Hype Cycle for Agile and DevOps, 2022

Published 13 July 2022 - ID G00772103 - 112 min read

By Keith Mann, Joachim Herschmann, and 1 more

Agile and DevOps are critical parts of a modern software engineering practice. Software engineering leaders should use this research to better understand which innovations can take their capabilities to the next level.

Analysis

What You Need to Know

Agile and DevOps have moved from being emergent perspectives to proven collections of practices and tools that can enable all organizations to deliver customer value faster and more reliably. These approaches emphasize learning and continuous improvement, and they continue to inspire innovative practices and technologies. These include ideas that promote both the technical: automation, continuity and engineering — and the human: collaboration, community and understanding.

Scaling Agile and DevOps efforts to the enterprise level requires meeting the challenges of managing the overarching value streams and optimizing aspects of human factors, organization, process, technology and information exchanges (see recommended research for related notes). No two organizations will face exactly the same challenges in exactly the same sequence.

The Hype Cycle

Working at scale continues to be a driver of progress and adoption in the innovations profiled in this Hype Cycle. Agile and DevOps are the ubiquitous, default approaches to building and operating software. Their supporting practices and technologies must suit everyone, everywhere. At the same time, the scarcity of talent makes automation, optimization and machine learning indispensable. In short, software engineering leaders must, with fewer people, do more things involving more complexity for more stakeholders than ever before.

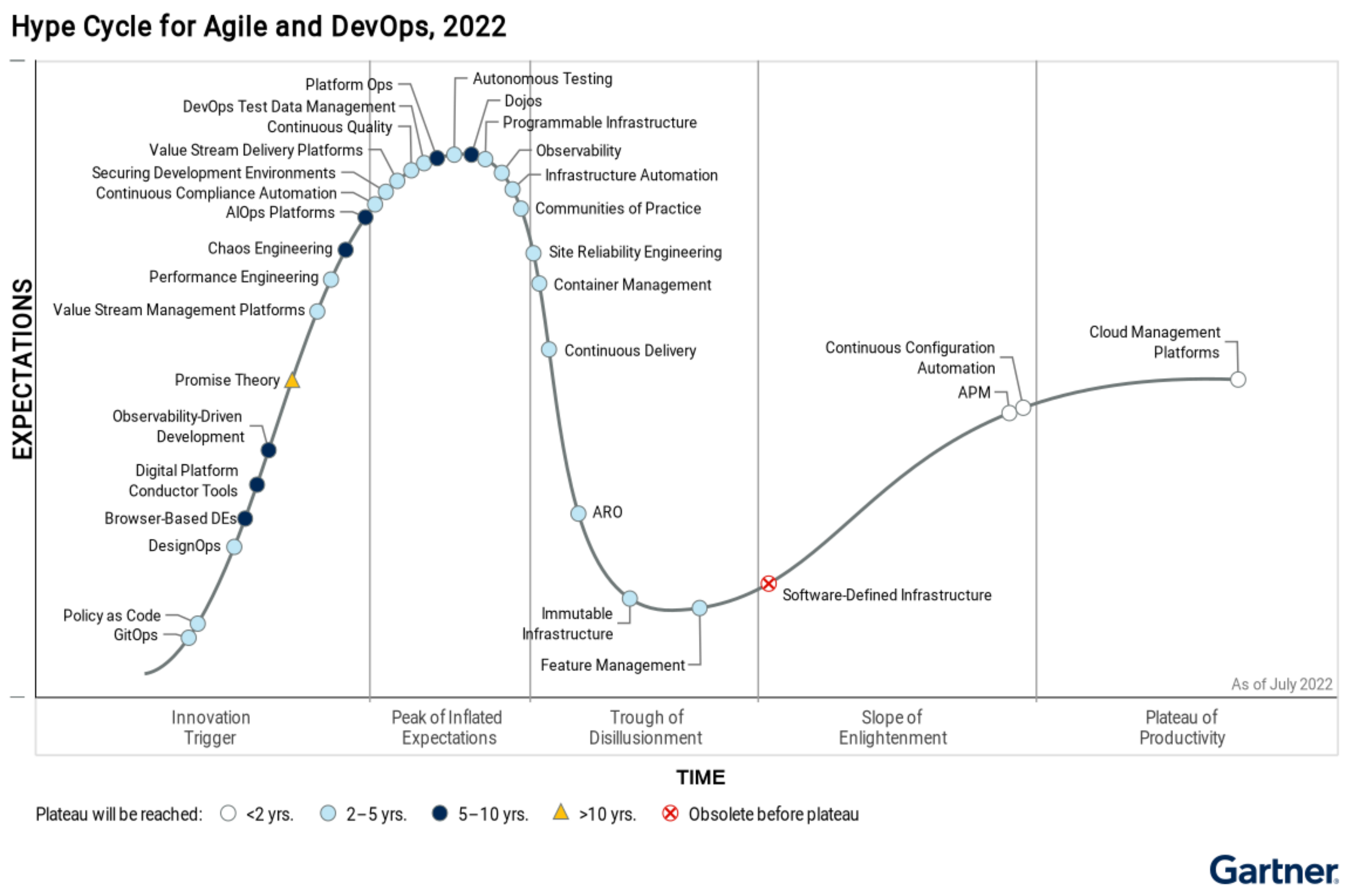
All of this has led innovations that focus on automation, artificial intelligence and autonomy — in other words, those that help to do more with less — to rise to the peak of the Hype Cycle. Software engineering leaders must carefully assess whether these technologies, however desperately needed, are suitable for their organization at their current maturity.

Continuous availability, reliability and resilience have been made popular by high-profile innovations like Chaos Engineering and Site Reliability Engineering. While nobody likes downtime, software engineering leaders must manage expectations for “always on” software with the expense and complexity of the practices and technologies needed to enable it.

Security remains a perennial concern. Fast-moving innovations like DevOps Test Data Management and Continuous Quality reflect this. Be aware, though, that they are no substitute for collaboration between security and software engineering teams.

Often, DevOps technology choices are driven by individual teams without attention to the broader enterprise scope and strategy. Leaders must devise a DevOps strategy and create a culture that enables organizational learning through the use of communities of practice. These communities must evaluate technologies that can be used to support a cohesive, federated DevOps initiative across autonomous teams. Avoid creating tool and capability “islands” that inhibit effective DevOps practices; platform engineering is an effective construct that helps in this regard. Toolchains are evolving from do-it-yourself approaches to platforms that maximize the flow of work.

Figure 1: Hype Cycle for Agile and DevOps, 2022



Source: Gartner (July 2022)

The Priority Matrix

The Priority Matrix maps the time to maturity of technologies and frameworks in an easy-to-read grid format. It answers two high-priority questions:

- How much value will an organization receive from an innovation?
- When will the innovation be mature enough to provide this value?

Application performance monitoring (APM) and continuous configuration automation (CCA) both offer high levels of benefit and are within two years of mainstream adoption. CCA greatly enhances the efficiency and reliability of deployments, while APM provides valuable insights into the behavior of software in production. Software engineering leaders who haven't yet done so should evaluate APM and CCA vendors. CCA in particular requires training and even cultural changes; these take time, so prepare now.

Table 1: Priority Matrix for Agile and DevOps, 2022

Benefit	Years to Mainstream Adoption			
	Less Than 2 Years	2 - 5 Years	5 - 10 Years	More Than 10 Years
Transformational		Observability Site Reliability Engineering	Digital Platform Conductor Tools Platform Ops	

Benefit	Years to Mainstream Adoption			
	Less Than 2 Years	2 - 5 Years	5 - 10 Years	More Than 10 Years
High	APM Continuous Configuration Automation	ARO Autonomous Testing Communities of Practice Container Management Continuous Delivery Continuous Quality DesignOps Feature Management GitOps Infrastructure Automation Performance Engineering Policy as Code Programmable Infrastructure Value Stream Delivery Platforms Value Stream Management Platforms	AIOps Platforms Browser-Based DEs Dojos Observability-Driven Development	Promise Theory

Benefit	Years to Mainstream Adoption			
	Less Than 2 Years	2 - 5 Years	5 - 10 Years	More Than 10 Years
Moderate		Continuous Compliance Automation DevOps Test Data Management Immutable Infrastructure Securing Development Environments	Chaos Engineering	
Low	Cloud Management Platforms			

Source: Gartner (July 2022)

Off the Hype Cycle

DevSecOps has been removed from the Hype Cycle as security is now largely considered an inherent part of DevOps, and corresponding, individual security innovations are emerging.

Hypothesis-driven development has been replaced by Feature Management, as the latter is now the dominant approach to experimentation in software engineering.

On the Rise

GitOps

Analysis By: Paul Delory, Arun Chandrasekaran

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

GitOps is a technique for operating a cloud-native application using only declarative constructs stored in Git. It is the latest name for extending CI/CD to software and infrastructure deployment. According to the canonical OpenGitOps standard, the state of any system managed by GitOps must be: (1) expressed declaratively, (2) versioned and immutable, (3) pulled automatically, and (4) continuously reconciled. These ideas are not new, but new tools and practices now bring GitOps within reach.

Why This Is Important

GitOps would be transformative. GitOps workflows deploy a verified container image to a cluster, bringing code to production with only a Git pull request. Infrastructure is code. All changes flow through Git, where they are version-controlled, immutable, auditable and reversible. Developers interact only with Git, using abstract, declarative logic. I&O uses it to deploy infrastructure. It extends a common control plane across K8s clusters, which is increasingly important as clusters proliferate.

Business Impact

GitOps can be used to improve version control, automation, collaboration and compliance. Artifacts are consistent and reusable. All of this translates directly to business agility and faster time to market. GitOps artifacts are centralized and version-controlled, making them easy to verify and audit. By implementing and centralizing infrastructure as code, GitOps enhances availability and resilience of services.

Drivers

- **Kubernetes adoption and maturity:** GitOps must be underpinned by an ecosystem of technologies, including tools for automation, infrastructure as code, CI/CD, observability and compliance. Kubernetes has emerged as a universal common substrate for cloud-native applications. This provides a ready-made foundation for GitOps. As Kubernetes adoption grows within the enterprise, so, too, can GitOps.

- **Need for increased speed and agility:** Speed and agility of software delivery is a critical metric that CIOs care about. As a result, IT organizations are pursuing tighter coupling of I&O and development teams to drive shorter development cycles, faster delivery and increased deployment frequency. This will enable organizations to respond immediately to market changes, handle workload failures better, and tap into new market opportunities. GitOps is the latest way to drive this type of cross-team collaboration.

- **Need for increased reliability:** Speed without reliability is useless. The key to increased software quality is effective governance, accountability, collaboration and automation. GitOps can enable this with process transparency and workflow commonality across development and I&O teams. Automated change management helps to avoid costly human errors that can result in poor software quality and downtime.

- **Talent retention:** Organizations adopting GitOps have an opportunity to upskill existing staff for more automation- and code-oriented I&O roles. This opens up opportunities for staff to learn new skills and technologies, resulting in better employee satisfaction and retention.
- **Cultural change:** By breaking down organizational silos, development and operations leaders can build cross-functional knowledge and collaboration skills across their teams to enable them to work effectively across boundaries.
- **Cost reduction:** Automation of infrastructure eliminates manual tasks and rework, improving productivity and reducing downtimes, both of which can contribute to cost reduction.

Obstacles

- **Prerequisites:** GitOps is only for cloud-native applications. Many GitOps tools and techniques assume the system is built on Kubernetes (likely also with a host of other technologies built on top of K8s). GitOps is possible outside Kubernetes, but in practice K8s will almost certainly be used. Thus, GitOps is necessarily limited in scope.
- **Cultural change:** GitOps is a cultural change that organizations need to invest in. IT leaders need to embrace process change. This requires discipline and commitment from all participants to doing things in a new way.
- **Skills gaps:** GitOps requires automation and software development skills, which many I&O teams lack.
- **Organizational inertia:** GitOps requires collaboration among different teams. This requires trust among these teams for GitOps to be successful.

User Recommendations

- **Use GitOps for cloud-native workloads:** Your initial target for GitOps must be a containerized, cloud-native application that is already using both Kubernetes and a continuous delivery platform such as Flux.
- **Embed security into GitOps workflows:** GitOps practices can and should embed security into workflows. Security teams need to “shift left,” so that the organization can build holistic CI/CD pipelines that impact software delivery and infrastructure configuration, with security embedded in every layer of the workflow.
- **Be wary of vendors trying to sell you GitOps:** GitOps isn’t a product you can buy; it is a workflow and a mindset shift that needs to be part of your overall DevOps culture.
- **Use GitOps to sell automation initiatives:** You can use the GitOps buzzword to get buy-in for adopting general automation best practices that the organization should already be following. In the absence of a standard, you can define what GitOps means to your organization, then deploy it iteratively.

Sample Vendors

GitLab; Harness; Red Hat; Upbound; Weaveworks

Policy as Code

Analysis By: Paul Delory

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Policy-as-code languages express governance and compliance rules as code, so they can be enforced programmatically by automation tools. PaC languages are often domain-specific and declarative. With PaC, policies are treated as software, making them subject to version control, code review and functional testing. The most mature PaC tools can render any business logic in code. You can use them to enforce architectural standards, corporate policies, regulatory requirements, budgets and more.

Why This Is Important

In the most mature automation pipelines, I&O engineers mostly spend time on optimization, governance and compliance. They no longer build infrastructure; that work has been automated and turned over to others. Now, I&O builds the guardrails around the infrastructure services that their end users consume. I&O must align with security and compliance teams. PaC brings policy enforcement into their automation pipelines, while preserving a separation of duties that mirrors a typical IT org chart.

Business Impact

Policy-as-code improves:

- **Security, compliance and automation:** PaC combined with infrastructure automation provides direct enforcement of policies with implicit compliance guarantees.
- **Alignment of security and Ops teams:** PaC allows security and compliance teams to interface directly with automation pipelines to ensure conformance.
- **Visibility and auditability:** PaC provides both documentation of policies and evidence they are being enforced.
- **Time and effort spent:** PaC means less toil for operators.

Drivers

- **New PaC tools:** Several dedicated PaC tools are now on the market, many of them open source. Most prominent is the Open Policy Agent, a Cloud Native Computing Foundation project. But other options abound, including InSpec (part of the Chef suite) and Sentinel (part of the HashiCorp suite). All of the major hyperscale public clouds also have their own native policy engines.

- **Increasing regulation:** The advent of new regulations such as GDPR has increased both the difficulty of compliance and the pressure on compliance teams. PaC allows compliance teams and auditors to document their policies in detail, and to verify that they are being enforced.
- **Security breaches:** Similarly, a spate of newsworthy security breaches at public companies has put every IT organization's security and compliance practices under increased scrutiny. No I&O team wants its security failures to be the reason that its company ended up in the headlines.
- **Growth of DevOps and DevSecOps:** More and more companies are embracing DevOps and DevSecOps — which means more and more companies are encountering the hard governance problems of automation. Many teams that implement infrastructure as code quickly find that they need better policy enforcement, and PaC will help.
- **Cloud optimization and cost control:** Beside their benefits for security and compliance, PaC tools can also be used to enforce the build standards for infrastructure, including enforcing budgets. In the public cloud, where oversized or unnecessary infrastructure incurs direct out-of-pocket costs, programmatically enforced policies can help to control spending.

Obstacles

- **Scarcity of downloadable content:** PaC tools will not gain real traction until they have an extensive library of community-generated content. Ideally, users would simply download the policies they need from a free, public repository, rather than having to write their own policies. Over time, as the user base expands and commercial offerings see increased adoption, PaC tools will reach a critical mass of downloadable content that enables real-world use cases.
- **Skill set:** Many I&O professionals lack the skills to operate automation and PaC tools effectively. Gartner clients routinely report that their automation and policy management are hindered primarily by people, not tools. PaC will magnify these existing skills challenges.
- **Organizational inertia:** PaC promises improved collaboration between I&O and security/compliance teams. But in some organizations, this change would be unwanted. Internal resistance of this kind will slow the rate, scope and scale of PaC initiatives.

User Recommendations

- **Start small:** Choose a pilot use case where PaC is likely to provide real business benefits. Once PaC has proven its value, expand to other use cases over time.
- **Prioritize existing templates:** Focus your PaC efforts on use cases that have ready-made implementation templates — ideally, publicly-available downloadable content. For example, almost every PaC tool on the market has a canned implementation of the CIS benchmarks.

- **Break down team silos:** Use PaC to build a common workflow for automation and policy enforcement that spans I&O, security and compliance teams.
- **Integrate PaC into automation pipelines:** To automate a process, you must know what to do before you can determine how to do it. PaC creates specific and enforceable guidelines for automation tools to follow, thus solving this problem.
- **Measure before and after:** Use observability tools and value stream mapping to define your starting state, then compare it to the end state. Collect real data to quantify the value of PaC.

Gartner Recommended Reading

Using Cloud-Native 'Policy as Code' to Secure Deployments at Scale

How to Protect Your Clouds With CSPM, CWPP, CNAPP and CASB

Innovation Insight for Continuous Compliance Automation

Innovation Insight for Cloud-Native Application Protection Platforms

Market Guide for Value Stream Delivery Platforms

DesignOps

Analysis By: Brent Stewart

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

DesignOps is a set of operational practices that enables design team management and product-level delivery of design assets. The team management side stresses strategic alignment with business operations for the central design function and career development. The product delivery side combines user experience (UX), product management and technology operations to enable efficient and DevOps-compatible plans, estimates and processes that support quality, collaboration and ongoing innovation.

Why This Is Important

DesignOps introduces formalized approaches to governance, operations and people management. As a set of easy-to-use operational standards, DesignOps continues to gain in popularity. Digital product companies (e.g., Airbnb, Adobe and InVision) and agencies are discovering the tremendous value of a proven operational approach for UX team management and design delivery on product teams.

Business Impact

DesignOps is notable mainly for the value it creates during the delivery of design assets. Here, DesignOps does not alter the core skills and activities of a UX team; rather, it reorganizes them in a way that supports ongoing feature enhancement and idea generation without interrupting the continuous workflow of development teams. DesignOps represents the first widespread implementation of operational methods and techniques created for designers, as well as developers.

Drivers

Modeled to be compatible with DevOps and agile practices, DesignOps structures and organizes design work to enable early and frequent feedback. This takes place via collaboration among the user, the designer and the developer, as well as ongoing, iterative delivery of assets and design decisions to the development team. This allows product teams to run parallel tracks of work (dual-track agile) in which UX teams employ “continuous discovery” to understand the user, engage in research, explore various design directions, and test possible solutions and document outcomes. It also enables them to progressively support early development activities, such as tech design and story creation.

There are three key drivers behind DesignOps:

- **Innovation:** When coupled with DevOps, DesignOps leads to more innovative solutions. As a practice, DesignOps employs dual-track agile, which sets aside ongoing tracks of work dedicated to new discovery, idea generation and design exploration. This work acts as a constant source of evidence-based, multidisciplinary innovation.

- **Speed:** DesignOps reduces the time to market for major updates and incremental feature enhancements alike. Due to the concepts of continuous discovery and continuous delivery, developers engage in tech design, architectural explorations and proofs of concept (POCs) sooner than before, and with a deeper understanding of the overall vision.

- **Collaboration:** DesignOps increases communication and camaraderie between design and development teams. The design-development gap exists for many reasons, one of them being culture. DesignOps promotes multidisciplinary teams in workshop settings, design sprints or one-on-one “pairing and sharing” that promotes understanding, empathy and relationship building between these two crucially important groups.

Obstacles

To a large extent, the growth of DesignOps is inhibited by key gaps in planning, estimation and tracking knowledge:

- Few UX practitioners are educated in detailed planning and estimation, using a common work breakdown structure (WBS).

- Few product managers are trained in UX planning, estimating and tracking, and many of the design platforms lack robust change control solutions.

- Popular enterprise agile planning (EAP) tools are not designed with UX practitioners, activities and deliverables in mind (although this is changing).

User Recommendations

Software engineering leaders should:

- Educate themselves about the practice of DesignOps
- Train their UX teams in the basics of agile
- Pilot the approach with a high-performing, multidisciplinary feature team

Following a successful pilot, application leaders and the pilot team members should:

- Engage in a product-wide rollout that involves training, updated product plans and the allocation of one or more persons to the role of design manager — essentially, a UX-focused product manager.

It should be noted that a successful rollout of DesignOps at the product level requires complete buy-in from product management, design and development teams, as well as robust logistical and administrative skills.

Gartner Recommended Reading

DesignOps: Organize, Collaborate and Innovate Product UX at Speed

Technology Insight for Digital Product Design Platforms

3 Key Practices to Enable Your Multiexperience Development Strategy

Software Engineering Technologies Primer for 2022

Browser-Based DEs

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Browser-based development environments (DEs) provide remote, ready-to-use access to a complete development environment without setup and configuration hassles. This decoupling of the development workspace from the physical workstation enables a consistent developer experience. Browser-based DEs comprise elements of a traditional IDE — such as code editing, debugging, code review and code collaboration. In addition, they also integrate with source control repositories and CI/CD pipelines.

Why This Is Important

Browser-based DEs provide consistent, secure access to preconfigured development workspaces to developers. This frees them from setting up their own environments, eliminating the need to install and maintain dependencies, software development kits (SDKs), security patches and plug-ins. Browser-based DEs are prepackaged with language tooling for multiple programming languages, enabling teams to write code for different application stacks with standardized and templatized workflows.

Business Impact

Browser-based DEs are becoming popular, due to their native integration with Git-based repositories and continuous integration/continuous delivery (CI/CD) tools, accelerating DevOps adoption. They enable IT administrators to centrally manage and secure access to development environments, even from personal devices, minimizing code exfiltration from user machines. Browser-based DEs are an important component of internal developer portals to help improve developer experience.

Drivers

Gartner predicts that, by 2026, 60% of cloud workloads will be built and deployed using browser-based DEs. Four factors are driving their increased adoption:

- Remote work and remote onboarding of software developers create a need for a frictionless onboarding experience. The ability to share the development environment among distributed team members makes remote debugging and pair programming easier.
- Development environment setup issues can impede developer productivity and hurt the onboarding experience.
- The ability to centrally manage, govern and secure development environments becomes especially important with the increasing threat of software supply chain attacks. In addition, browser-based DEs make it easier to support and secure BYOD use cases.
- Automating DevOps workflows introduces more plug-ins, extensions and API integrations, which make it cumbersome to manage on local machines.

Obstacles

- Browser-based DEs incur costs in addition to what an organization may already be paying for DevOps tooling. The cost can be prohibitive for teams that rely only on open-source development tools for application development and delivery needs on local machines.
- Connectivity presents another obstacle. Poor or inconsistent internet speeds adversely affect developer experience. Some in-browser IDEs support offline access, but only after initially logging in. In many cases, developers simply like to use their own local machines to get their

work done. That allows them to use their own plugins, editors and personalized scripts all of which will be difficult through a browser-based interface.

- Security and compliance policies can prevent the use of cloud for development needs in some organizations. This can rule out the use of browser-based DEs that rely on cloud services. Note that browser-based DEs don't necessarily have to be provisioned in the cloud.

User Recommendations

- Pilot the use of browser-based DEs when their teams are developing with cloud-hosted source repositories. Plan for downtime resulting from service outages since browser-based DEs rely on the remote development environment being up and running.
- Ensure that browser-based DEs are part of an overall developer self-service platform with centralized governance. The benefit of centralized governance and developer agility can be a win-win for platform and product teams.
- Use browser-based DEs as one of the "quick wins" to improve the onboarding experience for new developers and reduce ramp-up time.
- Work with security leaders and enforce strong authentication and authorization policies to mitigate security risks. Browser-based DEs present an additional, high-value attack vector, as they become the pipeline through which intellectual property flows.

Sample Vendors

Amazon Web Services; Codeanywhere; Coder; GitHub; Gitpod; Fastly (Glitch); Red Hat; Replit; StackBlitz

Gartner Recommended Reading

Quick Answer: How to Create a Frictionless Onboarding Experience for Software Engineers

Infographic: Platforms and Tools to Scale the Delivery of High-Quality Software

Innovation Insight for Internal Developer Portals

Market Guide for Value Stream Delivery Platforms

Digital Platform Conductor Tools

Analysis By: Roger Williams, Dennis Smith, David Cappuccio

Benefit Rating: Transformational

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

Digital platform conductor (DPC) tools coordinate the various infrastructure tools used to plan, implement, operate and monitor underpinning technology and services for applications and digital products. They enable digital business, regardless of the environments used or who owns them. DPC tools provide a unified view of underpinning technologies and their connection to applications. This augments strategic decision making and improves the value obtained from technology investments.

Why This Is Important

Traditional, cloud and hybrid infrastructure management tools are not designed to provide an integrated view of infrastructure across all environments. Moreover, as infrastructure and operations leaders struggle to manage their portfolio of investments to enable composable business, optimize costs and reduce risks, they need help to fill the gaps in visibility, assurance and coordination. These pressures are fueling the rise of DPC tools that help organizations close these capability gaps.

Business Impact

DPC tools address the following gaps in infrastructure management toolsets:

- Providing visualizations of digital platform performance across all life cycle stages — planning, implementing, operating and monitoring.
- Enabling continual optimal performance and placement of workloads across all environments — on-premises, in the cloud or at the edge.
- Ensuring that improvement initiatives show tangible business value across all technology architectures — compute, storage, middleware and network layers.

Drivers

- Difficulty in maintaining an inventory of all technology infrastructure resources and their dependencies that are aligned with changes to services, applications and components, as well as the configuration of their promised performance levels
- Lack of transparency into spending for hybrid digital infrastructure and how resource capacity aligns to actual application workload demand
- Need to guide where workloads are processed (data center, public cloud, colocation facility, etc.) based on requirements, including capacity, cost and dependency dynamics
- Challenges with estimating the value, efficiency, quality and compliance delivered by hybrid digital infrastructure based on aggregated data from performance analysis tools and other hybrid digital infrastructure management (HDIM) toolset data feeds
- Desire for a single point of entry and reporting for digital platform resource requests, and routing them to appropriate HDIM tooling for fulfillment

- Gaps, duplication and conflicts in data to support application workload migration and business continuity goals, as well as protection of data from accidental deletion or malicious activities
- Inability to confirm compliance of application workloads and digital platforms to identity requirements and security baselines as part of the organization’s cybersecurity mesh approach
- Poor credibility of business cases for digital platform improvements, including: assessing business impact; measuring gaps between current and desired performance; providing oversight of improvement efforts; and validating benefits delivered

Obstacles

- Lack of interoperability: Tool sprawl and difficulties in integration will limit DPC tool adoption. The technology landscape is littered with failed approaches that were intended to support data sharing between vendors.
- Lack of data credibility: The desire for a complete, accurate view of all technology as a precondition for decision making has been around for decades, yet is no closer to being realized. Customers that demand perfect data before they act, and vendors that require complete and accurate data for their tools to function properly, will continue to co-create expectations that will not be met.
- Lack of budget: DPC tools may be viewed as “overhead” that does not have a compelling business case. No one likes paying for something that does not appear to address specific pain points felt today.
- Lack of vendor commitment: Many vendors will be tempted to “DPC wash” their existing offerings and claim that these capabilities are already addressed or can be added for very little cost.

User Recommendations

- Build a DPC tooling strategy that supports digital business ambitions by defining the management elements, environments and technology layers required to meet the organization’s infrastructure needs now and in the future.
- Address measurement and coordination gaps by working with key stakeholders to identify infrastructure value and risk and cost objectives, and making targeted investments in integration, dependency mapping and continuous improvement capabilities.
- Plan for DPC tooling investments by determining which DPC capability aspects are needed in the short, medium and long term. Compare these capabilities to current and future vendor offerings for infrastructure management tooling that can provide initial DPC tool functionality.
- Ensure that DPC tooling investments can deliver sustained value by requiring that DPC tool marketers show how the tool will address current organizational pain points and how it will adapt to future needs as organizational requirements evolve.

Sample Vendors

Amazon Web Services (AWS); Cloudsoft; Flexera; IBM (Turbonomic); LeanIX; Microsoft; OpsRamp; ReadyWorks

Gartner Recommended Reading

Innovation Insight for Digital Platform Conductor Tools

To Maximize the Value of Data Centers, Combine DCIM Tools With Other Sources

Observability-Driven Development

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Observability-driven development (ODD) is a software engineering practice that provides fine-grained visibility and context into system state and behavior by designing systems to be observable. It relies on instrumenting code to unravel a system's internal state, using externally observed data. As part of a shift-left approach to software development, ODD makes it easier to detect, diagnose and resolve unexpected system anomalies early in the development life cycle.

Why This Is Important

Building observable systems can expedite issue resolution because observability data serves as a useful debugging aid. Designing for observability also amplifies the benefits from other resilience engineering practices such as site reliability engineering (SRE) and chaos engineering. ODD enables software engineers and product owners to understand how the software is performing, as well as how it is being used. However, the tools and practices to institutionalize ODD are still emerging.

Business Impact

Built-in observability helps engineering teams ship code confidently, because they can more easily pinpoint likely root causes of failure. This makes it easier to comply with customer-specific and regulatory SLAs. Also, systems designed for observability help unravel code quality issues of performance, security, latency and other nonfunctional attributes early in the software development life cycle (SDLC). This creates software engineering teams that are focused on reliability and security.

Drivers

ODD is driven by the following factors:

- Reliability as a key differentiator — Organizations adopting SRE and chaos engineering practices need the data provided by designing for observability, because these practices are fundamentally data-driven.
- User experience is subjective and is difficult to measure, because it depends on a variety of factors that span the complete technology stack. Some factors, such as last-mile network connectivity, affect user experience (UX), but are difficult to optimize for — unless systems are designed to be observable to extract data related to such signals as response rates and latency.
- Mobile and edge environments present unforeseeable challenges because apps can run on potentially unknown devices and untrusted environments. These production environments can significantly differ from the local environments used to test the application. Therefore, ODD is valuable for capturing and investigating the "unknown unknowns" at runtime.
- Distributed system architectures increase the need for observability, because issues can arise due to previously untested and unexpected component interactions. Troubleshooting issues in distributed applications requires fundamentally different techniques, compared with monolithic or client/server applications. Building in observability narrows down the problem domain and helps engineers inspect the specific problematic component.
- The rise of vendor-agnostic, open-source standards — such as OpenTelemetry (resulting from a merger of OpenCensus and OpenTracing) — has made ODD more accessible. OpenTelemetry standards include protocols to ingest data from traces, metrics and logs. Open standards supported by open-source communities and commercial implementations are increasing developer adoption.

Obstacles

Organizations must overcome the following obstacles in adopting observability-driven development:

- Monitoring (and by association, observability) is commonly viewed as an operational responsibility. Software engineers often lack expertise with observability as a practice and with the tools and frameworks that are used to implement observability.
- The perception that observability can be achieved merely by implementing an "observability" tool. Observability is a property that must be designed and built into systems to ensure that it provides business benefits.
- A piecemeal approach to observability may thwart efforts to adopt ODD at scale. ODD as a standard practice has greater benefits when all components of a system are designed with an observability mindset. For example, distributed tracing requires that all components contributing to the trace be "instrumented" and propagate context for diagnosing response time issues.

User Recommendations

Software engineering leaders responsible for building reliable systems must:

- Adopt ODD as a standard software engineering practice to prepare their teams to handle unexpected and unforeseeable system behaviors and anomalies.
- Treat observability as a critical property that provides insight to resolve errors in production, and provides continual feedback to understand how the software is being used.
- Keep up with the pace of innovation in observability by using open standards and open-source-based technologies, such as OpenTelemetry. Gartner predicts that, by 2025, 70% of new cloud-native applications will adopt OpenTelemetry for observability, rather than vendor-specific agents and SDKs.
- Be wary of vendor hype regarding observability merely as a way to provide access to logs, metrics and traces. Use ODD as a fundamental software engineering practice that improves UX and resilience with granular insight into system state and behavior.

Sample Vendors

Cisco (Appdynamics); Datadog; Dynatrace; Honeycomb; IBM (Instana); Logz.io; New Relic; ObservIQ; ServiceNow (Lightstep); Splunk

Gartner Recommended Reading

Predicts 2022: Modernizing Software Development is Key to Digital Transformation

Innovation Insight for Observability

Monitoring and Observability for Modern Services and Infrastructure

Promise Theory

Analysis By: Roger Williams

Benefit Rating: High

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

Promise theory is an approach to modeling the interactions among entities to understand uncertainty, improve coordination and solve problems. Agents (entities in a system that can independently change) make explicit, voluntary promises about unverified behaviors, which enables them to examine a system and assess the likelihood of a desired outcome. Examples include a promise that work will finish by a certain time, or a system's response time for requests will be less than 50 ms.

Why This Is Important

Promise theory can provide a common language for coordinating expectations among various entities, due to the bottom-up nature of this approach. It provides IT leaders a scalable approach to enforce consistent, declarative and repeatable methods for deploying and managing systems.

Business Impact

Promise theory's open structure, simple concepts and acknowledgment of the uncertainty inherent in promises can enhance trust in digital business. Given the importance of building trust to succeed with agile and DevOps, promise theory can be a catalyst for these efforts. Promises can improve performance management and coordination among product team subject matter experts, and tools such as digital platform conductors, enabling nimble, resilient, decentralized and distributed systems.

Drivers

The use of promises differs from a command-and-control approach to uncertainty, which requires obligations and uses punishments and rewards to enforce them. Command and control often fails, because it can't guarantee results (only consequences), and it is prone to passive resistance that results in actions, rather than desired outcomes. For instance, for reporting purposes, we may prohibit an incident record from being saved unless a category is selected. However, the report will be worthless if it leads to the default or first available option being selected for convenience, rather than the correct value. Similarly, we may insist on 99.999% uptime from a server, yet that, in and of itself, will not make it so. Allowing entities to interact through a public promise offers multiple benefits:

- A declarative model (as opposed to a procedural model)
- Idempotency — repeatable actions that are safe to execute any number of times without causing a change in the state of the system
- Autonomy and scale — for example, agents that manage the state of the system locally through delegation of control, instead of command and control

Obstacles

- Lack of awareness of the concept, making it unlikely that it will become more relevant unless interest in this concept noticeably increases.
- Rejection due to "not invented here" syndrome and/or an unwillingness to accept the uncertainty required for digital business and DevOps success.
- An incorrect perception that the topic is merely theoretical and "ivory tower," rather than practical.
- Pushback from experts on configuration management in various domains who are unwilling or unable to understand why new terminology is needed, rather than other domains simply adopting their approach to configuration management.

- “Promise washing” of work already done, without recasting in terms of business outcomes.
- Reluctance to set clear targets and provide transparency into performance versus those targets.
- Conflict with other behaviors that undermine trust, such as command-and-control management techniques.

User Recommendations

- Begin by identifying the agents that relate to the situation at hand, including the devices and staff that enable the system to deliver the desired result.
- Determine the promises (either explicitly or inherent) that contribute to the result for each agent.
- Identify each of these attributes: the intention and the agent undertaking it voluntarily; how the intention is being communicated to another agent; an assessment of the level and intensity of commitment; what benefits other agents can expect to obtain when the promise is kept.
- Review the network of promises for gaps and conflicts. Agents can then identify what new and changed promises they are willing to make to fill gaps, reduce friction or eliminate conflicts.

Sample Vendors

Cisco; Cloudsoft; Northern.tech; The Apache Software Foundation

Gartner Recommended Reading

Innovation Insight for Digital Platform Conductor Tools

Value Stream Management Platforms

Analysis By: Hassan Ennaciri, Akis Sklavounakis

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Value stream management platforms (VSMPs) enable organizations to manage their software delivery process as a value stream to maximize the delivery of customer value. They provide visibility and traceability to every process in software delivery — from ideation through development to release and production, and extending to feedback from customers. VSMPs continuously measure flow, surface constraints and non-value-adding work, to improve customer value delivery.

Why This Is Important

As organizations scale their agile and DevOps practices, higher-level metrics that assess performance and efficiency of their product delivery is essential. VSMPs integrate with multiple data sources to provide DevOps-related telemetry. These insights enable stakeholders to make data-driven decisions in an agile manner and to correct course as needed. The visualization capabilities of VSMPs help product teams analyze customer value metrics against the cost required to deliver that value.

Business Impact

VSMPs help organizations bridge the gap between business and IT by enabling stakeholders to align their priorities to focus on delivering customer value. VSMPs can provide CxOs with strategic views of product delivery health and pipelines, allowing them to expedite data-driven decisions about future investments in products. These platforms also provide product teams with end-to-end visibility and insight into the flow of work to help them address constraints and improve delivery.

Drivers

- Scaling agile and DevOps initiatives.
- Need for visibility into business, development and operational metrics.
- Optimization of delivery flow by mapping end-to-end processes involved in software delivery to reduce waste and bottlenecks due to cross-team dependencies.
- Need to improve quality and velocity of product deployments.
- Visibility to security and compliance status of products.
- Need for orchestration capabilities to have better traceability.

Obstacles

- VSMPs are not focused on continuous integration/continuous delivery (CI/CD) capabilities. Execution of the delivery pipeline requires use of a custom toolchain or a value stream delivery platform (VSDP).
- VSMPs may lack native plugins for some existing tools and that will require custom integration.
- Rationalization of acquiring another tool with more dashboards.
- VSMPs are still evolving and not all vendors have all the critical capabilities.

User Recommendations

- Improve business outcomes by leveraging real-time, data-driven metrics and value stream insights provided by VSMPs.

- Leverage VSMPs AI-/machine learning (ML)-powered analytics and insights to surface constraints, detect bottlenecks and improve flow.
- Ensure all stakeholders in the value streams are involved in the selection of a VSMP: business leaders, product owners, application leaders and I&O leaders.
- Examine your existing EAP tools or VSDPs as some vendors are expanding their capabilities to include VSMP functionality.
- Pilot the VSMP with a product that has mature DevOps practices to best evaluate what insight can be generated.

Sample Vendors

CloudBees; Digital.ai; Opsera; Plandek; Plutora; Tasktop

Gartner Recommended Reading

The Future of DevOps Toolchains Will Involve Maximizing Flow in IT Value Streams

Hype Cycle for Agile and DevOps, 2021

Flattening the Application Organization — Everyone Must Be Part of the Agile Value Stream

Market Guide for Value Stream Delivery Platforms

Use the Right Metrics in the Right Way for Enterprise Agile Delivery

Performance Engineering

Analysis By: Joachim Herschmann

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Performance engineering is a systematic approach for continuous application performance improvement that involves practices, techniques and activities during each DevOps phase to achieve the performance quality goals of the business. It focuses on the architecture, design and implementation choices that will affect application performance and encompasses the practices that help mitigate performance risks before progressing to subsequent phases.

Why This Is Important

The ability to consistently deliver products that satisfy end-user expectations of scalability, stability, quality of service (QoS), availability and performance has become crucial for digital businesses. Performance engineering promotes a holistic and proactive approach with performance requirements driving the design, development and delivery of products as part of a continuous quality strategy.

Business Impact

Performance engineering includes both “shift left” and “shift right” testing practices and significantly improves an organization’s ability to consistently deliver solutions that delight customers by exceeding their performance expectations. It provides the framework for application performance excellence that drives value and supports the realization of business outcomes for customers.

Drivers

- Raised end-user expectations for application quality, specifically nonfunctional characteristics, such as performance efficiency.
- The need to ensure business continuity under changing usage patterns, network topologies and data volumes.
- The need to optimize the use of modern microservices-based architectures, multicloud deployments, and automation and integration capabilities of modern application platforms.
- Support for different migration scenarios, such as lift and shift, replatforming or refactoring the architecture of packaged or on-premises apps.
- The need to manage performance and scalability across different cloud providers, such as Amazon Web Services (AWS), Google or Microsoft Azure, to ensure the ability to shift from one operator to another without a change in user experience.
- Cost optimization for SaaS/PaaS services that makes use of dynamic infrastructure to spin up (and down) testing resources as needed.

Obstacles

- Focusing only on tools and technology: Performance engineering requires a change in organizational culture. Tools enable quality but won’t solve problems on their own.
- Organizational inertia: Departmental silos, traditional top-down management structures and a lack of experience with managing quality continuously can impede adoption.
- Lack of clear goals: Successful performance engineering requires clear goals that are aligned with the priorities of the business.
- Internal pushback: Performance engineering requires engaging stakeholders throughout the organization and empowering them to be more accountable and to seek out opportunities for improvement. Such a holistic approach can be seen as restrictive and requires consensus across all team members.

- Limitation to performance testing only: Performance engineering includes designing with performance in mind, building the product with clear performance objectives and facilitating the discovery of issues early in development.

User Recommendations

- Create awareness for nonfunctional characteristics such as performance efficiency. ISO/IEC 25010 provides a template for understanding quality characteristics and includes performance efficiency as one of the top-level nonfunctional domains.
- Foster a proactive performance quality strategy that makes performance an explicit requirement and verifies that performance goals are met and user satisfaction meets expectations.
- Allocate ownership and appoint staff with the required skills needed for performance engineering by identifying the required roles, technologies and practices.
- Establish relevant performance quality metrics based on the joint objectives that the business and IT are trying to accomplish.
- Collaborate with I&O leaders and establish a feedback loop by leveraging performance information from production and real users.

Sample Vendors

AppDynamics; Dynatrace; Keysight (Eggplant); Micro Focus; Perforce; Tricentis

Gartner Recommended Reading

Innovation Insight for Performance Engineering

How to Identify and Resolve Application Performance Problems

Quick Answer: Which Non-Functional Software Quality Characteristics Can Make or Break Your Product?

Quick Answer: What Are the Essential Shift-Right Practices for Testing in Production?

Improve Software Quality by Building Digital Immunity

Chaos Engineering

Analysis By: Jim Scheibmeir, Dennis Smith

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Chaos engineering is the use of experimental and potentially destructive failure testing or fault injection to uncover vulnerabilities and weaknesses within a distributed complex system. Chaos engineering tools provide the ability to systematically plan, document, execute and analyze an attack on components and whole systems throughout the life cycle of the system. This planning may include the injection of random timing or attack executions.

Why This Is Important

Many organization's stake success on test plans that overemphasize functionality and underemphasize validating the system's reliability. The distribution and complexity of systems makes understanding them more elusive. Chaos engineering (CE) moves the focus of testing a system from the "happy path" — to testing how the system can gracefully degrade or continue to be useful and secure while under various levels of impact. By using the CE practice, system knowledge and documentation are improved.

Business Impact

With chaos engineering, we aim to minimize time to recovery and change failure rate, while maximizing uptime and availability. Addressing these elements helps improve customer experience, customer satisfaction, customer retention and new customer acquisition.

Drivers

- Gartner's 2020 Achieving Business Agility with Automation, Continuous Quality and DevOps Survey found that 18% of respondents were currently using or planning to use chaos engineering to improve software quality.
- Complexity in systems and increasing customer expectations are the two largest drivers of this practice and use of the associated tools.
- As systems become more rich in features, they also become more complex in their composition, and more critical to digital business success.
- Overall, chaos engineering helps organizations become more resilient across their processes, knowledge and technology.

Obstacles

- The first obstacle to chaos engineering is perception. Within many organizations, the predominant view is that the practice is random, done first in production, and due to these leads to more risk than less.
- Another obstacle to chaos engineering is organizational culture and attitude toward quality and testing. When quality and testing are only viewed as overhead, then there will be a focus on feature development over application reliability.

- Another common obstacle is simply gaining the time and budget to invest in learning the practice and associated technologies. Organizations must reach minimum levels of expertise where value is then returned.

User Recommendations

- Utilize a test environment-first approach by practicing chaos engineering in preproduction environments.
- Incorporate chaos engineering into your system development, CI/CD or testing processes.
- Build-out incident response protocols and procedures, as well as monitoring, alerting and observability capabilities in tandem with advancement of the chaos engineering practice.
- Utilize scenario-based tests — known as "game days" — to evaluate and learn about how individual IT systems would respond to certain types of outages or events.
- Investigate opportunities to use chaos engineering in production to facilitate learning and improvement at scale as the practice matures. However, be warned that Gartner believes that there are still very few organizations purposely using chaos engineering in their production environments.
- Formalize the practice by adopting a platform or tool to track the activities and create metrics to build feedback for continuous improvements.

Sample Vendors

Amazon Web Services (AWS); ChaosIQ; Gremlin; Harness; Microsoft; Steadybit; Verica

Gartner Recommended Reading

Improve Software Quality by Building Digital Immunity

Innovation Insight for Chaos Engineering

How to Safely Begin Chaos Engineering to Improve Reliability

At the Peak

AIOps Platforms

Analysis By: Gregory Murray, Pankaj Prasad

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

AIOps platforms analyze monitoring data, events and operational information to augment, accelerate or automate IT operations. AIOps Platforms have five characteristics: cross-domain data ingestion; topology assembly from implicit and explicit sources of asset relationship and dependency; correlation between related or redundant events associated with an incident; pattern recognition to detect incidents, their leading indicators or probable root cause; and association of probable remediation.

Why This Is Important

The combination of increasing application complexity, monitoring tool proliferation, and increasing volumes and varieties of telemetry has shifted complexity from gathering data to interpreting data. AIOps Platforms apply machine learning and data analytics to classify and cluster diverse telemetry signals in near-real time, at scale, and in ways that can exceed human capacity. These inferences can augment human analysis, accelerate human response, or automate a process to resolve an issue.

Business Impact

AIOps Platforms deliver value through:

- Agility and productivity: By reducing alert fatigue through identification and correlation of related events, operators can focus on fewer, more critical events.
- Service availability and triage cost: By reducing the time and effort required to identify root causes and augmenting, accelerating, or automating remediation.
- Increased value from monitoring tools: By unifying events from siloed tools and learning actionable event patterns across domains.

Drivers

Demand for AIOps Platform capabilities is accelerating and is fueled by:

- Increasing complexity: Organizations use an increasingly complex mix of IT assets that rely on a highly integrated combination of on-premises assets, cloud IaaS/PaaS providers and SaaS platforms to deliver solutions.
- Increasing monitoring expectations: Investments and improvements in monitoring and the pursuit of observability are generating more data from more sources. Increasing demand and advances in monitoring trends, like application performance management (APM) and digital experience monitoring (DEM), present operators with extremely detailed views into their business applications and the end-user experience. Effective use of this additional data requires near-real-time analysis and rationalization with telemetry from related assets and services.
- Demands for reliability: Shifts in roles and responsibilities driven by modern operating models, like DevOps and SRE, in the pursuit of greater availability and faster incident resolution. AIOps Platforms enable agility by offloading some of the mechanical tasks of event triage, root cause

analysis and solution identification. This both accelerates response for common issues and frees up human creative capacity for novel events and business priorities.

As these trends, themselves, are accelerating, expect the case for AIOps Platforms to continue to accelerate.

Obstacles

- Unrealistic expectations: Hype is a major obstacle to AIOps Platform adoption. Surveyed clients struggle to separate claims of AI and magical automation from practical, achievable use cases. Success with AIOps requires clarity on the true capabilities and practical limitations of the solutions.
- Time to value for impactful use cases: AIOps Platforms learn through observation. They learn normal data ranges and patterns, and associate a solution with these patterns. This can take time depending on the frequency of occurrence. Developing accurate detection models for rare events can take months.
- Market shifts and maturity: The AIOps market is in flux. Monitoring vendors are moving up the stack, AIOps Platform vendors are reaching into monitoring domains, and ITSM vendors see AIOps capabilities as extending their reach. Expect both technology advancements and market shifts to change the definition of "state of the art."

User Recommendations

- Establish clear, realistic use cases for an AIOps Platform pilot and validate them individually, rather than all at once. This approach helps reveal pockets of potential value that might be missed when evaluating only the aggregate impact. Ultimately, this fundamental step underpins an eventual strategy, while scoping the vendor landscape, clarifying the telemetry requirements and separating hype from reality.
- Layer the domain-centric AIOps features within monitoring tools with the cross-domain analysis of an AIOps Platform. This approach enables efficient data ingestion and analysis, and the surfacing of insights across domains.
- Do not require automation outcomes for all AIOps applications. There is tremendous value in accelerating and augmenting human activity. These approaches often avoid the challenge of the probabilistic uncertainty combined with automated change in production environments.

Sample Vendors

BigPanda; IBM; Moogsoft; ServiceNow; Splunk

Gartner Recommended Reading

Market Guide for AIOps Platforms

Infographic: Artificial Intelligence Use-Case Prism for AIOps

Solution Path for Adopting AIOps

Infographic: AIOps Architecture for Analyzing Operational Telemetry

Solution Criteria for AIOps Platforms

Continuous Compliance Automation

Analysis By: Daniel Betts, Hassan Ennaciri

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Continuous compliance automation (CCA) integrates compliance and security policy enforcement into DevOps delivery pipelines. CCA codifies and continuously applies compliance policies and controls, while monitoring, correcting and protecting against vulnerabilities resulting from coding defects and misconfiguration. It reduces the number of manual execution steps involved in adhering to regulatory requirements, enhancing consistency, traceability and auditability.

Why This Is Important

Increased focus on security and compliance improvements drives enterprise investments in compliance automation used to secure the software supply chain. CCA improves release velocity and reliability while simplifying compliance enforcement via policy-driven, automated controls,. This improves compliance without slowing the software delivery pipelines. Traditional compliance practices are incompatible with continuous software delivery processes — leading to slower delivery and unexpected, expensive remediation work.

Business Impact

Organizations evolving DevOps practices can minimize risks and penalties by embedding automated compliance into their delivery pipelines. CCA enables organizations to integrate compliance into all phases of the delivery pipeline and consistently enforces compliance policies without sacrificing operational agility.

Drivers

- Organizations are facing an increasing number of regulatory obligations and more stringent enforcement, so automating compliance will become even more valuable to I&O leaders as they strive to maximize flow.

- Additional compliance requirements continue to be added, and require support with limited delay.

- Compliance activities are increasingly executed through automated testing, which delivers increased efficiency for developers and reduces the risk of compliance audit failures.
- As cloud-native application architectures and development models become more pervasive, integrating compliance into the toolchain will become more feasible and common.

Obstacles

- Most CCA tools only target one development or delivery activity. No vendor provides capabilities across all elements of the delivery value stream. DevOps teams must integrate multiple tools into their value streams to provide compliance coverage across development and delivery activities.
- Failure to engage with compliance and security subject matter experts (SMEs) early in the development life cycle can lead to problems. Early input from compliance and security SMEs will help I&O leaders account for security and compliance requirements and audit failures.
- A lack of rule set understanding and consistent implementation can be an impediment to CCA. While it is important to capitalize on the acceleration that vendor rule sets can provide, it is vital that they are understood by organizational compliance teams and implemented consistently in order to provide maximum value.
- Poorly implemented CCA presents a business risk. If it is assumed that by implementing CCA, delivered software becomes compliant without additional effort, organizations will face increased risk of compliance failure.

User Recommendations

- Adhere to compliance, governance and security requirements while creating a leaner operating environment. CCA tools enable DevOps teams to achieve both goals — improving value stream delivery and mitigating risks.
- Implement a shift-left approach to ensure compliance controls are understood earlier in the development process. Implement automated compliance checks at every phase of the pipeline, demonstrating a “shift-secure” approach.
- Invest in tools that enable CCA at scale and can provide a continuous approach to prevent, detect and correct audit failures.
- Select tools that can integrate into value stream delivery platforms to enable security and compliance checking.
- Enforce security and compliance across all domains, including databases, application code, infrastructure and open-source software. Since there is no single vendor tool that covers all these domains, DevOps teams must use multiple tools and integrate across all phases of the delivery pipeline.

Sample Vendors

Anitian; JFrog; Prisma Cloud; Rapid7; Redgate; Snyk; Sonatype; Styra; WhiteSource

Gartner Recommended Reading

Innovation Insight for Continuous Compliance Automation

3 Steps to Ensure Compliance and Audit Success With DevOps

3 Steps to Integrate Security Into DevOps

How to Build and Evolve Your DevOps Toolchains

Market Guide Value Stream Delivery Platforms

Securing Development Environments

Analysis By: Manjunath Bhat

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Securing development environments involves protecting the complete software development environment including but not limited to source code repositories, CI/CD pipelines, application artifacts and user identity information. Development environments become a primary attack vector since they contain IP, trade secrets and user credentials. The increased incidence of supply chain attacks not only puts the affected organization at risk but also organizations participating in the ecosystem.

Why This Is Important

IT security teams are often unaware of the security posture and risks associated with software development and delivery tools. However, given the increased risks due to software supply chain attacks, widespread use of open-source tools and exposure due to remote ways of working, securing the development environment becomes paramount. As organizations digitize their processes and launch new digital services, the software environment is where most of the intellectual property originates and resides.

Business Impact

Securing development environments helps organizations improve compliance with government and vertical-specific regulations. Software supply chain attacks subject the organization to both financial and reputational damage. The complexity and heterogeneity of DevOps pipelines make it difficult to secure the environment while organizations are planning to support hybrid ways of working, expanding the attack surface. Easy access to open-source tools belies the need to govern and secure their usage.

Drivers

- Software supply chain attacks are becoming increasingly sophisticated, with malicious actors exploiting weaknesses at every stage in the software procurement, development and delivery life cycle. This includes everything from injecting malicious code into open-source packages to installing back doors in postdeployment software updates.

- Git is now the de facto version control system (VCS). Although Git-based VCSs including BitBucket, GitHub and GitLab support access policy controls, branch protection and secrets scanning capabilities, these controls are not enabled by default and must be explicitly set.

- Software development environments span multiple distributed systems, platforms and tools, communicating with each other in the software delivery life cycle using privileged service accounts. For example, build machines communicate with source code repositories to pull source code and artifact repositories to pull common packages, and connect to deployment targets to deploy binaries. The risk is that the tools and services default to running with elevated system privileges without privilege access controls in place.

- The COVID-19 pandemic forced a lot of software developers to work from home, keeping proprietary IP on laptops connected through their home network. Current trends suggest this has become a permanent option for many companies.

- With software becoming the engine for innovation, intellectual property is increasingly reflected in easily copied assets like scripts, source code and configuration files.

Obstacles

- The need for collaborative approaches such as innersourcing and the need for governing source code access often requires fine-grained permissions that may not be supported by current collaboration and source code repositories.

- The need to provide remote access to continuous integration/continuous delivery (CI/CD) pipelines, version control systems and artifact registries creates blindspots for security teams when developers seek to access them from personal devices or use unsecure “client-side” apps that can exfiltrate content.

- Development teams that view security controls as creating needless friction might look for ways to circumvent them or use unapproved tools. This is often cited in relation to developers requiring root access on their local machines to install experimental development tools, making it easy to thwart controls.

- Attack vectors from new tools like CI/CD pipelines and code repositories are often difficult to recognize by developers who don’t see them as unsafe and by security who may not have a deep familiarity with the tools.

User Recommendations

- Configure CI/CD pipelines with the elevated security and access controls, as they may be turned off by default. CI/CD systems, if left unsecured, enable attackers to manipulate build pipeline definitions allowing malicious code to pass through or redirect releases to a malicious delivery target.
- Harden the software delivery pipeline by configuring security controls in CI/CD tools, securing secrets using secrets management tooling, and signing code and container images.
- Adopt an immutable infrastructure approach, leverage hardened baselines within infrastructure automation practices and maintain vetted builds in VCSs for organizational teams to provide consistent, "known good" baselines and to avoid configuration errors and unauthorized alterations.
- Secure the development environment by governing access to resources using principles of least privilege and a zero-trust security model.

Sample Vendors

Apiiro; BluBracket; Chainguard; Cider; Cycode; GitGuardian; Legit Security

Gartner Recommended Reading

Best Practices for Securing Continuous Delivery Systems and Artifacts

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Innovation Insight for SBOMs

Value Stream Delivery Platforms

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Value stream delivery platforms (VSDPs) aim to provide fully integrated capabilities that enable continuous delivery of software. These capabilities include planning, version control, continuous integration, test automation, release orchestration, continuous deployment, monitoring, security automation and flow analytics using value stream metrics. VSDPs integrate with infrastructure and compliance automation tools to automate infrastructure deployment and policy enforcement.

Why This Is Important

Value stream delivery platforms enable organizations to simplify building and managing delivery pipelines. They reduce the complexity involved in orchestrating, integrating and governing pipeline activities. In addition, the prebuilt integration between different components of the platform improves management, updates and eases maintenance. A platform approach leads to improved visibility and traceability in the complete application development value stream.

Business Impact

VSDPs are the software delivery pipeline that enables continuous delivery of business value. The seamless integration, automation, extensibility and shared visibility between development and operations workflows help bridge the silos that exist between development and operations teams. Using a common platform for development, security and operations accelerates agile transformation and helps organizations adopt a product and platform team operating model.

Drivers

- Agile ways of working with automation and collaboration being the focal points.
- Need to address toolchain technical debt as organizations adopt cloud-native architectures.
- Need for improved developer experience, satisfaction and productivity.
- Need for deeper collaboration and shared visibility across all IT teams (Dev, Sec and Ops).
- The emergence of product and platform engineering teams drives digital transformation initiatives.
- Competitive pressure demands a faster time to market for digital products and services.
- The proliferation of tools and the ensuing complexity of integrating those tools as well as the lack of skills to do so at scale.
- Industry traction around value stream management — VSDPs will increasingly bundle capabilities to analyze value stream metrics and extend the usability of the platforms to business stakeholders.
- The growing importance of open-source software (OSS) as customers benefit from the latest innovation from the OSS community. VSDPs take advantage of OSS tooling with their solutions.
- Increased maturity of “as code” approaches to automate security and compliance (via policy as code) and infrastructure (via infrastructure as code).

Obstacles

- Organizations that want to unlock the full benefits of VSDPs must be willing to replace an existing toolchain — either completely or in part. However, this impedes teams and product owners that are resistant to change or view the change as a disruption to their ways of working.
- Organizations accrue technical and skill debt over time due to outmoded automation workflows and legacy applications. This hinders teams from adopting new tools.
- The potential for vendor lock-in presents a constraint as well. Dependency on a single provider for a majority of their software development needs increases concentration risk and lowers

bargaining leverage. Gartner cautions VSDP buyers that none of the vendors currently provide the full set of software delivery capabilities.

- VSDPs currently fall short in providing the required value stream analytics that organizations can use to measure and improve the flow of value in the software delivery life cycle.

User Recommendations

- Scale deliver capability by providing VSDPs as a self-service platform to reduce overhead, lower complexity, and ensure consistent and templatized workflows across multiple teams.
- Improve the flow of value by streamlining the software delivery life cycle with VSDPs that provide enhanced visibility, traceability, auditability and observability across the DevOps pipeline.
- Support inner source efforts across multiple development teams by adopting VSDPs as a part of internal developer portals for knowledge sharing, code reviews, code sharing and issue tracking.
- Reduce inconsistency between CI/CD pipeline definitions between teams by leveraging declarative and shareable pipeline capabilities in VSDPs.
- To fully reap the business benefits of VSDPs, organizations must adopt agile methods and practices.

Sample Vendors

Atlassian; CloudBees; CodeNOW; Copado; Digital.ai; GitHub; GitLab; Guide-Rails; Harness; JFrog

Gartner Recommended Reading

Market Guide for Value Stream Delivery Platforms

Beware the DevOps Toolchain Debt Collector

Solution Path for Agile Transformation

Solution Path for Continuous Delivery With DevOps

Keys to DevOps Success

Continuous Quality

Analysis By: Joachim Herschmann, Jim Scheibmeir

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Continuous quality is a systematic approach toward process improvement to achieve the quality goals of business and development. A continuous quality strategy fosters a companywide cultural change to achieve the goal of making “quality” the responsibility of all. It synchronizes quality assurance and testing with DevOps processes, and encompasses the practices that help mitigate risks, before progressing to subsequent stages of the software development life cycle.

Why This Is Important

Many DevOps organizations are practicing continuous integration and continuous deployment, yet a continuous approach to quality is often missing. The ability to consistently deliver business value with high quality has become critical for organizations seeking to mature their DevOps processes. Continuous quality encourages a holistic and proactive approach with functional and nonfunctional requirements driving the design, development and delivery of products.

Business Impact

The adoption of a continuous quality strategy significantly improves an organization’s ability to serve and delight its customers. Continuous quality helps to deliver solutions at a greater release rate and with fewer defects than traditional quality control practices. It provides a framework for operational excellence that drives value, supports the realization of business outcomes for customers, and streamlines operational processes.

Drivers

- Raised end-user expectations for application quality require a shift to a more holistic view of what constitutes superior quality that delights users.
- The pressure to innovate rapidly in order to launch differentiated products in the market quickly without compromising on quality.
- The ability to consistently deliver business value with consistently high quality to mature DevOps processes.
- The need to ensure that teams are equipped to create a superior user experience, build features that fit the market’s timing, and enable the characteristics of an application that deliver value faster than they create technical debt.

Obstacles

- Lack of clear goals: Successful continuous quality requires clear goals aligned with the priorities of the business.
- Internal pushback: Continuous quality requires engaging stakeholders across the organization and empowering them to be more accountable. Such a holistic approach can be seen as restrictive and requires consensus on usage across all team members.

- Loss of productivity: Changing organizational culture and engaging in new practices require significant investment and time. This will impact current timelines and can cause a decrease in productivity prior to reaching steady productivity.
- Limitation to testing only: Continuous quality includes designing a product with quality in mind, building it with clear quality objectives, and facilitating the discovery of issues early in development.
- Focusing only on tools: Continuous quality requires a change in organizational culture. Tools are enablers of quality but tools on their own won't solve problems.

User Recommendations

- Move away from the traditional application- or project-centric model of quality to a holistic quality approach by adopting an ecosystem-centric view of quality and a focus on business outcomes.
- Accelerate product delivery by championing a continuous quality mindset and involving stakeholders across the organization.
- Allocate ownership and appoint staff with skills needed for continuous quality by identifying the required roles, technologies and practices.
- Enable collaboration with user experience (UX) designers and customer experience (CX) teams to infuse quality right from the inception of an idea.
- Establish relevant quality metrics based on the joint objectives that the business and IT are trying to accomplish.
- Task teams with developing continuous quality practices before choosing tools.

Gartner Recommended Reading

Innovation Insight for Continuous Quality

Tool: Craft and Communicate a Continuous Quality Strategy

Infographic: Sketching a Continuous Quality Strategy Over Coffee

Improve Software Quality by Building Digital Immunity

Quick Answer: What Are the Key Skills to Drive Software Quality?

DevOps Test Data Management

Analysis By: Dale Gardner

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

DevOps test data management is the process of providing DevOps teams with data to evaluate the performance, functionality and security of applications. It typically includes copying production data, anonymization or masking, and, sometimes, virtualization. In some cases, specialized techniques, such as synthetic data generation, are appropriate. Given potential compliance and privacy issues, the effort frequently involves members of application and data security teams.

Why This Is Important

Test data management is inconsistently adopted across organizations, with many teams copying production data for use in test environments. As organizations shift to DevOps and the pace of development increases, this traditional approach is increasingly at odds with requirements for efficiency, privacy and security, and even the increased complexity of modern applications. This opens organizations to a variety of legal, security and operational risks.

Business Impact

Quick provisioning of test data helps ensure the pace of development isn't slowed. It's also increasingly important to remain compliant with the growing number of privacy mandates to which organizations are subject. This helps avoid fines and remediation and mitigation costs, along with the inevitable delays associated with audits and investigations. Finally, by providing application teams with anonymized or synthetic data, the risk of data breaches is reduced.

Drivers

- Test data management is generally viewed as a mature, relatively uncomplicated, practice. However, the reality is the combination of the increased pace of development from DevOps and a growing number of privacy mandates and constraints have stressed traditional approaches — prompting use of virtualization as well as alternative masking and protection techniques.

- More traditional test data management has been inconsistently adopted, with many organizations either simply using copies of production data in unsafe environments or generating "dummy data" (distinct from emerging synthetic data generation techniques) that doesn't accurately reflect production data. The data privacy requirements and complexity issues noted have prompted organizations to revisit and update their processes with an eye toward scalability and automation. Updated technologies may also be a requirement. For example, requirements for speed and agility have created a need for data virtualization tools.

- Data protection is cited by most Gartner clients in inquiries regarding test data management. Privacy and data protection requirements mean it's no longer safe to simply provide development teams with a copy of production data. The practice leaves organizations open to increased risk of regulatory violations, data breaches and other security issues.

- With modern applications relying on an increasing number of interconnected data stores (many of which are technologically vastly more complex), applications and APIs to function, testing has become more complex. Such complexity demands that tools support the ability to coordinate and synchronize changes across different data stores to ensure relational consistency while still addressing security and speed mandates.

Obstacles

- In the absence of a strong culture of security, processes and technologies to protect sensitive information during development and testing will encounter friction. Conflicting needs for rapid development and privacy require attention to a mix of organizational and cultural issues to strike a balance across groups.
- Responsibility for test data management in organizations has been shared by application development and database administration. New technologies and processes may shift those responsibilities to include security, complicating organizational dynamics and potentially creating the need for additional staffing.
- Implementation can be a burden, especially where little or no data sensitivity classification has been done. This must be accomplished before teams can proceed with required data transformation and masking. These efforts are typically combined with an analysis of data relationships so that relational integrity can be assured.

User Recommendations

- Involve stakeholders such as application development and test teams (to understand consumption patterns and needs), data management, privacy and security teams, compliance teams, other security teams, and legal counsel — as appropriate.
- Document existing test data management practices so tools and processes can be evaluated against data protection mandates.
- Coordinate with other teams to avoid duplication of effort and tooling since data masking tools may also be used by analytics teams (e.g., to provide data for machine learning or other purposes).
- Evaluate data masking tooling by considering support for databases and other stores, data discovery capabilities, types of masking supported, and the ability to coordinate change to ensure consistency (e.g., key fields) across multiple sources.
- Evaluate data virtualization for DevOps use cases where frequent updates to test data are required. Virtualization can speed the process of providing copies of safe data.
- Determine whether synthetic data is appropriate in cases where suitable data doesn't exist or reidentification risks are high.

Sample Vendors

Actifio; BMC Compuware; Broadcom (CA Technologies); Delphix; Hazy; Informatica; K2View; Mage; Micro Focus; Solix Technologies

Gartner Recommended Reading

Market Guide for Data Masking

Elevating Test Data Management for DevOps

Innovation Insight for Synthetic Data

Platform Ops

Analysis By: Daniel Betts, George Spafford

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Platform ops is an approach to scaling DevOps that involves dedicating a team led by a product owner to the provisioning and operation of a codified, highly automated, shared self-service customer-driven platform. This platform team enables multiple product teams to accelerate delivery while managing quality, platform security and compliance, and standardization.

Why This Is Important

Platform ops practices are adopted by organizations needing to scale DevOps initiatives in order to reduce overlap and redundancy, enable economies of scale, and establish high standards of security, compliance and governance. These practices require a culture that supports continuous learning and improvement, highly skilled automation practices, and usage of infrastructure as code (IaC) practices.

Business Impact

Using a platform and product approach can help enable the business to respond faster to threats and opportunities while also managing cost and risk. The product teams can focus on customer requirements and, in turn, the platform teams can specialize in enabling the product teams. This makes it well-suited to support digital business initiatives such as analytics, mobile applications, portals and so forth.

Drivers

- Many large organizations are adopting platform models to scale DevOps, where platform teams provide a known, good path to consistently delivering customer value.
- Improve developer experience to attract and retain software engineering talent, as well as more effectively delivering software.

- Difficulty in providing enough operations expertise in product teams as they scale, resulting in slower delivery cycles, software defects and frustration.
- Full-stack DevOps team structures are not scalable and often lead to duplication and competing demands that thwart long-term success.
- Challenges to ensure high standards of governance and production efficiency when product teams recreate platforms' capabilities inconsistently from team to team.

Obstacles

- Before embarking on platform ops — picking technologies, hiring a team — you should determine that platform ops is right for your organization to scale DevOps.
- If your organization is just starting DevOps or struggling to find DevOps maturity, platform ops may not be the right approach. Efforts should be focused on an iterative learning approach and scale when ready.
- Having sufficient resources to fulfill the roles required to deliver a shared platform is a challenge for many organizations.
- There can be considerable ramp-up time to enable platform team members with the software engineering, testing and programming skills needed to effectively deliver a platform.
- Organizational size and maturity to adopt platform ops can impact the decision to use platform ops.

User Recommendations

- Appoint a product owner for the platform to collaborate with the platform's customers, prioritize platform feature development, lead marketing and championing the platform.
- Establish dedicated platform teams to maintain and continuously improve shared, self-service platforms.
- Create a cross-functional team to support the evolving needs of product teams. This platform team needs longevity and dedicated funding throughout the entire life cycle of the platform.
- Encourage platform team members to work closely with product teams. They must prioritize tools the teams need to be more productive, provide education on how to use the platform and implement architectures that are conducive to reusable shared platforms.
- Collaborate with other stakeholders — such as security, compliance, finance — to ensure that additional requirements are integrated into the capabilities that the platform offers.

Gartner Recommended Reading

Why DevOps Success Requires Platform Teams

Platform Teams and AIOps Will Redefine DevOps Approaches by 2025

Case Study: Infrastructure Platform Teams for Self-Service Delivery

Building a Platform for Product Team Productivity (adidas)

Autonomous Testing

Analysis By: Joachim Herschmann, Jim Scheibmeir

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Autonomous testing comprises AI- and ML-based technologies and practices to make software testing activities independent from human intervention. It continuously improves testing outcomes by learning from the data collected from performed activities. It extends traditional test automation beyond the automated execution of test cases to include fully automated planning, creation, maintenance and analysis of tests.

Why This Is Important

Software engineering leaders seeking to release faster without degrading quality are looking for more efficient ways of testing, which includes all phases of the testing life cycle. As such, autonomous testing is a driver for a holistic approach to automating the broader set of quality and testing activities related to requirements quality, design quality, code quality, release quality and operational resilience in an integrated way. This increases the degree of autonomy for those activities.

Business Impact

The adoption of autonomous testing has the potential to significantly improve an IT organization's ability to serve and delight its customers. It can be an enabler for adjusting testing scenarios and overall software quality parameters as part of a continuous quality initiative aimed at optimizing end-user experience. It will also help to constitute a closed-loop system that provides continuous feedback about critical quality indicators.

Drivers

- A high dependency on human expertise and interaction limits how quickly modern digital businesses can design, build and test new software.
- Where automated testing is already in place, current levels of automation often remain below expectations due to a continued dependency on human intervention in maintaining the automation as applications under test (AUT) evolve.

- The pressure to innovate quickly for market differentiation without compromising on quality relies both on a higher velocity and a higher degree of autonomy of the related activities.
- While delivery cycle time is decreasing, the technical complexity required to deliver a positive user experience and maintain a competitive edge is increasing. The answer is not more testing but more intelligent testing enabled by AI technologies.

Obstacles

- Risk of doing nothing: Waiting until feature-complete autonomous testing solutions are available leads to a loss in competitive advantage, and reduced agility and innovation.
- Unrealistic goals: Underestimating the time required to acquire new skills and setting wrong expectations about the time required to become successful can be obstacles.
- Managing data quality: Gathering, cleaning and processing of data and training of the model are not trivial tasks and require adequate skills. Moreover, they are not yet autonomous processes.
- Internal pushback: Autonomous testing requires significant investment in new areas such as data science and analytics. While this will motivate some team members, others may see the approach as a threat to their known way of working and they may be afraid to adjust.
- Immaturity of tools: Currently available tools are still relatively new, have a narrow scope of technology coverage and still need to prove their value.

User Recommendations

- Set the right expectations about where autonomous testing can provide value, what its current limitations are and what is needed for it to be successful.
- Maximize the impact of autonomous testing by leveraging it as an enabler of a systematic approach to achieve the quality goals of business and development. Focus on key business value enablement and determine where it can help with revenue, cost and risk management.
- Familiarize yourself with advanced analytics and ML. Invest in augmented analytics tools that employ ML algorithms to mine existing data for current and future projects.
- Allocate ownership and appoint staff with the required skills such as data analytics by identifying the required roles, technologies and practices.
- Select tools that best match available skills and development style by assessing application profiles and different teams' needs.

Sample Vendors

ACCELQ; Applitools; Avo Automation; Diffblue; Functionize; mabl; ProdPerfect; test.ai; testRigor

Gartner Recommended Reading

Market Guide for AI-Augmented Software Testing Tools

Innovation Insight for Continuous Quality

Improve Software Quality by Building Digital Immunity

Infographic: Artificial Intelligence Use-Case Prism for Software Development and Testing

Dojos

Analysis By: George Spafford

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

A dojo is a gathering place (real or virtual) where students learn and practice together in an experiential manner in accordance with their skill level. Students can expand their skill set by learning about new methods and tools, while those with more experience can share and refine skills.

Why This Is Important

Continuous learning and growth are essential for DevOps progress and scalability. Without experiential approaches to embed and support a continual, iterative approach to mindsets, skills and culture, many DevOps initiatives tend to fail. The use of a dojo can be layered with other methods to improve organizational learning for people, ranging from newcomers to experienced ones.

Business Impact

Dojos are experiential learning opportunities that can support shifts in culture, enable new ways of working and help adjust mindsets and behaviors. These are all critical elements for success when it comes to digital transformation, and a foundational shift in focus from building systems, components and code to delivering business value to internal customers.

Drivers

- Digital business initiatives are requiring new approaches and technologies from product teams.
- These approaches include moving systems to the cloud, DevOps, infrastructure as code and platforms — all of which require the staff to learn and change.
- I&O leaders are seeking organizational learning methods that can scale to help large numbers of employees learn.

Obstacles

- General misconceptions around how to approach dojos include huge investments being required to start dojos.
- People wanting to start dojos tend to start their initiative in an expanded manner requiring too many approvals, which leads to unwanted delays.
- Finding skilled practitioners who can also be effective instructors for the dojo is challenging.

User Recommendations

- Start small with the resources you have available and demonstrate value.
- Attract people to the dojos by understanding and delivering value that resonates with them.
- Use practitioners with real-world knowledge to teach in an experiential style so that students can learn by doing.
- Integrate the dojo into the DevOps strategy, and understand which classes need to be held to help prepare people to move in the required direction.
- Co-create a dojo charter with all stakeholders by establishing a shared vision for the dojo, and using metrics that track progress and success.
- The dojos are but one method of organizational learning. Clients tend to use three or more approaches. Investigate others, such as communities of practice, job rotations, mentoring, internal technical conferences and so forth.

Gartner Recommended Reading

Agile Learning Manifesto

Rebalanced Technical Skills Portfolio (Nationwide)

4 Steps to Create a DevOps Dojo That Accelerates Learning and Cultural Change

Programmable Infrastructure

Analysis By: Nathan Hill, Philip Dawson

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Programmable infrastructure is the concept of using and applying methods and tooling from the software development area to management of IT infrastructure. This includes, but is not limited to, APIs, immutability, resilient architectures and agile techniques.

Why This Is Important

Programmable infrastructure ensures optimal resource utilization while driving cost-efficiencies. A continuous-delivery approach requires continuous insight and the ability to automate application responses. Moving to an API-driven infrastructure is the key first step to enabling anti-fragile and sustainable automation through programmatic techniques.

Business Impact

Greater value (rather than cost reduction) can be achieved via programmable infrastructure's ability to drive adaptive automation — responding faster to new business infrastructure demands, driving service quality and freeing staff from manual operations. Programmable infrastructure helps reduce technical debt and enables a sustainable and highly responsive IT infrastructure service to the business.

Drivers

- Programmable infrastructure strategies can be applied to private cloud, hybrid cloud and infrastructure platforms, as well as public cloud. Demand for programmable infrastructure grows as heterogeneous infrastructure strategies are embraced.
- Programmable infrastructure is needed to manage the life cycle of infrastructure delivery from provisioning, resizing and reallocation to reclamation, and in the case of external resources, manage elasticity and the termination of consumption.
- Programmable infrastructure is needed to optimize and reduce the dependency on the infrastructure life cycle. More importantly, it enables the desired (performance, cost, speed) infrastructure provisioning and orchestration in line with business demands.

Obstacles

- Cost of refreshing API-enabled infrastructure components on-premises after initial implementation.
- Applying automation to existing monolithic infrastructure components will fail due to lack of platform agility.
- Lack of maturity in APIs that enable integration across different infrastructure platforms.
- Lack of open APIs/API compatibility across vendor platforms.
- The scarcity of programmable infrastructure experience within I&O, and the shortage of skilled resources to comprehensively exploit it, especially in web technologies (such as HTTP and JSON) to develop these APIs.

User Recommendations

- Deploy a programmable infrastructure to further abstract application from infrastructure delivery and pursue an agile digital business outcome.

- Implement a programmable infrastructure by investing in infrastructure automation tools and continuous delivery (example vendors for these markets are listed below, but no single vendor or platform can enable an organizationwide programmable infrastructure strategy).
- Invest in infrastructure and DevOps, and modernize legacy IT architectures to implement an API-driven infrastructure.
- Examine reusable programmable building blocks as they extend their programmable infrastructure strategy on repeatable and available skills from providers.

Sample Vendors

Alibaba Cloud; Amazon Web Services (AWS); CU Coding; Google; IBM; Microsoft; Oracle; proteanTecs; Tencent Cloud; VMware

Gartner Recommended Reading

Programmable Infrastructure Is Foundational to Infrastructure-Led Disruption

How to Lead Digital Disruption With Programmable Infrastructure

Understand the Hype, Hope and Reality of Composable Infrastructure

How to Lead Digital Disruption With Programmable Infrastructure

Drive Administration, Application and Automation Capabilities of Infrastructure-Led Disruption

Observability

Analysis By: Padraig Byrne, Gregg Siegfried

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Observability is the characteristic of software and systems that enables them to be understood, based on their outputs, and enables questions about their behavior to be answered. Tools that facilitate software observability enable observers to collect and quickly explore high-cardinality telemetry using techniques that iteratively narrow the possible explanations for errant behavior.

Why This Is Important

The inherent complexity of modern applications and distributed systems, and the rise of practices such as DevOps, has left organizations frustrated with legacy monitoring tools and techniques. These can do no more than collect and display external signals, which results in monitoring that is, in effect, only reactive. Observability acts like the central nervous system of a digital enterprise.

Observability tools enable a skilled observer to explain unexpected system behavior more effectively.

Business Impact

Observability tools have the potential to reduce both the number of service outages and their severity. Their use by organizations can improve the quality of software, because previously invisible (unknown) defects and anomalies can be identified and corrected. By enabling product owners to better understand how their products are used, observability supports the development of more accurate and usable software, and a reduction in the number and severity of events affecting service.

Drivers

- The term "observability" is now ubiquitous, with uses extending beyond the domain of IT operations. Although the 2020s are shaping up to be the "decade of observability," care must be taken to ensure the term retains relevance when used beyond its original range of reference.
- OpenTelemetry's progress and continued acceptance as the "observability framework for cloud-native software" raises the profile of observability and its toolchain.
- Traditional monitoring systems capture and examine signals (possibly adaptive) in relative isolation, with alerts tied to threshold or rate-of-change violations that require prior awareness of possible issues and corresponding instrumentation. Given the complexity of modern applications, it is unfeasible to rely on traditional monitoring alone.
- Observability tools enable a skilled observer — a software developer or a site reliability engineer — to explain unexpected system behavior more effectively, provided enough instrumentation is available. Integration of software observability with artificial intelligence for IT operations (AIOps) to automate subsequent determinations is a potential future development.
- Observability is an evolution of longstanding technologies and methods, and established monitoring vendors are starting to reflect observability ideas in their products. New companies are also creating offerings based on observability.

Obstacles

- In many large enterprises, the role of IT operations has been to "keep the lights on," despite constant change. This, combined with the longevity of existing monitoring tools, means that adoption of new technology is often slow.
- Enterprises have invested significant resources in their existing monitoring tools, which exhibit a high degree of "stickiness." This creates nontechnical, cultural barriers to adopting new practices such as those based on observability.

User Recommendations

Users starting a DevOps journey should:

- Assess software observability tools to integrate into their continuous integration/continuous delivery (CI/CD) pipelines and feedback loops.
- Investigate problems that cannot be framed by traditional monitoring by using observability to add flexibility to incident investigations.
- Enable observability by selecting vendors that use open standards for collection, such as OpenTelemetry.
- Tie service-level objectives to desired business outcomes using specific metrics, and use observability tools to understand variations.
- Ensure IT operations and site reliability engineering teams are aware of updates to existing monitoring tools and how they may take advantage of them. Many traditional application performance monitoring vendors are starting to incorporate observability features into their products.
- Avoid the conclusion that observability is synonymous with monitoring. At minimum, observability represents the internal perspective, rather than external.

Sample Vendors

Chronosphere; Grafana Labs; Honeycomb; Lightstep; Observe; VMware

Gartner Recommended Reading

Monitoring and Observability for Modern Services and Infrastructure

Magic Quadrant for Application Performance Monitoring and Observability

Cool Vendors in Monitoring, Observability and Cloud Operations

Innovation Insight for Observability

Infrastructure Automation

Analysis By: Chris Saunderson

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Infrastructure automation (IA) allows DevOps and I&O teams to design and implement automated delivery services across on-premises and cloud environments. IA enables DevOps and I&O teams to manage the life cycle of services through creation, configuration, operation and retirement. These infrastructure services are then made available through self-service catalogs, direct invocation, application programming interface (API) integrations and DevOps toolchains.

Why This Is Important

IA delivers speed, quality and reliability, with scalable approaches for deploying and managing infrastructure. DevOps and I&O teams are using IA tools as a part of their delivery pipelines targeting deployment topologies that range from on-premises to the cloud.

Business Impact

Supporting IA, provisioning, configuration and operation will enable:

- Agility — continuous infrastructure integration and delivery
- Productivity — version-controlled, faster and repeatable deployment
- Cost improvement — reductions in manual effort via increased automation
- Risk mitigation — compliance driven by standardized configurations
- Efficiency — reducing work and adding value
- Environments that product teams require to deliver features
- Better security and compliance

Drivers

Infrastructure and operations (I&O) leaders must automate processes and leverage tools to mature beyond simple deployments of standardized platforms and deliver the systemic, transparent management of platform deployments. This same discipline must be applied to the operation of these deployed platforms, ensuring that efficient operations (including automation incident response) can be achieved. IA tools deliver the following key capabilities to support this maturation:

- Multicloud/hybrid cloud infrastructure orchestration
- Support for immutable and programmable infrastructures
- Predictable delivery enabling automated operations
- Self-service and on-demand environment creation
- Support DevOps initiatives (continuous integration/delivery/deployment)
- Resource provisioning
- Operational configuration management efficiencies
- Policy-based delivery and assessment/enforcement of deployments
- Enterprise-level framework to enable tooling strategy maturation

Obstacles

- The combination of tools needed to deliver IA capability can increase tool count.
- Software engineering skills and practices are required to get maximum value from tool investments.
- The migration from point activities to infrastructure capability releases entails a steep learning curve.
- IA vendor capability overlaps obscure the tool landscape.
- Steep learning curves can cause developers and administrators to revert to familiar scripting methods to deliver required capabilities.

User Recommendations

- Identify existing IA tools in use to catalog capabilities and identify use cases.
- Assess existing internal IT skills to incorporate training needs that enable IA more fully.
- Baseline how managed systems and tooling will be consumed (e.g., engineer, self-service catalog, API or on-demand).
- Integrate security and compliance requirements into evaluation criteria.
- Develop an IA tooling strategy that incorporates current needs and near-term roadmap evolution.

Sample Vendors

Amazon Web Services; HashiCorp; Microsoft; Perforce (Puppet); Pliant; Progress (Chef); Pulumi; RackN; Upbound; VMware

Gartner Recommended Reading

Market Guide for Infrastructure Automation Tools

Innovation Insight for Continuous Infrastructure Automation

How to Build Agile Infrastructure Platforms That Enable Rapid Product Innovation

The Future of DevOps Toolchains Will Involve Maximizing Flow in IT Value Streams

To Automate Your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations

Communities of Practice

Analysis By: Thomas Murphy

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

A community of practice (CoP) is a community- or stakeholder-owned forum — often initiated by an organization's leaders — to enable collaborative problem solving, knowledge sharing and organizational learning. Agile and DevOps teams require a shift in culture to support greater collaboration and cross-functional learning and to break down traditional silos and their associated centers of excellence (COEs).

Why This Is Important

As organizations shift to a product model and adopt DevOps practices, CoPs are a key success factor. While CoPs are widely used by agile and DevOps and cloud leaders, adoption by mainstream clients is mixed. Many IT organizations continue to rely on top-down, functionally aligned COEs that focus on policy adherence and governance instead of problem solving and learning. In contrast, CoPs are motivational if they provide a clear vision and space for the community to collaborate, learn and evolve.

Business Impact

Communities of practice enable knowledge management, professional development and capability improvement.

Our research finds that CoPs:

- Shorten the learning curve for employees
- Provide higher levels of employee satisfaction, leading to higher motivation and innovation
- Respond more rapidly to customer needs and inquiries
- Reduce duplication of effort
- Spawn new ideas for products and services
- Help members develop capabilities that align with organizational needs

Drivers

- DevOps product and platform team members must gain new skills and behaviors, and improve the pace of delivery. CoPs provide a community-focused approach for knowledge sharing and organizational learning.
- CoPs are a continued source of strength through interactions and learning that should be captured and shared digitally.

- CoPs are voluntary groups that enable practitioners to create common guidelines and practices based on proven practice.
- Transversal knowledge management through CoPs prevents unintentional functional and product silos from forming.

Obstacles

- Challenges associated with hybrid work models will disrupt organizations that are initiating new CoPs.
- Hierarchical structures and existing reporting chains can feel threatened by the self-directed nature of CoPs wanting to lean on COEs and business as usual.
- While CoPs are a powerful concept, organizations will also need other structures and practices, such as DevOps Dojos or InnerSourcing, to drive the best long-term benefits.
- CoPs require management support and active coaching to be effective, continuously create value and prevent rapid obsolescence.

User Recommendations

I&O leaders and applications and software engineering leaders should:

- Empower cross-functional teams to learn and discover through experimentation, collaboration and knowledge sharing by creating and communicating a clear vision for the community with time and space.
- Encourage experimentation and risk taking by dedicating time to the community to build a culture where failure and new approaches are celebrated as learning opportunities.
- Complement the CoP by organizing job rotations, hackathons and technology conferences.

Sample Vendors

Atlassian; GitHub; Microsoft; Slack; StackOverflow; Zoom

Gartner Recommended Reading

Community of Practice Essentials

Case Study: Kick-Starting a Low-Code/No-Code Community of Practice (Heathrow Airport)

13 Best Enterprise Architecture Practices to Ensure Program Success

Drive Innovation by Enabling Innersource

Sliding into the Trough

Site Reliability Engineering

Analysis By: George Spafford, Daniel Betts

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Site reliability engineering (SRE) is a collection of systems and software engineering principles used to design and operate scalable resilient systems. Site reliability engineers work with the customer or product owner to understand operational requirements and define service-level objectives (SLOs). Site reliability engineers work with product or platform teams to design and continuously improve systems that meet defined SLOs.

Why This Is Important

SRE emphasizes the engineering disciplines that lead to resilience; but individual organizations implement SRE in widely varying ways such as a defined role or a set of practices. SRE teams can serve as an operations function, and nearly all such teams have a strong emphasis on blameless root cause analysis. This is to decrease the probability and/or impact of future events and to enable organizational learning, continual improvement and reductions in unplanned work.

Business Impact

The SRE approach to improving reliability is intended for products and platforms that need to deliver customer value at speed at scale while managing risk. The two primary use cases are to improve the reliability of existing products/platforms or to create new products or platforms that need reliability from the start.

Drivers

- Clients are under pressure to meet customer requirements for reliability while scaling their digital services and are looking for guidance to help them.
- While Google originated what became known as SRE and continued to evolve it, practitioners are developing and sharing new practices as well. Potential practitioners looking for pragmatic guidance to improve the reliability of their systems have a rich body of knowledge they can leverage that works well with agile and DevOps.
- Organizations are adopting highly skilled automation practices (usually DevOps), and usage of infrastructure-as-code capabilities (which usually requires a cloud platform) to deliver digital business products reliably.
- The most common use-case based on inquiry calls with clients is to leverage SRE concepts to improve the reliability of existing systems that are not meeting customer requirements for availability, performance or are proving difficult to scale.

Obstacles

- Insufficient internal marketing to understand what agile, DevOps or Product teams need or would value and then explaining how the value SRE can deliver will justify the costs and risks incurred. Without marketing its benefits, SRE adoption tends to be less certain or slower. “SRE” the concept by itself is insufficient — people must continuously believe it is worthwhile.
- Finding SRE candidates who have the right mix of development, operations and people skills is a big challenge for clients. Impacts on initial adoption and scaling efforts as well.
- Clients have voiced problems with product owners who overly focus on functional requirements and not nonfunctional requirements thus slowing support of SRE within the organization. This impacts both the ability to start and then scale as well because those people performing a SRE role on a team will have limited opportunities to improve reliability and without sufficient value being demonstrated, then others will be reluctant to adopt SRE.

User Recommendations

- Leverage practices pragmatically based on need. Don’t feel that you must implement SRE exactly the way Google does it, learn what works for you.
- Detect an opportunity to begin that is politically friendly, will demonstrate sufficient value and has an acceptable risk profile.
- Start small, focus, learn, improve, and demonstrate value — do not try to change everything at once.
- Work with the customer or product owner to define clear, obtainable SLOs based on their needs.
- Implement monitoring and improve observability to objectively report on actual performance relative to the SLOs.
- Product owners must be accountable for functional and non-functional requirements of their products.
- Instill collaborative working between site reliability engineers, developers and other stakeholders to help them learn how to design, build and evolve their products to meet SLOs.
- Create a community, implement effective organizational learning practices and evolve SRE practices.

Sample Vendors

Atlassian; Blameless; Datadog; Dynatrace; New Relic; OpsRamp; PagerDuty; Splunk

Container Management

Analysis By: Dennis Smith, Michael Warrilow

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Container management tools automate provisioning, starting/stopping and maintaining of runtime images for containers and dependent resources via centralized governance and security policies. They include a management interface, registration of container images and integration with container management brokers.

Why This Is Important

Container runtimes simplify use of container functionality and enable integration with DevOps tooling and workflows. Productivity and/or agility benefits of containers include accelerating and simplifying the application life cycle, enabling workload portability between different environments and improving resource utilization. Container management makes it easier to achieve scalability and production readiness for containerized workloads.

Business Impact

Gartner surveys and client interactions show that demand for containers continues to rise. This trend is due to application developers' and DevOps teams' preference for container runtimes, which have introduced container packaging formats. Developers have quickly progressed from leveraging containers on their desktops to needing environments that can run and operate containers at scale, introducing the need for container management.

Drivers

- Container runtimes, frameworks and other management software provide capabilities such as packaging, placement and deployment, and fault tolerance (for example, clusters of nodes running the application).
- A vibrant open-source ecosystem and competitive vendor market has culminated in a wide range of container management offerings. Many vendors enable management capabilities across hybrid cloud or multicloud environments by providing an abstraction layer across on-premises and public clouds. Container management software can run on-premises, in public infrastructure as a service (IaaS) or simultaneously in both.
- Container-related edge computing use cases have increased in industries that need to get compute and data closer to the activity (for example, telcos, manufacturing plants, etc.).
- Data analytics use cases have emerged over the past few years, as have operational control planes that enable the management of container nodes and clusters.
- All major public cloud service providers now offer on-premises container solutions.

- Independent software vendors (ISVs) are starting to package their software for container management systems.
- Some enterprises have scaled sophisticated deployments, and many more have recently begun or are planning container deployments. This trend is expected to increase as enterprises continue application modernization projects.

Obstacles

- More abstracted, serverless offerings may enable enterprises to forgo container management. Among these services are Knative-based services, AWS Lambda and Fargate, Azure Functions and Google's Cloud Run. These services embed container management in a manner that is transparent to the user.
- Third-party container management software faces huge competition in the container offerings from the public cloud providers, both with public cloud deployments and the extension of software to on-premises environments. These offerings are also challenged by ISVs that choose to craft open-source components with their software during the distribution process.
- Organizations that perform relatively little app development or make limited use of DevOps principles are served by SaaS, ISV and/or traditional application development packaging methods.

User Recommendations

- Determine if your organization is a good candidate for container management software adoption by weighing organizational goals of increased software velocity and immutable infrastructure, and its hybrid cloud requirements, against the effort required to operate third-party container management software.
- Leverage container management capabilities integrated into cloud IaaS and PaaS providers' service offerings by experimenting with process and workflow changes that accommodate the incorporation of containers.
- Avoid using upstream open source (e.g., Kubernetes) directly unless the organization has adequate in-house expertise to support.

Sample Vendors

Amazon Web Services (AWS); Google; IBM; Microsoft; Mirantis; SUSE (Rancher Labs); Red Hat; VMware

Gartner Recommended Reading

Market Guide for Container Management

Continuous Delivery

Analysis By: Hassan Ennaciri

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Continuous delivery (CD) is a software engineering approach that enables teams to produce valuable software in short cycles while ensuring that the software can be reliably released at any time. Through dependable, low-risk releases, CD makes it possible to continuously adapt software to incorporate user feedback, shifts in the market and changes to business strategy. This approach requires engineering discipline to facilitate the complete automation of the software delivery pipeline.

Why This Is Important

Growing DevOps initiative success continues to drive investments in CD capabilities. CD improves release velocity and reliability, while simplifying compliance enforcement via automation. It is a prerequisite and a first step to continuous deployments for organizations that aspire to push changes with zero downtime.

Business Impact

CD is a key practice for a DevOps initiative that reduces build-to-production cycle time. This accelerates the positive impact of new applications, functions, features and fixes by increasing velocity across the application life cycle. The positive impacts include improved business delivery and end-user satisfaction, improved business performance and agility, and risk mitigation via rapid delivery of updates.

Drivers

- Agile and DevOps adoption to deliver solutions
- Need to improve release velocity and reliability
- Need to shift left and simplify compliance enforcement via automation
- Need to improve delivery outcomes to more consistently deploy application builds and updates, by extending the benefits of continuous integration (CI) and automated testing to continuously build deployable software
- CD is a prerequisite and first step to continuous deployments for organizations aspiring to push changes with zero downtime

Obstacles

- Organizational culture and collaboration between teams with different roles and skills is a major barrier to CD success. Agile practices that helped bridge the gap between business and development need to be extended to deployment, environment configuration, monitoring and support activities.
- Lack of value stream mapping of product delivery hinders visibility and quick feedback loops needed for continuous improvements. Teams struggle to improve and focus on value work as they don't have insights into the critical steps in the process, the time each step takes, handoffs and wait states.
- Manual steps and processes involved in deploying to production environments.
- Other challenges that impact success of CD include application architecture and lack of automation in all areas of testing, environment provisioning, configuration security and compliance.

User Recommendations

- Evaluate all associated technologies when you start a CD initiative, and take an iterative approach to adoption. This will require collaboration with all stakeholders from product, development, security and operations.
- Establish consistency across application environments for a higher likelihood of success, and implement a continuous improvement process that relies on value stream metrics.
- Evaluate and invest in associated tooling, such as application release orchestration tools, containers and infrastructure automation tools. These tools provide some degree of environment modeling and management, which can prove invaluable for scaling CD capabilities across multiple applications.
- Explore value stream delivery platforms (VSDPs) to provide fully integrated capabilities that enable continuous delivery of software.

Sample Vendors

Broadcom; CloudBees; GitLab; Guide-Rails; Harness; JFrog; Red Hat

Gartner Recommended Reading

How to Build and Evolve Your DevOps Toolchains

Market Guide for Value Stream Delivery Platforms

Beware the DevOps Toolchain Debt Collector

ARO

Analysis By: Daniel Betts

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Application release orchestration (ARO) combines deployment automation, pipeline and environment management with release orchestration capabilities to simultaneously improve the quality, velocity and governance of application releases. ARO tools enable organizations to scale release activities across multiple diverse teams (e.g., DevOps), technologies, development methodologies (e.g., agile), delivery patterns (e.g., continuous), pipelines, processes and toolchains.

Why This Is Important

Demand is growing, and will continue to grow, for new applications and features delivered faster to support business agility. The resulting tumultuous and transformative activity (often in the form of DevOps initiatives) has created multiple buyers for ARO solutions. These buyers often desperately need ARO's cohesive value, yet are challenged to articulate and/or gain consensus around the business criticality of release activities to drive adoption.

Business Impact

ARO tools provide increased transparency in the release management process by making bottlenecks and wait states visible in areas such as infrastructure provisioning or configuration management. Once these constraints are visible and quantifiable, business value decisions can be made to address them and measure improvement. This speeds the realization of direct business value, as new applications and enhancements/bug fixes can be more quickly and reliably delivered.

Drivers

- Agility and productivity gains — faster delivery of new applications and updates in response to changing market demands.
- Cost reduction — significant reduction of manual interactions by high-skill and high-cost staff, freeing them to work on higher-value activities.
- Risk mitigation — consistent use of standardized, documented processes and configurations across multiple technology domains.
- Improvement and remediation — use of dashboard views over metrics outlining and predicting release quality and throughput.
- Improved visibility and traceability into the release process.

Obstacles

- The need to map capabilities to client's internal delivery challenges or opportunities.
- Vendors in the DevOps toolchain market are building feature-comparable products, with some incremental differences, affecting ARO adoption.
- Challenges in capturing the release orchestration space, due to maturity (feature parity across supplier platforms) and disruption (the environments around the ARO space are pressing inwards on the core functionality) in the market.

User Recommendations

- Organize activities into three categories: deployment automation, pipeline and environment management, and release orchestration.
- Explore value stream delivery platforms that provide ARO along with other native capabilities.
- Prioritize capabilities for current and future needs prior to evaluating vendors. When evaluating ARO tools for selection, prioritize tool features and map capabilities to requirements. Where legacy environments exist, those should be weighted more heavily.
- Simplify and speed up the transition to automated workflows by documenting current application release procedures, activities and artifacts performed by both traditional and DevOps teams.
- Confirm the availability of an ARO platform dashboard, with release performance and underlying platforms metrics.

Sample Vendors

CloudBees; Digital.ai; GitLab; IBM; Microsoft; Plutora

Gartner Recommended Reading

Market Guide for Value Stream Delivery Platforms

Market Guide for Value Stream Management Platforms

The Future of DevOps Toolchains Will Involve Maximizing Flow in IT Value Streams

Beware the DevOps Toolchain Debt Collector

Platform Teams and AIOps Will Redefine DevOps Approaches by 2025

Immutable Infrastructure

Analysis By: Neil MacDonald, Tony Harvey

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Immutable infrastructure is a process pattern (not a technology) in which the system and application infrastructure, once deployed, is never updated in place. Instead, when changes are required, the infrastructure and applications are simply replaced from the development pipeline.

Why This Is Important

Immutable infrastructure ensures the system and application environment is accurately deployed and remains in a predictable, known-good-configuration state. It simplifies change management, supports faster and safer upgrades, reduces operational errors, improves security, and simplifies troubleshooting. It also enables rapid replication of environments for disaster recovery, geographic redundancy or testing. This approach is easier to adopt and often applied with cloud-native applications.

Business Impact

Taking an immutable approach to workload and application management simplifies automated problem resolution by reducing the options for corrective action to, essentially, just one — repair the application or image in the development pipeline and rerelease. The result is an improved security posture with fewer vulnerabilities and a faster time to remediate when new issues are identified.

Drivers

- Linux containers and Kubernetes are being widely adopted. Containers improve the practicality of implementing immutable infrastructure and will drive greater adoption.
- Interest in zero-trust and other advanced security postures where immutable infrastructure can be used to proactively regenerate workloads in production from a known good state (assuming compromise), a concept referred to as "systematic workload reprovisioning."
- For cloud-native application development projects, immutable infrastructure simplifies change management, supports faster and safer upgrades, reduces operational errors, improves security, and simplifies troubleshooting.

Obstacles

- The use of immutable infrastructure requires a strict operational discipline that many organizations haven't yet achieved, or have achieved for only a subset of applications.
- IT administrators are reluctant to give up the ability to modify or patch runtime systems.
- Although immutable infrastructure may appear simple, embracing it requires a mature automation framework, up-to-date blueprints and bills of materials, and confidence in your ability to arbitrarily recreate components without negative effects on user experience or loss of state.

- Many application stacks have elements that are deployed in the form of virtual machine images. VM replacement is slower and requires greater coordination than other workload components such as containers.

User Recommendations

- Reduce or eliminate configuration drift by establishing a policy that no software, including the OS, is ever patched in production. Updates must be made to individual components, versioned in a source-code-control repository, then redeployed for consistency.
- Prevent unauthorized change by turning off all normal administrative access to production compute resources, for example, by not permitting Secure Shell (SSH) or Remote Desktop Protocol (RDP) access.
- Adopt immutable infrastructure principles with cloud-native applications first. Cloud-native workloads are more suitable for immutable infrastructure architecture than traditional on-premises workloads.
- Treat scripts, recipes and other codes used for infrastructure automation similar to the application source code itself, as this mandates good software engineering discipline.
- Include immutable infrastructure scripts, recipes, codes and images in your backup and ransomware recovery plans as they will be your primary source to rebuild your infrastructure after an infection.

Sample Vendors

Amazon Web Services; Google; HashiCorp; Microsoft; Perforce; Progress (Chef); Red Hat; Snyk (Fugue); Turbot; VMware

Gartner Recommended Reading

How to Make Cloud More Secure Than Your Own Data Center

2022 Strategic Roadmap for Compute Infrastructure

Innovation Insight for Continuous Infrastructure Automation

To Automate Your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations

Innovation Insight for Cloud-Native Application Protection Platforms

Feature Management

Analysis By: Keith Mann

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

A feature is a discrete unit of software that provides a single, valuable function or capability within a larger system. Feature management combines the selective enabling or disabling of features via feature toggles (also known as feature flags) with the monitoring, assessment, and comparison of features and feature variants.

Why This Is Important

Feature toggles, though powerful, can be cumbersome to track and maintain, and can introduce technical debt. Feature management enables the use of feature toggles on a very large scale. This makes it practical to use feature toggles extensively, which in turn enables them to be used for more purposes. Although feature toggles were originally used to disable features that were not ready for deployment, they are now also used for experimentation, testing and customization.

Business Impact

Almost all businesses can benefit from faster, more reliable delivery of software — the impact of feature management in that sense could be universal. Experimentation has already had a large impact on digital marketing and digital consumer product development, where human responses are hard to predict. The biggest potential impact comes from the improved ability to select valuable features for development, based on feedback loops from feature monitoring in production to development planning.

Drivers

- Software engineering teams are reaching high levels of maturity in their use of agile and DevOps practices. This enables them to deliver software even faster than it can be deployed and consumed. Software engineering organizations can use feature management to decouple delivery from deployment and avoid slowing down teams.

- Demand for continuously available systems is increasing. This drives a need for instant recovery from defective features, which, in turn, fuels a need for feature toggles and feature management.

- With new technologies and corresponding new user experience options comes uncertainty about which types of experience are most effective for users. Feature management enables experiments that reduce this uncertainty by presenting various experiences to various sample audiences and measuring their responses.

- Advanced software engineering organizations are finding that fast delivery alone does not satisfy stakeholders. They are beginning to realize that they must establish feedback loops from production to feature design, so that they can learn which designs produce the greatest value. Feature management enables the value of features to be monitored, tracked and compared.

Obstacles

- Even years after its genesis, the feature management market is still dominated by fairly small vendors. This limits user awareness and adoption.
- Feature management vendors continue to have a technical rather than a business focus in their messaging, leaving it to technology buyers like software engineering leaders to justify the investment.
- The vision of feature management as a tool for experimentation and value feedback is new. The focus has long been on the speed and reliability of deployment, so buyers view that as the benefit of feature management. Shifting the message is hard.
- Although value feedback loops could be the most important application of feature management, the concept and corresponding practices are still emerging.
- As with many areas of software engineering, low talent availability is a problem. Data scientists and other professionals skilled in experimentation are in short supply and high demand.

User Recommendations

- Ensure that development teams are familiar with feature management techniques. Ideally, developers will already be using feature toggles, where appropriate, to improve deployment speed and reliability.
- Promote the use of feature management for experimentation. This may mean selecting a feature management vendor and training or hiring staff.
- Introduce the concept of value feedback loops to software designers.
- Have software designers clearly identify features during solution design and define the value of each feature.
- Establish value feedback loops as tools and practices mature.

Sample Vendors

LaunchDarkly; Optimizely; Split

Gartner Recommended Reading

Software Engineering Teams Must Learn to Deliver More Value

Solution Path for Continuous Delivery With DevOps

Quick Answer: What Are the Essential Shift-Right Practices for Testing in Production?

Climbing the Slope

Software-Defined Infrastructure

Analysis By: Philip Dawson

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Obsolete

Definition:

Software-defined infrastructure (SDI) enables abstraction of the physical infrastructure, with its services exposed via APIs enabling greater levels of automation, policy-based orchestration and reuse. SDI includes software-defined data center, network, storage, compute and SD edge infrastructure.

Why This Is Important

Software defined is the further abstraction of software from hardware. It enables businesses to be more agile and flexible by enabling programmatic control of the infrastructure through software interfaces. SDI combines compute (SDC), network (SDN) and storage (SDS), but SDI also extends to non-data center infrastructure, with the use of either monitoring devices or machines that are software-defined.

Business Impact

While data center SDI is embedded in other data center initiatives like cloud and hyperconverged infrastructure, SDI is now focused in key verticals operating in multiple, edge locations, such as retail, manufacturing, retail banking, distribution and utilities. It also continues to extend IoT and non-data-center SDI initiatives for new IT operations and functions.

Drivers

- SDI data center infrastructure is well-covered with compute (SDC), network (SDN, now obsolete), edge (SD-WAN) and storage (SDS), but SDI also extends to non-data-center infrastructure with the use of monitoring devices or machines that are software-defined.

- SDI reaches beyond and between SDDCs, and leverages SDI benefits and features for new multimode applications and edge and/or IoT endpoints.

- In 2022, SDI's lingering presence of hype is enabled through the use of sensors and adapters that are abstracted through software, stretching SDI to the edge, IoT and operational technology (e.g., retail POS), rather than traditional, IT-driven SDI through data center or cloud.

- Key verticals operating in multiple, geographically distributed locations, such as retail, manufacturing, retail banking, distribution and utilities, are extending IoT and non-data-center SDI initiatives for new IT operations and functions.

Obstacles

- SDI is now tied to extending vendor technology, not interoperability.

- SDI overlaps other integrated systems taxonomy like hyperconvergence, as it drives cloud to data center and edge adoption.
- In 2022, we are seeing SDI continue releasing vendor-specific silo technology (not heterogeneous service-driven) and, hence, it continues to be obsolete as multivendor interoperability standards and technology silos persist and limit SDI integration between vendors.
- SD Wan segmentation is driving SDI to the edge and is architecturally different from SDN, which is focused more around data Center Infrastructure convergence.

User Recommendations

- Include the integration and measurement of non-data-center edge infrastructure, as SDI initiatives roll out tied to SD-WAN and edge initiatives.
- Focus on core IT SDI for compute, network, storage and facilities, but expand the impact of SDI on IoT, edge computing, remote office/branch office (ROBO) and other operational technologies.
- Anticipate SDI to be tied to a specific vendor or technology silo, such as storage and network hardware or virtualization software. Be cautious not to commit to a vendor's SDI without realizing the specific area of lock-in.

Sample Vendors

IBM; Intel; Microsoft; Red Hat; VMware; Wipro

Gartner Recommended Reading

The Road to Intelligent Infrastructure and Beyond

Drive Administration, Application and Automation Capabilities of Infrastructure-Led Disruption

APM

Analysis By: Padraig Byrne, Mrudula Bangera

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Application performance monitoring (APM) enables the observation of application behavior and its dependencies, users and business KPIs throughout the application's life cycle. The applications being observed may be developed internally, as packaged applications or as software as a service (SaaS).

Why This Is Important

The APM market continues to evolve beyond its core root of server-side application monitoring as organizations seek to optimize business outcomes, enhance user experience and improve application performance. It is no longer sufficient to monitor one aspect of the technology stack; nor is it enough to deploy proprietary technologies to collect performance data. The exponential growth in mobile, cloud-native applications continues to fuel the APM market.

Business Impact

APM solutions enable businesses to examine modern applications' end-to-end performance, coupled with detailed inspections to quickly identify service-impacting outages. As organizations continue to embrace digital transformation, their need for agility in order to succeed with these transformation initiatives increases. APM solutions can be perceived as more than another monitoring tool, supporting the need for agility and aiding in its acceleration and effectiveness.

Drivers

- Unified monitoring: New application monitoring tools are becoming more unified. This approach requires platforms that share common data models to conduct correlation analysis and other critical functions of APM.
- Holistic monitoring: Modern monitoring platforms are becoming more holistic in terms of the types of data they can ingest, analyze and integrate. The continued adoption of new application development and operations technologies requires monitoring teams to constantly test the limits of their monitoring products.
- Shift-left monitoring: Testing in preproduction and integration with continuous integration/continuous delivery (CI/CD) tools have become the new monitoring norm, increasing the quality and robustness of the finished product.
- Intelligent monitoring: The use of logs, traces, metrics and multiple other types of telemetry is enabling operations and monitoring teams to find unexpected patterns in high-volume, multidimensional datasets using artificial intelligence for IT operations (AIOps) technologies.
- Business monitoring: Digital transformation, along with the effects of remote work due to the COVID-19 pandemic, has highlighted the need to monitor technology as well as user experience and its impact on business outcomes.

Obstacles

- APM software often fails to provide a complete solution, requiring organizations to pivot between tools, wasting time and resources while struggling to find the root cause of the problem.
- Containers/microservices, Internet of Things (IoT), cloud and software-defined anything (SDx) changes are coming to IT operations environments faster than monitoring strategies are evolving to handle them. This is leading to visibility gaps and performance challenges.

- IT monitoring teams still rely on manually invoked runbooks or ad hoc scripts to remediate problems, hindering infrastructure and operations (I&O) leaders' ability to deploy and monitor new technologies.
- Vendors promise a lot when it comes to AIOps, but the reality is that most uses of AI/ML in APM are still evolving and remain immature.
- There is a major disconnect between what I&O monitors and what the business cares about, which can have significant negative implications for the business.

User Recommendations

- Choose APM vendors that assist in relating application performance to business objectives and serve not only ITOps, but also DevOps, application owners and lines of business, providing value throughout the application life cycle. Select a vendor that provides actionable answers and not just endless drill-downs to more data.
- Choose APM vendors based on their ability to support: mapping and monitoring of customer and business journeys; bidirectional integration with the DevOps toolchain; new emerging standards in instrumentation, such as OpenTelemetry; cloud-native monitoring with an API-first approach; application security; and integrations with your existing or planned IT service management (ITSM) and configuration management database (CMDB) tools.

Sample Vendors

Cisco; Datadog; Dynatrace; Instana; New Relic; Splunk

Gartner Recommended Reading

Magic Quadrant for Application Performance Monitoring

Critical Capabilities for Application Performance Monitoring

Continuous Configuration Automation

Analysis By: Chris Saunderson

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Continuous configuration automation (CCA) tools enable infrastructure administrators and developers to automate the deployment, configuration and operation of systems and software. They support the description, assessment and maintenance of configuration states and settings. Most CCA tools have open-source heritage, and some offer commercial support. Commercial CCA tools have vendor support, role-based administration and advanced management capabilities.

Why This Is Important

CCA tools have proven critical to delivery and operational efficiency. They enable automation and DevOps initiatives by delivering and managing infrastructure and associated software and configuration changes as code. In combination with infrastructure automation tools, CCA tools expand their reach into networking, containers, patching, compliance and security use cases. They also enable the automation of Day 2 operations of delivered systems.

Business Impact

By enabling automation of the deployment and configuration of settings and software programmatically, organizations realize:

- **Agility improvements —** Continuous integration/continuous delivery for I&O infrastructure management.
- **Productivity gains —** Repeatable, version-controlled infrastructure deployment and operation.
- **Cost optimization —** Reductions in manual interventions by skilled staff.
- **Risk mitigation —** Automated compliance using standardized, documented configurations and associated processes.

Drivers

- Organizations need a broader set of deployment and automation functions beyond configuration management, including infrastructure as code (IaC), patching, application release orchestration (ARO), configuration auditing (e.g., for regulatory or internal policy compliance) and orchestration of operational tasks.
- CCA provides an automation framework that enables deployment automation and Day 2 operational automation of changes, compliance and response to incidents or problems.
- CCA tools are imperative for I&O administrators to mature from task-based scripting to a more structured approach to automation and delivery.
- There is a need for repeatable, standardized deployments to be made available to end users or I&O administrators that enable quick delivery of infrastructure to meet end-user needs and I&O policies and baselines.

Obstacles

- Confusion around the capabilities of automation tools leads to assumptions and misunderstandings.
- Developers and administrators may use CCA in a silo, further inhibiting enterprisewide adoption.

- IT skill sets hinder adoption of these tools, requiring source code management and software engineering skills to make full use of capabilities.
- The growing use of IaC tools has created confusion about the role of CCA tools; however, CCA tools are necessary to deliver effective and efficient IaC.

User Recommendations

- Clarify the role that CCA tools fulfill in their toolchain and make selections based on the tasks that are in scope.
- Evaluate the availability of content against organizational use cases. Prioritize CCA tools that provide out-of-the-box content that addresses current pain points and accelerates time to value.
- Include both professional services for enablement and training requirements in cost evaluations. Costs associated with CCA tools extend beyond just the licensing cost.
- Expect to invest in training beyond tool implementation to fully realize the benefits of these tools.
- Guard against developers and administrators reverting to known scripting methods to complete specific tasks in place of using CCA capabilities.
- For DevOps and I&O leaders: Maximize the value of CCA tool investments by ensuring that your organization's culture can embrace CCA tools strategically.

Sample Vendors

Inedo; Perforce (Puppet); Progress (Chef); Red Hat (Ansible); VMware (SaltStack)

Gartner Recommended Reading

Market Guide for Infrastructure Automation Tools

To Automate Your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations

Innovation Insight for Continuous Infrastructure Automation

Entering the Plateau

Cloud Management Platforms

Analysis By: Dennis Smith

Benefit Rating: Low

Market Penetration: 5% to 20% of target audience

Maturity: Mature mainstream

Definition:

Cloud management platforms (CMPs) enable organizations to manage private, public and multicloud services and resources. Their functionality combines provisioning and orchestration; service request management; inventory and classification; monitoring and analytics; cost management and resource optimization; cloud migration, backup and disaster recovery; and identity, security and compliance. Functionality can be provided by a single product or a set of offerings with some degree of integration.

Why This Is Important

Managing a workload after it has migrated to the cloud is as important as the initial cloud migration. Enterprises will deploy CMPs (often as a part of a larger product suite) to increase agility, reduce the cost of providing services and increase the likelihood of meeting service levels. Costs are reduced, and service levels are met, because CMP deployments supports adherence to standards, as well as increased governance and accountability.

Business Impact

- CMPs provide the functionality to address initial provisioning of cloud resources, as well as ongoing management.
- The recent CMP market continually focuses on preventing enterprises from overspending or leaving themselves vulnerable to security issues — two key items to avoid when adopting cloud services.

Drivers

- Organizations need to address hybrid and multicloud requirements. Often they want tooling that provides consistency and uniformity across these environments. This is particularly so for organizations that have mature on-premises deployment and are selectively moving workloads to multiple cloud environments.
- Vendors are addressing the key issues enterprises face. Many vendors are looking to combine cost management and security functionality into governance tooling. A few vendors are also looking to provide infrastructure as code (IaC) assistance by overlaying cloud management functionality to this capability.
- In addition, many vendors have added container management to their CMP offerings. The ability to serve both application developers and I&O personas is key. This requires CMPs to be linked into the application development process to enable I&O teams to enforce provisioning standards, without imposing a workflow that inhibits agility.
- Desirable IT outcomes include: policy enforcement; reduced lock-in to public cloud providers, although at the cost of CMP vendor lock-in, which can slow innovation; enhanced ability to broker services from various cloud providers and make informed business decisions as to the providers to use; ongoing optimization of SLAs and costs; management of SLAs and enforcement of compliance requirements; health and performance monitoring of cloud applications; accelerated development, enabling setup/teardown of infrastructure that mimics

production, resulting in lower overall infrastructure costs and higher quality (this can be in support of DevOps initiatives); and movement of complex workloads into the cloud that require best-in-class capabilities from multiple public cloud providers resulting in a multicloud adoption model.

Obstacles

- **Market evolution:** The CMP market continues to change; it is increasingly becoming a smaller portion of the overall cloud management tooling market. In this case being superseded by more single focused specialized tooling.
- **Evolving customer requirements:** Challenges vendors face include interfacing with multiple public clouds, cost transparency with workload optimization to remediate cost overruns, handling newer functions (e.g., containers and serverless deployments), and edge computing.
- **Market consolidation:** Many CMP vendors have acquired cost management vendors, incorporating their functionality into their products. Vendors in adjacent markets have acquired CMP vendors; combining this functionality with asset management and SaaS operational management. Cloud service providers (CSPs) and management service providers (MSPs) have entered the market, and many long-standing vendors have introduced next-generation products that target gaps in their previous products.

User Recommendations

- Weigh your full vertical (infrastructure as a service [IaaS], platform as a service [PaaS] and SaaS) and horizontal (on-premises, public cloud and edge) needs.
- Examine single focused specialized tooling if your requirements do not expand beyond a limited scope.
- Select between native cloud services and CMPs by identifying a preferred provider model. Favor native cloud services if you value depth with a cloud provider; choose CMPs if you want breadth across cloud providers and your on-premises deployment.
- Determine the utility of functionally focused tools by defining the organization’s functionality needs. Integrate cloud management or traditional management tools, because no vendor has a total cloud management solution.
- Ensure staffing to operate CMP platforms by planning new roles (e.g., cloud engineers and/or operators).
- Develop skills in financial and capacity management.

Sample Vendors

CloudBolt; Flexera; Morpheus Data; Scalr; Snow Software; VMware

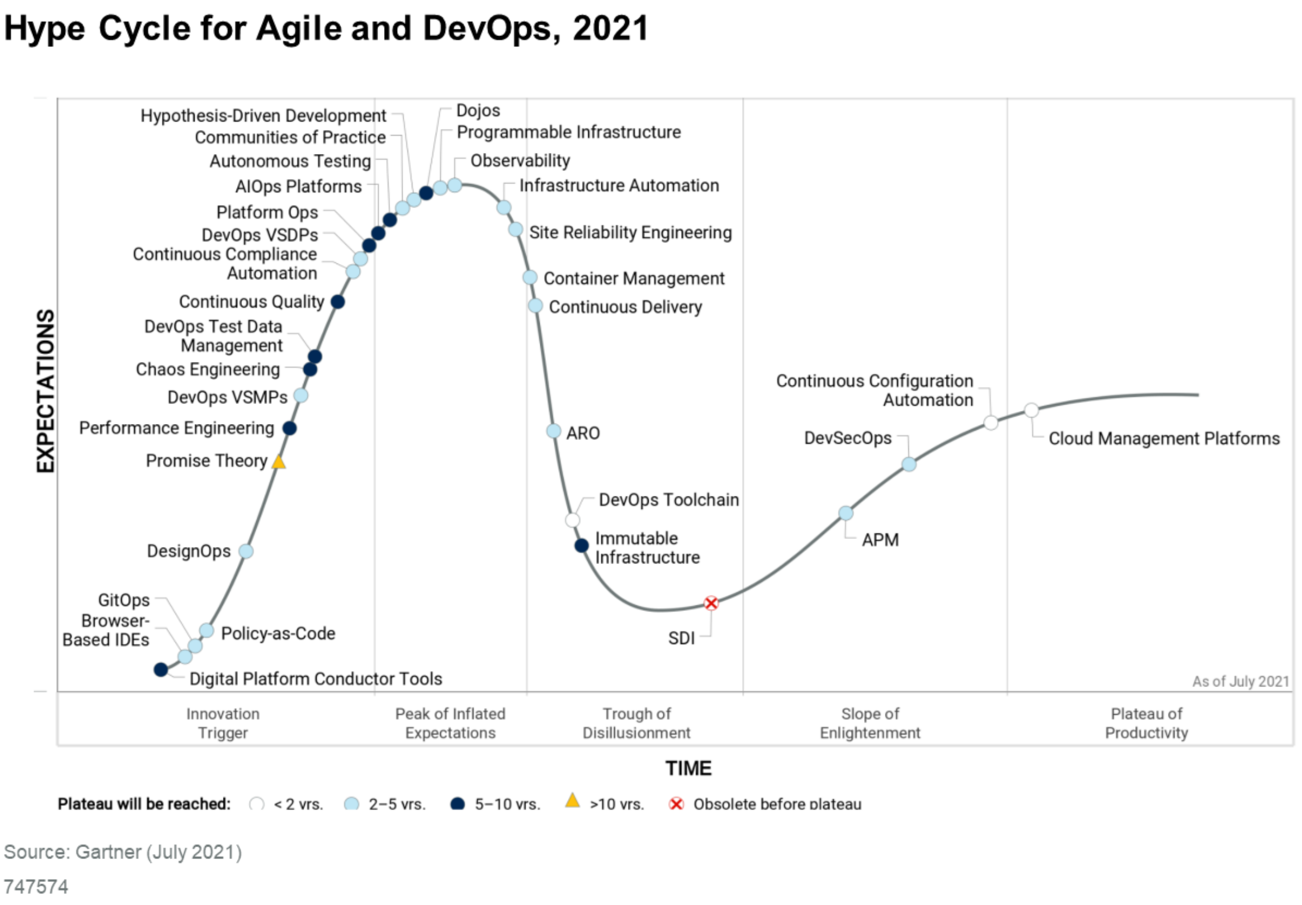
Gartner Recommended Reading

Market Guide for Cloud Management Tooling

Market Guide for Container Management

Appendixes

Figure 2: Hype Cycle for Agile and DevOps, 2021



Source: Gartner (July 2022)

Hype Cycle Phases, Benefit Ratings and Maturity Levels

Table 2: Hype Cycle Phases

Phase	*Definition*
Innovation Trigger	A breakthrough, public demonstration, product launch or other event generates significant media and industry interest.

Phase ↓	*Definition* ↓
Peak of Inflated Expectations	During this phase of overenthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes, but more failures, as the innovation is pushed to its limits. The only enterprises making money are conference organizers and content publishers.
Trough of Disillusionment	Because the innovation does not live up to its overinflated expectations, it rapidly becomes unfashionable. Media interest wanes, except for a few cautionary tales.
Slope of Enlightenment	Focused experimentation and solid hard work by an increasingly diverse range of organizations lead to a true understanding of the innovation's applicability, risks and benefits. Commercial off-the-shelf methodologies and tools ease the development process.
Plateau of Productivity	The real-world benefits of the innovation are demonstrated and accepted. Tools and methodologies are increasingly stable as they enter their second and third generations. Growing numbers of organizations feel comfortable with the reduced level of risk; the rapid growth phase of adoption begins. Approximately 20% of the technology's target audience has adopted, or is adopting, the technology as it enters this phase.
Years to Mainstream Adoption	The time required for the innovation to reach the Plateau of Productivity.

Source: Gartner (July 2022)

Table 3: Benefit Ratings

Benefit Rating ↓	*Definition* ↓
Transformational	Enables new ways of doing business across industries that will result in major shifts in industry dynamics.

Benefit Rating ↓	*Definition* ↓
High	Enables new ways of performing horizontal or vertical processes that will result in significantly increased revenue or cost savings for an enterprise.
Moderate	Provides incremental improvements to established processes that will result in increased revenue or cost savings for an enterprise.
Low	Slightly improves processes (e.g., improved user experience) that will be difficult to translate into increased revenue or cost savings.

Source: Gartner (July 2022)

Table 4: Maturity Levels

Maturity Levels ↓	*Status* ↓	*Products/Vendors* ↓
Embryonic	In labs	None
Emerging	Commercialization by vendors Pilots and deployments by industry leaders	First generation High price Much customization
Adolescent	Maturing technology capabilities and process understanding Uptake beyond early adopters	Second generation Less customization
Early mainstream	Proven technology Vendors, technology and adoption rapidly evolving	Third generation More out-of-box methodologies

Maturity Levels ↓	*Status* ↓	*Products/Vendors* ↓
Mature mainstream	Robust technology Not much evolution in vendors or technology	Several dominant vendors
Legacy	Not appropriate for new developments Cost of migration constraints replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2022)

Acronym Key and Glossary Terms

AIOps	Artificial intelligence for IT operations (AIOps)
APM	Application performance monitoring
ARO	Application release orchestration
SDI	Software-defined infrastructure
VSDP	Value stream delivery platform
VSMP	Value stream management platform