Hype Cycle for Container Technology, 2023

Published 13 July 2023 - ID G00791913 - 107 min read

By Analyst(s): Dennis Smith

Container technologies supply the speed and agility required to enable digital business strategies. Infrastructure and operations leaders should leverage this Hype Cycle to achieve innovation, value, agility and efficiency.

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 Assumption

By 2027, more than 90% of Global 2000 (G2000) organizations running containerized applications in hybrid deployments will be leveraging container management tooling, which is a significant increase from fewer than 20% in 2023.

Analysis

What You Need to Know

Container technology is fundamental to the ability of infrastructure and operations (I&O) organizations to build agile infrastructure and application platforms pipelines, while meeting the increasing demands of digital business. This is particularly critical to achieving the needs of the software engineering organization as it delivers cloud-native architectures. Enabling speed to market and increased agility remain the core business drivers; container technology investments can help meet these objectives. Container technology affects the roles, skills and expectations of IT staff and leaders, and offers a means to improve developer engagement, productivity and experience. This Hype Cycle provides insight into technologies that are pivotal to successful container adoption.

I&O leaders responsible for container technologies must:

- Continually engage with the application development community, while jointly developing a container strategy and ensuring that it is consistent with the enterprise's cloud strategy.
- Increase automation maturity, because it is a key building block for all successful container deployments.
- Incorporate security in all aspects of container adoption planning.
- Consider monitoring a key requirement for container deployment.
- Weigh the degree of serverless requirements when developing a container strategy, because serverless technologies can complement, as well as compete with aspects of containerization.

The Hype Cycle

Container technologies empower digital initiatives to deliver value. I&O leaders should view container technologies as enablers of the overarching trends of DevOps, cloud computing and engineering-driven approaches to operations.

The four forces primarily responsible for driving the growth of container technologies are:

- Adoption of DevOps practices
- Cloud-native application deployments

- Emergence of platform engineering principles
- Cloud-native infrastructure adoption

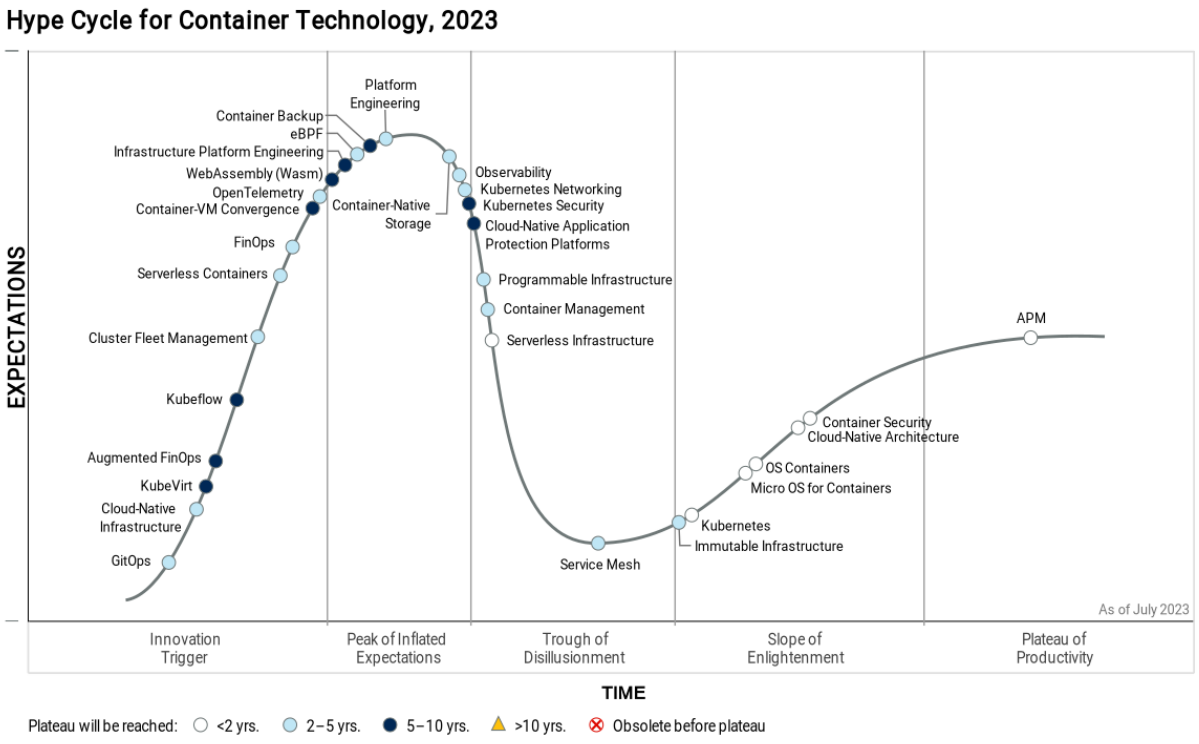
Container technology remains important to deliver agility and efficiency in support of DevOps adoption. Vendors and the open-source community have provided a wide range of technologies to enable delivery pipelines, broader sets of integrations, and better alignment between infrastructure and applications. Container technology adoption continues to expand to support digital initiatives, increasing the value of container technologies to business-centric domains.

Cloud computing characteristics continue to influence how I&O manages technologies that support product delivery. This includes the continued adoption of immutable, composable and automated approaches to managing infrastructure in support of on-premises, colocation, edge and cloud environments. In addition, the growing need for container technology skills in I&O to support these initiatives has forced I&O leaders to develop or acquire these skills in a competitive market, often with mixed results.

Platform engineering principles continue to gain traction among Gartner clients, supporting and scaling an agile approach to software delivery. Container platform adoption is, for many I&O teams, the first foray into platform engineering principles.

As organizations assess how to handle their critical legacy applications, they are also considering their infrastructure strategies. Container technologies can be critical elements of these strategies to support modern application hosting and development.

Figure 1: Hype Cycle for Container Technology, 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. How much value will an organization receive from an innovation?

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

During the next two to five years, container adoption will continually grow. This adoption has already expanded beyond the public cloud realm to on-premises and the edge. As a result, it will require the continued evolution and the enhancement of container management tooling to deploy and support the operation of containers across these environments.

The overlap of traditional container technologies and serverless solutions that abstract much of the underlying container technology will continue. This requires vetting and providing the correct technology (e.g., containers/Kubernetes versus a more abstracted serverless offering) for the right end-user persona.

An overlap of container and virtualization technologies already exists, and it will be particularly impactful in edge computing deployments, where a smaller footprint is available to provide multiple functionalities.

Container security and monitoring has been an ongoing challenge for years. Technologies and approaches in this area, such as OpenTelemetry (OTel), will improve during the next two to five years.

Enterprises will continually evaluate and adopt technologies to modernize their infrastructures. In some cases, this includes looking at container-related technology to replace existing hypervisor deployments (e.g., KubeVirt). However, much of this technology still does not provide the capabilities that enterprises have grown accustomed to with their longstanding virtualization environments. Despite this, container technologies will continue to be a part of these considerations.

Table 1: Priority Matrix for Container Technology, 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	OS Containers Serverless Infrastructure	Observability Platform Engineering	Augmented FinOps Kubeflow WebAssembly (Wasm)	
High	APM Kubernetes	Cluster Fleet Management Container Management Container-Native Storage FinOps GitOps OpenTelemetry Programmable Infrastructure	Cloud-Native Application Protection Platforms Container-VM Convergence Infrastructure Platform Engineering	
Moderate	Cloud-Native Architecture Container Security Micro OS for Containers	Cloud-Native Infrastructure eBPF Immutable Infrastructure Kubernetes Networking Serverless Containers	Container Backup Kubernetes Security	
Low		Service Mesh	KubeVirt	

Source: Gartner (July 2023)

On the Rise

GitOps

Analysis By: Paul Delory, Arun Chandrasekaran

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

GitOps is a type of closed-loop control system for cloud-native applications. The term is often used more expansively, usually as a shorthand for automated operations or CI/CD, but this is incorrect. According to the canonical OpenGitOps standard, the state of any system managed by GitOps must be: (1) expressed declaratively, (2) versioned and immutable, (3) pulled automatically, and (4) continuously reconciled. These ideas are not new, but new tools and practices now bring GitOps within reach.

Why This Is Important

GitOps can be transformative. GitOps workflows deploy a verified and traceable configuration (such as a container definition) into a runtime environment, bringing code to production with only a Git pull request. All changes flow through Git, where they are version-controlled, immutable and auditable. Developers interact only with Git, using abstract, declarative logic. GitOps extends a common control plane across Kubernetes (K8s) clusters, which is increasingly important as clusters proliferate.

Business Impact

By operationalizing infrastructure as code, GitOps enhances availability and resilience of services:

- GitOps can be used to improve version control, automation, consistency, collaboration and compliance.
- Artifacts are reusable and can be modularized.
- Configuration of clusters or systems can be updated dynamically. All of this translates to business agility and a faster time to market.

- GitOps artifacts are version-controlled and stored in a central repository, making them easy to verify and audit.

Drivers

- **Kubernetes adoption and maturity:** GitOps must be underpinned by an ecosystem of technologies, including tools for automation, infrastructure as code, continuous integration/continuous deployment (CI/CD), observability and compliance. Kubernetes has emerged as a common substrate for cloud-native applications. This provides a ready-made foundation for GitOps. As Kubernetes adoption grows within the enterprise, so can GitOps, too.
- **Need for increased speed and agility:** Speed and agility of software delivery are critical metrics that CIOs care about. As a result, IT organizations are pursuing better collaboration between infrastructure and operations (I&O) and development teams to drive shorter development cycles, faster delivery and increased deployment frequency. This will enable organizations to respond immediately to market changes, handle workload failures better, and tap into new market opportunities. GitOps is the latest way to drive this type of cross-team collaboration.
- **Need for increased reliability:** Speed without reliability is useless. The key to increased software quality is effective governance, accountability, collaboration and automation. GitOps can enable this through transparent processes and common workflows across development and I&O teams. Automated change management helps to avoid costly human errors that can result in poor software quality and downtime.
- **Talent retention:** Organizations adopting GitOps have an opportunity to upskill existing staff for more automation- and code-oriented I&O roles. This opens up opportunities for staff to learn new skills and technologies, resulting in higher employee satisfaction and retention.
- **Cultural change:** By breaking down organizational silos, development and operations leaders can build cross-functional knowledge and collaboration skills across their teams to enable them to work effectively across boundaries.
- **Cost reduction:** Automation of infrastructure eliminates manual tasks and rework, improving productivity and reducing downtimes, both of which can contribute to cost reduction.

Obstacles

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

User Recommendations

- **Target cloud-native workloads initially:** Your first use case for GitOps should be operating a containerized, cloud-native application that is already using both Kubernetes and a continuous delivery platform such as Flux or ArgoCD.
- **Build an internal operating platform:** This is the foundation of your GitOps efforts. Your platform should manage the underlying infrastructure and deployment pipelines, while enforcing security and policy compliance.
- **Embed security into GitOps workflows:** Security teams need to shift left, so the organization can build holistic CI/CD pipelines that deliver software and configure infrastructure, with security embedded in every layer.
- **Be wary of vendors trying to sell you GitOps:** GitOps isn't a product you can buy, but a workflow and a mindset shift that becomes part of your overall DevOps culture. Tools that expressly enable GitOps can be helpful; but GitOps can be done with nothing more than standard continuous delivery tools that support Git-based automation.

Sample Vendors

GitLab; Harness; Red Hat; Upbound; Weaveworks

Gartner Recommended Reading

Innovation Insight: Top 4 Use Cases for GitOps

Is Using GitOps-Style Automation With Kubernetes Right for Me?

How to Scale DevOps Workflows in Multicluster Kubernetes Environments

Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes

Automate the Application Delivery Value Stream

Cloud-Native Infrastructure

Analysis By: Tony Iams, Michael Warrilow

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

Cloud-native infrastructure is optimized for developing and/or delivering applications based on cloud-native architecture. Cloud-native infrastructure is programmable, resilient, immutable, modular, elastic and configured with a declarative approach that supports self-service. Although there are many possible ways to deploy cloud-native infrastructure, in practice, large-scale cloud-native infrastructure is typically based on containers and Kubernetes.

Why This Is Important

Cloud-native infrastructure delivers a higher degree of modern functionality than traditional IT environments. Examples include service discovery, programmability, automation, observability, robust network communications and security. For many leading-edge organizations, containers and Kubernetes have become the foundation for the next generation of their IT infrastructure. Combined with new operational practices, cloud-native infrastructure will better serve the needs of modern applications and organizations' digital ambitions.

Business Impact

Cloud-native application architecture requires infrastructure that is optimized to increase software velocity; enable developer agility, application scalability and resilience; and reduce technical debt. Cloud-native infrastructure can extend the economics and operational benefits of cloud computing to infrastructure deployed on-premises. These options include consumption-based models (CBMs) for on-premises compute and storage, and public cloud Kubernetes services running on-premises.

Drivers

- Gartner clients seek to accelerate their digitization initiatives by adopting cloud-native architecture and platforms. These workloads require the underlying infrastructure and platforms to be scalable, elastic and programmable to an unprecedented degree.
- Modern applications require application-centric infrastructure rather than machine-centric virtual infrastructure. The scale-out orientation of cloud-native infrastructure, and flexible and modern networking, such as service meshes, provide a strong foundation for modern application architecture patterns, such as microservices.
- Organizations want to reduce vendor lock-in and enhance application portability by using higher-level abstractions that can decouple the applications from underlying infrastructure through common APIs and interfaces. Cloud-native infrastructure provides a way for organizations to deploy a common runtime and services across disparate environments.
- The immutability of cloud-native infrastructure helps to avoid configuration drift that can occur in traditional infrastructure where ad hoc changes can lead to system configurations that go unrecorded.
- Many organizations are investing in edge computing, which requires varying degrees of compute resources and network bandwidth. Cloud-native infrastructure supports the deployment and operation of edge environments in a more seamless manner and at the necessary scale of management.
- Hyperscale cloud providers and the open-source community are spawning a new round of infrastructure innovation. This provides the ability for organizations to reduce the gap for organizations that are constrained from using public cloud. For distributed hybrid environments, cloud-native infrastructure provides a more consistent platform and reduces the disparity between public cloud and traditional on-premises environments.

Obstacles

- The introduction of cloud-native infrastructure will require the adoption not just of new technologies, but also new processes and governance, as well as cultural and organizational changes. For example, the successful implementation of cloud-native infrastructure will require the practice of DevOps and adoption of operational practices based on declarative configuration. Many traditional and risk-averse organizations will struggle to succeed with this degree of transformation.
- Cloud-native infrastructure continues to mature but is still in the early stages of adoption by mainstream enterprises. Although Kubernetes is a solid foundation for cloud-native infrastructure, operational teams will need to supplement it with additional tooling for important capabilities.
- Adequate investment in skills such as platform engineering and DevOps will also be required. These skills can be difficult to develop or hire and to retain, given demand and the scarcity of experienced and skilled resources.

User Recommendations

- Plan and implement cloud-native infrastructure holistically by giving new operations and sourcing models the priority required to enable cloud-native architecture and technology.
- Focus on achieving the main benefit of cloud-native infrastructure: enabling competitive advantage in support of the organization's digital ambitions.
- Build relevant expertise in platform engineering by iteratively enhancing cloud-native capabilities based on developer feedback and metrics.
- Maximize agility needed to support the upper layers of the cloud-native platform by automating the operation of cloud-native infrastructure.
- Quantify the potential cost, performance and productivity benefits by evaluating emerging cloud-native infrastructure solutions such as serverless computing and bare-metal environments.
- Improve IT operational efficiency and reduce technical debt by prioritizing vendors that offer consistent management, licensing and support for their hybrid offering both on-premises and in the cloud.

Gartner Recommended Reading

Solution Path for Cloud-Native Infrastructure With Kubernetes

KubeVirt

Analysis By: Dennis Smith, Michael Warrilow

Benefit Rating: Low

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

KubeVirt is an open-source project that allows enterprises to create and operate virtual machines within Kubernetes clusters, allowing virtualized and containerized applications to run alongside each other within a single Kubernetes cluster.

Why This Is Important

As enterprises expand into cloud-native infrastructure while still operating a more traditional VM environment, they find that supporting containers and VMs separately is difficult and inefficient. KubeVirt offers enterprises the potential for virtual machines to be combined with their container environment, allowing infrastructure and operations (I&O) teams to operate VMs using the same management APIs used for containers.

Business Impact

I&O organizations face challenges while adopting and efficiently supporting their new Kubernetes deployments along with their existing virtual machine environments. KubeVirt begins to bring these two environments together by combining some of the operational needs for new and traditional application workloads. This enables enterprise digital strategies by allowing I&O to develop modern tooling and processes while maintaining legacy and evolving into new infrastructure.

Drivers

- **Enabling digital business:** Enables I&O teams to gain experience with modern technology and accelerate the evolution from monolithic applications to microservices architectures and the delivery of digital products.
- **Infrastructure modernization:** Interests in modernizing infrastructure by adopting cloud-native infrastructure, with Kubernetes being the foundation of such infrastructure. KubeVirt enables Kubernetes to orchestrate VMs in pods in the same manner as pods of containers so that administrators can apply one management workflow for both types of virtualization.
- **Edge computing deployments:** Where hardware limitations cause a desire to leverage Kubernetes on bare metal hardware.
- **Contingency planning:** Exploring alternatives to VMware's server virtualization platform in response to concerns about the pending acquisition of VMware by Broadcom.

Obstacles

- KubeVirt is an early-stage technology (incubating Cloud Native Computing Foundation [CNCF] project) with a few enterprise solutions that have been proven in production environments.
- Lack of operational tooling, in comparison to what teams have grown accustomed to with VMs.
- Creates a risk of perpetuating legacy workloads by delaying the need to address technical debt.
- Desire of some enterprises to caretake their legacy environment and do their new development in an infrastructure environment solely devoted to Kubernetes.

User Recommendations

- Assess the value of KubeVirt in the context of opportunity cost. With limited resources, I&O leaders must determine the potential benefit of investing in the transformation of legacy production workloads relative to building a modern I&O platform.
- Leverage KubeVirt for workloads that are not easily moved to containers, but where there is business benefit from cloud-native orchestration with Kubernetes.
- Engage with vendors offering KubeVirt solutions to understand their plans/roadmap for the technology and also client reference successes.
- Ensure KubeVirt is supported by your strategic technology partners. This includes independent software vendors (ISV) certification and support for virtual workloads running in production. Similarly, ensure compatibility and certification for hardware and operating systems that will host workloads running KubeVirt.
- Perform proofs of concept to validate functionality before making buying decisions.

Sample Vendors

D2iQ; Google; Platform9; Red Hat; SUSE

Gartner Recommended Reading

How to Run Containers and Kubernetes in Production

Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes

Solution Path for Cloud-Native Infrastructure With Kubernetes

Market Guide for Container Management

Modernize and Run Existing Applications on Red Hat OpenShift Container Platform

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

Kubeflow

Analysis By: Dennis Smith, Sumit Agarwal

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Kubeflow is an open-source platform built on top of Kubernetes that contains a curated set of compatible tools and frameworks specific to developing and deploying machine learning (ML) systems.

Why This Is Important

As enterprise AI initiatives mature from pilots and proofs of concept into operationalization of AI and/or MLOps, platforms like Kubeflow will be crucial to managing the complexity of framework model management and deployment. Additionally, more enterprises see a need to consolidate a ML platform on Kubernetes as more and more organizations use Kubernetes as the foundation for cloud-native platforms.

Business Impact

Scaling AI into production requires enterprises to build systems that manage ML pipelines comprising capabilities including frameworks, data ingest and model life cycle management. The process of building ML projects is often complex and has many stages, with each stage often built from customized building blocks. Kubeflow builds on Kubernetes to enable a system for deploying, scaling and managing ML pipelines across development, training, validation and deployment of models.

Drivers

- Increased interest in AI throughout the IT industry.
- Kubeflow allows AI/ML workloads to be operated on Kubernetes as well as other workloads, thus enabling a common infrastructure layer.
- An active Kubeflow development community with a stable set of contributors.
- Kubeflow provides integration with a wide range of powerful AI toolchains.
- Allows the utilization of state-of-the-art infrastructure agility technologies.
- Wide Kubeflow support across cloud and systems vendors.
- Kubeflow is open standard-based, which mitigates the risk of vendor lock-in (compared to MLaaS like SageMaker).
- There is a desire to leverage the immutability of cloud-native infrastructure based on Kubernetes. Where pipeline code is version-controlled, and all tasks are run in containers.

Obstacles

- Enterprise adoption of AI continues to be nascent.
- Kubeflow is still evolving in its functionality and often needs to be combined with complementary technologies for productive utilization.
- The learning curve associated with Kubeflow can be intimidating for organizations that do not have prior context with technology stacks, such as Kubernetes.

User Recommendations

- Establish an objective and structured process to validate and develop quick prototypes using new open-source components within the ML development life cycle.
- Develop standards and patterns for the use of open-source components within the entire organization by standardizing enterprise AI operationalization platforms and tools, demonstrating prototypes and celebrating implementation success to motivate application developers. This will enable more participation, skills development and standardization.
- Prioritize cloud and systems vendors that provide Kubeflow-optimized and preintegrated infrastructure stacks.

Gartner Recommended Reading

- Predicts 2022: Artificial Intelligence Core Technologies
- Building Data Orchestration and Workflows

Cluster Fleet Management

Analysis By: Tony Iams

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Cluster fleet management is an emerging set of tools and processes to manage the life cycle and state of multiple Kubernetes clusters. The key capabilities for cluster fleet management include deployment and upgrade of orchestration software and distributing containerized applications and operational policies across clusters. Offerings can support single and/or heterogeneous Kubernetes distributions.

Why This Is Important

As Kubernetes adoptions grow in an organization, infrastructure and operations (I&O) teams increasingly find that they must deploy and manage multiple clusters, which may be located in a single region, on-premises, in the cloud, and/or across multiple regions. An organization's approach for managing fleets of clusters can impact hybrid and multicloud initiatives, the resiliency of containerized workloads, and the design of developer workflows for delivering containers.

Business Impact

Success in digital transformation often requires the use of leading-edge cloud-native technologies. With success comes the need to improve scalability, resilience and elasticity. Because many cloud-native technologies are utilizing containers and Kubernetes, cluster fleet management becomes an important technical consideration to sustain or accelerate growth in digital products.

Drivers

- **Hybrid and multicloud container initiatives** (where separate Kubernetes clusters must be deployed): These initiatives may be based either on a single distribution or multidistribution approach. Depending upon the deployment, it is possible to use a Kubernetes distribution from a single vendor in all environments. With a multidistribution (heterogeneous) approach comes added complexity and the need to ensure consistency in management across multiple distributions and/or cloud container services.
- **Multitenancy** (where separate clusters are deployed in one location for different teams or classes of applications): By deploying separate clusters for teams or applications, the blast radius can be limited in the event a catastrophic outage makes an entire cluster unresponsive. In the cloud, assigning separate clusters to different teams also simplifies cost showback and chargeback.
- **Disaster recovery and/or resilience** (where clusters are maintained in separate data centers to take over when a primary data center becomes inaccessible).
- **Edge deployments** (where containers need to run in many remote locations, each of which requires a separate lightweight Kubernetes cluster deployed on minimal hardware).

Obstacles

- Kubernetes does not currently define a standard approach for distributing its functionality across multiple clusters, and open-source projects for extending clusters are not yet mature. Managing cluster fleets now requires the use of proprietary commercial solutions, introducing concerns about operational lock-in and vendor viability (if/when standard approaches are adopted).
- Some public cloud services charge separately for each Kubernetes control plane, which increases costs if multiple clusters are deployed.
- Deploying containerized applications to fleets of clusters based on heterogeneous Kubernetes distributions is complex. Heterogeneous fleet management tools are limited in the range of operations they support. Developer workflows may not be sufficiently flexible to account for operational differences in each distribution.

User Recommendations

- Select a management platform to control the life cycles and operation of multiple Kubernetes clusters, and to automate the distribution of Kubernetes resource definitions, policies and identities across multiple clusters.
- Determine whether to deploy homogeneous cluster fleets based on a single distribution, or use a platform engineering approach to normalize operations and application deployment workflows across clusters based on different Kubernetes distributions.
- Automate cluster life cycle management as much as possible, using an immutable approach to define abstract classes of clusters that can be used to launch specific cluster instances on demand. Avoid specialized clusters and target the ability to deploy applications on any cluster when demand arises.

Sample Vendors

D2iQ; Google; Microsoft; Platform9; Rafay; Rancher; Red Hat; Spectro Cloud; VMware; Weaveworks

Serverless Containers

Analysis By: Richard Watson

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Serverless containers are container management environments that abstract infrastructure from their users, who can run containers without managing servers or clusters. With serverless container platforms, the service automatically provisions infrastructure resources on demand, prices by units of execution and can scale resources to zero. In Kubernetes (K8s) platforms, serverless containers run enterprises' containers on worker nodes in the data plane that are not preprovisioned.

Why This Is Important

The demand for containers and K8s is running ahead of enterprises' skill sets and experience. Serverless technologies enable faster software building and delivery, low operational overheads and a truly flexible pricing model. Serverless containers help developers and operators focus on the containers they need to run instead of the undifferentiated supporting infrastructure.

Business Impact

Developing the right operational model for container deployment is challenging and requires organizational development and new skills. Serverless containers help enterprises benefit from container management and require fewer skills to adopt and operate. Similar to other serverless offerings, serverless containers enable potential cost savings as they do not require preprovisioning and scale-to-zero resources.

Drivers

- **Evolution:** Cloud services continue to evolve toward more managed, serverless operational and cost models. Today, K8s is the core of cloud-native infrastructure, but it will become less visible and more embedded in software over time. As the need for visibility and control over any aspect of infrastructure decreases, developers expect container platforms to become increasingly serverless, in which capacity management details are offloaded to the platform. Serverless container platforms are blurring the lines between traditional container-as-a-service orchestration models and serverless function abstractions.
- **Mitigating complexity:** With serverless data plane offerings, such as Amazon EKS with AWS Fargate, Google Kubernetes Engine (GKE) Autopilot and AKS virtual nodes, the cloud service provider will automatically deploy and upgrade worker nodes in K8s clusters.
- **Event-based architectures:** K8s extensions, such as Knative and K8s Event-Driven Autoscaling (KEDA), enable event-driven applications to be deployed with automatic scale-to-zero resource management.
- **Cloud-cost sensitivity:** With serverless containers, customers pay only for the vCPU and memory resources the container execution uses and not for servers idling and waiting for workloads.
- **Security:** Serverless containers have a reduced security attack surface due to elevated isolation mechanisms in implementations — for example, microVMs.
- **Network edge:** Beyond public clouds, serverless container execution is now being used in network edge to achieve low-latency, just-in-time, personalized user experiences (UXs). Also, serverless container execution is being employed in the rapid processing of event data closer to the point of collection.

Obstacles

- **Standards compliance:** Depending on the implementation, serverless containers may not comply with Cloud Native Computing Foundation (CNCF).
- **Lock-in:** Potential for other portability challenges with differing provider implementations.
- **Limitations:** Cannot be used for all container scenarios — for example, ISV software delivered in containers, K8s deployments or services that demand 24/7 processing. Also, users will be unable to use all the technologies provided in the K8s ecosystem. For example, extensions that require access to the cluster data planes for low-level monitoring or security controls.
- **Complex scaling:** Scaling up from a small number of serverless container deployments to massive numbers can be challenging.
- **Operational characteristics:** Diagnosing runtime operations and security problems is more challenging when compute for the container nodes is ephemeral.

User Recommendations

- Choose serverless containers as your default platform abstraction. Move down to more traditional, self-managed compute only when necessary.
- Prioritize workloads that fit the serverless model. Identify use cases where serverless containers offer a strategic benefit from a cost or agile perspective — for example, microservices deployments and applications with highly variable or unpredictable capacity requirements.
- Plan for the coexistence of serverless and more traditional, persistent compute implementations for container execution. Prioritize vendors that enable both models seamlessly.
- Calculate resource cost holistically. Aggregate very low container execution instance cost with realistic execution estimates and other cloud services in the environment before deciding on a serverless or virtual machine (VM)-based container execution.

Sample Vendors

Alibaba; Amazon; Google; Huawei; IBM; Microsoft Azure; Oracle; Tencent

Gartner Recommended Reading

Solution Path for Cloud-Native Infrastructure With Kubernetes

Emerging Tech Impact Radar: Cloud-Native

A CTO's Guide to Navigating the Cloud-Native Container Ecosystem

A CTO's Guide to Serverless Computing

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

Container-VM Convergence

Analysis By: Michael Warrilow, Tony Iams

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Container-virtual machine (VM) convergence refers to the fusion of hypervisor and operating system (OS)-based virtualization technologies. By integrating and optimizing the most desirable features of containers and virtual machines, container-VM convergence delivers improved workload isolation and greater infrastructure utilization. Underlying technologies include optimized virtual machine monitors (hypervisors) integrated with container runtimes.

Why This Is Important

Containers appeal to the need for modern, agile infrastructure, whereas virtual machines are a foundational element of data center infrastructure. Container-VM convergence promises the "best of both worlds" and new options for future infrastructure needs. Using innovation from public cloud providers and open-source communities, container-VM convergence creates opportunities to improve infrastructure elasticity, scalability and efficiency, while supporting modern application development.

Business Impact

Container-VM convergence enables greater infrastructure agility without sacrificing security. By infusing a cloud-inspired approach, it enables I&O teams to manage hypervisor and OS-based virtualization together. It also supports initiatives to increase developer and operational productivity. By supporting a gradual transition to cloudlike infrastructure for both on- and off-premises, it preserves existing infrastructure investments and reduces additional spend over the medium term.

Drivers

- Increasing interest is being driven by disruption in the server virtualization market and the desire for increased competitive negotiation power. This has resulted from the market dominance of a small number of server virtualization providers with few alternatives for on-premises requirements. Container-VM convergence has the potential to do this and further disrupt the competitive landscape before 2025.
- I&O teams struggle to meet the demands for cloudlike infrastructure in noncloud environments. A major factor for this deficiency is that traditional IT infrastructure is limited in the ability to deliver modern requirements, such as immutable infrastructure, agile delivery and API-driven management. Container-VM convergence provides an infrastructure technology to bridge these needs.
- CIOs require technologies such as containers to satisfy digital business requirements. This is because they are investing in areas such as digital business transformation, API integration and legacy application modernization. Containers are becoming a foundational element required to leverage many emerging technologies (cloud-native infrastructure, AI/ML and edge computing).
- Container-VM convergence can support current and future I&O requirements, and help balance rising financial pressures that will increasingly require I&O to refocus on IT cost optimization. If existing infrastructure assets are maintained for longer than was originally anticipated, container-VM convergence can reduce resulting pressure to continue supporting digital transformation. This is achieved by avoiding investment in new tools, skills and processes.
- The financial outlay is limited to upgrading existing infrastructure software; however, the benefit will be achievable across traditional and future requirements.

Obstacles

- Practices and technologies related to container-VM convergence favor Linux. For non-Linux enterprise workloads, a different operation/production support model must be maintained.
- Successful adoption of new operational approaches requires cultural change. This may constrain the uptake of container-VM convergence, leading to friction with application developers, business-led IT projects and/or circumvention of existing I&O capabilities. In turn, this will lead to compliance issues in regulated, high-security environments.
- Similarly, financially constrained organizations may struggle to see the benefit of additional investment over and above existing sunk costs.
- Container-VM convergence is not an established technology in enterprise environments.
- Container-VM convergence is yet to establish itself as being suited to the complexity of on-premises legacy environments.

User Recommendations

- Plan for container-VM convergence to deliver modern, cloud-inspired infrastructure supporting enterprise developers, while maintaining operational management and security for production environments.
- Monitor developments in enterprise adoption of open-source projects related to container-VM convergence, including Kata Containers, gVisor and Firecracker.
- Establish and maintain consensus on the role of virtual infrastructure in data center, distributed cloud, public cloud and edge — recognizing that various virtualization technologies may be necessary.
- As a precursor to selecting container-VM convergence, conduct or utilize a skills inventory. An I&O skills roadmap will help to verify existing and future skills needed to support emerging technology trends like this. Where skills deficiencies are identified, adopt container-VM convergence in a more gradual fashion that supports future requirements without detracting or unnecessarily complicating existing service-level requirements.

Sample Vendors

Alibaba Cloud; Amazon Web Services (AWS); Fly; Google; Microsoft; Red Hat; SUSE; VMware

Gartner Recommended Reading

Market Guide for Server Virtualization

Market Guide for Container Management

OpenTelemetry

Analysis By: Gregg Siegfried

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

OpenTelemetry (OTel) is a collection of specifications, tools, APIs and SDKs that describe and support the implementation of an open-source instrumentation and observability framework for software. The initiative, curated by the Cloud Native Computing Foundation (CNCF), defines multiple flavors of telemetry — traces, metrics, logs and, soon, performance profiles. OTel is widely supported by commercial and open source monitoring solutions.

Why This Is Important

OpenTelemetry introduces a portable way to instrument, generate, collect and export telemetry data about application health and performance. It has changed the way application performance monitoring (APM) solutions are assessed, deployed and used. OpenTelemetry has garnered widespread support and adoption. It is currently second in velocity only to Kubernetes within the CNCF ecosystem. Many software vendors, cloud providers and observability tools have released or announced support for OTel.

Business Impact

OpenTelemetry promises to:

- Enable deeper visibility into application health and performance, even across application and service provider boundaries.
- Facilitate the use of multiple vendors or the migration between vendors.
- Benefit product owners, SREs and platform operators by allowing them to “instrument once, analyze anywhere.”
- Standardize encoding, transport, and delivery of telemetry between sources and targets, thus improving reliability and scalability as more vendors become OpenTelemetry Protocol (OTLP) compliant.

Drivers

- **Uniformity of instrumentation:** Traces provide a rich, sequenced perspective to request handling in distributed software, but are not always enough to fully identify and resolve anomalies. By including support for correlating metrics and logs with traces, OpenTelemetry incorporates a more complete dataset for application observability use cases.
- **Software architecture:** Microservices, containers and functions are powerful constructs that serve as the basis for modern applications. Loose coupling facilitates the build, test and release of independent components.
- **Cost of APM solutions:** Many organizations cannot afford to monitor all of their applications with commercial APM offerings and either reduce the monitoring footprint or leverage a low-cost, secondary APM tool. OpenTelemetry facilitates the latter by allowing the same telemetry to be flexibly routed to different solutions.
- **Site reliability engineering:** In many organizations, Site Reliability Engineers (SREs) are responsible for health and performance management and are the most likely to demand insights from OpenTelemetry. Interest in building an SRE role is on the rise.

Obstacles

- **Maturity:** OpenTelemetry is evolving rapidly, and is beginning to be used widely in the field. It is still most suitable for early adopters. There may be a cost in time and effort associated with adopting it now that will diminish over time.
- **Implementation variations:** The need to ship support for an emerging set of specifications has led some vendors to make assumptions about work in progress, which risks multivendor compatibility. In April 2023, Elastic donated their Elastic Common Schema to the OpenTelemetry project, which may help reduce these instances.
- **Roadmap:** While the tracing and metrics specifications are completely stable, OpenTelemetry logging is partially "experimental" as of this writing. These specifications often stabilize earlier, but full implementation of all flavors will be spotty through development. An initiative to add continuous performance profiling as a fourth type of telemetry is in the very early stages.

User Recommendations

- Prefer vendors that are committed to OpenTelemetry when selecting monitoring solutions.
- Embrace OpenTelemetry for distributed tracing today when building trace instrumentation into your custom application software.
- Instrument your cloud-native applications using OpenTelemetry SDKs as they are available for your languages and frameworks.
- Augment existing APM solutions with OpenTelemetry for hybrid workloads.

Sample Vendors

Amazon Web Services (AWS); Cisco Systems; Dynatrace; Elastic; Honeycomb; Lightstep; New Relic; Splunk

Gartner Recommended Reading

Assessing OpenTelemetry's Impact on Application Performance Monitoring

Monitoring and Observability for Modern Infrastructure and Applications

Solution Path for Modern Infrastructure and Application Monitoring

At the Peak

WebAssembly (Wasm)

Analysis By: Oleksandr Matvitskyy, Gregg Siegfried

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

WebAssembly (Wasm) is a lightweight virtual stack machine and binary code format designed to support secure, high-performance applications on webpages. A growing number of programming languages can generate Wasm as a target, and applications beyond the web are becoming more common. Nonbrowser use cases range from Lua-like application extensibility mechanisms to server-side application services, as an alternative to containers or as a platform for serverless and edge applications.

Why This Is Important

Wasm has potential to disrupt runtime environments like VMs and containers by improving software portability, efficiency, performance and security. As a W3C standard, developed in partnership with vendors, web browsers support it today. The server-side Wasm ecosystem is emergent, with standardization via the CNCF and Bytecode Alliance. One of the co-founders of Docker has been quoted as saying, “If WASM+WASI existed in 2008, we wouldn’t have needed to create Docker. That’s how important it is.”

Business Impact

Similar to the benefits of Java VM, container environments or public cloud infrastructure, Wasm benefits are associated with technology, so business impacts are aligned with technology transformations like application replatforming and rearchitecting. Wasm represents technology disruption of the application runtimes with new risks for incumbent application longevity, security and compliance.

Drivers

- Browser performance: Modern, browser-based UIs can be complex and heavy with substantial application and presentation logic delegated to the client side. This requires a runtime that is faster than the interpreted JavaScript VM and able to deliver native or near-native compute performance.

- Portability of code and browser limitations: The JavaScript VM can be used on both browser and server side. However, restricting browser-side implementation to JavaScript creates obstacles for developers proficient in other languages or code sharing between browser and server. Wasm allows development in most popular languages, for both the browser and server side. It also supports multiple processor architectures including ARM, and is compatible with the Kubernetes ecosystem.
- Edge computing: The need to deliver and execute latency-sensitive code closer to the user is increasingly a requirement for many modern workloads. Wasm is a near-perfect vehicle for meeting this type of requirement due to its compact packaging and very low resource requirements.
- Security: The capability model supported by the Wasm runtimes allows an extremely granular model for managing the “sandbox” within which the code executes that minimizes the attack surface. Unlike Java, Wasm is designed to be secure by default.
- Scalability: The startup time for Wasm applications is near instantaneous (below one millisecond). In a server-side use case, rather than keeping idle request handlers waiting for traffic, the request handlers are created at the time that the requests are received.
- Language flexibility: Many programming languages can compile into Wasm. Rust is a particularly popular choice, but support is available today for JavaScript, Go, Python and C/C++, among others.

Obstacles

- Developer tooling: Wasm represents lower abstraction level compared to the modern popular runtimes, so developers need improved tooling, component libraries and frameworks to keep development productivity high.
- Architecture and skills: While Wasm is designed for interoperability, it’s not just a matter of switching it on or off when developers compile their code directly for Wasm. Usage of Wasm in the existing applications requires significant changes in application architecture and design.
- Security risks: Wasm supports much more granular control and can be safer than JavaScript engine in browser implementation. However, current browsers’ DOM reliance on JavaScript is blocking the opportunities for addressing browser security concerns with Wasm implementation.

- Toolchain maturity: The DevOps toolchain for building, testing, deploying and releasing Wasm has not yet reached a level of maturity sufficient for enterprise use outside of the experimental.

User Recommendations

- Pilot the use of Wasm for performance-sensitive client-side software rather than defaulting to JavaScript. This will acquaint product teams with the differences in developer experience and the language/toolchain requirements for incorporating it into your stack.
- Prepare for transition to Wasm implementations by selecting platforms and frameworks that already support Wasm or have it on their roadmap. This can be an easy win for the organizations choosing to minimize the impact on application architecture and implementation.
- Explore the server-side Wasm ecosystem by encouraging a small team to prototype with the platforms and tools available from Cosmonic and/or Fermyon.

Sample Vendors

Cloud Native Computing Foundation (WasmEdge); Cosmonic; Dylibso; Fermyon; Google (Flutter with CanvasKit renderer, Node.js); Microsoft (Blazor)

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

Container Backup

Analysis By: Jerry Rozeman

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Container backup helps back up persistent volumes of containerized application data and Kubernetes configuration data like K8S objects. Container backup solutions are offered either as part of a traditional backup solution, as part of the primary storage solution or as a containerized application.

Why This Is Important

Container backup is an emerging and largely nascent technology that protects organizations against data and configuration loss in a containerized environment. It is different from a physical or virtual machine (VM) backup as there is no direct mapping between the application and the underlying storage.

Business Impact

Container backup can help:

- Business owners responsible for application data in addressing the data and application configuration loss risk associated with containerized application environments, because such protection is not available by default.
- Platform and application engineering teams in protecting their data as they become the new owners responsible for containerized application data protection.

Drivers

- Containerized application adoption will increase at a rapid pace during the next few years primarily driven by the rapid growth of cloud adoption, application modernization leveraging containerization and the need for application mobility. This will fuel the need to protect data in these environments.
- There is a steady increase in use of containers to run stateful applications — the 2022 CNCF Annual Survey shows that over 63% of respondents are running stateful applications in containers in production (an increase from 55% in the 2020 CNCF Annual Survey).
- DevOps processes require tight integration with backup tools that support Kubernetes and can be embedded in the continuous integration/continuous delivery (CI/CD) workflow. Container backup solutions can deliver on this need by delivering data management features that are configured with declarative and immutable artifacts.
- Recovery strategies designed for physical infrastructure and virtual machines (virtualization) don't work well with containers and Kubernetes.

Obstacles

- Data protection of new workloads has always been an afterthought and that especially applies to container-based applications.
- While container technology seems like the next evolution of server virtualization, the technology and operating model are completely different compared to operating and protecting VMs and hypervisors.
- Container technology requires new buyers in application or DevOps teams who do not see backup to be as high a priority as the infrastructure team does.
- It will still take a long time for the majority of organizations to move away from traditional hypervisors and VMs to container technology in production. This limits the growth potential of container technology and as such it will limit the growth of data protection in containerized environments.
- The current ecosystem for container backup is quite overcrowded with a lack of standards, while demand is running behind.

User Recommendations

- Determine the need for container backup based on application criticality as not every containerized app requires backup.
- Invest in container knowledge to understand the need for backing up Kubernetes objects like namespaces, secrets, keys and configuration maps, in addition to just persistent volumes.
- Align container backup requirements with the organizational structure as, unlike traditional infrastructure, container backup operations will be performed by the platform or application engineering teams.
- Dedicate budget for protecting containers as it requires additional investments in backup solutions and infrastructure.
- Adopt a strategy for container backup just as with every other data source in your enterprise.
- Select specialized container backup solutions first while traditional backup solution capabilities mature over time.

Sample Vendors

Catalogic Software; Cohesity; Commvault; Dell Technologies; Druva; IBM; Pure Storage; Rubrik; Trilio; Veeam Software

Gartner Recommended Reading

Magic Quadrant for Enterprise Backup and Recovery Software Solutions

Critical Capabilities for Enterprise Backup and Recovery Software Solutions

Innovation Insight for Backup and Recovery for Kubernetes-Based Containerized Applications

Comparing Backup and Disaster Recovery Approaches for Kubernetes

Solution Path for Cloud-Native Infrastructure With Kubernetes

eBPF

Analysis By: Simon Richard

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Extended Berkeley Packet Filter (eBPF) is an enhancement to the Linux operating system kernel that allows specific instruction sets to run (sandboxed) inside the kernel. It allows companies to add features to Linux without changing kernel source code or requiring kernel modules.

Why This Is Important

eBPF increases the extensibility of Linux. It allows users to create hooks that are triggered by Linux kernel events. This offers a safer and simpler way to add capabilities, such as performance, security and visibility, in Linux. Technology vendors like ISVs and cloud providers use eBPF to avoid kernel-level modules, which carry inherent risks. eBPF is used in production at scale by hyperscalers, including AWS, Facebook and Netflix, and content delivery networks (CDN) such as Cloudflare.

Business Impact

eBPF improves observability, security and performance for applications. However, most enterprises will not use eBPF directly. Technology vendors do use eBPF as an underpinning technology in their products and services to improve the performance and safety of programs that run on Linux. eBPF allows extremely technically savvy organizations to safely and quickly make changes to Linux, compared to using alternative approaches, such as Linux kernel modules or upstreaming to the Linux distribution.

Drivers

- eBPF usage is driven by hyperscalers using it to deliver more efficient cloud offerings, as well as networking, monitoring and security vendors that use it in their products.
- Hyperscalers use eBPF to remediate kernel vulnerabilities without patching to address Day 0 vulnerabilities, and to more efficiently handle distributed denial of service (DDoS) attacks.
- Organizations are looking to accelerate the development speed of software that runs on Linux via avoidance of the requirement for upstream inclusion into the Linux distribution.
- Organizations are looking to improve the performance, security and monitoring capabilities of software running on Linux.
- eBPF is popular among technologically advanced companies, including technology vendors and hyperscalers, because it provides a standardized interface, supported kernel portability and requires less in-depth kernel programming knowledge.
- eBPF helps overcome scale and visibility limitations of iptables, which is the default networking stack in Linux. eBPF helps optimize and customize Linux network packet handling by processing them earlier in the cycle.
- Vendors are increasingly using eBPF in their career network infrastructure (CNI) software to improve performance, security and network visibility.

Obstacles

- While it is realistic for technology vendors and hyperscalers, most enterprises lack the expertise and skills necessary to build and integrate eBPF-based functions.
- Most enterprises do not have the awareness, need or risk tolerance to tackle Linux kernel challenges directly.
- Many older Linux kernels don't support eBPF, or only partially support the latest features.
- Security and system reliability concerns will severely limit what organizations are willing to deploy using eBPF, as poorly written eBPF programs can directly impact the operation of the Linux kernel.
- Integration challenges and backward compatibility with existing non-eBPF-enabled products.

User Recommendations

- Migrate to more modern platforms for organizations that are still using Linux distributions with limited or no eBPF support.
- Seek eBBF-based Kubernetes CNI solutions when scale, performance, visibility and security are top of mind.
- Use Linux variants that provide eBPF support to enable network performance, visibility and security products.
- Explore whether eBPF can meaningfully address the organization's performance or visibility challenges by supporting technologically advanced enterprises.
- Invest in eBPF to improve performance and visibility, to avoid falling behind competitors, for networking and network security vendors.

Sample Vendors

Aqua; Cloudflare; Isovalent; New Relic; Splunk; Sysdig; Tigera

Gartner Recommended Reading

Cool Vendors in Cloud Networking

Using Emerging Service Connectivity Technology to Optimize Microservice Application Networking

Platform Engineering

Analysis By: Bill Blosen, Paul Delory

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Platform engineering is the discipline of building and operating self-service developer platforms for software development and delivery. A platform is a layer of tools, automations and information maintained as products by a dedicated platform team, designed to support software developers or other engineers by abstracting underlying complexity. The goal of platform engineering is to optimize the developer experience and accelerate delivery of customer value.

Why This Is Important

Digital enterprises need to respond quickly to customer and internal demands; therefore, flexible, complex distributed software architectures have become popular. Software product teams struggle to focus on features due to this complexity, which results in poor developer experience. Platform engineering provides a self-service, curated set of tools, automations and information driven by developer priorities to accelerate value delivery in line with internal stakeholders, such as security and architecture.

Business Impact

Platform engineering empowers application teams to deliver software value faster. It removes the burden of underlying infrastructure construction and maintenance and increases teams' capacity to dedicate time to customer value and learning. It makes compliance and controls more consistent and simplifies the chaotic explosion of tools used to deliver software. Platform engineering also improves the developer experience, thus reducing employee frustration and attrition.

Drivers

- Scale: As more teams embrace modern software development practices and patterns, economies of scale are created, whereby there is enough value to justify creating a platform capability shared by multiple teams.
- Cognitive load: Adoption of modern, distributed architectural patterns and software delivery practices means that the process of getting software into production involves more tools, subsystems and moving parts than ever before. This places a burden on product teams to build a delivery system in addition to the actual software they are trying to produce.
- Need for increased speed and agility: The speed and agility of software delivery is critical to CIOs. As a result, software organizations are pursuing DevOps which is a tighter collaboration of infrastructure and operations (I&O) and development teams to drive shorter development cycles, faster delivery and increased deployment frequency. This will enable organizations to respond immediately to market changes, handle workload failures better and tap into new market opportunities. Platform engineering can drive this type of cross-team collaboration.
- Emerging platform construction tools: Many organizations have built their own platforms, but to date, these platforms have been homegrown, individual efforts tailored to the unique circumstances of the organizations that build them. Platforms generally have not been transferable to other companies or sometimes even to other teams within the same company. However, a new generation of platform-building tools is emerging to change that.
- Infrastructure modernization: During digital modernization, some forward-looking I&O teams embrace a new platform engineering role as a way to deliver more value, increasing their relevance to the business.

Obstacles

- Lack of skills: Platform engineering requires solid skills in software engineering, product management and modern infrastructure, all of which are in short supply.
- Platform engineering is easily misunderstood: Traditional models of mandated platforms with limited regard for developer experience can easily be relabeled and thus not achieve the true benefits of platform engineering.
- Outdated management/governance models: Many organizations still use request-based provisioning models. Those need to give way to a self-service, declarative model, with the primary focus being the effectiveness of the end users developing and operating solutions using the platform.
- Internal politics: There are many intraorganizational fights that could derail platform engineering. Product teams may resist giving up control of their customized toolchains. There might also be no appetite to improve the developer experience. Enterprises may also refuse to fund platform engineering without a clear ROI.

User Recommendations

- Start small with cloud-native workloads: Begin platform-building efforts with thinnest viable platforms for the infrastructure underneath cloud-native applications such as containers and Kubernetes.
- Embed security into platforms: Enable shift-left security within DevOps pipeline platforms, which will provide a compelling paved road to engineers.
- Don’t expect to buy a complete platform: Any commercially available tool is unlikely to provide the entirety of the platform you need. Thus, the job of the platform team is to integrate the components necessary for the platform to meet your needs.
- Implement a developer portal as part of your platform: An internal developer portal (IDP) serves as the user interface that enables self-service discovery and access to internal developer platform capabilities. Consider Backstage open-source or other commercial tools. Note: “IDP” has multiple meanings in this context, as well as in the industry.

Gartner Recommended Reading

How to Start and Scale Your Platform Engineering Team

Guidance Framework for Implementing Cloud Platform Operations

Adopt Platform Engineering to Improve the Developer Experience

Innovation Insight for Internal Developer Portals

Container-Native Storage

Analysis By: Julia Palmer, Arun Chandrasekaran

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Container-native storage (CNS) is designed specifically to support container workloads and focus on addressing unique cloud-native architecture, granularity and performance demands, while providing deep integration with the container management systems. CNS is aligned with microservices architecture principles and adheres to the requirements of container-native data services, including being hardware-agnostic, API-driven and based on distributed software architecture.

Why This Is Important

CNS solutions are specifically designed to provide persistent storage to cloud-native applications. The common foundation is typically based on a distributed, software-defined and unified pool of storage and has container-level granularity of data services, while providing enterprise data management features. In addition, the entire stack is most often orchestrated with Kubernetes to manage container life cycle integration and enable self-service operations for developers.

Business Impact

- CNS enables the deployment of stateful cloud-native applications; this enhances elasticity, availability and multicloud integration.
- Infrastructure and operations (I&O) leaders require storage that can adhere to principles of container-native infrastructure as they support stateful applications, share application data and provide advanced data services.
- CNS eliminates bottlenecks to achieving portable infrastructure agility when building and deploying modern, cloud-native applications.

Drivers

- Organizations are building new cloud applications using cloud-native principles and rearchitecting traditional applications on Kubernetes platforms, both of which are driving significant momentum in the adoption of CNS.
- Due to the increased popularity of deploying and operating container environments by orchestration platform, most IT leaders require a persistent storage solution that can be tightly integrated with container orchestrators, such as Kubernetes.
- I&O leaders require new tools and processes for data management to provide storage services accessed by stateful applications running in containers and orchestrated by Kubernetes.
- CNS solutions can be deployed on-premises or in the cloud, making them optimal for hybrid and multicloud deployment infrastructure. Because CNS functions are based on software, they can be implemented in containers, enabling them to be managed with the same orchestration functions as containerized applications.

Obstacles

- A CNS solution will not be adopted by every enterprise, because it remains most appropriate for new deployments of cloud-native applications, or for applications that will be revised with significant refactoring.
- Although embracing the CNS paradigm will yield agility benefits, adopting a CNS solution is likely to increase operational complexity in the short term for traditional enterprise environments.
- CNS vendor landscape and technology is constantly evolving with a mix of early- and late-stage startups. During the past year, we have observed few acquisitions and product repositioning in the CNS market.
- Given the fragmented nature of the vendor ecosystem, I&O leaders risk creating a technology silo with CNS solutions, which is a common obstacle to large-scale adoption.

User Recommendations

- Choose storage solutions that align with microservices architecture principles and adhere to the requirements of container-native data services, such as being hardware-agnostic, API-driven, based on distributed architecture, and can support edge, core or public cloud deployments.
- Align storage solutions with cloud strategies, making sure you take into consideration CNS applicability in the public cloud and hybrid cloud scenario of your choice.
- Select storage products closely aligned with the developer workflow tools that can be directly integrated with the application layer for portability, scaling and data protection.
- Validate your vendor's capability of continuous innovation delivery, quality customer support and a consistent pricing model, given that the container ecosystem is rapidly evolving with unproven vendor business models.
- Ensure that the CNS solution is tested and qualified for specific Kubernetes platforms.

Sample Vendors

DataCore Software; Diamanti; IBM; ionir; Pure Storage; Red Hat; SUSE; VMware

Gartner Recommended Reading

A CTO's Guide to Navigating the Cloud-Native Container Ecosystem

Solution Path for Cloud-Native Infrastructure With Kubernetes

Market Guide for Container Management

2022 Strategic Roadmap for Storage

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

Kubernetes Networking

Analysis By: Simon Richard

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Kubernetes networking addresses three requirements: pod-to-pod communications, external entities to services communications (often called north-south traffic) and pod-to-service traffic communication (often called east-west traffic). Container networking interface (CNI) plug-ins handle pod-to-pod networking and pod-to-service networking; ingress controllers/gateway API handle north-south traffic; and service mesh provides enhanced east-west service traffic control and security.

Why This Is Important

Kubernetes has become the de facto standard system for container orchestration. The native networking capabilities within K8s don't suffice for most enterprise production workloads at scale. CNI software and ingress controllers bridge this gap.

Business Impact

Many organizations are investing in Kubernetes as a foundational technology to their digital business strategies. To securely scale networking within K8s, a CNI and an ingress controller are needed. K8s' CNIs enable developers to implement network policies and multitenancy without requiring developers to concern themselves with lower-level network tasks. Some K8s CNI also support network policies, which effectively implement a clusterwide distributed firewall and multitenancy.

Drivers

- The default networking capabilities in K8s do not scale to meet production enterprise requirements and/or fall short from a security/visibility and management-at-scale perspective. Kubernetes ingress controllers provide several benefits, including automatically encrypting traffic and increasing network performance and visibility.
- Open-source options reduce acquisition friction and initial cost (compared with commercial vendors), and align with the culture of many advanced customers who prefer OSS.
- Network virtualization vendors and data center networking vendors have CNI plug-ins that extend their policy model to Kubernetes applications and are supported by commercial Kubernetes platforms.
- For on-premises Kubernetes deployments, third-party CNI plug-ins and ingress controllers are necessary. However, commercial platforms include support for a default CNI plug-ins.
- On the ingress-controller side, most load balancing as well as some API gateway solutions extend their offerings to provide ingress controller functionality.

Obstacles

- Public cloud providers offer their own natively integrated CNI plug-ins and ingress controllers that integrate with their cloud platform and load balancers. Moreover, their Layer 7 load balancers can act as ingress controllers.
- Kubernetes networking vendors are pivoting to become CNAPP or service mesh vendors.
- Many enterprises lack deep K8s expertise. Moreover, many enterprises also lack deep K8s networking expertise.
- Organizations not using K8s don't need networking capabilities for it.
- Commercial vendors offering CNIs have difficulty maintaining pace with the K8s releases, which can limit the value to enterprises.
- With Kubernetes networking, pods do not have fixed IP addresses, which breaks many existing network security processes.
- The technology landscape associated with container networking solutions is fragmented, with many commercial vendors and open-source CNI plug-ins — which can confuse customers — and delay adoption for fear of making a wrong choice.

User Recommendations

- Determine if your requirements include network policies or pod-to-pod encryption. If so, look for an advanced CNI offering.
- Avoid making Kubernetes networking decisions in isolation. When sourcing, designing and deploying container networking solutions, ensure that the networking, cloud and development teams are part of a cross-functional effort.
- Hide network complexity from the application team (e.g., an ingress controller exposing Layer 7 policy rules to the application team but hiding the load balancing minutia).
- Eliminate manual network provisioning in Kubernetes-based environments. The automation and self-service provisioning of network resources are required.
- View first to extend your deployed network virtualization or data center fabric vendors/solutions from the cloud or data center networking during deployment.
- Prefer ingress controllers that extend your existing load balancing or API gateway vendors.Include requirements for turnkey Kubernetes network integration in network RFPs.

Sample Vendors

Amazon; Avesha; Azure; Cisco; F5; HAproxy; Isovalent; Juniper Networks; Tigera; VMware

Gartner Recommended Reading

Solution Path for Cloud-Native Infrastructure With Kubernetes

Using Emerging Service Connectivity Technology to Optimize Microservice Application Networking

Kubernetes Security

Analysis By: Charlie Winckless, Thomas Lintemuth

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Kubernetes security refers to the implementation of security processes, testing and controls for the Kubernetes container orchestration platform. It operates in close conjunction with individual container security, but focuses on Kubernetes configuration and admission control.

Why This Is Important

Kubernetes expands the potential attack surface for an organization, as it provides the capabilities to deploy, scale, monitor and manage container infrastructures. This increase in attack surface and the broad adoption of Kubernetes by developers — and in managed cloud environments — means security teams need tools to provide visibility and control in these environments. Unsecure Kubernetes can expose containers and its ecosystem to various attacks.

Business Impact

Kubernetes is the most common container orchestration platform, and is in many ways a de facto standard. Kubernetes involves significant complexities in terms of access control and RBAC, container name spaces, Kubernetes networking, segmentation, admission control and other configuration, requiring automated tools that work closely with container security systems to protect the environment. Kubernetes security tools help address these potential exposures.

Drivers

- Developer adoption of containers and Kubernentes has been driven by their fit to DevOps style microservices. This adoption has expanded the attack surface, forcing security teams to use different tools and approaches to address orchestration exposures.
- Container as a service (CaaS) offerings built on Kubernetes, such as Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS) and Google Kubernetes Engine (GKE), are on the rise. These require security integrations to provide coverage.
- Security and risk management leaders must address the security of their container orchestration — primarily Kubernetes — to mitigate misconfigurations and overly permissive access.
- Container environments spread the attack surface across many containers, each of which is a unique network endpoint. East-west traffic is vastly increased, and the value of segmentation is correspondingly higher. Kubernetes security solutions can address these segmentation requirements.
- Shift-left and general container security is bolstered by effective use of admission control to limit the deployment of malicious or vulnerable container images.
- Cloud-native application protection platforms are incorporating kubernetes security capabilities.

Obstacles

- Kubernetes security blurs the line between application security, infrastructure security and container security, creating overlap in vendors, offerings and responsibilities within the organization.
- Kubernetes security alone cannot address the overall security of a containerised application. Multiple other attack points exist.
- Kubernetes is often just one part of an architecture that also includes PaaS, and even serverless functions. Teams struggle to understand the whole system in these cases — not just the Kubernetes risks.
- While Kubernetes is widely adopted for container orchestration, other orchestration platforms do exist and may not be supported by the same products as Kubernetes.
- Unless the Kubernetes security solution is designed to provide minimal friction for developers, their adoption will be at best, resisted and, at worst, actively circumvented.

User Recommendations

- Use and enforce admission controls to prevent vulnerable or malicious containers from being deployed into your environment. Use these as a guard rail for your developers.
- Ensure that your tools support validating the Kubernetes orchestration environment for proper patch levels and correct and compliant configuration, both in development and in production.
- Ensure that the Kubernetes tools you adopt have the ability to do more than forward logs, and to support both native Kubernetes and Kubernetes-based CaaS options.
- Enforce least privilege and minimal access — avoid wildcard permissions and use name spaces to reduce permission access. Use RBAC for all users and service accounts.
- Use an industry standard baseline like the CIS benchmarks and automate scanning of your environment to ensure continuous compliance.
- Address Kubernetes networking risks by adopting tools that support network policy controls.
- Collaborate with engineering teams to implement security controls that require application context.

Sample Vendors

Aqua Security; ARMO; Lacework; Orca; Palo Alto Networks; Red Hat; Sysdig; Trend Micro; Wiz

Gartner Recommended Reading

Market Guide for Cloud Workload Protection Platforms

How to Make Cloud More Secure Than Your Own Data Center

Guidance Framework for Securing Kubernetes

Container Supply Chain: 10 Security Vulnerabilities and How to Address Them

Sliding into the Trough

Cloud-Native Application Protection Platforms

Analysis By: Neil MacDonald, Charlie Winckless

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Cloud-native application protection platforms (CNAPPs) are an integrated set of security and compliance capabilities designed to help secure and protect cloud-native applications across development and production. CNAPPs consolidate a large number of previously siloed capabilities, including container scanning, cloud security posture management, infrastructure as code scanning, cloud infrastructure entitlements management and runtime workload protection.

Why This Is Important

Comprehensively securing cloud-native applications requires the use of multiple tools from multiple vendors that are rarely well-integrated. This lack of integration and automation slows developers down and creates fragmented visibility of risk and friction. CNAPP offerings allow an organization to use a single integrated offering to protect the entire life cycle of a cloud-native application.

Business Impact

Cloud-native application protection platforms consolidate disparate fragmented security testing and protection tools that increase cost and complexity for IT. Using a CNAPP offering will improve developer and security professional efficacy. It will also reduce complexity and costs while maintaining development agility and improving the developer's experience.

Drivers

CNAPPs:

- Reduce the chance of misconfiguration, mistake or mismanagement as cloud-native applications are rapidly developed, released into production and iterated.

- Converge and reduce the number of tools and vendors involved in the continuous integration/continuous delivery (CI/CD) pipeline.
- Reduce the complexity and costs associated with creating secure and compliant cloud-native applications.
- Facilitate the reporting and auditing of cloud security posture/status.
- Improve developer acceptance with security-scanning capabilities that seamlessly integrate into their development pipelines and tooling.
- Place an emphasis on scanning proactively in development and rely less on runtime protection, which is well-suited for container as a service and serverless function environments.

Obstacles

- Cloud workload protection platform (CWPP) vendors that are good at runtime protection aren't necessarily good at integrating into development and vice versa.
- Cloud-native workloads in the form of containers and serverless functions don't require heavyweight runtime protection capabilities.
- There is no single CNAPP offering that does everything. Convergence of capabilities will occur, but will take place over several years.
- Organizations may have siloed purchases of application security testing tooling that is chosen by a different team that manages the runtime protection of workloads. Even at runtime, a separate team may be responsible for web application protection.
- Organizational immaturity in terms of cloud-native application development may inhibit adoption and fragment buying motions.
- Buying centers and influencers are shifting to newer roles such as DevOps architects and cloud security engineering, requiring information security teams to coordinate with these users.

User Recommendations

- Sign contracts of only one to two years because the market for CNAPP is changing rapidly.

- Solicit CWPP vendors to scan containers in development and add cloud security posture management (CSPM) capabilities, including infrastructure-as-code scanning.
- Select integrated offerings with flexible licensing models that allow you to only pay for the capabilities your organization is prepared to use.
- Evaluate the CSPM vendor's ability to add scan of Kubernetes security posture management (KSPM) as well as provide runtime Kubernetes protection capabilities.
- Consolidate open-source software (OSS) vulnerability scanning and software composition analysis through integrations or replacement within a CNAPP offering.
- Scan containers proactively in development for all types of vulnerabilities, not just vulnerable components, including hard-coded secrets, malware and Kubernetes misconfiguration.

Sample Vendors

Aqua Security; Cisco; Lacework; Microsoft; Orca Security; Palo Alto Networks; Rapid7; Sysdig; Trend Micro; Wiz

Gartner Recommended Reading

Market Guide for Cloud-Native Application Protection Platforms

How to Select DevSecOps Tools for Secure Software Delivery

How to Make Cloud More Secure Than Your Own Data Center

Programmable Infrastructure

Analysis By: Philip Dawson, Nathan Hill

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

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

Why This Is Important

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

Business Impact

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

Drivers

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

Obstacles

- The ongoing cost of refreshing API-enabled infrastructure components on-premises after initial implementation adds financial pressure to organizations.
- Applying automation to existing monolithic infrastructure components fails due to the lack of platform agility and vendor lock-in.
- While APIs enable integration across different infrastructure platforms, the lack of open APIs/API compatibility across vendor platforms creates a siloed mentality.
- The implementation of programmable infrastructure is hampered by the early adoption of it within infrastructure and operations (I&O), and the shortage of skilled software engineering resources to comprehensively exploit it (especially in web technologies such as HTTP and JSON to develop these APIs).

User Recommendations

- Deploy a programmable infrastructure to further abstract application from infrastructure delivery and pursue an agile digital business outcome.
- Implement a programmable infrastructure by investing in infrastructure automation tools and continuous delivery (example vendors for these markets are listed below, but no single vendor or platform can enable an organizationwide programmable infrastructure strategy) leading to API-led programmable platforms.
- Invest in infrastructure and DevOps, and modernize legacy IT architectures to implement an API-driven infrastructure.
- Examine reusable programmable infrastructure building blocks leveraging programmable infrastructure strategy built on repeatable and available skills from providers.

Sample Vendors

Amazon Web Services; CU Coding; Google; IBM; Microsoft; Oracle; Quality Technology Services; RackN; Tencent; VMware

Gartner Recommended Reading

Market Guide for Servers

Predicts 2023: XaaS Is Transforming Data Center Infrastructure

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

Container Management

Analysis By: Dennis Smith, Michael Warrilow

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Gartner defines container management as offerings that enable the development and operation of containerized workloads. Delivery methods include cloud, managed service and software for containers running on-premises, in the public cloud and/or at the edge. Associated technologies include orchestration and scheduling, service discovery and registration, image registry, routing and networking, service catalog, management user interface, and APIs.

Why This Is Important

Container management automates the provisioning, operation and life cycle management of container images at scale. Centralized governance and security are used to manage container instances and associated resources. Container management supports the requirements of modern applications, including platform engineering, cloud management and continuous integration/continuous delivery (CI/CD) pipelines. Benefits include improved agility, elasticity and access to innovation.

Business Impact

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

Drivers

- The adoption of DevOps-based application development processes.
- The rise of cloud-native application architecture based on microservices.
- New system management approaches based on immutable infrastructure, which gives the ability to update systems frequently and reliably maintained in a “last known good state” rather than repeatedly patched.
- Cloud-based services built with replaceable and horizontally scalable components.
- A vibrant open-source ecosystem and competitive vendor market have culminated in a wide range of container management offerings. Many vendors enable management capabilities across hybrid cloud or multicloud environments. Container management software can run on-premises, in public infrastructure as a service (IaaS), or simultaneously in both.
- Container-related edge computing use cases have increased in industries that need to get compute and data closer to the activity (for example, telcos, manufacturing plants, etc.).
- AI/ML use cases have emerged over the past few years, leveraging the scalability capabilities of container orchestration.
- Cluster management tooling that enables the management of container nodes and clusters across different environments is increasingly in demand.
- All major public cloud service providers now offer on-premises container solutions.
- Independent software vendors (ISVs) are increasingly packaging their software for container management systems through container marketplaces.
- Some enterprises have scaled sophisticated deployments, and many more are planning container deployments. This trend is expected to increase as enterprises continue application modernization projects.

Obstacles

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

User Recommendations

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

Sample Vendors

Alibaba Cloud; Amazon Web Services; Google; IBM; Microsoft; Mirantis; Red Hat; SUSE; VMware

Gartner Recommended Reading

Market Guide for Container Management

Serverless Infrastructure

Analysis By: Jeffrey Hewitt

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Serverless infrastructure is a model of IT service delivery in which the underlying enabling resources are used as an opaque, unlimited, shared pool that is continuously available without advance provisioning and priced in units of the consumed IT service. The runtime environment (that is, the compute, storage, networking and language execution environment) required to execute an application or service is automatically provisioned and operated.

Why This Is Important

Accelerating the development and delivery of software is a core imperative for IT leaders. Not only do serverless technologies enable organizations to build and deliver software faster, but they also offer low operational overheads, resource scaling as needed and an elastic pricing model. Cloud providers and open-source software vendors are all innovating and making serverless products available for a broad set of use cases.

Business Impact

Serverless technologies enable organizations to build cloud-native applications with newer application architectures — such as microservices, which can usher in higher degrees of resiliency, elasticity and agility for digital workloads. Serverless technologies also enable consumption of platform services by developers and business users, with the infrastructure provisioning and life cycle management abstracted away from the consumer.

Drivers

- **Evolution:** In the past few years, the term “serverless infrastructure” has evolved to include much more than function as a service (FaaS) products. Currently, it refers not only to a programming model such as FaaS, but also to an operational model where all provisioning, scaling, monitoring and configuration of compute infrastructure are delegated to the platform. Examples of such architectures include serverless containers and serverless databases.

- **Operational simplicity:** Serverless infrastructure obviates the need for IT departments to perform infrastructure setup, configuration, provisioning and management.

- **"Built-in" scalability:** Infrastructure scaling is automated and elastic.
- **Cost-efficiency:** You only pay for infrastructure resources when they are needed to support requested transactions.
- **Developer productivity and business agility:** Serverless infrastructure abstracts infrastructure architecture and allows developers to focus on writing code and designing applications.

Obstacles

- **Vendor lock-in:** As with most cloud functionality, the leading serverless implementations are proprietary to specific cloud providers. If an application has to move from one cloud platform to another, it will have to be significantly reengineered.
- **Low degree of control:** The managed service model and the runtime virtualization of serverless technologies bestow huge benefits, but at the cost of little or no control over the service. The environment is a "black box" that must be used as it is.
- **Skills gap:** Serverless operations require a major shift in skills and best practices, with much more code and API-oriented service delivery.

User Recommendations

- Ensure cost governance and budget control by evaluating the cost implications of event-driven application architectures and the pricing models of different vendors. Be aware of API gateway, network egress and other costs.
- Revise data classification policies and controls to account for the fact that objects in a content store can now represent code as well as data.
- Rethink IT operations from infrastructure management to application governance, with an emphasis on ensuring that security, monitoring, debugging requirements and application SLAs are being met. In cases where on-premises deployment is merited, IT teams can support FaaS in the role of service provider.

Sample Vendors

Alibaba; Amazon Web Services; Apache Software Foundation; DigitalOcean; Google; IBM; Knative; Microsoft; OpenFaaS; Oracle

Gartner Recommended Reading

A CTO's Guide to Serverless Computing

When to Use Serverless Computing to Optimize Cloud Costs?

Decision Point for Selecting Virtualized Compute: VMs, Containers or Serverless

Service Mesh

Analysis By: Anne Thomas

Benefit Rating: Low

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

A service mesh is a distributed computing middleware that manages communications between application services — typically within managed container systems. It provides lightweight mediation for service-to-service communications and supports functions such as authentication, authorization, encryption, service discovery, request routing, load balancing, self-healing recovery and service instrumentation.

Why This Is Important

A service mesh is lightweight middleware for managing and monitoring service-to-service (east-west) communications — especially among microservices running in container management systems, such as Kubernetes. It provides visibility into service interactions, enabling proactive operations and faster diagnostics. It automates complex communication concerns, thereby improving developer productivity and ensuring that certain standards and policies are enforced consistently across applications.

Business Impact

Service mesh is one of many management technologies that provide software infrastructure for distributed applications. Service meshes are most often used with services deployed in container management systems, such as Kubernetes. This type of middleware, along with other management and security middleware, helps provide a stable environment that supports "Day 2" operations of containerized workloads. However, the technology is complex and often unnecessary for smaller deployments.

Drivers

- Microservices and containers: Service mesh adoption is closely aligned with microservices architectures and container management systems like Kubernetes. Service mesh supports useful functionality in ephemeral environments, such as dynamic service discovery and mutual Transport Layer Security (mTLS) between services.
- Observability: As microservice deployments scale and grow more complex, DevOps teams need better ways to track operations, anticipate problems and trace errors. Service mesh automatically instruments the services and feeds logs to visualization dashboards.
- Resilience: A service mesh implements the various communication stability patterns (including retries, circuit breakers and bulkheads) that enable applications to be more self-healing.
- Bundled feature: Many container management systems now include a service mesh, inspiring DevOps teams to use it. The hyperscale cloud vendors provide a service mesh that is also integrated with their other cloud-native services.
- Federation: Independent vendors such as Buoyant, greymatter.io, HashiCorp, Kong and Solo provide service meshes that support multiple environments.

Obstacles

- Not necessary: Service mesh technology can be useful when deploying microservices in Kubernetes, but it's never required.
- Complexity: It's complex to use and administer, and there are increasing discussions on why not to use a service mesh in technology discussion groups and social media.
- Redundant functionality: Users are confused by the overlap in functionality among service meshes, ingress controllers, API gateways and other API proxies. Management and interoperability among these technologies is still nascent within the vendor community.
- Overhead: Service mesh technology consumes resources and typically adds overhead to the interactions it manages. Some vendors now support alternate architectures, such as a shared-agent model to reduce overhead, but this solution reduces the observability benefits.
- Competition with "free": Independent service mesh solutions face challenges from the availability of platform-integrated service meshes from the major cloud and platform providers.

User Recommendations

- Determine whether the value you might get from a service mesh in terms of improved security or observability is worth the increase in complexity and administration of the service mesh. A service mesh becomes more valuable as the number of service-to-service (east-west) interactions increases.
- Favor the service meshes that come integrated with your container management system unless you have a requirement to support a federated model.
- Reduce cross-team friction by assigning service mesh ownership to a cross-functional platform engineering team that solicits input and collaborates with networking, security and development teams.
- Accelerate knowledge transfer and consistent application of security policies by collaborating with I&O and security teams that manage existing API gateways and application delivery controllers.

Sample Vendors

Amazon Web Services; Ambient Mesh; Buoyant; Google; HashiCorp; Istio; Kong; Microsoft; Solo.io

Gartner Recommended Reading

How a Service Mesh Fits Into Your API Mediation Strategy

Climbing the Slope

Immutable Infrastructure

Analysis By: Neil MacDonald, Tony Harvey

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Immutable infrastructure is a process pattern (not a technology) in which the system and application infrastructure, once deployed, are never updated in place. Instead, when changes are required, the infrastructure and applications are simply updated and redeployed through the CI/CD pipeline.

Why This Is Important

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

Business Impact

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

Drivers

- Linux containers and Kubernetes are being widely adopted. Containers improve the practicality of implementing immutable infrastructure due to their lightweight nature, which supports rapid deployment and replacement.

- The GitOps deployment pattern, which emphasizes continuously synchronizing the running state to the software repository, has become an effective way to implement immutable infrastructure in Kubernetes-based, containerized environments.
- Infrastructure as code (IaC) tools (including first-party cloud provider IaC tools) have increasingly integrated configuration drift detection and correction, improving the practicality of implementing immutable infrastructure across an application's entire stack and environment.
- Interest in zero-trust and other advanced security postures where immutable infrastructure can be used to proactively regenerate workloads in production from a known good state (assuming compromise), a concept referred to as "systematic workload reprovisioning."
- For cloud-native application development projects, immutable infrastructure simplifies change management, supports faster and safer upgrades, reduces operational errors, improves security, and simplifies troubleshooting.

Obstacles

- The use of immutable infrastructure requires a strict operational discipline that many organizations haven't yet achieved, or have achieved for only a subset of applications.
- IT administrators are reluctant to give up the ability to directly modify or patch runtime systems.
- Applying the immutable infrastructure pattern is most easily done for stateless components. Stateful components, especially data stores, represent special cases that must be handled with care.
- Implementing immutable infrastructure requires a mature automation framework, up-to-date blueprints and bills of materials, and confidence in your ability to arbitrarily recreate components without negative effects on user experience or loss of state.
- Many enterprise applications are stateful applications deployed on virtual machines. These applications are oftentimes commercial off-the-shelf and are not designed for fully automated installation when redeployed.

User Recommendations

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

Sample Vendors

Amazon Web Services; Google; HashiCorp; Microsoft; Perforce; Progress; Red Hat; Snyk; Turbot; VMware

Gartner Recommended Reading

Comparing DevOps Architecture to Automate Infrastructure and Operations for Software Development

2022 Strategic Roadmap for Compute Infrastructure

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

Innovation Insight for Continuous Infrastructure Automation

Market Guide for Cloud-Native Application Protection Platforms

Kubernetes

Analysis By: Wataru Katsurashima

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Kubernetes is an orchestrator and scheduler for containerized workloads. Originally developed by Google, it has been open-sourced and is now governed by Cloud Native Computing Foundation (CNCF). Kubernetes is at the core of many container management offerings, which are over 90, certified by CNCF currently available.

Why This Is Important

As the de facto standard for container orchestration and scheduling, Kubernetes is a key technology that underpins container management. Almost all major container management vendors, including all hyperscale cloud providers, offer Kubernetes-based products. Many open-source and/or cloud-native ecosystems are being built around Kubernetes, which is a key reason for its importance.

Business Impact

Kubernetes can be a foundation for delivering new forms of agility and digital innovation. It is an underpinning for cloud-native and/or modern applications that allow organizations to respond to change faster and to improve customer and employee experience while mitigating vendor lock-in risk.

Drivers

- Kubernetes solves a problem many organizations have: Orchestrating multiple containers across multiple servers across various infrastructures, to support a cloud-native architecture.
- Recognized as the de facto standard, Kubernetes has far more supporting products, service providers and skilled engineers than any other container orchestration technologies.
- Kubernetes has very large and diverse technology ecosystems covering adjacent areas, such as container-based networking/storage, DevOps toolchains, observability, security and so forth. Independent software vendors (ISVs) are increasingly packaging their software as Kubernetes applications.
- As an open project hosted by CNCF, Kubernetes alleviates concerns about the risk of vendor lock-in. CNCF empowers and extends its ecosystem by offering certification programs for Kubernetes products, engineers and service providers.
- As all major cloud infrastructure and platform services (CIPS) providers offer Kubernetes-based container services, a Kubernetes-managed environment is available regardless of your CIPS choice.
- A growing number of tools support multicluster management of Kubernetes in hybrid and multicloud environments.
- Enhancements to Kubernetes continue to support new use cases. Examples include artificial intelligence/machine learning, serverless services and edge environments.

Obstacles

- Architecting and managing Kubernetes systems requires advanced knowledge and experience, and its learning curve is steep. Kubernetes users often raise concerns about operational complexity and lack of skills.
- Operating Kubernetes in a multicloud, heterogeneous environment remains complex, immature and unstandardized.
- From a developer's perspective, Kubernetes is a primitive, low-tier function. On its own, it does not provide the cloud-native platform that developers want.
- There are too many offerings already certified by CNCF (approximately 90). Vendor consolidation is inevitable and will cause confusion in the market.
- Maintaining security in a Kubernetes environment is complex. It requires securing not only the container runtime and management components but also the container images and continuous integration/continuous delivery (CI/CD) pipelines.
- Kubernetes and containers suit modern workloads. The diversity of most enterprise IT environments means that the key benefits are difficult to attain for legacy workloads.

User Recommendations

- Take advantage of new sources of innovation from Kubernetes' open ecosystem by using Kubernetes as a foundation when developing new cloud-native applications and modernizing existing applications.
- Meet the diverse requirements of multiple application teams efficiently by building a Kubernetes-based platform that enables consistent governance and management in hybrid or multicloud environments.
- Invest in self-service capabilities to encourage developer adoption and as a key way to avoid "Kubernetes" sprawl.
- Determine the necessary degree of investment in sourcing or developing Kubernetes expertise. Assess the ability of infrastructure and operations (I&O) teams to maintain and operate Kubernetes and the cost of external assistance needed to address any skills gaps.

Sample Vendors

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

Gartner Recommended Reading

Market Guide for Container Management

CTOs' Guide to Containers and Kubernetes — Answering the Top 10 FAQs

A CTO's Guide to Navigating the Cloud-Native Container Ecosystem

Micro OS for Containers

Analysis By: Thomas Bittman

Benefit Rating: Moderate

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

A micro operating system (OS) for containers needs to be small and lightweight, and be designed specifically to support containers. Designed for clustered microservices architecture applications, it is often deployed in cloud environments. A micro OS for containers is intended for rapid deployment and horizontal scaling, with a typical image footprint ranging from 150MB to 500MB.

Why This Is Important

Modern, general-purpose OSs often have large footprints, are cumbersome to deploy and require relatively large hardware platforms. As containers, microservices and edge computing deployments develop a range of smaller form factors, more agile and clusterable micro OSs are required to support them.

Business Impact

Micro OSs will enable business agility through more rapid application development, easier and more efficient platform management, and high levels of horizontal scaling to meet business needs. Micro OSs will be core enablers to most new digital business applications — both in the cloud and at the edge.

Drivers

- Agile deployment models that require a small footprint software foundation
- Small, container-based solutions that don't require a rich general-purpose OS
- Microservices architectures and DevOps models that are designed for many small footprints
- Processing-constrained edge computing deployments using containers

Obstacles

- Micro OSs face decades of skills, process and application architectures centered on rich, general-purpose OSs and vendor business models surrounding existing OSs.
- Many micro OSs are being developed as subsets of existing OSs, but many new ones are also emerging. Not all of them will survive.
- Applications may need refactoring to operate with a micro OS.

User Recommendations

Micro OS technologies should be:

- Evaluated based on their technical maturity and footprint, and their interoperability with intended cloud providers and orchestration technologies.
- Assessed according to their feature sets, for example, too much, not enough or just right, and their fit with chosen container frameworks.
- Examined based on the viability of the vendor or the level of community support for open source.
- Rated according to the support and update technology provided by the vendor, which can be a subscription update service.

Sample Vendors

Amazon Web Services; Google; Microsoft; Red Hat; SUSE; VMware

Gartner Recommended Reading

Market Guide for Container Management

Prioritizing Security Controls for Enterprise Servers and End-User Endpoints

Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes

OS Containers

Analysis By: Thomas Bittman, Philip Dawson

Benefit Rating: Transformational

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

OS containers are a shared OS virtualization technology that enables multiple applications to share an OS kernel without conflicting. A “container daemon” provides logical isolation of processes. This enables several applications to share an OS kernel while maintaining their own copies of specific OS libraries.

Why This Is Important

Containers were previously used to increase the density of lightly used workloads, for improved infrastructure management. Now containers are focused on developer requirements for agile development, rapid provisioning and real-time horizontal scaling, especially for microservices architecture applications and cloud-native computing.

Business Impact

Container technologies are part of a development architecture that helps enterprises become more agile, with applications that can change quickly, and scale rapidly to demand. In production, containers will often be used for new applications designed for agile development. However, for developer ease of use, containers will also be used as wrappers for traditional, monolithic workloads.

Drivers

- Lightweight overhead for small applications (improving capacity utilization and density)
- Portabability — containers package up the code and its dependencies making it easier to migrate workloads reliably and predictably
- Ease of use and reuse by application developers
- Alignment with microservices architecture and agile development

Obstacles

- Reliance on the OS for application isolation can create security concerns, especially in multitenant environments.
- Containers are not direct replacements for hypervisors and, unlike with hypervisors, existing applications require redesign to take full advantage of the benefits of containers.
- Container use is constrained by the immaturity and complexity of tools and operations, especially in security, monitoring, data management and networking.
- Developing the right operational model for Kubernetes deployments is difficult, and requires organizational evolution and new skills.

User Recommendations

Infrastructure and operations leaders responsible for data center infrastructure should:

- Use containers when security and manageability concerns are easily mitigated.
- Combine containers with virtual machines (VMs) to separate developer concerns from capacity management, and when the performance overhead of VMs is an acceptable trade-off.

Sample Vendors

Canonical; Docker; Microsoft; Mirantis; Oracle; Red Hat; Virtuozzo; VMware

Gartner Recommended Reading

Market Guide for Container Management

Prioritizing Security Controls for Enterprise Servers and End-User Endpoints

Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes

Cloud-Native Architecture

Analysis By: Anne Thomas

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Cloud-native architecture is the set of application architecture principles and design patterns that enables applications to fully utilize the agility, scalability, resiliency, elasticity, on-demand and economies of scale benefits provided by cloud computing. Cloud-native applications are architected to be latency-aware, instrumented, failure-aware, event-driven, secure, parallelizable, automated and resource-consumption-aware (LIFESPAR).

Why This Is Important

Many organizations are moving to cloud-native architecture as they modernize their applications and migrate to cloud and cloud-like infrastructures. Cloud-native principles and patterns enable applications to operate efficiently in a dynamic environment and make the most of cloud benefits. Organizations adopting cloud, containers and serverless infrastructures must apply these principles to ensure their applications perform well, consume resources efficiently and handle failures gracefully.

Business Impact

Cloud-native architecture has the following business impacts:

- It ensures that applications can take full advantage of a cloud platform's capabilities to deliver agility, scalability and resilience.
- It enables DevOps teams to use cloud self-service and automation capabilities more effectively to support continuous delivery of new features and capabilities.

- It can also improve system performance and business continuity, and can lower costs by optimizing resource utilization.

Drivers

- **Cloud adoption:** Organizations want to make the most of cloud platforms, including serverless platforms and container-based systems, to support their digital business initiatives, but they can't fully exploit cloud benefits without cloud-native architecture (summarized as LIFESPAR).
- **DevOps automation:** Software engineering teams are adopting cloud-native architecture to support cloud-native DevOps automation. A basic set of rules known as the " twelve-factor app" was first published in 2011. It is still a good summary of basic practices that support cloud-native DevOps, although many teams have moved beyond this foundation and are using cloud-native architecture to support continuous integration/continuous delivery (CI/CD).
- **Modern architecture practices:** Cloud-native architecture complements modern architecture practices such as application decomposition (following the mesh app and service architecture [MASA] structure and microservices architecture), containerization, configuration as code, and stateless services.

Obstacles

- **Application renovation:** Many existing applications require significant revisions to convert them to cloud-native architecture.
- **Complexity:** Cloud-native architecture adds a level of complexity to applications, and development teams require new skills, new frameworks and new technology to be successful.
- **Skills gap:** Without proper education, architects and developers can apply the principles poorly and deliver applications that fail to deliver the expected benefits. This leads to developer frustration in adopting the new patterns and practices.
- **Applicability:** Not every cloud-hosted application needs to be fully cloud-native, and developers may be confused about when they need to use particular patterns to address their specific application requirements.

User Recommendations

- Follow LIFESPAR architecture principles when building cloud-native applications. Adhere to the twelve-factor-app rules to ensure that the application conforms to basic DevOps practices.
- Incorporate basic cloud-native design principles in all new applications, irrespective of whether you currently plan to deploy them in the cloud. All new applications should be able to safely run on a cloud platform.
- Apply cloud-native design principles as you modernize legacy applications that will be deployed on a cloud platform to ensure that they can tolerate ephemeral or unreliable infrastructure.
- Select an application platform that matches your cloud-native architecture's maturity and priorities. Low-code platforms enable rapid development of cloud-ready applications, but they don't fully apply LIFESPAR and twelve-factor principles. Container platforms require cloud-friendly architecture. Serverless platforms require a stricter adherence to the LIFESPAR principles.

Sample Vendors

Alibaba Cloud; Amazon Web Services (AWS); Google; Microsoft; Red Hat; Salesforce; VMware

Gartner Recommended Reading

A CTO's Guide to Cloud-Native: Answering the Top 10 FAQs

8 Architectural Principles for Going Cloud-Native

How to Modernize Your Application to Use Cloud-Native Architecture

Essential Skills for Cloud-Native Application Architects

9 Principles for Improving Cloud Resilience

Container Security

Analysis By: Charlie Winckless, Michael Warrilow, Thomas Lintemuth

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Container security refers to the technologies and processes, testing, and controls in container-based environments. Full-life-cycle container security starts in development by assessing the risk or trust of the container's contents, secrets management and configuration of container instances. This extends into production with runtime container segmentation, threat protection and access control.

Why This Is Important

By enabling greater speed and agility to streamline development, container-based applications have become mainstream. Rapid adoption of orchestration platforms, such as Kubernetes, has left traditional vendors and some security teams without appropriate tools to ensure secure application deployment. Container security requires a life cycle approach, starting with scanning of containers in development and protection of containers at runtime.

Business Impact

Containers are not inherently insecure, but they are being deployed insecurely with known vulnerabilities and configuration issues. Without proper controls, developers can introduce vulnerabilities into development and, subsequently, production environments. This can expose organizations to avoidable risk. Furthermore, security has been slow to embrace secure container development practices and tools, leaving organizations unaware of potential risks and unprepared to respond to attacks.

Drivers

- Developer adoption of containers and Kubernetes has shifted threats from traditional environments to containerized ones, forcing security teams to use different tools and approaches to address these new threats.
- Container-as-a-service (CaaS) offerings, such as Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS) and Google Kubernetes Engine (GKE), are on the rise. These require security integrations to provide coverage for the clusters and containers they host.
- With rising adoption of containers, security and risk management leaders need to address container-related security issues around vulnerabilities, visibility, compromise, and compliance. This would help meet the needs of digital business and application modernization.
- Multiple point solutions can now integrate transparently into the continuous integration/continuous delivery pipeline and DevOps practices, to proactively scan containers for security and compliance issues. However, organizations must carefully manage these point solutions to minimize the complexity they introduce.
- Microservices architectures are proliferating and driving container deployments in DevOps processes, which causes some of the responsibility for securing the environment to shift left to developers. DevOps pipeline integrations provide opportunities to secure against supply chain and other development risks in these environments.

Obstacles

- Container security must start in development, yet many security vendors and enterprises treat container security as a runtime-only problem. Worse, some vendors are simply placing an agent on a container, forwarding logs and calling this “container security.”
- If container image governance policies are not introduced early on, applying standards becomes increasingly difficult, as different software product teams start to implement their own processes for building container images.
- Organizations and teams may resist the use of active runtime container security for fear of disrupting applications and business.

User Recommendations

- Create and maintain a minimum set of hardened and immutable container images as the basis for all container workloads. In doing so, prioritize the use of a container-optimized operating system distribution.
- Scan containers in development for configuration and vulnerability issues of all code types, before deploying to production. Integrate with admission controllers to prevent these vulnerable containers being deployed.
- Take advantage of continuous scanning provided by code repositories and cloud providers.
- Use automated tools to analyze the processes expected to run in containers, along with their behaviors. Use this information to replace signature-based deny-listing with allow-listing-based lockdown.
- Require container security solutions to explicitly support and integrate with your container management tooling and/or Kubernetes.
- Design single-purpose containers and clear tagging mechanisms to track data sensitivity.

Sample Vendors

Aqua Security; Lacework; Palo Alto Networks; Red Hat; Snyk; SUSE, Sysdig; Tigera; Trend Micro; Uptycs

Gartner Recommended Reading

Container Supply Chain: 10 Security Vulnerabilities and How to Address Them

Market Guide for Cloud-Native Application Protection Platforms

Entering the Plateau

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

Critical Capabilities for Application Performance Monitoring and Observability

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 Author

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

Understanding Gartner's Hype Cycles

Tool: Create Your Own Hype Cycle With Gartner's Hype Cycle Builder

Market Guide for Container Management

Table 1: Priority Matrix for Container Technology, 2023

Benefit	*Years to Mainstream Adoption*			
↓	*Less Than 2 Years* ↓	*2 - 5 Years* ↓	*5 - 10 Years* ↓	*More Than 10 Years* ↓
Transformational	OS Containers Serverless Infrastructure	Observability Platform Engineering	Augmented FinOps Kubeflow WebAssembly (Wasm)	
High	APM Kubernetes	Cluster Fleet Management Container Management Container-Native Storage FinOps GitOps OpenTelemetry Programmable Infrastructure	Cloud-Native Application Protection Platforms Container-VM Convergence Infrastructure Platform Engineering	
Moderate	Cloud-Native Architecture Container Security Micro OS for Containers	Cloud-Native Infrastructure eBPF Immutable Infrastructure Kubernetes Networking Serverless Containers	Container Backup Kubernetes Security	
Low		Service Mesh	KubeVirt	

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)