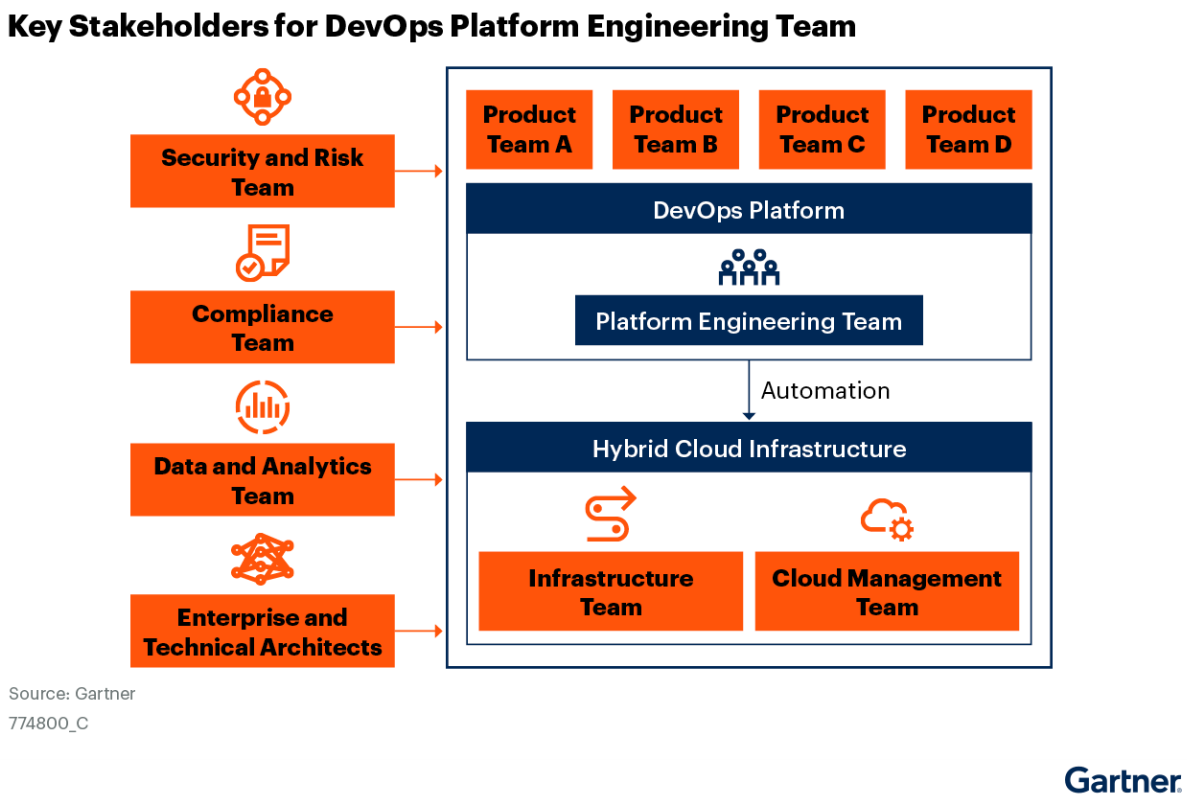
- Optimizing customer value, cost, compliance and risk in the timeframes required.
- Accountable for the whole product life cycle management and technology/solution selection for DevOps platform across hybrid cloud environments.
- Collaborate with product teams or software engineering teams closely to identify business future use cases across different environments and pain points to build a thinnest viable platform (TVP). Must be built with self-service in mind to not become a bottleneck.

- Building feedback loops for developer experience metrics, such as customer satisfaction, adoption and retention. Make the product delivery metrics available for product teams, such as deployment frequency, lead time for changes in different environments (e.g., on-premise, private cloud or public cloud/multi-clouds), change failure rate across different environments or mean time to restore (MTTR).
- Regularly promoting the catalog of tools and services that platform offers and marketing all platform development initiatives that are in progress to ensure it meets customers' needs.

Define the Goal and Engage Key Stakeholders

The platform owner must engage with the stakeholders to define goals and align DevOps platform requirements to ensure stakeholder acceptance and support (see Figure 2).

Figure 2: Key Stakeholders for DevOps Platform Engineering Team



To create a platform and evolve the core platform capabilities for hybrid cloud, the platform team must understand the needs and pain points of their customers and create reusable capability to meet those needs, as well as the ones of others as they scale. Typical goals or objectives for establishing or scaling the DevOps platform into hybrid cloud environments include:

- Establish a DevOps platform that covers all phases of product life cycle (i.e., plan, build, secure and deploy), using a standardized process to drive business deliverables quickly with complete transparency, consistency and traceability.
- Empower organizations to maximize the business value of software development by integrating with agile practice management processes, shortening software development cycles, and improving the delivery and operational efficiency. Enabling product teams to create better software or products in less time and at lower costs.
- Develop a sustainable infrastructure for specific applications and ensure scalability by leveraging the capability of infrastructure as code.
- Eliminate issues surrounding toolchain sprawl, maintenance and integration. Provide a single source of truth to streamline the technical development process across hybrid infrastructure.
- Build a platform foundation with out-of-box tools or processes to enable software automatic deployment for different workloads into the right environment software developer's need for cloud migration.

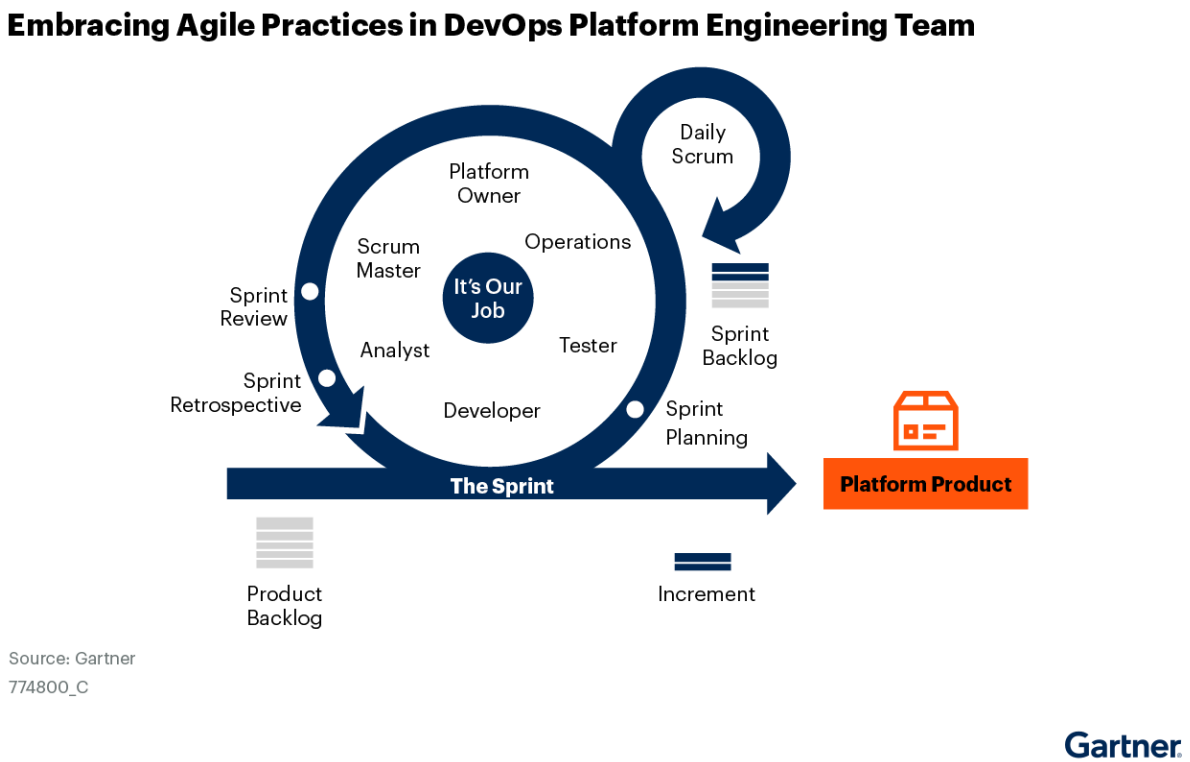
Adopt Agile Practices to Evolve the Product Via Changing Requirements

Agile practices have proliferated into platform engineering to boost self-organization and stakeholder or team member engagement and collaboration with product teams or software engineering teams. The following steps are key to adopting agile practices (see Figure 3):

- Adoption of new collaboration mindsets and employee behaviors (see Improve Infrastructure Agility — A Story for a New Era).
- Use a planning tool (e.g., Atlassian Jira) to manage requirements and create visibility and transparency of work.
- In collaboration with product/development teams, create a platform backlog of requirements and prioritize these as user stories/needs. The typical template of user story is with the format, *"As a <role or personal>, I can <goal or need>, so that <why>"*
- Break initiative-based tasks into iterations (that is, sprints) and define the timeframe for each sprint — normally, two or three weeks will be defined as one sprint, based on the skills and experience of the platform engineering team (see Adopt an Iterative Approach to Drive DevOps Success in Large Organizations).

- Work in iterations or sprints with agreed-upon efforts and acceptance criteria by establishing the definition of done (DOD) and definition of ready (DOR) with platform users.
- Schedule daily 15-minute standing up meetings like daily scrum, which focus on progress toward sprint goals, velocity, capacity and an actionable plan for the next day of work.

Figure 3: Embracing Agile Practices in DevOps Platform Engineering Team



Define Hybrid Cloud Architecture That Your DevOps Platform Must Support

Toolchains should evolve and scale to meet business priorities and product teams or software engineering teams' needs in hybrid cloud infrastructure. To do so, the platform owner — in collaboration with consumers of the platforms — should adopt the following evaluation methods to identify the target solution and architecture for a future DevOps platform in hybrid cloud:

- Collaborate with software engineering leaders or business product owners to identify the existing tools in use in accordance with the critical capability of DevOps toolchains (see Note 1 and How to Build and Evolve Your DevOps Toolchains). Identify the pain points and constraints of SWE teams existing toolchains.

- Identify the application architectures and future use cases of technology stacks of cloud workloads, including CI/CD pipelines and data of applications to do gap analysis to identify future tools needed (i.e., tools to support cloud native architecture or serveless).
- Align future toolchain platform architecture planning with the organization's cloud strategy. If cloud strategy is focused on private cloud for most cloud workloads while public cloud or multi-clouds are used to enable technology innovation due to regulatory compliance and security requirements, then the master instance of the DevOps platform should be deployed on-premises. Agents can be deployed to the public clouds as cloud workloads require (e.g., a Jenkins agent to support the CI/CD task to deploy the workload to public cloud or multi-cloud as one scenario (see Comparing DevOps Architecture to Automate Infrastructure and Operations for Software Development).
- The platform engineering team may lack the skills to develop an entire, comprehensive DevOps platform, or the existing technology stacks and tools of the DevOps toolchain may not meet the needs of product or development teams. If cases like this, evaluate the third party or cloud-native DevOps platform solution or product together with platform customers based on the organization's functional needs, non-functional needs (i.e., compliance and security requirements, customer support, service and delivery experiences) and total cost of ownership (TCO).
- When planning the security functionality for the platform, prioritize security as part of a continuous-improvement process spanning development and operations.
- When scaling DevOps platform to a hybrid cloud environment, mitigate toolchain technology debt for DevOps platform. Use a hybrid architecture to mitigate growing costs incurred by scaling up your existing pipeline or vendor lock-in.

Simplify Standardized Pipelines

For each DevOps activity (i.e., compile and build software source code, run the auto-testing, build a container image and deploy the code package to a container), a single tool or multiple tools (depending on the complexity of DevOps environment and applications) may be used. While all activities are interconnected, different tools are often designed to address specific activities. Thus, the platform engineering team should follow the principles below to orchestrate and integrate these tools to optimize and streamline flows and optimize the developer experience throughout the whole process, like:

- **Keep the thinnest viable platform (TVP) concept in mind.** The platform should provide as thin and simple an abstraction as possible so that it does not cost too much to build/maintain, and so that it does not become deeply proprietary to the organization.
- **User-centered design.** The platform product design should base on the needs of platform users rather than seeking to change the user's behavior around how a platform product works.
- **Self-service paved road.** Platform should provide an easy, self-service paved road and reusable tools or services for users.
- **Demand-driven.** Based on product team or development team needs, improves and simplifies the pipeline by integrating automation tools and workflows to build, compile, test and release the application.

Meanwhile, to simplify and standardize pipelines, platform owners should ensure that the platform in a hybrid cloud environment:

- Establishes a developer portal to minimize friction and enables automation processes to eliminate repetitive tasks across the delivery pipeline. The platform must automate activities, like build, test, configuration, reporting, compliance, provisioning and monitoring across hybrid cloud infrastructure.
- Leverages the capability of container-native to establish a more stable and popular way to centralize the build process and use container orchestration and management services to launch and manage generated images.
- Leverage IaC tools (i.e terraform) to enable the capability of infrastructure as code for the scenarios of serverless applications or fPaaS, and initiate the CI/CD pipeline integrating with CSP's native tools (i.e., AWS codebuild) and CLI.
- Implements governance policies and continuously monitors and mitigates risk without burdensome administrative controls that impede agility. A solid foundation of building, managing and using CI/CD pipelines can be established via governance process and management (e.g., version management for code, testing cases, etc).

- Adheres to regulatory compliance requirements mandated by internal and external audit committees by establishing measurable governance parameters, such as peer reviews, access controls and audit trails for all actions performed in the DevOps pipeline.
- Use specialized tools to automate packaging of ML models and their associated dependencies into containers with model-serving frameworks and linking them to CI/CD processes if the platform consumer plans to build ML-enabled applications.
- Provides great documentation, guidance and support to users of the platform:
  - Provides an overview of the existing pipeline types and offers agreed-upon pipeline templates for different scenarios to enable reuse and improve efficiency for running the application delivery pipeline.
  - Establishes guidelines for product teams or development teams to create new pipelines, how to configure a pipeline or how to merge request pipelines and multiple ways to run a pipeline (e.g., by using a URL query string, running a pipeline manually and how to add manual interaction to your pipeline for different target environments).
  - Maintains other artifacts or documentation (e.g., pipeline architecture, the way to create a repository either for a new project or existing project, jobs management, docker integration, etc.) and shares this with all development teams or product teams.

Evidence

[1] DevOps and Hybrid Cloud: Life in the Fast Lane? DevOps.com.

Note 1: Critical Capability of DevOps Toolchain

The critical Capability of DevOps Toolchain that Gartner Defined as below:

- **Planning tools**, which are used to manage the requirements for product, metrics gleaned from production, ongoing change requirements, developer effort, time to production and product life cycle management, like Jira or Confluence.
- **Create tool,** the activities associated with the creation of the release candidate, which includes design and code, build, and configure, (i.e., Gitlab, Atlassian, Maven).

- **Verify tool,** the critical activities for verifying are automatic test, static analysis, acceptance testing and security, and the tools for such areas are likely CAST, SonarCode, Selenium and JUnit.
- **Preproduction** is the key stage for the product to be deployed to the production or ready for release. The integration testing, penetration testing and stress testing will be conducted to ensure the quality and safety for the product.
- **Release,** schedule, orchestration, provision and deploy code to production are the key activities for release, which aims to ensure the target environment is ready for release. The typical tool for it is like Jenkins or Chef.
- **Configure** is an integral activity throughout the DevOps toolchain. It will be conducted to support IT infrastructure provisioning and configuration activities required that cannot be configured during “Create” and “Verify.” Here, we are also involving configuration for the shared server platform, storage, databases and network. The example for configuration tools are Ansible, puppet, etc.
- **Monitor,** DevOps monitoring factors in any testing or benchmark performance metrics to measure the effectiveness of ongoing releases. Relevant monitoring information is factored back into “plan,” with any required changes added to the next release cycle, like APM, ITOM, NPM, etc.

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

Comparing DevOps Architecture to Automate Infrastructure and Operations for Software Development

Top Strategic Technology Trends for 2023: Platform Engineering

Innovation Insight for Internal Developer Portals

Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes

Case Study: Developer-Centric Approach to Building a High-Value DevOps Toolchain (LexisNexis)

Use Platform Engineering to Scale and Accelerate DevOps Adoption

Actionable, objective insight

Position your IT organization for success. Explore these additional complimentary resources and tools for I&O and IT leaders:

Guide

Ignition Guide to Creating a Workforce Development Plan for I&O

Use this guide to help you plan and develop a workforce for the future.

Download Guide

Research

Common Pitfalls to Avoid When Developing Your Workload Placement Strategy

Analyze and overcome common pitfalls when designing a workload placement strategy.

Download Research

Article

The 4Rs Cost Optimization Strategy

Read about a cost optimization strategy for I&O leaders in uncertain times.

Read Article

Resource Center

Infrastructure and Operations

Explore insights, advice and tools to help I&O leaders address their top challenges.

Learn More

Already a client?
Get access to even more resources in your client portal. Log In

Connect With Us

Get actionable, objective insight to deliver on your mission-critical priorities. Our expert guidance and tools enable faster, smarter decisions and stronger performance. Contact us to become a client:

U.S.: 855 811 7593

International: +44 (0) 3330 607 044

Become a Client

Learn more about Gartner for IT Leaders
gartner.com/en/information-technology

Stay connected to the latest insights

Attend a Gartner conference
View Conference

Gartner®