Hype Cycle for Site Reliability Engineering, 2023

Published 17 July 2023 - ID G00791221 - 102 min read

By Analyst(s): Hassan Ennaciri, Chris Saunderson, Daniel Betts

SRE practices enable organizations to drive agile software delivery while optimizing reliability, performance and cost. I&O leaders should leverage this Hype Cycle to assess their SRE maturity and invest in new practices and tools, and to upskill their teams to improve their SRE implementation.

More on This Topic

This is part of an in-depth collection of research. See the collection:

- 2023 Hype Cycles: Deglobalization, AI at the Cusp and Operational Sustainability

Strategic Planning Assumptions

By 2027, 75% of enterprises will use site reliability engineering practices across their organizations to optimize product design, cost and operations to meet customer expectations, up from 10% in 2022.

By 2025, 40% of organizations will implement chaos engineering practices as part of site reliability engineering initiatives, improving mean time to repair (MTTR) by an average of 90%.

By 2026, 70% of organizations that successfully applied observability will achieve shorter latency for decision making, enabling competitive advantage for target business or IT processes.

By 2025, organizations that invest in building digital immunity will increase customer satisfaction by decreasing downtime by 80%.

By the end of 2025, 30% of enterprises will establish new roles focused on IT resilience and boost end-to-end reliability, tolerability and recoverability by at least 45%.

Analysis

What You Need to Know

Given the sprawling complexity of digital infrastructure, supporting business-critical applications has become challenging. Site reliability engineering (SRE) practices are foundational to enable innovation and rapid deployments of software products while improving availability, reliability and performance. SRE is a discipline that combines systems, reliability and software engineering principles, and is used to design and operate scalable resilient systems.

Site reliability engineers work with their customers or product owners to understand operational requirements and define service-level objectives (SLOs). They also proactively work with product or platform teams to design and continuously improve systems resilience and prevent issues that impact customer experience.

The Hype Cycle

Organizations are under pressure to drive innovation, relying heavily on digital channels to reach their customers anywhere. The reliability of digital products and services is more critical than ever, hence the need to explore modern approaches such as SRE to balance reliability and change velocity.

The Key Principles of SRE

Embrace risk: Managing services is about balancing the risk of unavailability against the need to maintain and innovate the services.

For more information, see:What Is the Risk of Actually Losing Your Cloud Provider?

9 Principles for Improving Cloud Resilience

Focus on measure of customer experience: SLOs are the objectives that must be achieved for any service level to meet user reliability and performance expectations. An SLO continually measures the performance of a key performance metric, a service-level indicator (SLI), that is important to the customer. In combination with error budgets (see below), they provide a proactive mechanism to maintain or improve reliability:

- SLI: This is a quantitative measure of some aspect of the level of service being provided.

- SLO: This is a target value or range of values for a service level that is measured by an SLI.

For more information, see:

Improve Product Reliability by Applying SRE Principles to Service Operations

Eliminate toil: The SRE approach is to automate as much toil as possible, leaving more opportunities to perform higher-value work. It is also about automating reliability improvement at scale, by shifting left and codifying SLOs, and enforcing testing of service-level indicators.

For more information, see:

Innovation Insight for Continuous Infrastructure Automation

Ignition Guide to Creating an Automation Strategy for Financial Services Operations Leaders

Embed observability into distributed systems: To manage risk and assess whether the service is being impacted requires a dedicated focus on the use of monitoring systems to proactively measure SLO performance.

For more information, see:

Consider These Key Functional Areas for Application Performance Monitoring and ObservabilityTop Strategic Technology Trends for 2023: Applied Observability

Adhere to release engineering as a discipline: Release engineering focuses on high velocity of change by utilizing techniques including the automation of the release process. This is likely to be out of scope for service operations.

For more information, see:

Improve Product Team Speed and Agility by Minimizing Dependencies: Approaches From 3 Leading Organizations

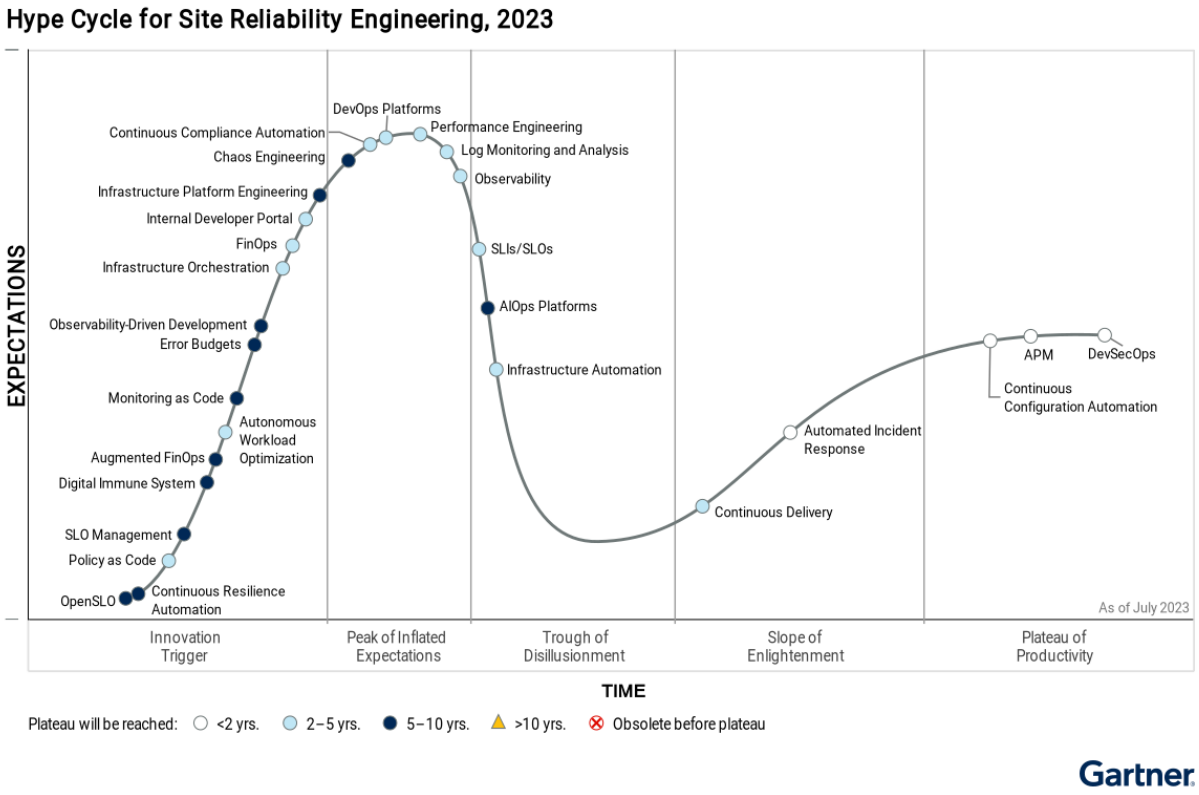
Magic Quadrant for DevOps Platforms

Embrace blameless postmortems for incidents: SREs practice blameless postmortems that remove the assumption that outages are caused by operators, and focus on finding out what caused the human to make the error rather than blaming an individual or a team.

For more information, see:

7 Steps to Start and Evolve an SRE Practice

Figure 1: Hype Cycle for Site Reliability Engineering, 2023



The Priority Matrix

The Priority Matrix maps the time to maturity of a technology/framework on a grid in an easy-to-read format. It answers two high-priority questions:

1. What are the potential benefits and value from an innovation?

2. When will the innovation be mature enough to provide this value?

During the next two to five years, customer experience, and expectations of reliability and performance of digital products will require increased adoption of SRE practices. The combination of cloud-native applications, SaaS, platform as a service (PaaS), and third-party services and dependencies will require SRE approaches to meet these expectations.

APM, observability and AIOps tools are powerful analytics platforms that ingest multiple telemetry feeds and provide critical insight that SRE rely on for health, performance and, increasingly, security of deployed systems. They require SREs with advanced skills and system-level understanding for successful implementation.

Increased use of AI techniques and emergence of generative AI across the technology spectrum will enable SRE to take advantage of these capabilities. This will drive greater capability to optimize operations by leveraging tools that enable autonomous workload optimizations, maximizing performance and reliability while minimizing resources.

Table 1: Priority Matrix for Site Reliability Engineering, 2023

(Enlarged table in Appendix)

Benefit	Years to Mainstream Adoption			
↓	Less Than 2 Years ↓	2 - 5 Years ↓	5 - 10 Years ↓	More Than 10 Years ↓
Transformational	DevSecOps	Observability	Augmented FinOps Continuous Resilience Automation Error Budgets SLO Management	
High	APM Automated Incident Response Continuous Configuration Automation	Autonomous Workload Optimization Continuous Delivery DevOps Platforms FinOps Infrastructure Automation Infrastructure Orchestration Internal Developer Portal Log Monitoring and Analysis Performance Engineering Policy as Code SLIs/SLOs	AIOps Platforms Digital Immune System Infrastructure Platform Engineering Monitoring as Code Observability-Driven Development OpenSLO	
Moderate		Continuous Compliance Automation	Chaos Engineering	
Low				

Source: Gartner (July 2023)

On the Rise

Continuous Resilience Automation

Analysis By: Hassan Ennaciri, Chris Saunderson

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Embryonic

Definition:

Continuous resilience automation (CRA) is a process that enforces and integrates operational requirements continuously throughout a software life cycle. CRA can be used to dynamically manage all aspects of modern software products to ensure that they're in line with the organization's IT resilience program. This is an important element for organizations looking to improve their overall resilience strategy in today's complex and turbulent environments.

Why This Is Important

Many organizations are becoming increasingly aware of business disruptions due to IT. Because most of these organizations rely on digital channels to conduct business, even a minor IT outage can cause a major impact on customers and revenue. The CRA provides a consistent approach to different stakeholders involved in software value streams to integrate controls to meet operational requirements for software products.

Business Impact

CRA seeks to automate IT product resilience and be able to dynamically adjust, as defined factors change. This includes defining and instrumenting service-level objectives (SLOs) and disaster recovery (DR) objectives, while continuously monitoring and validating the targets. By automating operational requirements in all phases of a product's life cycle, site reliability engineers (SREs) can ensure optimal product resilience that meets business requirements, while optimizing cost.

Drivers

- **IT resilience:** Organizations that rely heavily on complex digital products are at a greater risk of IT failures. They are designing an IT resilience program, and need a way to automate and enforce operational requirements at scale.

- **Regulatory requirements:** Organizations that need to adhere to operational requirements, such as business continuity/disaster recovery (BC/DR), from a regulatory compliance perspective.
- **Team health:** Product teams and SREs can leverage CRA to improve response to failures and disruptions and enable their businesses to gain a competitive advantage by adapting to changing conditions.
- **Customer expectations:** Customers have high expectations when using digital products from their mobile devices or home computers. Leveraging CRA can ensure the delivery of reliable and performant services by minimizing the impact to the services due to IT system failures or unexpected events.
- **Cost optimization:** CRA can help reduce costs by eliminating manual processes and downtime, as well as optimizing the cost of high-availability architecture and DR solutions.
- **Agile systems:** Automating operational requirements and integrating them in the early phases of development will improve system design and enable product teams to focus more on features that meet customer needs.

Obstacles

- **Proliferation of tools:** Too many tools with overlapping capabilities used in siloed approaches that make it hard to have end-to-end solutions. No single tool currently enables all the practices of CRA, so SREs will need a few.
- **Cultural:** CRA is a modern approach that requires teams to shift their approach to work on operational requirements. This can be the hardest obstacle to overcome, because teams may have other priorities that make it difficult to start CRA. A lack of management support can also be an issue.
- **Skills required:** CRA requires many advanced skills in automation, cloud native architecture and SREs. These resources are hard to find and hire, because demand is high.
- **Awareness:** Many teams are not aware of the different CRA practices, and resilience is managed with siloed, traditional processes and tools.

User Recommendations

- Collaborate with other stakeholders: CRA is a complex process that requires SREs to continuously collaborate with product owners and development teams to prioritize operational requirements with software delivery velocity.
- **Start small:** Start with an important system that has acceptable levels of risk, but can demonstrate real value to stakeholders, then build from there and evolve.
- **Integrate resilience into the entire life cycle of products:** It is important to integrate operational requirements at an early stage of design and development of products.
- **Select the right tools:** There are so many different tools for testing, monitoring and observability. Perform an assessment of existing tools and select tools that provide end-to-end visibility and can integrate with your continuous integration/continuous delivery (CI/CD) toolchains.

Sample Vendors

AWS; Blameless; Nobl9

Gartner Recommended Reading

IT Resilience — 7 Tips for Improving Reliability, Tolerability and Disaster Recovery

Improve Software Quality by Building Digital Immunity

Improve Product Reliability by Applying SRE Principles to Service Operations

Assessing Site Reliability Engineering (SRE) Principles for Building a Reliability-Focused Culture

OpenSLO

Analysis By: Hassan Ennaciri, Chris Saunderson

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

OpenSLO is an open-source project providing a standard format to define and measure a service-level objective (SLO) for modern software products. It relies on YAML Ain't Markup Language (YAML) specification to provide a declarative language for defining reliability and performance targets. It enables codifications of SLOs by creating a vendor-neutral implementation. SLOs can be treated as version-controlled software suitable for continuous integration/continuous delivery (CI/CD) tools.

Why This Is Important

Many organizations struggle to implement SLOs at scale. Most of them start their site reliability engineering (SRE) journey by optimizing the operations of their products. But they encounter obstacles when trying to evolve the practice and take a "shift left" approach to enforce operational requirements. OpenSLO is a viable solution that provides standards to automate, define and measure SLOs. It is gaining traction and has potential to become a differentiator when selecting SRE tools.

Business Impact

OpenSLO allows site reliability engineers to collaborate with product owners and define SLOs for every critical use case of products that map to a user journey. This provides stakeholders from product teams, site reliability engineers and platform teams with a consistent approach to manage SLOs and enable efficiencies by sharing SLOs across multiple products. OpenSLO makes it easy to monitor business metrics and identify issues that impact user experience.

Drivers

- **SLO enforcement:** By using OpenSLO to codify SLO targets for their complex products, product teams and site reliability engineers can optimize development and operations and ensure reliability and performance targets meet customer expectations.
- **SLO management:** Because SLOs can be treated as software, changes are version controlled, visible, auditable and suitable for common CI/CD tools.
- **Scalability and automation:** OpenSLO enables organizations to make better decisions at scale in an automated way to set and monitor reliability and performance targets.

Obstacles

- **Skills required:** OpenSLO is new and many teams lack the expertise or maturity to adopt it.
- **Acceptance of the standard:** As OpenSLO is a new way of creating, governing and managing SLOs, teams may face challenges as it will require considerable integration with existing tools and processes.
- **Lack of broad support from vendors:** Not many vendors support OpenSLO.
- **Cost of implementation:** Implementing OpenSLO would require investment in new tools and additional training for teams.

User Recommendations

- Scope a strategy for leveraging OpenSLO. Collaborate with stakeholders from SRE, product teams and platform teams to develop the adoption framework with clear objectives.
- Select the right tools based on the objectives.
- Start small by implementing OpenSLO for one or two systems, iterate and improve until you achieve the desired outcomes before you broaden adoption. SLOs are dynamic and need to be continuously managed and adjusted.
- Use version control as OpenSLO works better with infrastructure as code. Leverage version control and existing CI/CD tools to manage changes to the OpenSLO instrumentation.

Sample Vendors

GitLab; Lightstep; Nobl9; Sumo Logic

Gartner Recommended Reading

Quick Answer: What Is Site Reliability Engineering?

7 Steps to Start and Evolve an SRE Practice

Improve Software Quality by Building Digital Immunity

Improve Product Reliability by Applying SRE Principles to Service Operations

Assessing Site Reliability Engineering (SRE) Principles for Building a Reliability-Focused Culture

Policy as Code

Analysis By: Paul Delory

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Policy as code (PaC) 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 today to enforce infrastructure compliance, authorization, Kubernetes admission control, and more.

Why This Is Important

In the most mature automation pipelines, infrastructure and operations (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, the I&O function 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 implements policies automatically, with implicit compliance guarantees.
- **Alignment of security and operations 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

- **PaC tooling:** Several dedicated PaC tools are now on the market, many of them are open-source. The Open Policy Agent, a Cloud Native Computing Foundation project, has become the *de facto* standard for PaC. Indeed, even some other PaC tools now use Open Policy Agent policies alongside or instead of their own policy engines.
- **Increasing regulation:** New regulations such as GDPR have 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 — caused by infrastructure misconfigurations — 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 for its company getting negative 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 are finding that they need better policy enforcement, and PaC can 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 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 supports 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 or 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, expanding to others once PaC has proven its value.
- **Upskill staff:** PaC languages are not always intuitive. Technical staff will need practice to reach proficiency.
- **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:** Use PaC to build guardrails for automation tools, so that they cannot take actions that are out of compliance.
- **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.

Sample Vendors

HashiCorp; Palo Alto Networks; Progress; Pulumi; Styra

Gartner Recommended Reading

Using 'Policy as Code' to Secure Application Deployments and Enforce Compliance

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

Magic Quadrant for DevOps Platforms

SLO Management

Analysis By: Hassan Ennaciri, Daniel Betts

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Service-level objective (SLO) management involves setting initial SLOs for a service or product and continuously monitoring and improving them. It is an important process for organizations that adopt site reliability engineering (SRE) principles, as it allows them to set SLOs for each service and select their service-level indicators (SLIs) to track and measure their performance. SLO management helps optimize customer experience.

Why This Is Important

As customers have high expectations while using digital products, many organizations are seeking to scale adoption of SRE practices to balance reliability and the need to innovate faster. SLO management focuses on the features customers care about and the way they feel when they use the services. However, setting these services at scale can be challenging, hence the need for process and tooling to facilitate its management.

Business Impact

Leveraging SLIs and SLOs is key to ensuring SRE success. One of the main benefits is to enable SRE to provide product teams and operations teams with metrics that focus on customer experience. SLO management provides a scalable approach that enables site reliability engineers to collaborate with multiple product teams to define, iterate and communicate service levels. This helps them to continuously set and adjust SLOs to balance the need for releasing features and making operational improvements with keeping the customer happy.

Drivers

- **Customer expectations:** Customers have high expectations while using digital products. SLOs measure the availability and performance levels that meet those customers' expectations. This allows the teams to monitor these levels and proactively identify and correct them to keep the customers happy.
- **Complexity of modern software and platforms:** Leveraging SLOs is needed to enable complex modern software architectures. Site reliability engineers need a platform to help them manage this at scale and ensure they continuously improve to meet customer expectations.
- **Cost optimization:** By ensuring a robust SLO management process, site reliability engineers can continuously optimize design and operations of the services and reduce cost of design and operations.
- **Compliance requirements:** SLO management can be used to ensure products meet business continuity compliance requirements, and to ensure availability during failure or unexpected events.
- **Service-level agreements (SLAs):** SLIs and SLOs are metrics that are used internally and create targets that exceed SLAs. This helps meet customer expectations and ensures meeting the contractual agreements.

Obstacles

- **Lack of understanding of SLOs:** Many leaders from business and IT don't understand what SLOs are, how they are different from SLAs, and how they work. This leads to resistance to implementing modern practices.
- **Skills required:** Many teams are either unfamiliar with SLO implementation or don't have the skills required to do it properly. Implementation requires experienced site reliability engineers, but they are hard to find and can be very expensive resources.
- **Lack of collaboration and acceptance of change:** It is challenging to initially implement SLOs as they require involvement and collaboration from all the teams involved in software product delivery. They also require investments on tools and additional resources that many organizations don't have.
- **Lack of telemetry:** Setting SLOs is a complicated process to do without enough historical data. This is especially hard in complex systems with interdependencies.

User Recommendations

- **Establish an SLO process:** SLOs need to be continuously evaluated and updated, so it is critical to establish an SLO process and ensure other stakeholders are aligned with it. The process needs to include defining the SLOs, monitoring them, reporting results and agreeing on actions when they are breached.
- **Educate all stakeholders:** A key to success is educating everyone on the concept of SLO management, so that they understand how it can be used to scale SLO implementation and how it will help optimize adoption to achieve the desired results.
- **Invest in a platform that supports SLO management:** Adopting a platform that can integrate with your existing observability and monitoring tools will facilitate SLO management. It can also provide visibility into SLOs and their performance to enable collaboration across teams. This will give teams the data they need to continuously improve.
- **Start small:** Start small by implementing SLO management on a couple of products, focus on clear objectives, learn and grow iteratively.

Sample Vendors

Blameless; Datadog; Dynatrace; Grafana; New Relic; Nobl9; PagerDuty; Prometheus

Gartner Recommended Reading

7 Steps to Start and Evolve an SRE Practice

Improve Software Quality by Building Digital Immunity

Improve Product Reliability by Applying SRE Principles to Service Operations

Assessing Site Reliability Engineering (SRE) Principles for Building a Reliability-Focused Culture

Autonomous Workload Optimization

Analysis By: Pankaj Prasad, Manjunath Bhat, Hassan Ennaciri

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Embryonic

Definition:

Autonomous workload optimization tools maximize performance while minimizing resources through one or more of the three approaches. First, resource and cost optimization, including minimizing compute resources. Second, performance, including optimizing code for running applications, e.g., fine-tuning configuration of the underlying runtime. Third, dynamic workload placement, including reshuffling jobs and allocating resources to workflows for maximum utilization and minimizing idle time.

Why This Is Important

Optimizing IT costs has always been on the agenda for infrastructure and operations (I&O) leaders. Additional pressures include becoming energy efficient and simultaneously ensuring optimal performance to customers. For the results to be impactful, this has to be done at scale while also dealing with the trade-offs required to optimize across the dimensions of cost, performance and energy. The scale and trade-off challenge is better addressed via autonomous operations and algorithms.

Business Impact

Automated workload optimization tools enable organizations to optimize their IT costs, and avoid wastage of IT resources, while balancing the performance requirements of applications to ensure that customers are not impacted. These tools can enable organizations to become more energy efficient, and in the process, take a step closer toward their sustainability goals.

Drivers

Once organizations achieve their targets for availability and performance, the next goal is optimization along a few dimensions:

- **Optimize performance:** Enterprises should aim for an optimal range of performance where cost is balanced while avoiding negative impact to customer experience, for example, by leveraging error budgets.
- **Optimize workloads and cost:** Mapping application resource utilization to demand patterns, especially in virtualized and cloud-native architectures, enterprises can dynamically project resource requirements thereby also optimizing their IT spend.
- **Energy efficiency:** I&O leaders are committing to sustainability goals and achieving greater energy efficiency is one of their primary targets. This is also a driver for sustainable software engineering.
- **Speed and scale:** Manual intervention in optimization goals will not be sustainable in the long run. The need to speed and scale is a driver for autonomous tools and processes.

Obstacles

- **Lack of maturity:** Ad hoc workloads and lack of appropriate tools to capture the right metrics are inhibitors toward realizing the full potential of workload optimization tools.
- **Standardization:** Many enterprises have a nonstandardized IT architecture where these tools will only tackle simple optimizations and require high efforts for significant results.
- **Collaboration challenges:** Data sharing between customer-facing, preproduction and production teams is crucial to ensure no negative impact due to optimization efforts. For e.g., to identify the appropriate error budgets and to ensure trade-offs do affect other operation areas like increase in errors.
- **Disconnect with business:** Business leaders' buy-in is crucial for continued investment, and is hampered due to a lack of appropriate reporting, i.e., translating resource optimization gains in dollar terms, proper tracking and trend analysis for comparisons.
- **Narrow focus:** Most of the tools in this space have a narrow focus on virtual operating systems or Kubernetes, limiting their adoption to workloads that leverage such architectures.

User Recommendations

- Start small and get some perspective on resource utilization vs. performance to identify patterns and correlations. Use this data to pilot autonomous optimization for non-mission-critical workloads to prove value and capability before rolling out autonomous optimization for all applications.
- Use autonomous optimization as a driver for improving maturity in monitoring metrics and to speed up standardization initiatives across IT architectures and processes to maximize the value of optimization objectives.
- Improve collaboration across customer-facing, preproduction and production teams to ensure regular review and appropriate data exchange, including, but not limited to customer engagement, performance, IT resource requirement and utilization patterns, and capacity.
- Collaborate with business leaders and IT stakeholders to ensure measurements and benefits are suitably conveyed and business and technical objectives are appropriately captured.
- Optimize continuously, which means reviewing results, revisiting targets and refining goals since this has to be an ongoing effort.

Sample Vendors

Akamas; CAST AI; Cisco Systems (Opsani); Control Plane; Google; Granulate; IBM; Sedai; StormForge

Gartner Recommended Reading

Market Guide for Digital Platform Conductor Tools

Market Guide for AIOps Platforms

Augmented FinOps

Analysis By: Adam Ronthal, Dennis Smith

Benefit Rating: Transformational

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

FinOps applies the traditional DevOps concepts of agility, continuous integration and deployment, and end-user feedback to financial governance, budgeting and cost optimization efforts. Augmented FinOps automates this process through the application of artificial intelligence (AI) and machine learning (ML) practices — predominantly in the cloud — to enable environments that automatically optimize cost based on defined business objectives expressed in natural language.

Why This Is Important

In the cloud, it is now possible to assess the cost of a specific workload or collection of workloads assigned to a project. However, price/performance — the primary measure of cloud efficiency — is difficult to assess due to the complexity and diversity of choice in underlying cloud infrastructure and service offerings and a lack of consistency in pricing models. Augmented FinOps can automate this process by applying AI/ML techniques.

Business Impact

The automation of cloud budget planning and financial operations will allow businesses to express their objectives — ideally in natural language — and allow their cloud ecosystems to automatically optimize the underlying cloud resources to meet those objectives. This will result in more efficient use of resources and, therefore, optimal spend by reducing/eliminating misaligned or poor use of cloud infrastructure and service offerings.

Drivers

- Practitioners are increasingly realizing that cloud is fundamentally a complex cost optimization exercise.
- Cloud adopters have a strong desire for transparency into cloud spending.
- Buyer inexperience is leading to either under-provisioning and associated resource contention or overprovisioning and spending more than is needed.
- Vendors are positioning cost-effectiveness as a competitive differentiator in their go-to-market strategies.
- Practitioners need to reduce the unpredictability of cloud spending when using cloud infrastructure and services for analytics, operational database management systems (DBMSs), data lakes and other applications, including custom IT infrastructure.
- Consumption-based usage remains common in earlier stages of cloud adoption, driving the need for augmented FinOps, although commit-based usage mitigates some unpredictability.
- Cost overruns are often obscured, downplayed, or dismissed by line of business implementers, requiring augmentation to achieve holistic and comprehensive cost optimization.
- Automation of financial governance controls in cloud environments provides increased predictability and cost optimization with less operational effort.
- Solid financial governance frameworks are positioning organizations to take advantage of FinOps.
- Emergence of specific roles — like FinOps practitioner or cloud economist — focused on FinOps practices and cost optimization means organizations have the expertise to address augmented FinOps.
- Owing to their complexity, cloud environments are ideally suited for the application of ML and AI methods to automate processes and track price and performance.
- Core FinOps capabilities are being delivered in three ways: Homegrown solutions, cloud service provider (CSP) instrumentation and third-party vendors. Increasingly practitioners are seeking out third-party or CSP tools to address their needs. All of these have a broad objective of adopting augmented capabilities as a means of competitive differentiation.

Obstacles

- Cloud service provider pricing models remain needlessly complex and diverse.
- Cloud ecosystems are (and will remain) open to third-party participants, which implies multiple commercial arrangements with multiple providers.
- Standards for cloud cost, usage and billing data like the FinOps Foundation's FOCUS proposal have yet to be broadly adopted. APIs for communicating performance data within the context of a broader ecosystem have yet to emerge. Both of these are required to assess the primary measure of success: price/performance.

User Recommendations

- Seek out service offerings to automate (via AI/ML) performance, consumption and pricing options. Increasingly, incorporate these capabilities into cloud data ecosystems that will learn from consumption patterns as they seek to optimize the underlying resources, and by extension, cloud spending through orchestration and optimization.
- Apply Gartner's FinOps Maturity Model to assess FinOps offerings in terms of their ability to address the following core capabilities: Observe, report, recommend, predict and optimize. The last three introduce augmented FinOps capabilities.
- Plan to use multiple tools to address the full scope of requirements. Many tools are broad in reach, but do not go deep into prescriptive recommendations. Others are tightly scoped and provide very targeted optimizations. Expect to spend time combining multiple tools to achieve broad and deep capabilities.

Sample Vendors

Acceldata; Anodot; Apptio; Capital One Software; Densify; Enteros; Finout; OtterTune; Sync Computing; Unravel Data

Gartner Recommended Reading

How to Identify Solutions for Managing Costs in Public Cloud IaaS

A Guidance Framework for Selecting Cloud Management Tools

Emerging Tech: Data Management Product Leaders Must Implement Augmented FinOps in Their Cloud Solutions

CDAOs and CFOs Must Drive Business Value in the Cloud Through Collaboration

Financial Governance Is Essential to Successful Cloud Data and Analytics

Digital Immune System

Analysis By: Joachim Herschmann

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

The digital immune system (DIS) approach interlinks practices from six areas: observability, software testing, chaos engineering, automated remediation, site reliability engineering (SRE) and software supply chain security. This ensures high resiliency and quality of applications. It focuses on making applications more resilient so that they recover quickly from failures, rather than impacting users and business performance.

Why This Is Important

Software development teams are under pressure to deliver faster, while dealing with increasingly complex architectures, compliance needs and constantly evolving technology stacks. This exposes organizations to operational and business risks, when applications and services are severely compromised or stop working altogether. The DIS approach combines and amplifies several resilience engineering practices, to mitigate these risks and deliver a high level of business impact.

Business Impact

A digital immune system strategy helps engineering teams build and deploy software confidently, as they can continuously assess system health, remediate issues and optimize system performance. Also, systems designed for digital immunity help identify code quality issues related to performance, security and other operational attributes early in the software development life cycle. This creates software engineering teams that are focused on connecting software development to business value.

Drivers

- **Highly distributed, cloud-based applications:** Software engineering leaders struggle to plan for all eventualities of how modern, highly distributed software systems may fail. Troubleshooting issues in distributed applications requires fundamentally different techniques, compared with monolithic or client/server applications.
- **Business continuity:** As an enabler of business operations, IT plays a significant role in business continuity and enterprise success. In response to pressure from business stakeholders, software engineering leaders are looking for new ways to improve the resilience of business-critical systems and reduce their vulnerability.
- **Risk mitigation:** Prominent examples of how leading digital enterprises have suffered from software glitches, outages or security incidents put additional pressure on software engineering leaders to mitigate risks to the business.
- **Continuous improvement:** Organizations need a frame of reference to improve the resilience of digital systems, measure progress and use failures as learning opportunities. Teams must continuously improve their skills to minimize the impact of a failure, with a focus on preserving the overall health of the system.
- **Developer experience:** Developers often get disrupted in their daily work, as they have to deal with incidents and frequent firefighting exercises, which are detrimental to developer experience. A DIS approach fosters a mindset of continuous quality improvement, and enables a shift toward building quality and security into the product.
- **Reliability as a key differentiator:** Organizations that have recognized reliability as a differentiator in the market need key insights, operational agility and competence. They acquire these through the combination of observability, AI-augmented testing, chaos engineering, SRE and security practices.

Obstacles

- **Technology focus:** Tools are enablers of a digital immune system strategy, but just throwing technology at a problem will not solve it. Tools automate and facilitate your processes and practices, but they need to be in place first and evolve over time.
- **Failure to measure:** The core elements of a digital immune system are not new. But most organizations are only using some parts of it, often in uncoordinated ways, and without a plan to close gaps and evolve them into a more holistic strategy. It is, therefore, important to measure progress in the evolution of the different practices and the success of technologies that enable them.
- **Internal pushback:** Implementing a digital immune system requires engaging stakeholders across the organization and seeking out opportunities for collaboration and improvement. Such a holistic approach can be seen as too far-reaching and lead to turf wars.

User Recommendations

- Adopt a DIS strategy as a standard software engineering approach to prepare their teams to handle unexpected and unforeseeable system behaviors, quickly remediate software defects, and minimize the impact on users. Support a “you build it, you run it” (YBIYRI) approach.
- Assess which business capabilities have the highest priority or will benefit the most from DIS investments.
- Establish a platform engineering team to build standard platform support for a DIS. Create dedicated communities of practice (CoPs) to share lessons learned, guiding principles, reusable assets, standards, tools and any AI-based insights realized.
- Encourage and reward resilience improvements across the organization, especially collaboration on DIS opportunities.
- Foster a collaborative culture between development, security and operations teams to ensure ongoing support for these initiatives.

Gartner Recommended Reading

Top Strategic Technology Trends for 2023: Digital Immune System

Improve Software Quality by Building Digital Immunity

Error Budgets

Analysis By: Hassan Ennaciri, Chris Saunderson

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

An error budget is a key site reliability engineering (SRE) principle used to govern how to make changes to a system, while balancing the need for rapid innovation with risk. An error budget is the number of errors a service can accumulate during a certain period without affecting users. The error budget represents the difference between “perfect” and the service-level objective (SLO).

Why This Is Important

Customers have high expectations when using digital products. They continuously demand more features in products, while expecting them to be available and reliable. Because error budgets represent the number of errors a product or service can accumulate before users start to become unhappy, they can be used to automate governance, when delivering and operating products.

Business Impact

Leveraging error budgets is an essential practice to optimize operations, maximize delivery speed and automate the governance of software releases, especially for organizations implementing DevOps and SRE practices. The primary benefit is to provide product teams with a metric that can help them balance the need to release features and introduce operational improvements, while safeguarding the SLOs.

Drivers

- **Reliability:** The main reason SRE teams use error budgets is to ensure optimal operations of systems. The acceptable levels of error offer product and operations teams flexibility to design and operate systems that are reliable and resilient within an measurable range of risks.
- **Minimized change failure rates:** Error budgets help measure high frequency of changes. while ensuring that change failure rates are maintained at or below acceptable levels.
- **Continuous improvements:** The primary goal of leveraging error budgets is to have all stakeholders embrace a culture of continuous improvements, by agreeing on target levels of acceptable errors, and collaborate to improve when the targets have been exceeded.
- **Customer expectations:** Although customers have high expectations when using digital products, they also have a certain tolerance for errors and latency. The negative impact and loyalty to products can be avoided by staying within those tolerance levels.
- **Cost optimization:** By designing and operating systems in a way that is good enough to meet customer expectations, SRE teams can optimize architectural and operational costs.

Obstacles

- **Acceptance of error budgets:** Many leaders from business and IT don't connect the term "error" with improvement or value and don't fully understand the benefits. Because of this disconnect, they resist the adoption of error budgets.
- **Skills required:** Implementing error budgets requires a mature SRE practice that many organizations lack. Experienced SRE teams are hard to find and can be expensive resources.
- **Complexity of setting and managing error budgets:** It is difficult to set error budgets initially, because they directly correlate with SLOs that are also hard to set. Targets that are too strict can affect agility and innovations, and teams would focus more on nonfunctional requirements.
- **Lack of telemetry:** Setting error rates is an onerous task to perform without sufficient historical data. This is especially challenging in complex systems with interdependencies.

User Recommendations

- **Start with teams adopting SRE practices:** Implement error budgets with product teams that have already embraced SRE practices. Start small with a few services, continuously iterate and improve to demonstrate value.
- **Educate all stakeholders:** A key to success is to educate everyone on the concept of error budgets and how this relates to customer experience (CX) and how they can be important to improving reliability. Contrast error budgets with customers' perception of value to drive home the point that error budgets need not have negative connotations.
- **Manage error budgets along with SLOs:** Like SLOs, error budgets need to be continuously evaluated and adjusted. You can't set them once and for all — they need to be reviewed periodically. It is also critical to ensure that they are consistent with customer feedback.
- **Tools and telemetry:** Invest in tools that enable SLO and error budget management, as well as tools that ingest telemetry and provide insight into error rates.

Sample Vendors

Blameless; Harness; Nobl9; SquadCast

Gartner Recommended Reading

7 Steps to Start and Evolve an SRE Practice

Improve Software Quality by Building Digital Immunity

Improve Product Reliability by Applying SRE Principles to Service Operations

Assessing Site Reliability Engineering (SRE) Principles for Building a Reliability-Focused Culture

Monitoring as Code

Analysis By: Hassan Ennaciri, Pankaj Prasad

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Emerging

Definition:

Monitoring as code (MaC) is the process of applying software principles to monitoring, meaning the configuration of monitoring is designed to enable its management, like software. With MaC, the configuration of monitoring is codified, version-controlled, tested and automated. This flexibility offers DevOps teams the option to apply a shift-left approach for fast and consistent monitoring across systems.

Why This Is Important

Traditional monitoring approaches require manual and, in some cases, inflexible configurations that hinder DevOps teams' ability to support continuous delivery with frequent deployments. Manually changing monitoring introduces risk of misconfigurations and inconsistencies that can extend the mean time to detect failures, causing a larger impact on customers.

Business Impact

By leveraging MaC, teams can collaborate, share, optimize and speed up monitoring configurations to ensure a consistent, efficient approach to monitoring that can support high-velocity changes, while reducing the risk of missed alerts. MaC enables DevOps teams to automate monitoring configuration by leveraging the continuous integration/continuous delivery (CI/CD) toolchains or DevOps platforms they use to deliver features or infrastructure.

Drivers

- **Customers' Expectations for Reliability and Performance:** This makes monitoring very important, as it ensures reduced impact on customers during unexpected events. Monitoring as code provides a consistent way to manage monitoring configuration at scale.
- **Agile, DevOps and GitOps:** These methodologies and new ways of working drive the need for automation to support the velocity and high frequency of changes. MaC meets these needs because it leverages the modern architecture's attribute of instrumentation, while adding agility and efficiency to monitoring configuration.
- **Cloud-Native Architectures:** Modern cloud-native architecture, microservices and distributed nature drive the need for complex monitoring configurations.
- **Infrastructure Platform Engineering:** Delivering infrastructure services via infrastructure-as-code pipelines drives the need to automate monitoring configurations and take advantage of CI/CD processes to change, test and deploy the applications.

Obstacles

- **Resistance to Change:** Traditional monitoring is done by the IT operations team and is manual. This new modern approach is a paradigm shift that requires collaboration and automation.
- **Security and Compliance:** In-depth monitoring requires privileged and granular role-based access. MaC is relatively new and will require security teams to reassess governance and apply more stringent evaluations to ensure the organization will not be in breach of compliance requirements.
- **Skills and Expertise:** MaC is an advanced practice that relies on software engineering skills, which many I&O teams lack. IT organizations will require new skilled resources or training.
- **Lack of Support With Traditional Monitoring Tools:** This will require investment in new tools that support MaC.

User Recommendations

- **Collaborations:** Monitoring complex systems with modern architecture is a team activity. Teams need to continuously collaborate to ensure the relevant features and metrics are monitored.
- **Tools:** Invest in tools that support infrastructure as code.
- **Configurations via DevOps:** Treat the MaC as software, taking advantage of version control, testing and automated deployments. Teams should leverage the same DevOps toolchains to manage the monitoring code.
- **Continuous Improvements**: Continuously seek operational feedback, and use qualitative and business KPIs to improve the quality of monitoring configurations, as well as the process.

Sample Vendors

AppDynamics; Checkly; Datadog; Dynatrace; Grafana; New Relic; Splunk; Zoho

Observability-Driven Development

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% 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 observable systems. ODD works by instrumenting code to unravel a system's internal state with externally observable data. As part of a shift-left approach to software development, ODD makes it easier to detect, diagnose and resolve unexpected anomalies early in the development life cycle.

Why This Is Important

Building observable systems can expedite issue resolution as observability data is a useful debugging aid. Designing for observability also amplifies the benefits of 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 and how it is being used. However, the practices to institutionalize ODD are still emerging.

Business Impact

Built-in observability helps develop reliable software. ODD reduces the time to troubleshoot issues in production and preproduction environments. This feature helps software engineering teams reduce cycle time, making developers more productive during testing. ODD also helps ship code confidently since observable data makes it easier to troubleshoot production issues, thus minimizing downtime. In business terms, it translates to compliance with regulatory and contractual SLAs.

Drivers

- Organizations adopting SRE and chaos engineering practices need the data provided by observability design. These practices are fundamentally data-driven and support software reliability.
- User experience (UX) is subjective and difficult to measure. It depends on various factors that span the complete technology stack. Some factors, such as last-mile network connectivity, affect UX but are difficult to optimize unless systems are designed to be observable and extract data related to such signals, like response rates and latency.
- Mobile and edge environments present unforeseeable challenges since applications 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 can help capture and investigate the “unknown unknowns” at runtime.
- Distributed systems exhibit emergent behavior and are difficult to predict. Therefore, the need for observability increases as issues can arise due to unexpected component interactions. Troubleshooting issues in distributed applications requires fundamentally different techniques than monolithic or client and server applications. Building observability narrows down the problem domain and helps engineers inspect the problematic component.
- OpenTelemetry is an open standard that has seen increased open-source implementations and vendor adoption over the past year. This has improved consistency when adopting ODD across products. Organizations also favor using open standards and protocols to ingest telemetry data. This makes ODD accessible to most developers even when they lack the budget for commercial tools.

Obstacles

- 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 used to implement observability.
- Software engineering leaders must overcome the perception that observability can be achieved merely by implementing an observability tool. Observability is a domain 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. As a standard practice, ODD provides greater benefits when all system components are designed with an observability mindset. For example, distributed tracing requires that all components contributing to the trace are “instrumented” and propagate context for diagnosing response-time issues. Platform teams entrusted with driving ODD at scale can help overcome this obstacle.

User Recommendations

- Adopt ODD as a standard software engineering practice to handle unexpected and unforeseeable system behaviors and anomalies.
- Make observability a priority since it provides critical insights to resolve production errors. It 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 technologies, such as OpenTelemetry.
- Be wary of vendor hype regarding observability merely 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

Chronosphere; Datadog; Dynatrace; Grafana; Honeycomb; Logz.io; New Relic; observIQ; ServiceNow; Splunk

Gartner Recommended Reading

How to Identify and Resolve Application Performance Problems

Improve Software Quality by Building Digital Immunity

FinOps

Analysis By: Lydia Leong

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

FinOps is the model proposed by the FinOps Foundation to implement cloud financial management (CFM). FinOps applies DevOps principles and practices to cloud financial operations. The FinOps Foundation defines FinOps (which it has trademarked) as “an evolving cloud financial management discipline and cultural practice that enables organizations to get maximum business value by helping engineering, finance, technology and business teams to collaborate on data-driven spending decisions.”

Why This Is Important

All organizations with meaningful spend on cloud IaaS or PaaS need to engage in cloud financial management (CFM) so that they are aware of what they are spending and why. Organizations must undertake greater cost governance efforts if they have substantial spending, significant self-service, cost complexity or variability. Further, CFM helps organizations optimize their costs and extract greater value from their spending. FinOps is one possible implementation model for CFM.

Business Impact

FinOps is a partial solution to CFM needs. Cloud economics spans the continuous process of CFM as well as one-off activities, and is focused on maximizing the value of cloud computing to the business, rather than minimizing cloud expenses. For example, business leaders may reasonably make the decision to spend more to deliver a better user experience, or to ignore cost-related technical debt so application teams can focus on delivering more features.

Drivers

- Public cloud costs are of concern to many customers. Many customers experience unexpectedly high cloud costs because they have fulfilled far greater business demand for cloud services than the IT organization had forecast.
- Ungoverned cloud adoption can lead to uncontrolled and careless spending. Organizations without an effective CFM practice cannot properly plan, track or optimize their cloud costs.
- Organizations that do not effectively manage cloud economics cannot assist the business in making thoughtful decisions about the top line versus the bottom line in cloud-enabled digital products and services.
- The FinOps Foundation (FOF), founded in 2019 by several cloud cost management tool vendors, created and popularized the use of the FinOps term. At that time, it defined FinOps as “the practice of bringing financial accountability to the variable spend model of cloud, enabling distributed teams to make business trade-offs between speed, cost and quality.”
- FOF changed the definition of FinOps in November 2021. The new definition is more closely aligned to Gartner’s recommended practices for the management of cloud economics. However, not all entities use the FinOps term in a consistent way.
- In mid-2020, FOF became a project of the Linux Foundation. Its membership has grown over time, and now includes many vendors that sell FinOps tools and services (which FOF will badge as FinOps Certified Platforms), numerous cloud providers, several global systems integrators and some enterprises (mostly financial services institutions). These members promote FinOps concepts to their users, increasing hype.
- FOF has consistently promoted the adoption of a centralized FinOps team, which in turn purchases and uses FinOps tools, leading customers to believe they should adopt this approach.
- FOF launched the FinOps Open Cost Usage Specification (FOCUS) in 2023. It is intended to be an open standard for cloud billing data, and will be broadly useful if widely adopted by cloud providers.

Obstacles

- Not all organizations have a business case for CFM, although almost all organizations that have meaningful cloud adoption need to perform cost management.
- CFM needs to be a cross-functional and cultural practice, not solely the responsibility of a dedicated FinOps team. In addition, many organizations cannot cost-justify having such a team.
- Application teams and the business owners of applications need motivation to optimize their cloud spend. This usually requires coupling showback to incentives (or penalties) or performing chargeback, forcing these entities to be accountable for what they spend.
- Although FinOps tools can often recommend optimizations for cloud infrastructure, they have limited capabilities for PaaS.
- Many organizations do not undertake activities that reduce their cloud spending because these activities do not have an adequate ROI. That is, the spend reduction is insufficient to justify the time and labor necessary to technically implement the optimizations.

User Recommendations

- Establish a cross-functional CFM practice — not just a FinOps team — once you have public cloud IaaS and PaaS adoption that has proceeded beyond the pilot stage.
- Your cloud center of excellence (CCOE) or other cloud governance function should own the CFM practice. The CCOE should set the policies and guidance, but cloud operations teams are typically responsible for implementing CFM tools and assisting application teams with optimizations.
- When custom-developing cloud solutions, design cost-aware architecture. Application design and implementation will be the primary influence on cloud costs.
- When migrating existing applications to the cloud through a lift-and-shift (that is, a rehost) approach, use performance-based rightsizing and accept the smallest recommended size by default. Most organizations tend to oversize virtual machines when they migrate, due to unwarranted concerns about performance and “headroom.”

Sample Vendors

Apptio; Flexera; VMware

Gartner Recommended Reading

Is FinOps the Answer to Cloud Cost Governance?

Managing Cloud Economics: A Cloud Architect's Guide to Productive Relationships With Sourcing Leaders

Effective Cloud Sourcing Strategies Focus on FinOps, Not Cost Reductions

Beyond FinOps: The Gartner Framework for Public Cloud Financial Management

Infrastructure Orchestration

Analysis By: Chris Saunderson

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Infrastructure orchestration (IO) enables platform and I&O teams to design, deliver, operate and ensure orchestrated services across on-premises, cloud and edge deployments. IO enables templated service creation and management, spanning provisioning, Day-2 operations and integration with CI/CD, self-service portals, and API access to orchestrated services.

Why This Is Important

Infrastructure orchestration provides strategic workflow capabilities to drive life cycle delivery and ongoing maintenance of complex deployed infrastructure. These practices and tools enable agile, iterative automation delivery and execution of the processes required via self-service and API access. This investment improves the velocity and quality of infrastructure services, improves traceability and visibility of service delivery and reduces inconsistencies from manual activities.

Business Impact

Infrastructure orchestration drives consumer experience improvements of deploying and managing standardized infrastructure. I&O teams realize operational efficiencies through reduced manual efforts to deliver infrastructure, embedding security and compliance requirements into the delivered services, and offering cost optimization opportunities. I&O staff can transform their role into an automation-first focus and scale to meet increased business demands.

Drivers

- **Business agility:** Organizations must increase responsiveness to meet customer needs and adapt to market and technology changes. They must be able to deliver products that meet these changing demands and requirements quickly.
- **Cost optimization:** Infrastructure teams leverage orchestration to deliver scalable, reliable and secure platforms. This helps to improve delivery efficiency, reduce human work, and reduce downtime due to change failures.
- **Value extraction:** adoption of orchestration capabilities unlocks additional value from the automation tools already implemented, enabling incident response, request servicing and other tasks to be more richly automated and consumed.
- **DevOps:** Infrastructure orchestration is a key enabler of continuous software delivery, allowing the DevOps team to automate the provisioning and management of environments.
- **Infrastructure complexity:** Increasingly complex deployment topologies require greater automation to improve the consumability of infrastructure and the ongoing maintenance of deployments,
- **Security and compliance:** Increased automation enables the implementation of security and compliance controls through orchestration and avoids any audit failures. The end-to-end visibility and traceability of the provisioning and configuration can enable continuous compliance automation of the infrastructure.

Obstacles

- **Skill development:** Infrastructure orchestration practices and tools can be complex to implement and sustain, as they require skills beyond scripting to get maximum value. These tools leverage software engineering skills that can be challenging to find in I&O teams.
- **I&O operating models:** The organizational structure of many I&O teams is set up by domain specializations, making it hard to develop and deliver end-to-end services through orchestration. Perceptions of stability and reliability risks slow adoption.
- **Automation constraints:** To automate maintenance activities, a certain level of maturity needs to be reached within the organization. Orchestration requires that automated tasks be available to be able to realize maximum return on investment.

User Recommendations

- Identify and catalog use cases and constraints in your delivery workflows that are injecting delay into service delivery, especially for tasks that are executed manually.
- Benchmark existing service delivery execution time and quality problems to measure against to demonstrate improvement.
- Catalog operational tasks that are being executed manually today and are candidates to develop workflows to implement.
- Identify candidate orchestration platforms to execute proof of value testing with, ensuring that the candidates can be integrated into your existing operational environment.
- Monitor implementation to identify successes and opportunities for improvement and build a success story demonstrating velocity, quality, throughput and operational improvements.

Sample Vendors

Cloudsoft; Crossplane; Dell Technologies; env0; Itential; Morpheus Data; PagerDuty; Pliant; RackN; SpaceLift

Gartner Recommended Reading

Innovation Insight for Continuous Infrastructure Automation

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

Market Guide for Infrastructure Automation Tools

Market Guide for Continuous Compliance Automation Tools in DevOps

Internal Developer Portal

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Internal developer portals serve as the interface that enables self-service discovery and access to resources in complex, cloud-native software development environments. They can include software catalogs, scorecards to benchmark software quality, scaffold templates, product documentation, plug-ins for extensibility and automation workflows. Developer portals help improve developer productivity, operational efficiency and enhance governance by providing shared visibility across multiple teams.

Why This Is Important

Internal developer portals help software developers navigate infrastructure complexity, understand service interdependencies and enable faster release cadence in at least three ways. First, they serve as a common viewpoint for multiple teams of developers. Second, they provide developers with self-service access to underlying platform components and environments. Third, they provide a centralized place to score applications and measure progress against reliability and security requirements.

Business Impact

Developer portals can have the following business impacts:

- Developer experience and productivity: Help development teams improve their delivery cadence by improving developer experience, reducing cognitive load and shortening the onboarding time.

- Reliability and resilience: Aim to provide visibility to application health and include scorecards to assess their production readiness.
- Security and governance: Include prebuilt toolkits, templates and curated libraries that help create “paved roads” with built-in compliance, security and audit policies.

Drivers

- Platform engineering: Organizations are adopting platform engineering principles and creating platform teams to scale cross-cutting capabilities across multiple development teams. Platform teams curate internal developer platforms to abstract away the complexity of siloed systems and processes. Internal developer portals serve as an interface through which developers can consume the capabilities of internal developer platforms.
- Backstage: Backstage is one of the first open-source frameworks for building developer portals. It was created at Spotify and is now a Cloud Native Computing Foundation (CNCF) incubating project. The thriving open-source community supporting Backstage has largely contributed to its enormous mind share and rapid adoption. Hundreds of organizations have adopted Backstage since it was open-sourced in 2020. Backstage’s success continues to drive interest, momentum and competition in this space.
- Developer experience: With software at the core of all digital innovation today, a great developer experience that accelerates software development becomes a key competitive advantage. Therefore, software engineering leaders are increasingly focused on minimizing developer friction and frustration. The ability to curate and provide customizable, developer-friendly experiences within the developer portal and reign in complexity will drive their appeal for both product and platform teams.
- Innersource: To enable rapid innovation and facilitate greater collaboration and knowledge sharing, software engineering leaders are adopting innersource approaches to software development. However, innersource requires an easy way for other teams to discover and search for existing projects within their organization. This is why organizations adopting innersource are turning to internal developer portals to make projects available and discoverable by other teams. See InnerSource Portal.

Obstacles

- Prerequisites: The successful adoption of internal developer portals goes beyond deploying a tool and requires certain prerequisites to be in place. For example, application services and their dependencies must be organized with consistently defined metadata that helps track their usage, performance and team ownership.
- Absence of platform teams: A dedicated platform team to manage and evolve the portal as a product is necessary to ensure the portal meets desired objectives. The absence of a dedicated platform team, and more so, led by a platform product owner to manage the portal as a product results in a disconnect between developer expectations and the portal's capabilities.
- Lack of developer buy-in: Although the developer portal serves as the "window" to the underlying platform's capabilities, it should provide "paved roads" and not "forced marches" — portal use should remain the choice of the development team. Trying to force development workflows into organizationwide blueprints for building developer portals without involving developers is a recipe for failure.

User Recommendations

- Use internal developer portals to scale cross-cutting software engineering capabilities across multiple development teams and streamline the software delivery life cycle.
- Do not assume that internal developer portals are turnkey solutions — they require a lot of prerequisites to be in place and many cases involve several weeks of prework activities. For example, Backstage requires codification of service-related metadata in YAML files before the content shows up in the software catalog.
- Ensure that the platform team includes internal developer portals in their charter. Continuously innovate portal capabilities by appointing a platform product owner for the developer portal to manage its roadmap, gather feedback and market its capabilities.

Sample Vendors

Atlassian; Calibo; CodeNOW; configure8; Cortex; Mia-Platform; OpsLevel; Port; Roadie

Gartner Recommended Reading

Innovation Insight for Internal Developer Portals

Drive Innovation by Enabling Innersource

Adopt Platform Engineering to Improve the Developer Experience

Cool Vendors in Platform Engineering for Improving Developer Experience

At the Peak

Chaos Engineering

Analysis By: Jim Scheibmeir, Hassan Ennaciri

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Chaos engineering is the use of experimental and potentially destructive failure testing or fault injection to uncover vulnerabilities and weaknesses within a distributed 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 organizations rely on test plans that overemphasize functionality and underemphasize validating the system's reliability and resilience. The distribution and complexity of systems makes understanding them more difficult. Chaos engineering (CE) shifts the focus of testing a system from the "happy path" toward testing how it can degrade gracefully or continue to be useful and secure while under various levels of impact. Applying CE enables improvements to system knowledge and documentation.

Business Impact

CE is aimed to minimize time to recovery and the change failure rate, while maximizing uptime and availability. Addressing these elements helps improve customer experience, satisfaction, retention and acquisition. Gartner inquiries regarding CE increased by over 11% between 2021 and 2022.

Drivers

- Increased complexity of systems and increasing customer expectations are the two largest drivers of CE and 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, CE helps organizations become more resilient across their processes, knowledge and technology.
- Teams often lack the confidence to handle failures and the psychological safety to take action to resolve incidents. CE can help build that confidence.

Obstacles

- Within many organizations, the predominant view of CE is that the practice is random, first implemented during production, and increases, rather than reducing, risk.
- Organizational culture and attitudes toward quality and testing can present barriers to the adoption of CE. When quality and testing are only viewed as overheads, there will be a focus on feature development over application reliability.
- It can be challenging just to secure the time and budget to invest in learning CE and associated technologies. Organizations must reach minimum levels of expertise so that value is returned.

User Recommendations

- Utilize a test-environment-first approach by practicing CE in preproduction environments.
- Incorporate CE 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 the advancement of the CE 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 CE in production to facilitate learning and improvement at scale as the practice matures. However, Gartner believes that very few organizations purposely use CE 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; ChaosIQ; Gremlin; Harness; Microsoft; Steadybit; Verica

Gartner Recommended Reading

Quick Answer: What Metrics Should We Use to Assess and Improve Software Quality?

Predicts 2023: Observing and Optimizing the Adaptive Organization

Top Strategic Technology Trends for 2023: Digital Immune System

Predicts 2023: How Innovation Will Transform the Software Engineering Life Cycle

Infrastructure Platform Engineering

Analysis By: Hassan Ennaciri, Paul Delory

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Infrastructure platform engineering is the discipline of building internal software products that present IT infrastructure to users or other platforms in an easily consumable way. Infrastructure platforms are self-service tools that allow nonexpert users to deploy and manage infrastructure themselves while I&O retains governance, security and compliance. Infrastructure platforms are often used as the foundation of higher-order, self-service layers such as internal developer platforms.

Why This Is Important

Digital enterprises are pressured to innovate and deliver products faster to meet customer needs. This requires adopting new operating models and modern practices to deliver scalable, reliable platforms that enable faster product delivery. Infrastructure platform engineering provides automated delivery of curated secure, reliable and scalable infrastructure services that can be available via self services or APIs and reduce the effort and cycle time for users to request and access the products.

Business Impact

Infrastructure platform engineering abstracts the complexity of the digital infrastructure to deliver platforms that continuously evolve to meet customer needs. It is an agile approach necessary to enable software products' value streams to meet customer needs and expectations. It also provides on-demand, fast access to environments, services and tools that improve customer experience and productivity.

Drivers

- **Business agility and innovation:** Digital businesses are required to be responsive to customers' needs and changing market conditions. They must have the ability to quickly deliver products that meet these changing demands and requirements.
- **Cost optimization:** Infrastructure platform engineering teams leverage automation to deliver scalable, reliable and secure platforms. This helps to improve efficiency, reduce resource cost due to manual work and reduce downtime due to change failures. Standardizing tools and platforms also optimizes resource utilizations and reduces cost incurred in tool proliferation.
- **Digital infrastructure and platform complexity:** Public cloud IaaS and PaaS deliver extensive capabilities and are designed to be consumable by developers, but most enterprises need additional governance and management that is best delivered by a platform engineering team.
- **Improve developer experience and productivity:** Infrastructure platform engineering abstracts complexity from developers and provides them with quick access or self-service in the environments they need to develop and test their software. Services can be made via an internal developer portal (IDP) such as Backstage, Calibo or Humanitec.
- **Compliance and security:** Infrastructure platform engineering automates and integrates compliance and security controls into software delivery pipelines, improving the organization's security posture and reducing the burden from developers.

Obstacles

- **Confusion:** There is a lot of hype and confusion about platform engineering and what it means. Many vendors are defining it to help sell their products, causing uncertainty with teams trying to adopt it.
- **Cultural:** This operating model is a new, modern approach that requires a shift in how teams work and collaborate, which is the hardest obstacle to overcome for many organizations.
- **Lack of skills:** Infrastructure platform engineering requires software engineering and specialized skills that may not exist in the organization.
- **Structure of traditional I&O operating models:** The organizational structure of many I&O teams is set up by domain specializations, making it hard to develop and deliver end-to-end services.
- **IT service management approaches:** The current approaches are process-heavy and rely on tickets and handoffs.
- **Complexity:** Successful implementation of infrastructure platform engineering is challenging because it requires new roles and involvement from many stakeholders.

User Recommendations

- **Start small and evolve:** Define initial goals and objectives of the platform by understanding common user needs and delivering viable products that continuously evolve to meet those needs.
- **Build a dedicated team with the right skills:** Successful infrastructure engineering practice requires dedicated teams with diverse skills in infrastructure platforms and software engineering.
- **Identify and fill critical roles such as platform owner and platform architect.** Acquire new talent with the required technical skills, the right mindset and strong interpersonal skills. Develop existing resources by provisioning continuous learning opportunities.
- **Adopt a product mindset:** Thread platform users as customers and ensure that you talk to them and continuously get their feedback to meet their existing needs as well as anticipate their future needs. Enable users and reduce the level of effort required to use the platform products.

Gartner Recommended Reading

Adopt Platform Engineering to Improve the Developer Experience

Top Strategic Technology Trends for 2023: Platform Engineering

Innovation Insight for Internal Developer Portals

Quick Answer: How Can I Optimize the Use of Programmable Platforms for Effective Software Delivery?

Guidance Framework for Implementing Cloud Platform Operations

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, reporting on 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 code and infrastructure. Traditional compliance practices are incompatible with continuous software delivery processes — leading to slower delivery and unexpected, expensive remediation work. CCA improves release velocity and reliability while simplifying compliance enforcement and reporting via policy-driven, automated controls.

Business Impact

Organizations' evolving DevOps/DevSecOps practices can minimize risks and penalties by embedding automated compliance and reporting 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. CCA tools can offer benchmarks, assessments and self-service reporting to enable efficiencies in compliance auditing.

Drivers

- As organizations face an increasing number of regulatory obligations and more stringent enforcement, 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.
- Compliance reporting, benchmarking and assessments are often manual and slow.

Obstacles

- 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.
- A lack of rule set understanding and consistent implementation can be an impediment to CCA. Failure to consistently involve organizational compliance teams in implementation leads to a failure in delivering 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.
- Implement a shift-left approach to ensure compliance controls are understood and applied 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, and remove manual reporting activities.
- Select tools that can integrate into DevOps delivery platforms to enable security and compliance checking.
- Enforce security and compliance across all domains, including databases, application code, infrastructure and open-source software. No single vendor tool covers all these domains, so DevOps teams must use multiple tools and integrate across all phases of the delivery pipeline.
- Enable efficient compliance policy checking through compliance automation tools to measure benchmarks, perform assessments and report on compliance policy controls.

Sample Vendors

Anitian; Contrast Security; JFrog; Mend.io; Rapid7; Redgate; RegScale; Snyk; Sonatype; Styra

Gartner Recommended Reading

Market Guide for Continuous Compliance Automation Tools in DevOps

3 Essential Steps to Enable Security in DevOps

How to Build and Evolve Your DevOps Toolchains

Market Guide for Value Stream Delivery Platforms

DevOps Platforms

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

DevOps platforms provide fully integrated capabilities to enable continuous delivery of software using agile and DevOps practices. These span the software development life cycle (SDLC) and include product planning, version control, continuous integration, test automation, continuous deployment, release orchestration, automating security and compliance policies, monitoring, and observability. DevOps platforms support team collaboration, secure software development and software delivery metrics.

Why This Is Important

Organizations use DevOps platforms to minimize tool friction and operational complexity resulting from disparate toolchains, manual handoffs, and lack of consistent visibility throughout the SDLC. This enables product teams to deliver faster customer value without compromising quality. The DevOps platforms market reflects the consolidation of technologies across development, security, infrastructure and operations to streamline software delivery.

Business Impact

DevOps platforms are the software delivery pipelines that enable continuous delivery of business value. The seamless integration, automation, extensibility and shared visibility between development, security and operations workflows help bridge the silos that exist between these teams. Using a common platform for development, security and operations accelerates agile transformation, and helps organizations move toward a product and platform team operating model.

Drivers

- **Modernizing application architectures:** Modernizing applications to take advantage of emerging cloud-native architectures requires fundamental changes to underlying DevOps practices and tools.
- **Increased emphasis/focus on enhancing developer experience:** Improved developer experience, agility and the need to improve delivery cadence by reducing cognitive load due to constant context switches and repetitive low-value work.
- **An integrated approach to security and compliance:** Integrating and automating security, compliance and governance as part of the development and delivery process is becoming a priority. A few DevOps platform providers include SCA capabilities as features in their offerings. Example vendors include GitHub, GitLab and JFrog.
- **Improved visibility into the flow of work:** Organizations are under pressure to reduce friction and manual handoffs, and this requires complete visibility into software delivery pipelines from ideation to production.

Obstacles

- Organizations that want to unlock the full benefits of DevOps platforms must be willing to replace an existing toolchain — either completely or in part. Teams can view the change as a disruption to their established ways of working and resist any change to the tools they have been using.
- Organizations accrue technical and skill debt over time due to outmoded automation workflows and legacy applications. This hinders teams from adopting new tools.
- Dependency on a single provider for a majority of their software development needs increases concentration risk and lowers bargaining leverage.
- Most DevOps platforms currently fall short in providing the full set of software delivery capabilities that organizations require to build, deliver, measure and improve the flow of value in the software delivery life cycle.

User Recommendations

- To fully reap the business benefits of DevOps platforms, organizations must adopt agile methods and practices.
- Scale and deliver capability by providing DevOps platforms as self-service platforms 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 DevOps platforms that provide enhanced visibility, traceability, auditability and observability across the DevOps pipeline.
- Support InnerSource efforts by building InnerSource portals using source control repositories available in DevOps platforms.
- Reduce inconsistency in CI/CD pipeline definitions between teams by leveraging declarative and shareable pipeline capabilities in DevOps platforms.

Sample Vendors

Atlassian; CircleCI; CloudBees; GitHub; GitLab; Harness; JetBrains; JFrog; Red Hat; VMware

Gartner Recommended Reading

Keys to DevOps Success

Research Roundup for DevOps, 2022

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 to developing software applications to ensure they meet the application performance objectives of the business. It focuses on the architecture, design and implementation choices that will affect application performance, and encompasses practices that help mitigate associated 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 exceed customers’ performance expectations. Organizations can improve application observability by using the insights gained through performance engineering. This includes adding metrics and telemetry, or deploying new application monitoring tools so alerts can be raised before a performance bottleneck impacts users.

Drivers

- Increased end-user expectations for application quality, specifically operational 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, as well as the 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

- Many organizations focus only on tools and technology. Performance engineering requires a change in organizational culture. Tools enable quality but won't solve problems on their own.
- Departmental silos, traditional top-down management structures and a lack of experience with managing quality continuously can impede adoption.
- Successful performance engineering requires clear goals that are aligned with the priorities of the business. The absence of established service-level indicators (SLIs) and objectives (SLOs) leads to a lack of realistic, quantifiable service availability or performance metrics.
- 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.
- 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 of nonfunctional or operational characteristics, such as performance efficiency or the Application Performance Index (Apdex), an open standard developed by an alliance of companies to measure the performance of software applications in computing. ISO/IEC 25010 provides a template for understanding quality characteristics and includes performance efficiency as one of the top-level nonfunctional domains.
- Establish SLIs and SLOs as part of a performance engineering strategy. An SLI is a realistic, quantifiable measure of service availability or performance. An SLO is the target for the SLI over a fixed period.
- 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 skills needed for performance engineering by identifying the required roles, technologies and practices.
- Establish 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 using performance information from production and real users.

Sample Vendors

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

Gartner Recommended Reading

How to Identify and Resolve Application Performance Problems

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

Improve Software Quality by Building Digital Immunity

Log Monitoring and Analysis

Analysis By: Pankaj Prasad, Gregg Siegfried

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Log monitoring and analysis solutions provide log ingestion, storage, interactive search, dashboards, alerts and advanced diagnostic analysis. Log-monitoring solutions apply advanced analytics or machine learning (ML) to reduce the operator's cognitive load through contextualization, correlation and analysis of large volumes of log data from multiple data sources.

Why This Is Important

Storing and searching log files is reactive and relies on human operators, which is not effective for IT architectures currently being deployed. The focus of traditional log monitoring is entity-centric anomaly detection and manual diagnosis. Automated log analytics uses statistical analysis, pattern recognition, correlations and machine learning to accelerate identification and resolution of service-impacting anomalies.

Business Impact

Log monitoring and analysis will impact:

- **Customer journeys** — Log analysis can help in behavior analysis and mapping user journeys, enabling better decisions.
- **Enhanced anomaly detection** — This enables monitoring of logs for patterns, which provides better insights into potential anomalies over static thresholds.
- **Operational efficiency** — Logical and temporal correlations of log data across multiple data sources enables better diagnosis and faster root-cause analysis.

Drivers

- Data explosion with modern architectures — As organizations adopt geographically distributed, container- and microservice-based architectures, the amount of log data generated — already in the petabyte per day range at some organizations — will start approaching exabyte levels. At scale, collection, storage and gaining insights from logs is not feasible without automated analysis.
- Operational challenges — I&O organizations need to improve mean time to repair (MTTR) by reducing manual correlation effort time.
- Root cause analysis — I&O and site reliability engineering (SRE) teams looking for ways and means to enhance the resilience of their IT architectures need a faster way to identify correlating patterns and relevant logs beyond the system of interest.

Obstacles

- Implementation — ML models demand heavy investment for either supervised or unsupervised training to be accurate over time, thus increasing the time to value for these solutions. The noncentralized nature of log data in many organizations further hampers the quality of outcomes.
- Basic use cases — These products do not enable ease of higher-order use cases beyond I&O due to the lack of an advanced interface that can be used by a data-scientist-type persona.
- Cost — Whether a log-monitoring solution is deployed on cloud or on-premises (as either an open-source or proprietary option), the licensing or maintenance costs increase with the volume of log data and data-retention policies. Balancing total cost of ownership (TCO) for log monitoring solutions and preserving historical information for analysis is a prevalent problem for I&O organizations.

User Recommendations

- Simplify log analysis by establishing a log governance model that standardizes field names and data formats throughout multiple sources of log data. This facilitates the pooling of log data from multiple sources and results in more efficient queries.
- Evaluate advanced analysis and out-of-the-box capabilities and use cases during product assessment for log monitoring and analysis.
- Choose a solution that supports multiple sources of logs and broad use cases, and deploy them centrally. Log collection and analysis should not be deployed in silos.

Gartner Recommended Reading

Guidance Framework for Deploying Centralized Log Monitoring

Market Guide for AIOps Platforms

Infographic: AIOps Architecture for Analyzing Operational Telemetry

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 now 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 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.
- Costs associated with observability tools have grown as companies struggle to keep up with the explosion in volume and velocity of telemetry.

User Recommendations

- 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; Honeycomb; Lightstep; Observe; VMware

Gartner Recommended Reading

Monitoring and Observability for Modern Infrastructure and Applications

Magic Quadrant for Application Performance Monitoring and Observability

Sliding into the Trough

SLIs/SLOs

Analysis By: Hassan Ennaciri, Daniel Betts

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Service-level objectives (SLOs) provide visibility to product owners and business units about the reliability and performance of product features. SLOs are the objectives that must be achieved for any service level to meet user sentiments. Those objectives are measured by service-level indicators (SLIs) over a range of time. SLIs are quantitative measures of a specific aspect of the level of service being provided, such as availability, latency or error rate.

Why This Is Important

Customers have very high expectations when using digital products. They continuously demand more features in products and expect availability, performance and reliability.Traditional monitoring approaches fail to meet user sentiments when using digital services, because they mainly focus on availability and performance of applications. However, SLIs and SLOs focus on features customers care about and on their feelings when using the products. This helps optimize the operations of the services.

Business Impact

Using SLIs and SLOs is an important site reliability engineering (SRE) practice. The primary benefit is to provide product and operations teams with metrics that focus on customer experience. This creates a common understanding about what matters to the customer or product owner. Working with SLIs and SLOs also helps teams balance the need to release features and operational improvements to provide the best possible experience for customers.

Drivers

- **Customer expectations** — Customers have very high expectations when using digital products. SLOs measure which availability and performance levels meet their expectations. This allows the SRE teams to monitor these levels and proactively identify and correct them to keep customers happy.
- **Reliability** — The main reason SRE teams use SLIs or SLOs is to ensure optimal operation of systems. These acceptable SLO levels become key incentives for product and operations teams to design and operate systems that are reliable and resilient.
- **Continuous improvements** — The primary goal of using SLIs and SLOs is to have all stakeholders embrace a culture of continuous improvements. Stakeholders can reach that goal by agreeing on target levels that meet the needs of their customers.
- **Cost optimization** — By designing and operating systems in a way that is good enough to meet customer expectations, SRE teams can optimize cost by achieving operational efficiency.
- **Compliance requirements** — SLOs can be used to ensure that products meet business continuity requirements and are available during failure or unexpected events.

Obstacles

- **Resistance to change** — Many business and IT professionals don't understand what SLIs and SLOs are and how they work. Consequently, they may resist having to change and adopt the new practices.
- **Skills required** — Many teams are not familiar with how to implement SLIs and SLOs or don't have the skills to implement them properly. They require a mature SREs practice that not many organizations have.
- **Complexity of setting and managing SLIs and SLOs** — It is very difficult to initially implement SLIs and SLOs as it requires participation of all the teams involved in software product delivery. Managing SLIs and SLOs also requires investments in tools and additional resources that many organizations don't have.
- **Lack of telemetry** — Setting SLIs and SLOs is very difficult to do without having enough historical data. This is especially hard in complex systems with interdependencies.

User Recommendations

- **Start with teams adopting SRE practices** — Start by implementing SLIs and SLOs with product teams that have already embraced SRE practices. Start small with a few services, continuously iterate and improve to demonstrate value and grow adoption.
- **Educate all stakeholders** — A key to success is to educate everyone on the concept of SLIs and SLOs and how they can be used to ensure a great user experience and high customer satisfaction levels. More importantly, SLIs and SLOs can be used to continuously improve reliability and optimize operations.
- **Manage error budgets along with SLOs** — Just like service-level objectives, error budgets need to be continuously evaluated and adjusted.
- **Invest in tools and telemetry** — Invest in tools that enable SLO and error budget management, and also in tools that offer telemetry and provide insight into error rates.
- **Focus on cost optimization** — Use SLIs and SLOs to optimize the design and operations of services so that they meet customer expectations.

Sample Vendors

Atlassian; Blameless; Datadog; Dynatrace; New Relic; Nobl9; OpsRamp; PagerDuty; Splunk

Gartner Recommended Reading

7 Steps to Start and Evolve an SRE Practice

Improve Software Quality by Building Digital Immunity

Improve Product Reliability by Applying SRE Principles to Service Operations

Assessing Site Reliability Engineering (SRE) Principles for Building a Reliability-Focused Culture

AIOps Platforms

Analysis By: Matt Crossley, Matthew Brisse

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Gartner defines AIOps platform as the application of AI/ML and data analytics at the event management level in order to augment, accelerate and automate manual efforts in the event management process and associated procedures. AIOps platforms are defined by the key characteristics of cross-domain event ingestion, topology assembly, event correlation and reduction, pattern recognition, and remediation augmentation.

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 (ML) and data analytics to classify and cluster cross-domain events 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 of events 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.

Obstacles

- **Unrealistic expectations:** Hype is a major obstacle to AIOps platform adoption. Clients struggle to separate claims of AI and magical automation from achievable use cases. This impacts demonstrating value of AIOps platforms, specifically quantifiable return on investment.
- **Maturity of dependencies:** Benefits of AIOps platforms beyond event correlation requires maturity in dependencies such as automation.
- **Time to value:** AIOps platforms learn through observation, modeling normal data 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:** Monitoring vendors are moving up the stack, AIOps platform vendors are reaching into monitoring domains, and ITSM vendors use AIOps capabilities to extend their reach. Expect further convergence 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 technical and process dependencies, and separating hype from reality.
- Layer the 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; BMC Software; Digitate; IBM; Interlink; Moogsoft; OpsRamp; PagerDuty; ServiceNow; Splunk

Gartner Recommended Reading

Market Guide for AIOps Platforms

Deliver Value to Succeed in Implementing AIOps Platforms

Infographic: Artificial Intelligence Use-Case Prism for AIOps

Infographic: AIOps Architecture for Analyzing Operational Telemetry

How Do I Plan for Migrating My Data Center Infrastructure Into an XaaS Model?

Infrastructure Automation

Analysis By: Chris Saunderson

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Infrastructure automation (IA) enables DevOps and infrastructure and operations (I&O) teams to deliver automated infrastructure services across on-premises and cloud environments. This includes the life cycle of services through creation, configuration, operation and retirement. These infrastructure services are then made available through platform delivery, self-service catalogs, direct invocation and API integrations.

Why This Is Important

IA delivers velocity, quality, efficiency and reliability, with scalable, declarative approaches for deploying and managing infrastructure. These tools integrate into delivery pipelines targeting deployment topologies that range from on-premises to the cloud, and enable infrastructure consumers to build what is needed when they need it. Once deployed, IA provides day-2 and beyond operational automation, and extends to provide policy compliance and enforcement capabilities.

Business Impact

Implementing and maturing IA services will enable:

- **Agility** — continuous infrastructure delivery and operations
- **Productivity** — version-controlled, declarative, repeatable, efficient deployments
- **Cost improvement** — reductions in manual effort expended via increased automation
- **Risk mitigation** — compliance driven by standardized configurations
- **Collaboration** — delivering environments that product teams need with security, cost and compliance requirements baked in.

Drivers

I&O leaders must automate delivery through tool and skills investments to mature beyond simple deployments. The target should be standardized platforms that 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 automated incident response) can be achieved. IA tools deliver the following key capabilities to support this maturation:

- Multicloud/hybrid cloud infrastructure delivery
- Support for immutable and programmable infrastructures
- Predictable delivery enabling automated operations
- Self-service and on-demand environment creation
- Integration into DevOps initiatives (continuous integration/delivery/deployment)
- Resource provisioning, including cost optimization capabilities
- Operational configuration management efficiencies
- Policy-based delivery and assessment/enforcement of deployments against internal and external policy requirements
- Enterprise-level framework to enable maturing of automation strategies
- Skills and practice development inside infrastructure teams, enabling agile and iterative development and sustaining of services

Obstacles

- The combination of tools needed to deliver IA capability can increase tool count and complexity.
- Software engineering skills and practices are required to get maximum value from tool investments.
- IA vendor capability expansion overlaps and confuses the tool landscape, resulting in over-investment.
- 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, identify use cases and document overlaps to aid decision making.
- Assess existing internal IT skills to incorporate training needs that more fully enable IA, especially for an automation architect role to coordinate standards development and implementation.
- 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 scope for automation and delivery activities.
- Develop an IA tooling strategy that incorporates current needs and near-term roadmap evolution.

Sample Vendors

Amazon Web Services; HashiCorp; Microsoft; Perforce; Pliant; Progress; Pulumi; RackN; Upbound; VMware

Gartner Recommended Reading

Market Guide for Infrastructure Automation Tools

Innovation Insight for Continuous Infrastructure Automation

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

How to Start and Scale Your Platform Engineering Team

Climbing the Slope

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 build critical software quickly, while ensuring the software can be released reliably anytime. Through dependable, low-risk releases, CD allows continuous adaptation of the software to incorporate user feedback, market shifts and business strategy changes. This approach requires the engineering discipline to facilitate complete automation of the software delivery pipeline.

Why This Is Important

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

Business Impact

CD is a key practice for a DevOps initiative as it reduces the build-to-production cycle time. As a result, it 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

- Increased adoption of Agile and DevOps practices to deliver solutions.
- Pressure from digital business to improve release velocity and reliability.
- Additional compliance requirements that require automation and orchestration of release activities for better traceability and auditability.

- The need to improve delivery outcomes to deploy application builds and updates more consistently, by extending the benefits of continuous integration (CI) and automated testing to continuously build deployable software.

Obstacles

- Organizational culture and collaboration between teams with different roles and skills are major barriers to CD success. Agile practices that helped bridge the gap between business and development must be extended to deployment, environment configuration, monitoring, and support activities.
- Lack of value stream mapping of product delivery hinders visibility and quick feedback loops 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 impact software flow delivery.
- Other challenges impacting the success of CD include application architecture, 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 different stakeholders from the product, development, security and operations teams.
- 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 a DevOps platform that provides fully integrated capabilities and enables continuous delivery of software.

Sample Vendors

Broadcom; CloudBees; GitLab; Harness; JFrog; Red Hat

Gartner Recommended Reading

How to Build and Evolve Your DevOps Toolchains

Market Guide for Value Stream Management Platforms

Beware the DevOps Toolchain Debt Collector

Automated Incident Response

Analysis By: Pankaj Prasad, Padraig Byrne

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Automated incident response (AIR) centralizes alert or incident routing through a policy or rule-based engine, on-call scheduler and streamlined collaboration. AIR solution capabilities improve operational efficiencies with action-oriented insights, shorter incident durations and automated workflows for event routing, easier collaboration, remediation and escalations.

Why This Is Important

Manual processes for incident resolution is a challenge, especially when multiple experts need to be involved, time is of essence and the organization wants to improve efficiency. For DevOps teams, the juggling of contact lists and lack of seamless collaboration inhibit speedy delivery of application features, as well as the stability of features after the release. AIR solutions solve this by automating most of the incident response process and collaboration, and enabling iterative improvement.

Business Impact

AIR solutions deliver value through:

- Automated incident communication to the relevant recipient and visibility across the organization.

- Quick incident resolution minimizing customer impact.
- A well-integrated incident management practice that meets DevOps requirements.
- Insights into incidents and their responses, which helps improve process and operational efficiency.
- Automated workflows that eliminate fatigue and human errors and reduce the turnaround time.

Drivers

- **Incident communication and visibility challenges:** With geographically distributed teams, remote workforce, complex on-call schedules and notification channel preferences, incident triage teams often have difficulty engaging responders quickly. Incident communication itself may lack all the relevant inputs or rely on multiple sources of incident data.
- **Automation of incident response processes:** AIR reduces mean time to acknowledge (MTTA) by automating the process of identifying and contacting the relevant domain experts, and speeds up the resolution process.
- **DevOps and site reliability engineer (SRE) requirements:** Traditional incident management models cannot meet the needs of agile cultures because of manual tasks in the incident response workflow. AIR caters to the need for seamless collaboration across various groups enabling DevOps to underpin its offerings with an effective, consistent IT service management (ITSM) practice.
- **Transparent review and analysis:** AIR tools capture an incident's progress from identification through resolution, including the handoffs needed across various teams. This includes the time and action taken at each step of the incident, and provides vital information for postincident review (PIR) and process review for further enhancements.
- **Workflow automation:** These tools can automate workflows that are part of processes like creating incidents for actionable alerts, opening a communications channel in instant messengers for collaboration, updating on a web-portal and one-click remediation for existing runbooks.

Obstacles

- **Overlapping capabilities:** Although AIR solutions offer differentiating features, they also overlap with ITSM and event management systems, making it difficult to articulate the value of investing in AIR.
- **Service definitions:** Service definitions that connect alerts to responder teams are often challenging to configure as it involves interpretation of a problem based on the notification to identify the domain experts that need to be engaged. Service definitions are also a complex part of AIR onboarding.
- **Portability between solutions:** Migrating from one AIR vendor to another is a reset process, with no defined migration path. The integrations, team and service definitions, responder preferences, and role-based access controls must be reconfigured without sophisticated import/export mechanisms.
- **Maturity in I&O:** Few organizations have the required I&O maturity to quantify impact due to time lost in contacting the right personnel for resolving an issue to justify investing in these tools.

User Recommendations

- Invest in a centralized AIR solution for automating incident management workflows and on-call capability for major incidents and critical events with wide integrations for holistic incident response management.
- Integrate monitoring solutions and service desk systems with bidirectional synchronization to incident response systems, which keeps the incident status synchronized across systems.
- Leverage automation for remediation and to extend incident response capabilities that can integrate with DevOps toolchains.
- Improve incident communication and collaboration by integrating incident workflow processes with ChatOps tools, such as Slack or Microsoft Teams.

Sample Vendors

AlertOps; Atlassian; Derdack; Everbridge; OnPage; PagerDuty; ServiceNow; Splunk

Entering the Plateau

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 definition, deployment and maintenance of configuration states and settings. Most CCA tools have an open-source heritage, and most offer commercial support.

Why This Is Important

CCA tools are critical to delivering operational efficiency. They enable automation and DevOps initiatives by deploying and managing infrastructure elements, associated software and configuration changes as code. In combination with infrastructure automation tools, CCA tools form the core of infrastructure-as-code (IaC) capabilities. They also enable the automation of Day 2 operations of deployed systems, expanding their reach into networking, containers, compliance and security use cases.

Business Impact

By enabling automated deployment and configuration of systems, settings and software programmatically, organizations realize:

- **Agility improvements:** CI/CD enablement for infrastructure and operations (I&O) services.
- **Productivity gains:** Repeatable, declarative, version-controlled infrastructure deployment/operation.
- **Cost optimization:** Reductions in manual interventions by skilled staff.
- **Risk mitigation:** Compliance assessment and remediation using standardized processes Improved change success and reliability are also realized.

Drivers

- Organizations need a broader set of deployment and automation functions beyond configuration management, including IaC, patching, application release orchestration (ARO), configuration assessment and 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 essential 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, declarative, 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 and overlaps of automation tools causing conflict and overinvestment.
- Developers and administrators may use CCA in a silo, further inhibiting enterprisewide adoption.
- IT skill sets hinder the 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 imperative scripting methods to complete specific tasks in place of using CCA capabilities.
- Maximize the value of CCA tool investments by ensuring that your organization's culture can embrace CCA tools strategically and automate toil (for DevOps and I&O leaders).

Sample Vendors

Inedo; Perforce (Puppet); Progress; Red Hat; Rudder; VMware

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

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 an application's behavior, dependencies, users and business KPIs throughout its life cycle. The application being observed may be developed internally, as a packaged application 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. Modern APM implementations are becoming more tightly integrated with observability platforms.

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 increases for agility in order to succeed with their transformation initiatives. 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 and observability tools are becoming more unified. This approach requires platforms that share common data models to conduct correlation analysis and other critical functions of application performance monitoring.
- **Holistic monitoring:** Modern tools 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.
- **Integration with DevOps and site reliability engineering (SRE):** Testing in preproduction and integration with continuous integration/continuous delivery (CI/CD) tools have become the new 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:** APM tools can be used to derive business metrics, such as abandoned shopping carts or average spend per customer for retailers. Representing such critical business information means an increased likelihood of further investment in these tools.

Obstacles

- Traditional implementations of APM often fail 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.
- Modern architectures such as containers and microservices and cloud-native environments are coming to IT operations environments faster than monitoring strategies are evolving to handle them. This is leading to visibility gaps and performance challenges.
- Clients increasingly cite cost as a significant challenge for implementation of APM tools. Costs for larger organizations can run into millions per year, representing a significant percentage of IT spend.
- Many IT monitoring teams still rely on manually invoked runbooks or scripts to remediate problems, hindering I&O leaders' ability to deploy and monitor new technologies. There is often 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 vendors that assist in relating application performance to business objectives and serve not only IT operations (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 products 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; Elastic; Grafana; Instana; New Relic; Splunk

Gartner Recommended Reading

Magic Quadrant for Application Performance Monitoring and Observability

DevSecOps

Analysis By: Neil MacDonald, Mark Horvath

Benefit Rating: Transformational

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

DevSecOps is the integration and automation of security and compliance testing into agile IT and DevOps development pipelines, as seamlessly and transparently as possible, without reducing the agility or speed of developers or requiring them to leave their development toolchain. Ideally, offerings provide security visibility and protection at runtime as well.

Why This Is Important

DevSecOps offers a means of effectively integrating security into the development process, in a way that eliminates or reduces friction between security and development. The goal is to pragmatically achieve a secure, workable software development life cycle (SDLC) supporting rapid development. DevSecOps has become a mainstream development practice, although the specifics can vary between organizations based on their technology and the maturity of their development processes.

Business Impact

The goal of DevSecOps is to speed up development without compromising on security and compliance. Furthermore, the externalization of security policy enables business units and security organizations to define and prioritize policy guardrails and lets developers focus on application functionalities. Policy-driven automation of security infrastructure improves compliance, the quality of security enforcement and developer efficiency, as well as overall IT effectiveness.

Drivers

- Adoption of DevOps, and other rapid development practices, requires security and compliance testing that can keep up with the rapid pace of development.
- DevSecOps offerings are applied as early as possible in the development process, whereas traditional application security testing (AST) tools associated with older development models are applied late in the development cycle, frustrating developers and business stakeholders.
- Testing results need to be integrated into the development process in ways that complement developers' existing workflows and toolsets, and not require them to learn skills unrelated to their goals.
- The use of open source has greatly increased the risk of the inadvertent use of known vulnerable components and frameworks by developers.

Obstacles

- Incorrectly implemented, siloed and cumbersome security testing is the antithesis of DevOps. Due to this, developers believe security testing tools are slowing them down.
- Developers don't understand the vulnerabilities their coding introduces.
- Developers don't want to leave their development (continuous integration/continuous delivery [CI/CD]) pipeline to perform tests or to view the results of security and compliance testing tools.
- Historically, static application security testing (SAST) and dynamic application security testing (DAST) tools have been plagued with false positives or vague information, hence frustrating developers.
- The diversity of developer tools used in a modern CI/CD pipeline will complicate the seamless integration of DevSecOps offerings.

User Recommendations

- "Shift left" and make security testing tools and processes available earlier in the development process.
- Prioritize the identification of open-source software (OSS) components and vulnerabilities in development (referred to as software composition analysis).
- Opt for automated tools with fast turnaround times, with a goal of reducing false positives and focusing developers on the highest-confidence and most-critical vulnerabilities first.
- Ask vendors to support out-of-the-box integration with common development tools and support full API enablement of their offerings for automation.
- Evaluate emerging cloud native application protection platform (CNAPP) offerings for technical control implementation.
- Require security controls to understand and apply security policies in container- and Kubernetes-based environments.
- Favor offerings that can link scanning in development to correct configuration, visibility and protection at runtime.

Sample Vendors

Apiiro; Aqua Security; Contrast Security; Dazz; Lacework; Palo Alto Networks; Qwiet AI; Snyk; Sonatype; Wiz

Gartner Recommended Reading

How to Select DevSecOps Tools for Secure Software Delivery

Market Guide for Cloud-Native Application Protection Platforms

Magic Quadrant for Application Security Testing

12 Things to Get Right for Successful DevSecOps

How to Manage Open-Source Software Risks Using Software Composition Analysis

Appendixes

Hype Cycle Phases, Benefit Ratings and Maturity Levels

Table 2: Hype Cycle Phases

(Enlarged table in Appendix)

Phase	Definition
Innovation Trigger	A breakthrough, public demonstration, product launch or other event generates significant media and industry interest.
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 2023)

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
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 (for example, improved user experience) that will be difficult to translate into increased revenue or cost savings

Source: Gartner (July 2023)

Table 4: Maturity Levels

(Enlarged table in Appendix)

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
Mature mainstream	Robust technology Not much evolution in vendors or technology	Several dominant vendors
Legacy	Not appropriate for new developments Cost of migration constrains replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2023)

Recommended by the Authors

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

How Software Engineering Teams Should Work With Site Reliability Engineers

Assessing Site Reliability Engineering (SRE) Principles for Building a Reliability-Focused Culture

7 Steps to Start and Evolve an SRE Practice

9 Principles for Improving Cloud Resilience

Table 1: Priority Matrix for Site Reliability Engineering, 2023

Benefit ↓	*Years to Mainstream Adoption*			
	Less Than 2 Years ↓	***2 - 5 Years*** ↓	***5 - 10 Years*** ↓	***More Than 10 Years*** ↓
Transformational	DevSecOps	Observability	Augmented FinOps Continuous Resilience Automation Error Budgets SLO Management	
High	APM Automated Incident Response Continuous Configuration Automation	Autonomous Workload Optimization Continuous Delivery DevOps Platforms FinOps Infrastructure Automation Infrastructure Orchestration Internal Developer Portal Log Monitoring and Analysis Performance Engineering Policy as Code SLIs/SLOs	AIOps Platforms Digital Immune System Infrastructure Platform Engineering Monitoring as Code Observability-Driven Development OpenSLO	
Moderate		Continuous Compliance Automation	Chaos Engineering	

Benefit ↓	Years to Mainstream Adoption			
	Less Than 2 Years ↓	2 - 5 Years ↓	5 - 10 Years ↓	More Than 10 Years ↓
Low				

Source: Gartner (July 2023)

Table 2: Hype Cycle Phases

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

Phase ↓	Definition ↓

Source: Gartner (July 2023)

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
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 (for example, improved user experience) that will be difficult to translate into increased revenue or cost savings

Source: Gartner (July 2023)

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
Mature mainstream	Robust technology Not much evolution in vendors or technology	Several dominant vendors
Legacy	Not appropriate for new developments Cost of migration constrains replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2023)