Hype Cycle for Open-Source Software, 2023

Published 21 July 2023 - ID G00779797 - 93 min read

By Analyst(s): Oleksandr Matvitskyy, Mark Driver, Anne Thomas, Nitish Tyagi

Initiatives: Software Engineering Technologies; Adopt Modern Architectures and Technologies

2023 marks 40 years of free and open-source software as we know it, but powerful forces are still pushing modern OSS evolution. Software engineering leaders need to track changing usage rules, control supply chain risks, and tap into OSS communities to accelerate innovation and time to value.

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

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

Strategic Planning Assumptions

By 2028, more than 90% of governments and enterprises will adopt standards and regulations specifically aimed at promoting and governing the relationships with open-source software (OSS).

By 2027, 25% of Global 2000 organizations will have an open-source program office to support their open-source ambitions, up from less than 10% currently.

By 2027, 40% of open-source program offices will fail to deliver business value due to an absence of strategic vision and a lack of management support.

Analysis

What You Need to Know

In the early 1980s, Richard Matthew Stallman, then developer at the Artificial Intelligence Lab of Massachusetts Institute of Technology (MIT) and his colleagues were blocked by the proprietary closed-source software of a new printer. The software didn't allow them to improve the paper handling process with proactive notifications about paper jams.

"The thing that made it worse was knowing that we could have fixed it, but somebody else, for his own selfishness, was blocking us, obstructing us from improving the software."

— Richard Matthew Stallman's speech, "Free Software: Freedom and Cooperation," New York University, 29 May 2001

Disappointment with effects of proprietary commercial software prompted the creation of the GNU Project in 1983 [1] with the subsequent start of the Free Software Foundation (FSF) in 1985. The Open Source Initiative (OSI) branched off the FSF in the late 1990s to focus on the business potential of open-source software (OSS).

Despite the conflict and contradiction that still exist between the FSF and the OSI, time has proven that both movements were not wrong. Over the last 40 years, free and open-source software has boosted innovation and empowered the development of applications and tools, including the internet, the public cloud, and the recent proliferation of AI-powered technologies. OSS also provided significant support for talent acquisition and retention.

OSS maturity together with the initiatives like software composition analysis (SCA) and software bills of materials (SBOMs), OSS assessment tools, and government regulations that directly or indirectly target OSS [2] are marking the start of a new phase of OSS evolution — regulated OSS. Although OSS regulations and standards are still in the process of being established and gaining recognition, both government and commercial organizations are allocating significant effort and resources for the creation of centralized software engineering governance with a specific focus on OSS.

Successful evolution brings wider and more strategic reliance on OSS, prompting software engineering leaders to manage their relationship with OSS by keeping track of the tools and practices included in this Hype Cycle and its Recommended Reading section.

The Hype Cycle

In 2022, 85.7 million new OSS repositories were created on GitHub, with a developer community of nearly 94 million registered members. [3]

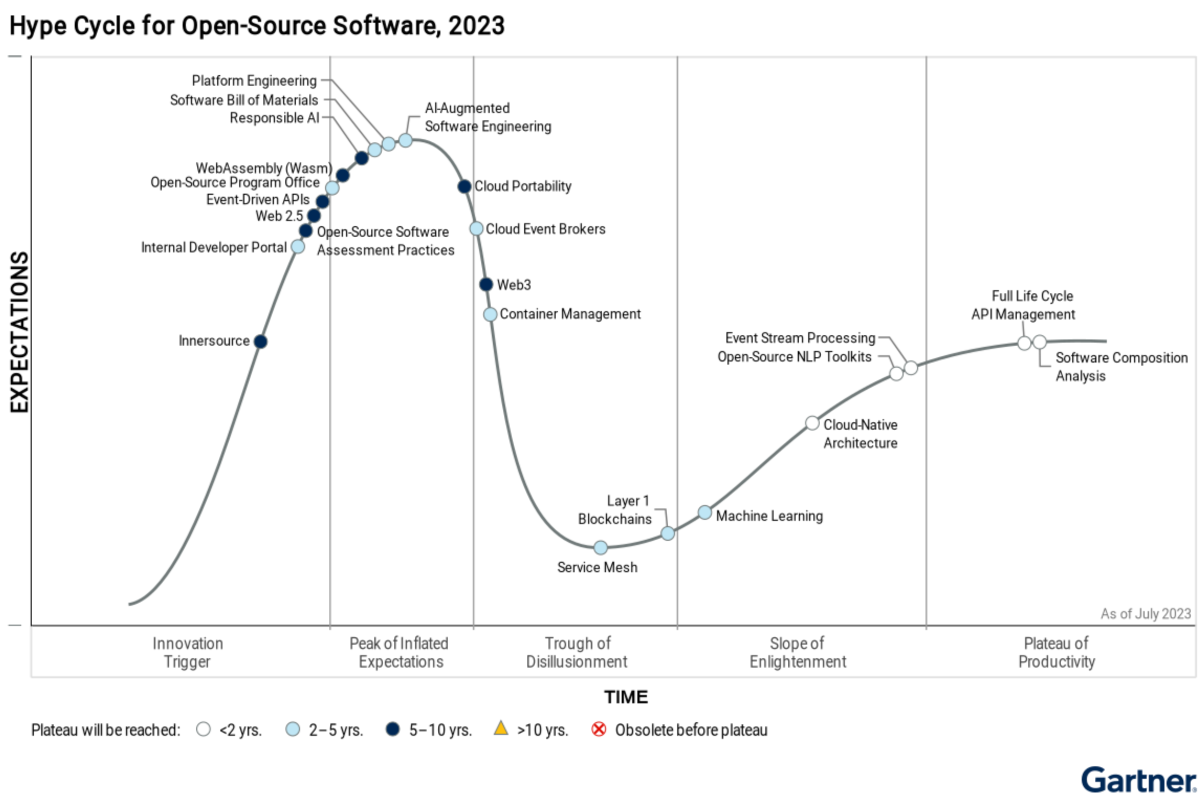
2022 saw no significant changes in security risk associated with OSS. [4] Audits found a slight increase in the number of codebases with at least one vulnerability and a slight decrease in the number of codebases containing high-risk vulnerabilities. At the same time, there is a notable increase in legal and operational risks as

- 31% of codebases include sources with no licenses or with a custom license, up from 20% in 2021.
- 91% of codebases include components that had no new development in the past two years, an increase from 88% in 2021.

This Hype Cycle includes the following notable changes:

- **Open-source software assessment practices** establish standards for the evaluation of OSS to support decisions about the selection and approval of specific OSS products for use in an organization.
- **Web 2.5** is split from the Web3 scope to acknowledge the popularity and readiness for blending selected Web3 components into the current Web 2.0 architectures.
- The inclusion of **WebAssembly (Wasm)** in the Hype Cycle is a bit overdue. Browser-based implementations are well standardized and close to the start of the mainstream adoption whereas server-side implementations have become a new hype, generating a lot of interest and opportunities for new ideas.
- The concept of an **internal developer portal** (IDP) is key to modern software engineering, and there is an interdependence between OSS and IDPs. IDPs often help to govern and manage OSS used in the organization. An example of the mutual dependence of IDPs and OSS is Backstage which wouldn't have taken off the way it did, had it not been made available as an open-source project.

Figure 1: Hype Cycle for Open-Source Software, 2023



The Priority Matrix

The Priority Matrix maps the benefit rating for each innovation against the amount of time each innovation requires to achieve mainstream adoption. The benefit rating indicates the innovation’s potential, but the rating may not apply to all industries and organizations. Applications and software engineering leaders should identify innovations that benefit their organization’s use cases significantly.

Practices and tools for OSS standardization and governance make a significant contribution to this year’s OSS Hype Cycle. Machine learning (ML) and artificial intelligence (AI) components and tools continue to lead the hype followed by the OSS-native decentralization initiatives of Web3, Web 2.5, blockchains, and popular software engineering choices such as asynchronous and cloud-based architectures. A notable increase of attention was observed for new and improved ways of working like platform engineering, innersource and internal developer portals.

Some of the innovations listed in this Hype Cycle are not exclusively offered as OSS, but open-source communities are driving innovation more significantly in each category. Evaluate these innovations based on the technological capabilities they can bring to your organization, not based on the licensing model.

Table 1: Priority Matrix for Open-Source Software, 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	Event Stream Processing Software Composition Analysis	AI-Augmented Software Engineering Layer 1 Blockchains Machine Learning Platform Engineering	Responsible AI Web 2.5 Web3 WebAssembly (Wasm)	
High	Full Life Cycle API Management Open-Source NLP Toolkits	Cloud Event Brokers Container Management Internal Developer Portal Open-Source Program Office Software Bill of Materials	Cloud Portability Open-Source Software Assessment Practices	
Moderate	Cloud-Native Architecture		Event-Driven APIs Innersource	
Low		Service Mesh		

Source: Gartner (July 2023)

Off the Hype Cycle

The 2023 OSS Hype Cycle narrows the scope to software engineering and application delivery so we delegate the following innovations to other, more specialized Hype Cycles:

- **Container-native storage** is more aligned with the Hype Cycle for Storage and Data Protection Technologies.
- **Cryptocurrencies** represent business-specific innovations covered by the Hype Cycle for Blockchain and Web3.

Other innovations removed or changed include:

- **Serverless fPaaS** that has graduated from the Plateau of Productivity.
- **CeDeFi/CeDeX** that has been retitled and updated to describe Web 2.5.
- **Blockchain platforms** that are now Layer 1 blockchains.

On the Rise

Innersource

Analysis By: Arun Chandrasekaran, Mark O'Neill, Arun Batchu, Anne Thomas, Mark Driver

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Innersource is the practice of bringing the open-source software principles of collaboration and openness into the organization for any form of software development.

Why This Is Important

Innersource can bring the benefits of collaborative, agile and robust software development principles to organizations. Every company is a software company, and applying innersource principles allows them to safely emulate the open-source model of collaborative development. Innersource can be a safe way to experiment with open-source practices while enabling governance at scale.

Business Impact

- Innersource practices lead to a culture centered on transparency and collaboration, which creates a healthy working environment.
- Innersource encourages professionals to think about software usage strategically rather than trying to solve tactical problems.
- Organizations can use innersource to align with external open-source communities and foundations to tap into their knowledge.
- Embracing open-source principles can encourage employee retention through the recognition of contributions.

Drivers

- The need for communication and collaboration between disparate software development teams within organizations drives the need for innersource.

- Software developers are now very familiar with open-source practices. They have been interacting with code repositories, using well-known open-source tools and even, in some cases, contributing to open-source projects. This creates the foundation for the effective application of open-source principles inside organizations.
- Code quality and velocity can be improved through the usage of an automated quality and consistency checking process. Such a process is often part of the technical infrastructure of an innersource effort, many times via the work of an open-source program office (OSPO).
- The increased reuse of functionality — such as code libraries or via APIs — are also key drivers of innersource initiatives.
- An important reason to adopt innersource is to increase knowledge sharing and potentially create a full-stack view through the democratization of know-how.

Obstacles

- Obtaining sponsorship from senior management can be challenging due to the long-term outlook and cultural transformation required for a successful implementation of innersource.
- Governance of innersource is a challenge since it may seem chaotic and fast-moving at first.
- Many organizations do not yet have self-service access to deploy code, or access APIs or shared libraries.
- Adoption of innersource requires a high degree of infrastructure automation, which can be challenging.
- Cultural resistance to innersource can inhibit broader adoption and maturity.

User Recommendations

- Encourage teams to open their code repositories for others to contribute to them by creating good documentation and contribution guidelines.
- Create smaller and flatter product teams that embrace DevOps, with an emphasis on egalitarian decision making and constant communication on product direction.

- Communicate and templatize best practices on coding standards, an API style guide, preferred tooling and common repositories, and a code release process.
- Enable self-service for infrastructure as a service (IaaS), and build automation, test and release pipelines.
- Create an innersource portal as a focal point for innersource initiatives.
- Explore gamification as a way to recognize and reward developers who contribute to innersource initiatives. Badges and recognition will improve employee motivation.
- Conduct hackathons to stimulate the creative process, and encourage collaboration across teams and codebases. This will also break down the silo mentality.

Gartner Recommended Reading

A CTO's Guide to Open-Source Software: Answering the Top 10 FAQs

2023 Planning Guide for Application Development

Cultivate a Culture of Ownership in Your Software Engineering Organization

Video Spotlight: Unlocking the Value of Open Source at Fannie Mae

Best Practices for Setting Up an Open-Source Program Office

Internal Developer Portal

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

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

Why This Is Important

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

Business Impact

Developer portals can have the following business impacts:

- Developer experience and productivity: Help development teams improve their delivery cadence by improving developer experience, reducing cognitive load and shortening the onboarding time.
- Reliability and resilience: Aim to provide visibility to application health and include scorecards to assess their production readiness.
- Security and governance: Include prebuilt toolkits, templates and curated libraries that help create “paved roads” with built-in compliance, security and audit policies.

Drivers

- Platform engineering: Organizations are adopting platform engineering principles and creating platform teams to scale cross-cutting capabilities across multiple development teams. Platform teams curate internal developer platforms to abstract away the complexity of siloed systems and processes. Internal developer portals serve as an interface through which developers can consume the capabilities of internal developer platforms.

- Backstage: Backstage is one of the first open-source frameworks for building developer portals. It was created at Spotify and is now a Cloud Native Computing Foundation (CNCF) incubating project. The thriving open-source community supporting Backstage has largely contributed to its enormous mind share and rapid adoption. Hundreds of organizations have adopted Backstage since it was open-sourced in 2020. Backstage's success continues to drive interest, momentum and competition in this space.

- Developer experience: With software at the core of all digital innovation today, a great developer experience that accelerates software development becomes a key competitive advantage. Therefore, software engineering leaders are increasingly focused on minimizing developer friction and frustration. The ability to curate and provide customizable, developer-friendly experiences within the developer portal and reign in complexity will drive their appeal for both product and platform teams.

- Innersource: To enable rapid innovation and facilitate greater collaboration and knowledge sharing, software engineering leaders are adopting innersource approaches to software development. However, innersource requires an easy way for other teams to discover and search for existing projects within their organization. This is why organizations adopting innersource are turning to internal developer portals to make projects available and discoverable by other teams. See InnerSource Portal.

Obstacles

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

User Recommendations

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

Sample Vendors

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

Gartner Recommended Reading

Innovation Insight for Internal Developer Portals

Drive Innovation by Enabling Innersource

Adopt Platform Engineering to Improve the Developer Experience

Cool Vendors in Platform Engineering for Improving Developer Experience

Open-Source Software Assessment Practices

Analysis By: Nitish Tyagi

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Open-source software (OSS) assessment practices are used to avoid the legal, business, viability and security risks that come with increased use of OSS within the organization. OSS assessment teams under the open-source program office (OSPO) develop policies and processes to assess OSS projects across three critical dimensions. These are alignment with governance policy, viability of the project's health and quality practices, and total cost of ownership (TCO) of a project.

Why This Is Important

More than 90% of organizations use OSS today (see GitHub Octoverse 2022) for rapidly building innovative and low-cost software products. However, OSS can introduce inherent risks if not properly vetted (see PyTorch Vulnerable to Arbitrary Code Execution and SolarWinds hack explained). Having assessment processes for OSS ensures that any OSS form — a packaged product, library/package or binary — went through an approved checklist before use, and will prevent the organization from any such risk.

Business Impact

OSS assessment practices are key for the secure and efficient use of OSS. Well-defined OSS assessment and tooling mitigate the risks of legal, security and viability issues by proactively vetting OSS products before they get introduced to the organization's technology portfolio. These processes underpin the consumption policy of open-source governance (see How to Create and Enforce a Governance Policy for Open-Source Software and Tool: OSS Governance Policy Template).

Drivers

- Organizations have formal or informal sets of rules for OSS governance; adopted OSS projects that don't align with the OSS governance policy can introduce complex legal obligations.
- The rise of upstream vulnerabilities introduced either inadvertently or maliciously (via typosquatting and other forms of dependency confusion attacks) in OSS poses huge security risks for the organization. The need to secure OSS is a key driver.
- Weak community and poor coding practices expose organizations to application's feasibility and security risks.
- OSS is considered to be free to use, although a certain TCO is always attached with each OSS project.
- Intended use cases change the assessment criteria of an OSS project altogether. For example, an organization using a GNU GPL license-based library for internal purposes only is not affected by GPL's restrictive terms and conditions.

Obstacles

- Ad hoc, poorly governed approaches to using OSS
- Lack of a properly situated open-source program office within the organization to build strategies on OSS governance and assessment
- Lack of a formal governance policy or unawareness of it across the entire organization
- Lack of presence of a one-stop customizable tool to assess any type of OSS across all three dimensions: security, health and cost
- Lack of collaboration between different stakeholders (such as security, legal, software engineering and procurement) when defining OSS assessment policies and processes

User Recommendations

- Establish an OSPO and create an OSS governance policy (if not already present) that defines the risk profile of the organization around consumption, contribution and creation of OSS.
- Create assessment policies under consumption of OSS, divided into three steps: assess governance compliance, assess the project's health and quality, and calculate the TCO (see 3 Steps for Assessing an Open-Source Software Project). Ensure that all key stakeholders (such as software engineering, security, legal and procurement) are included in defining these policies.
- Use tools such as GrimoireLab and Augur, sponsored by CHAOSS, or others such as Debricked, Synopsys and Tidelift to assess the health of an OSS project.
- Use OpenSSF tools to assess the quality of an OSS project. You can also reference Toolkit: Gartner's Open-Source Software Assessment Framework — 1.0 to do a top-level health and quality assessment of any OSS project.
- Build smooth and automated approval processes in accordance with the intended use cases.

Sample Vendors

Artifact Hub; Bionic; Community Health Analytics in Open Source Software (CHAOSS); Debricked; Google Open Source Insights; Open Source Security Foundation (OpenSSF); Tidelift

Gartner Recommended Reading

3 Steps for Assessing an Open-Source Software Project

Toolkit: Gartner's Open-Source Software Assessment Framework — 1.0

How to Create and Enforce a Governance Policy for Open-Source Software

Tool: OSS Governance Policy Template

Best Practices for Setting Up an Open-Source Program Office

Web 2.5

Analysis By: Avivah Litan

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Web 2.5 represents integration of Web3 blockchain technology artifacts, such as tokens, smart contracts and Web3 applications, with Web 2.0 applications and services. Web 2.5 application possibilities are endless. For example, decentralized peer-to-peer lending services (Web3) can be integrated into centralized bank (Web 2.0) product offerings. Similarly, news content can be tokenized (Web3) and sold by media companies using traditional Web 2.0 applications.

Why This Is Important

Web3 supports unique technological innovations that enable users to own and control their own content and money. It also supports a trustless shared system of record based on immutable data. However, enterprises are justifiably unwilling to give up governance, oversight and control of most business applications. Hence, they will mainly use Web3 innovations layered on top of their Web 2.0 applications and business models.

Business Impact

Blended Web3 and Web 2.0 applications bring together the best of centralized and decentralized systems by combining Web 2.0 business controls with decentralized blockchain technology. They support new business processes, such as automated management of tokenized assets across multiple parties in an ecosystem. Web 2.0 applications that use Web3 technologies buttress Web3 risks by supporting user onboarding, liability protection, customer service, insurance and regulatory compliance.

Drivers

- Enterprises are unwilling to give up control, which is needed to participate in Web3 applications. Hence, they are naturally inclined to adopt Web 2.5 instead.
- Innovative blockchain projects support the evolution of financial services systems into more modern infrastructure that represent a major technical upgrade of their rigid siloed and archaic rails.
- Digital asset tokenization used by traditional financial services firms are more efficiently managing fixed income assets. See Goldman Sachs' Tokenization Platform and Franklin Templeton Money Market Fund on Polygon. Other firms will follow their lead.
- Real-time, immutable and transparent payments and settlements are being supported by trusted regulated financial firms, such as Visa and Circle Internet Financial, using blockchain Web3 technology.
- Innovative blockchain projects are transmuting supply chains, document management, patent management and healthcare services, among others. Sponsoring organizations are not giving up organizational controls and are able to benefit from the technological innovations.
- Major drivers in nonfinancial sectors are: ESG and compliance, data integrity track and trace, provenance, retail/brand expansion, user-owned identity records (e.g., in healthcare and education).
- While Web 2.0 has enabled collaborative and social features that have transformed the internet, it still relies on centralized platforms and intermediaries to manage user data and transactions. This can lead to privacy and centralized gatekeeper concerns, which Web 2.5 alleviates.

Obstacles

- Most mainstream organizations do not understand Web3 or decentralized blockchain technology, and fail to articulate a clear business need for it. Organizations need to be clear on use cases that require Web3 technologies.
- Most Web3 technologies are still too hard to use. Progress is being made in improving development environments, user interfaces, and security controls, but are not widely available yet.
- Many brands are participating in Web3 applications by trading in various types of non-fungible tokens (NFTs), but most are still unclear on how they will profit.
- Decentralized applications are primary targets for hackers who usually exploit smart contract logic. These vulnerabilities must be dramatically minimized before organizations can risk funds held in smart contracts.
- Regulation of decentralized finance (DeFi) and NFT products and services varies widely across countries and, in most cases, will likely take three to five years to stabilize.
- “Web 2.5” is not yet a widely accepted industry standard or well-defined concept.

User Recommendations

- Work with organizational counterparts to understand how Web3 and blockchain technologies can improve your business opportunities. Gain an understanding of benefits, risks and requirements of decentralized applications as compared to centralized or traditional applications, and what the business benefits are of combining the two.
- Keep apprised of off-the-shelf Web 2.5 as they become more widely available, and prepare to use them if they suit your use case.
- Standardize on units of value or exchange by using tokens. Fungible and non-fungible tokens can be moved across disparate blockchains through various blockchain interoperability and smart contract options that support them.
- Ensure your security and risk management programs incorporate both Web 2.0 and Web3 technical controls.

Sample Vendors

Animoca Brands; Casper Labs; Context Labs; Finboot; Fujitsu; IPWe; NTT DATA; SettleMint; ShelterZoom; Sky Republic

Gartner Recommended Reading

FAQ for NFTs on Blockchains and Web3 Ecosystems

FAQ for Cryptocurrencies on Blockchains and Web3 Ecosystems

Quick Answer: What Is Web3?

Web3 and the Metaverse: Incomplete but Complementary Visions of the Future Internet

Event-Driven APIs

Analysis By: Max van den Berk

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Event-driven APIs describe the interfaces used in event-driven architecture, including subscriptions and channel creation. Event-driven APIs differ from traditional request/response REST APIs because they enable asynchronous communication. Popular standards for event-driven APIs include AsyncAPI.

Why This Is Important

Event-driven APIs differ from traditional request/response APIs because they enable real-time notification of changes to data and application state. They are also provided as part of an event broker infrastructure.

Industries like financial services require timely updates of data and application state, which have driven the usage of event-driven architecture and event-driven APIs.

Integration scenarios also drive the usage of event-driven APIs, including data synchronization across SaaS services.

Business Impact

Event-driven APIs open up business opportunities for faster response to change and streaming analytics.

They also facilitate, and help standardize, event-driven architecture (EDA). Event-driven APIs bring advantages such as enabling push notifications, which are much more efficient and less expensive (from both a time and networking perspective) than polling.

Drivers

- Mobile applications involve the widespread use of events, such as webhook notifications using server-sent events (SSEs) delivered over HTTP.
- Data analytics and artificial intelligence drive new use cases related to events and notifications, which in turn drive the need for event-based APIs.
- Open-source and cloud event broker technologies, especially Apache Kafka, have raised awareness of publish-subscribe infrastructure, as well as accessibility to it, and have encouraged adoption of EDA.
- The OpenAPI Specification (OAS) version 3, introduced in 2017 and now reaching widespread adoption as a standard for publishing APIs, includes a provision for callbacks as a means to describe event-driven APIs. OAS version 3.1 (February 2021) adds support for webhooks, the most common event-driven API approach for internet-facing APIs. It should be noted, however, that webhooks are a one-to-one pattern, as opposed to EDA, which also supports a many-to-many pattern.
- AsyncAPI, which defines a standard for how event-driven APIs are published, is supported by the Linux Foundation, which is the same standards body that houses OAS.

Obstacles

- API mediation products, such as API gateways, are often built or configured only for request-response APIs. In some cases, the patterns of event-driven APIs (including fire-and-forget or publish-and-subscribe methods) are not supported in the API mediation product.
- Protocol support for event-driven APIs is diverse, including webhook (HTTP/S), WebSockets, MQTT, advanced message queuing protocol (AMQP) and Apache Kafka. This makes decision making more difficult for software engineering leaders.
- Although API portals are widely used for publishing request/response APIs, their usage for event-driven APIs is nascent.
- Reuse of schema definitions, such as Protobuf, Avro and graphQL, is not standardized, and might require tooling from the end user to support it.
- Event-driven APIs, like all event-driven architecture, introduce challenges for debugging and testing.

User Recommendations

- Evaluate whether your chosen API mediation products, including API gateways as part of API management, support event-driven APIs as part of your technology selection process.
- Adopt a community-of-practice approach to raise your organization's overall skills on event-driven APIs, and their relationship to event-driven architecture and stream processing and analytics. This may be a specific EDA community of practice, or part of an API community of practice.
- Identify opportunities to replace request/response APIs with event-driven APIs within your API portfolio to enable both internal and external integration scenarios.

Sample Vendors

Ably; Axual; Axway; Confluent; Gravitee; Postman; SmartBear Software; Software AG; Solace; TIBCO Software

Gartner Recommended Reading

Using Event-Driven Integration With Enterprise Applications

Maturity Model for Event-Driven Architecture

At the Peak

Open-Source Program Office

Analysis By: Nitish Tyagi, Mark O'Neill, Mark Driver, Arun Chandrasekaran

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

An open-source program office (OSPO) is the center of competency to build strategies for governing, managing, promoting and efficiently using open-source software (OSS) and open-source data or models. The OSPO includes members from application delivery, legal, IT security, procurement and product management.

Why This Is Important

OSS is the backbone of digital innovation, with more than 95% organizations using it. However, ad hoc usage of OSS can lead to legal, security and viability risks. To manage these risks, OSPO or similar programs adoption has increased by 50% (see TODO's OSPO Survey 2022). Many large enterprises such as Alibaba, Apple, Bosch and Capital One have done so. OSPOs build cohesive strategies to efficiently use and promote OSS (see A CTO's Guide to Open-Source Software: Answering the Top 10 FAQs).

Business Impact

OSS drives accelerated software development, innovation, flexibility, cost savings and talent retention. A well-run OSPO ensures efficient consumption of OSS, implements effective governance policies, facilitates contributions and communicates the value of OSS to stakeholders. Embracing OSS, particularly contributing to or maintaining an OSS project, positively affects the retention of employees, especially developers. As the OSPO matures, it becomes a strategic partner in all technology decisions.

Drivers

- OSS is ubiquitous, but enterprises with an ad hoc approach to OSS have limited visibility into where and how OSS is used within their technology stack. An OSPO provides a strategic approach and required visibility by using correct tools.

- Poorly governed use of OSS, without proper assessment, exposes an enterprise to security, legal and viability risks. For example, ungoverned use of OSS components may violate licensing terms or infringe upon intellectual property rights, or OSS with a weak community may incur huge technical debt. An OSPO is responsible for governing the use of OSS.
- Software developers wish to contribute to open-source projects that they use in their day-to-day work. Actively supporting contributions to OSS projects helps talent acquisition and retention and ensures the longevity and relevance of the OSS your organization uses.
- Establishing the source and provenance of software components is essential, as legal requirements for software bills of materials grow (see Executive Order on Improving the Nation's Cybersecurity).

Obstacles

- The lack of funding and executive sponsorship is the biggest reason for the limited adoption of OSPOs. Establishing the correct metrics to predict the success of an OSPO is often challenging for organizations.
- Silos between different teams make it difficult for an OSPO to drive collaboration.
- Total cost of ownership is always attached with using an OSS project, but costs are often ignored when organizations do not plan their use of OSS.
- A lack of self-service tooling and automation in applying the policies related to consumption and publication can delay the software development life cycle.

User Recommendations

- Establish an enterprisewide OSPO and shift from ad hoc use of OSS to strategic use of OSS by building policies and processes to govern, assess and manage OSS. Fill up the core OSPO roles and include members from different stakeholders groups, such as application delivery, security, legal and procurement.
- Define the correct set of metrics such as developer productivity, and hiring and retention rates to measure the OSPO's success. Formulate and enforce a governance policy for consumption, contribution and creation of OSS by building an OSS governance committee.

- Work with platform engineering teams to provide the correct set of tools for artifact repository, security, issue tracking, continuous integration, continuous development, collaboration and knowledge management.
- Evaluate your OSPO's maturity level and execute strategies to gradually promote it to the next level as appropriate.

Sample Vendors

Bitergia; Cloud Native Computing Foundation (CNCF); Linux Foundation; TODO Group

Gartner Recommended Reading

Best Practices for Setting Up an Open-Source Program Office

A CTO's Guide to Open-Source Software: Answering the Top 10 FAQs

How to Create and Enforce a Governance Policy for Open-Source Software

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Video Spotlight: Unlocking the Value of Open Source at Fannie Mae

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)

Responsible AI

Analysis By: Svetlana Sicular

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Responsible artificial intelligence (AI) is an umbrella term for aspects of making appropriate business and ethical choices when adopting AI. These include business and societal value, risk, trust, transparency, fairness, bias mitigation, explainability, sustainability, accountability, safety, privacy, and regulatory compliance. Responsible AI encompasses organizational responsibilities and practices that ensure positive, accountable, and ethical AI development and operation.

Why This Is Important

Responsible AI has emerged as the key AI topic for Gartner clients. When AI replaces human decisions and generates brand-new artifacts, it amplifies both good and bad outcomes. Responsible AI enables the right outcomes by ensuring business value while mitigating risks. This requires a set of tools and approaches, including industry-specific methods, adopted by vendors and enterprises. More jurisdictions introduce new regulations that challenge organizations to respond in meaningful ways.

Business Impact

Responsible AI assumes accountability for AI development and use at the individual, organizational and societal levels. If AI governance is practiced by designated groups, responsible AI applies to everyone involved in the AI process. Responsible AI helps achieve fairness, even though biases are baked into the data; gain trust, although transparency and explainability methods are evolving; and ensure regulatory compliance, despite the AI's probabilistic nature.

Drivers

- Responsible AI means a deliberate approach in many directions at once. Data science's responsibility to deliver unbiased, trusted and ethical AI is just the tip of the iceberg. Responsible AI helps AI participants develop, implement, utilize and address the various drivers they face.

- Organizational driver assumes that AI's business value versus risk in regulatory, business and ethical constraints should be balanced, including employee reskilling and intellectual property protection.

- Societal driver includes resolving AI safety for societal well-being versus limiting human freedoms. Existing and pending legal guidelines and regulations, such as the EU's Artificial Intelligence Act, make responsible AI a necessity.

- Customer/citizen driver is based on fairness and ethics and requires resolving privacy versus convenience. Customers should exhibit readiness to give their data in exchange for benefits. Consumer and citizen protection regulations provide the necessary steps, but do not relieve organizations of deliberation specific to their constituents.

- With further AI adoption, the responsible AI framework is becoming more important and is better understood by vendors, buyers, society and legislators.

- AI affects all ways of life and touches all societal strata; hence, the responsible AI challenges are multifaceted and cannot be easily generalized. New problems constantly arise with rapidly evolving technologies and their uses, such as using OpenAI's ChatGPT or detecting deepfakes. Most organizations combine some of the drivers under the umbrella of responsible AI, namely, accountability, diversity, ethics, explainability, fairness, human centricity, operational responsibility, privacy, regulatory compliance, risk management, safety, transparency and trustworthiness.

Obstacles

- Poorly defined accountability for responsible AI makes it look good on paper but is ineffective in reality.
- Unawareness of AI's unintended consequences persists. Forty percent of organizations had an AI privacy breach or security incident. Many organizations turn to responsible AI only after they experience AI's negative effects, whereas prevention is easier and less stressful.
- Legislative challenges lead to efforts for regulatory compliance, while most AI regulations are still in draft. AI products' adoption of regulations for privacy and intellectual property makes it challenging for organizations to ensure compliance and avoid all possible liability risks.
- Rapidly evolving AI technologies, including tools for explainability, bias detection, privacy protection and some regulatory compliance, lull organizations into a false sense of responsibility, while mere technology is not enough. A disciplined AI ethics and governance approach is necessary, in addition to technology.

User Recommendations

- Publicize consistent approaches across all focus areas. The most typical areas of responsible AI in the enterprise are fairness, bias mitigation, ethics, risk management, privacy, sustainability and regulatory compliance.
- Designate a champion accountable for the responsible development and use of AI for each use case.
- Define model design and exploitation principles. Address responsible AI in all phases of model development and implementation cycles. Go for hard trade-off questions. Provide responsible AI training to personnel.
- Establish operationalize responsible AI principles. Ensure diversity of participants and the ease to voice AI concerns.
- Participate in industry or societal AI groups. Learn best practices and contribute your own, because everybody will benefit from this. Ensure policies account for the needs of any internal or external stakeholders.

Sample Vendors

Amazon; Arthur; Fiddler; Google; H2O.ai; IBM; Microsoft; Responsible AI Institute; TAZI.AI; TruEra

Gartner Recommended Reading

A Comprehensive Guide to Responsible AI

Expert Insight Video: What Is Responsible AI and Why Should You Care About It?

Best Practices for the Responsible Use of Natural Language Technologies

Activate Responsible AI Principles Using Human-Centered Design Techniques

How to Ensure Your Vendors Are Accountable for Governance of Responsible AI

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

AI-Augmented Software Engineering

Analysis By: Arun Batchu, Hema Nair, Oleksandr Matvitskyy

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

The use of artificial intelligence (AI) technologies (e.g., machine learning [ML] and natural language processing [NLP]) to help software engineers create, deliver and maintain applications is designated AI-augmented software engineering (AIASE). This is integrated with engineers' existing tools to provide real-time, intelligent feedback and suggestions.

Why This Is Important

Today's software development life cycle includes such routine and repetitive tasks as boilerplate functional and unit-test code and docstrings, which AIASE tools automate. AI-powered automation enables software engineers to focus their time, energy and creativity on such high-value activities as feature development. Emerging AI tools discover the configurations that meet operational goals. Software builders who use these tools remain productive and engaged, and they stay longer in their jobs.

Business Impact

AIASE accelerates application delivery and allocates software engineering capacity to business initiatives with high priority, complexity and uncertainty, helping quality teams develop self-healing tests and nonobvious code paths. These tools automatically generate test scenarios previously created manually by testers, and detect test scenarios often missed by test teams. AIASE tools detect issues with code security, consistency or maintainability and offer fixes.

Drivers

Demand drivers include:

- The increasing complexity of software systems to be engineered
- Increasing demand for developers to deliver high-quality code faster
- Increasing numbers of application development security attacks
- Optimizing operational costs

Technology solution drivers include:

- The application of AI models to prevent application vulnerabilities by detecting static code and runtime attack patterns
- The increasing impact of software development on business models
- The application of large language models to software code
- The application of deep-learning models to software operations

Obstacles

- Hype about the innovation has caused misunderstandings and unrealistic expectations about the benefits of AIASE.
- There is a lack of deep comprehension of generated artifacts.
- There is limited awareness about production-ready tools.
- Software engineers who fear job obsolescence have shown resistance.
- There is a lack of transparency and provenance of data used for model training.
- Uneven, fragmented solutions that automate only some of the tasks in the software development life cycle (SDLC).
- AI skills such as prompt engineering, training, tuning, maintaining and troubleshooting models.
- High model training and inference costs at scale.
- Intellectual property risks stemming from models trained on nonpermissive licensed code.
- Privacy concerns stemming from code, and associated proprietary data leaking as training data for AI models.
- Technical employees' fear of jobs being automated by AI.

User Recommendations

- Pilot, measure and roll out tools only if there are clear gains.
- Verify the maintainability of AI-generated artifacts, including executable requirements, code, tests and scripts.
- Track this rapidly evolving and highly impactful market to identify new products that minimize development toil and improve the experience of software engineers, such as those that ease security and site operations burden.
- Reassure software engineers that AIASE is an augmentation toolset for human engineers, not a replacement.
- Pick providers (including open-source vendors) that supply visibility to training data and transparency on how the model was trained.
- Establish the correct set of metrics, such as new release frequency and ROI, to measure the success of AIASE.

Sample Vendors

Akamas; Amazon Web Services; Diffblue; Google; IBM; Microsoft; OpenAI; SeaLights; Sedai; Snyk

Gartner Recommended Reading

Innovation Insight for ML-Powered Coding Assistants

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

Market Guide for AI-Augmented Software Testing Tools

Software Bill of Materials

Analysis By: Mark Driver

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

A software bill of materials (SBOM) is structured, machine-readable metadata that uniquely identifies a software package and the open source or proprietary components used to build it. SBOMs are designed to track and share the details of software components and their supply chain relationships across organizations. This enables greater transparency, auditability and traceability throughout the software supply chain, expediting resolution of security and compliance issues.

Why This Is Important

Gartner estimates that 40% to 80% of the lines of code in new software projects come from third parties (for example, runtime, libraries, components and software development kits [SDKs]). Most of this external code comes from myriad open-source projects; the remaining proprietary code comes from suppliers that provide little or no transparency to its status or condition. The quality and security of these external software assets (while leveraged across thousands of users) vary widely from one to another.

Business Impact

SBOMs aim to solve a fundamental problem with sharing open-source and third-party software. While organizations can use the same components, it’s inefficient to duplicate efforts around tracking vulnerabilities and analyzing compliance violations. SBOM standards such as Software Package Data Exchange (SPDX) and CycloneDX establish a common infrastructure and a data exchange format to share details about software packages. This standardization reduces time to remediate issues through better collaboration between organizations.

Drivers

- **Software License Identification and Risk Assessment —** Like commercial software, open-source software (OSS) is almost always distributed according to the terms of a license agreement articulating the ways the software may be used. Some are considered permissive, others restrictive. Restrictive licenses could pose substantial legal risk and the loss of intellectual property. SBOMs can indicate the license used to distribute a particular software package in order to support the assessment of legal risks.
- **Need for Improved Security and Compliance —** Regulatory authorities in the U.S. (like the Food and Drug Administration [FDA] and Federal Energy Regulatory Commission [FERC]) and in Europe (the European Union Agency for Cybersecurity [ENISA]) are mandating SBOMs as a prerequisite deliverable for all organizations transacting with government agencies and regulated organizations. The goal of these regulations is to provide greater visibility into software components and dependencies to accurately determine an organization's exposure to security risks and vulnerabilities.
- **Need for Improved Visibility and Traceability of Software —** SBOMs can provide software engineering and vendor risk management teams with increased transparency into how software gets built, which components make up that software, and how quickly security vulnerabilities can be identified and remediated. When teams discover flaws or vulnerabilities in a component, they can use SBOMs to quickly identify all software that is affected by the vulnerable component. SBOMs also provide them with information to assess the potential impact and risks introduced by the vulnerable component.

Obstacles

- SBOMs are not a panacea. They're only as useful as the processes and tools implemented to process, analyze and leverage the information they contain. The impediments to adopting SBOMs include sharing SBOM data due to nonstandard data formats; inability to automate software delivery workflows based on the contents in the SBOM; and the ability to generate, consume and securely distribute them at speed and scale.
- The seamless integration of SBOMs into software development, packaging and release activities will be critical for their widespread adoption. The challenges in scaling the use of SBOMs include policy automation based on the information in the SBOM, secure distribution across organizational boundaries and managing their life cycle. The attestation of SBOMs for provenance is a prerequisite to treating them as a trusted source of dependency metadata. The dynamic nature of dependencies and their versions requires regenerating and updating SBOMs every time the artifacts are updated.

User Recommendations

- Enhance visibility into the complete software supply chain by tracking dependencies between open-source components in the software development life cycle (SDLC) using SBOMs.
- Discover affected components and share vulnerability data by using SBOM standards to exchange information about components and their relationships.
- Ensure SBOMs are rebuilt along with software artifacts by generating SBOMs during the software build process, rather than relying on pregenerated SBOM data. Use software composition analysis tools to generate and verify SBOMs for third-party, open-source components.
- Mitigate software supply chain security risks by requesting commercial software providers to provide standards-based SBOMs (for example, software package data exchange [SPDX], software identification [SWID], CycloneDX) during the procurement process.

Sample Vendors

Apiiro; Codenotary; Cybeats Technologies; Eracent; Finite State; Invicti Security (formerly NetSparker); OX Security; Revenera (Flexera); Rezilion

Gartner Recommended Reading

Emerging Tech: A Software Bill of Materials Is Critical to Software Supply Chain Management

Innovation Insight for SBOMs

Market Guide for Software Composition Analysis

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Cloud Portability

Analysis By: Lydia Leong, David Smith

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Cloud portability is the ability for a customer to move a cloud-based application or workload from one cloud provider to another.

Why This Is Important

The ability to migrate between cloud providers (whether infrastructure as a service [IaaS], platform as a service [PaaS] or SaaS) is important for reducing both vendor lock-in and the impact of vendor concentration risk. Cloud portability offers customers more freedom and flexibility to alter workload placement based on evolving business or technical needs. It is sometimes perceived as offering a negotiation advantage when contracting with cloud providers, especially when coupled with the myth of container portability.

Business Impact

When a cloud-based application is not portable, the customer is dependent on a single cloud provider for that application. This exposes the customer to a range of vendor-related risks, and potential concerns about the provider's financial viability, service outages, alignment of the provider to the customer's long-term needs and negotiation leverage. However, achieving cloud portability decreases agility, slows cloud implementations and increases complexity and costs.

Drivers

- Flexibility: Organizations seek cloud portability to be able to potentially replace one provider with another (reducing “lock-in”) or, less commonly, to create the possibility of repatriating workloads on-premises. However, lock-in is not caused solely by differentiated capabilities or proprietary API dependencies. Processes, tools, data gravity, the cloud provider’s ecosystem, employee skills and contractual obligations all prevent customers from easily and economically switching between providers.

- “Cloud drift” not “cloud exit:” Organizations are increasingly shifting away from focusing on portability that would enable an immediate “disaster recovery” cloud exit. Rather, they have begun to focus on portability that would allow a gradual exit from a cloud provider over a period of two years or more. This allows the “drift” of workloads from one cloud to another cloud, based on the natural life cycle of each application. This is better aligned to shifts in the organization’s business needs and vendor preferences over time.

- Scenario-based plans: Organizations are building contingency plans that envision specific scenarios and risks that would result in a desire to switch cloud providers. Scenario-based cloud exit planning allows addressing cloud risks in a more specific fashion. Organizations may also be regulatorily required to address cloud provider concentration risks through this sort of planning. While portability needs to be a consideration from the beginning of cloud application architecture and development, shorter exit time frames increase architectural restrictions and portability complexity.

- Container myths: Cloud portability is aggressively hyped by vendors, especially in the context of containers and Kubernetes. However, containers provide limited portability benefits and do not address most of the underlying causes of cloud provider lock-in. Management tools that claim to commoditize the underlying cloud services generally reduce cloud benefits, add unnecessary cost and create a different, often riskier, point of lock-in.

Obstacles

- SaaS applications and platforms are normally entirely nonportable. An exit requires completely replacing the application.
- Open-source or commercial-off-the-shelf-based (COTS-based) PaaS may be replaced by running that software directly. Proprietary PaaS is nonportable and must be replaced by an alternative solution.
- The portability of cloud IaaS workloads may be limited by provider differentiation in infrastructure capabilities. Furthermore, the customer must replace the service's security, management and automation capabilities, including revising DevOps and continuous integration/continuous deployment (CI/CD) toolchains.
- Portability is a program requiring ongoing effort and investment. In addition to the direct technology impacts, customers must consider the impact of cloud provider replacement on the organization's internal skills, contractual relationships (including relationships with independent software vendors [ISVs] whose software they license to run on IaaS), and dependencies upon vendors in the cloud provider's ecosystem.

User Recommendations

- Address SaaS vendor risks through contractual means and integration strategies. SaaS portability is impractical.
- Treat the need for cloud IaaS and PaaS portability as an application portability challenge, not as an infrastructure lock-in problem.
- Balance portability costs and drawbacks against its benefits. Require a business case for larger investments in portability.
- Decide how much portability a particular application needs based on business requirements, and then make architectural choices for that application reflecting the portability requirements.
- Focus on scenario-based exit planning and take a risk management approach to cloud portability.
- Find the balance between desired short- and long-term business outcomes, and the potential future risks of cloud provider dependence. Cloud portability requires compromises between risk reduction and the negative impacts of increased complexity, cost and time to deliver cloud solutions.

Gartner Recommended Reading

Infographic: Mitigate Cloud Risks With Realistic Exit Planning

How to Create a Public Cloud Integrated IaaS and PaaS Exit Plan

Cloud Governance Best Practices: Managing Vendor Lock-In Risks in Public Cloud IaaS and PaaS

Quick Answer: How Should Executive Leaders Respond to Cloud Concentration Risk Concerns?

How to Manage Concentration Risk in Public Cloud Services

Sliding into the Trough

Cloud Event Brokers

Analysis By: Yefim Natis, Gary Olliffe, Max van den Berk, Keith Guttridge

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Event brokers play the role of an intermediary in event-driven architecture (EDA), administering the event topics, registering event publishers and subscribers, and facilitating the capture and distribution of event notifications. Cloud event brokers expand these capabilities with cloud-native characteristics of elastic scalability and multitenancy. They are optimized to support cloud-based applications and operations.

Why This Is Important

Most of the outcomes of digital business transformation depend, in part, on an organization's continuous awareness of relevant business events and its ability to respond in real time. Event brokers facilitate the monitoring and distribution of event notifications to application services for an automated response, dashboards for human action, or data stores for further analysis. Cloud event brokers bring these capabilities to cloud-based applications and operations.

Business Impact

Organizations aware of their ecosystem events are better prepared to manage unexpected interruptions and capitalize on opportunities in business moments. They are equipped to broadcast notable events for simultaneous, multitargeted responses. Event brokers enable organizations' versatility in monitoring multiple sources of events and communicating to many responders in parallel with strong scalability, integrity and resilience. Cloud event brokers extend this functionality to on-cloud operations.

Drivers

- Increased demand for real-time insights drives organizations to manage event streaming and stream analytics, leading, in turn, to event brokers for governance and coordination of event traffic.

- Increased adoption of Apache Kafka, by both businesses and leading technology vendors, promotes organizational awareness of the benefits and opportunities that event-driven application design brings.
- The migration of business applications to the cloud demands new platforms and communication infrastructure, driving many organizations to evaluate and adopt event broker services, paired with integration and API management offerings.
- The availability of multiple vendors' cloud event broker services, based on open-source standards such as Advanced Message Queuing Protocol (AMQP), Apache Pulsar, Apache Kafka, MQTT and NATS, provides competitive and differentiated options in event broker services for a better-tuned fit to customers' use cases.
- Increased maturity of cloud event brokers supports more advanced capabilities in performance, data and process management, and optimization of event-driven applications.
- Most leading SaaS offerings support some event processing, increasing awareness of the benefits and opportunities of event-driven application design in a large number of mainstream business and government organizations.
- The increasing popularity of digital integration hubs and data virtualization approaches deliver near-real-time data to applications by consuming event streams, instead of direct database lookups.
- Growing demand for advanced use patterns of "change cloud-data capture" drive new adoption of cloud event brokers.

Obstacles

- Cloud event broker offerings become too expensive as more proprietary features are added to help differentiate from the competition.
- Event broker functionality, embedded in some platform and application services, fragments control of event streaming across the organization, while delaying a systematic investment in event brokering.
- Some software engineering teams use webhook and WebSocket tools to set up event notifications, delaying the full many-to-many experience of EDA that's implemented via an event broker technology.

- Lack of universally supported standards for protocols or APIs for EDA implementation increases the costs and complexity of managing a large event-driven application infrastructure.
- Some organizations choose the request/reply communication where EDA would be a more effective model, just due to the unmitigated familiarity bias and lack of required skills.

User Recommendations

- Mix up the complementary strengths of the request/reply service-oriented architecture (SOA) and EDA, and encourage new projects to consider the combined use of both, as appropriate.
- Pilot experimental projects using cloud event brokers to gain insight and skills for upcoming, more advanced projects. Even a basic pub/submiddleware service is sufficient as a precursor for a full-featured event broker.
- Give preference to cloud event broker vendors demonstrating the understanding of the full life cycle of event processing functionality and responsibility.
- Plan for coordinated use of an event broker and an event stream processing (ESP) platform (also called stream analytics platform). The technologies are different but contribute to a shared business outcome.

Sample Vendors

Amazon Web Services (AWS); Confluent; Google; IBM; Microsoft; Solace; TIBCO Software

Gartner Recommended Reading

Using Event-Driven Integration With Enterprise Applications

Essential Patterns for Event-Driven and Streaming Architectures

Apply Event-Driven Approaches to Modernize IoT Data Integration

How to Identify Your Event-Driven Architecture Use Cases to Select the Best-Fit Event Broker

Maturity Model for Event-Driven Architecture

Web3

Analysis By: Avivah Litan, Adrian Leow

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Web3 is a new stack of technologies, business and governance constructs that support decentralized web applications that enable users to control their own identity and data. Technologies include blockchain as a trust verification mechanism, privacy-preserving and interoperability protocols, decentralized infrastructure and application platforms, decentralized identity, and support for applications like decentralized finance. These will eventually realize the vision of a partly decentralized web.

Why This Is Important

Web3 enables new business and social models. Smart contracts run applications that eliminate intermediaries and administrative overhead controlling centralized entities. Tokens (including cryptocurrencies) power the business models and economics of Web3 and are built into blockchain protocols. Web3 provides building blocks for new types of applications. It supports user ownership of their data and content, and new business opportunities, such as authenticating and tracking carbon emissions data.

Business Impact

Web3 offers new features to manage digital assets and ownership rights for content creators. Other benefits include:

- Trustless transaction verification
- Smart contract automation of shared processes
- Tokenization of digital or physical assets
- Self-sovereign identity

Existing Web3 applications, such as decentralized finance (DeFi), and non-fungible tokens (NFTs), have yielded previously unachievable benefits for everyday users, investors, artists, content creators and communities.

Drivers

- Blockchain infrastructure is becoming more mature, scalable and cost-efficient to support mainstream usage. For example, Ethereum transitioned to a new consensus mechanism in September 2022, resulting in over a 99% drop in its energy consumption.
- Web3 empowers content creators to sell their work in the form of NFTs that ensure they — and not intermediaries — are paid for their work based on contract terms they set themselves whenever, for example, they sell an artwork.
- Tokenization of real-world assets can benefit from Web3 applications, and early adopters are realizing new business benefits that were not possible before. For example, the tokenization of carbon certificates and credits is helping Williams, a natural gas infrastructure company, its partners and customers lower their carbon footprint.
- Web3 apps are useful for authenticating and trading data. For example, blockchain marketplaces are used for managing IP portfolios (see IPwe) and fixed-income portfolio management (see Goldman Sachs' Digital Asset Tokenization Platform).
- DeFi protocols, such as Aave and MakerDAO, provide users with lending and borrowing services run by smart contracts that eliminate the need for intermediaries, thereby enabling higher yields and returns, albeit with much more risk.
- Brands are realizing new revenue streams with Web3 apps, especially with NFT issuance and sales.

Obstacles

- Web3 poses many risks, such as lack of customer protections, security threats and swings toward centralized control. Although future internet technologies will be more decentralized, they will not eliminate centralized authorities from enterprise B2B and B2C applications.
- Enterprises are unwilling to give up control of most business applications. We expect enterprises to continue using Web 2.0 for most applications through 2030, and layer Web3 technologies on top of them in a Web 2.5 configuration.
- Some early Web3 activities have achieved success, but much work remains to improve performance, governance, risk management and user interfaces. Additionally, success in well-established industries remains sparse.
- The current lack of widely applicable Web3 business applications for enterprises is hindering enterprise adoption of Web3.
- Lack of clear regulatory frameworks in the United States and some other countries continues to hinder Web3 innovation and user adoption.

User Recommendations

- Pay attention to innovative and successful Web3 initiatives and use cases in areas such as tokenization of real-world assets, decentralized identity, DeFi, art, entertainment and sports for ideas and partnerships.
- Keep abreast of developments in Web3 protocols and standards by monitoring the initiatives of Ethereum and Web3 Foundation for blockchain and distributed ledger technologies.
- Evaluate and pilot blockchain applications that implement a Web3 vision by giving end users control of their own identity data and compliant access to Web3 apps. These applications include decentralized identity, verifiable claims, cryptocurrency payments and investments, and new apps that are yet to appear.
- Evaluate projects that use Web3 technologies layered on top of Web 2.0 applications, giving organizations the ability to leverage unique Web3 features, such as smart contracts and tokenized assets, retaining process control.

Gartner Recommended Reading

FAQ for NFTs on Blockchains and Web3 Ecosystems

FAQ for Cryptocurrencies on Blockchains and Web3 Ecosystems

Quick Answer: What Is Web3?

Web3 and the Metaverse: Incomplete but Complementary Visions of the Future Internet

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

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

Layer 1 Blockchains

Analysis By: Homan Farahmand, David Furlonger

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

The Layer 1 (L1) blockchains are a type of operating system that enable secure, decentralized computing. Blockchain establishes a shared trust infrastructure in a digital ecosystem, which enables organizations to run a business function as a protocol among participants. This includes the trusted issuance, registration and exchange, or verification and tracking of digital assets, by deploying smart contracts that define and manage appropriate tokens (assets) relevant to these business functions.

Why This Is Important

In the same way internet protocols decentralized connectivity, blockchain protocols are decentralizing computing as the trust foundation of the next-generation web (Web3). Blockchains provide features such as a distributed ledger, immutability, transparency, tokenization and support for smart contracts in a multilayer architecture. L1 blockchains address decentralization and security requirements, while Layer 2 (L2) blockchains (or enterprise blockchains) address the scalability.

Business Impact

Blockchains can change the terms and governance structures of business and society by enabling autonomous and peer-to-peer transactions, based on a variety of asset tokenization and value-exchange scenarios. Public L1 blockchains provide an open trust layer, enabling a plug-and-play platform for smart contracts or L2 chains. This is in contrast to enterprise blockchains, which provide a closed trust layer that requires ongoing governance, setup and maintenance among permitted participants.

Drivers

- Blockchain core technologies continue to evolve steadily to support key features such as more efficient consensus mechanisms, enhanced execution environments, enabling L2 chains, cost optimization and privacy improvements, which vary by each L1 blockchain.

- Blockchain developers are making progress in building traditional application stack capabilities to enable integration with blockchains. These include better modeling support, developer tools, frameworks, smart-contract templates, security audit, formal verification, interoperability, integration mechanisms and software development kits (SDKs).

- Feedback from early adopters indicate the blockchain design pattern is evolving around a multitier model. This model includes L1 public blockchains for decentralization and security, and L2 chains primarily for scalability. Also, there are emerging Layer 0 chains for cross-chain state interoperability.

- Development continues to progress in design, testing and piloting across different industries, and has gained more traction with the digital business, considering the promise of high speed and efficiency.

- As digital acceleration pervades all industries and the public sector, more attention is being paid to specific Web3 use cases that blockchains can support, such as digital currency, decentralized autonomous organization (DAO), decentralized applications, decentralized identity, decentralized finance, GameFi, TradeFi, non-fungible token (NFT), and metaverse.

- There is a proliferation of blockchain-inspired solutions that mimic blockchain features, but are using centralized components. These solutions are more concerned with transaction validation from a business-rule perspective. Decentralization and trust are not inherent in the protocol, and are achieved using traditional centralized processes and technologies.

- Many enterprise application use-case requirements that can be satisfied by blockchain-inspired technologies, instead of a true L1 blockchain.

Obstacles

- Blockchain may expose the enterprise to regulatory uncertainty, as governments are still working to understand the implications of blockchain solutions. Examples are asset securitization, money laundering and/or privacy compliance issues.
- Blockchain's vulnerabilities should be tracked carefully, considering its constant evolution. Attacks on blockchains can affect the operation or cause financial losses due to faulty smart contracts.
- Blockchain relies heavily on cryptographic algorithms, which require careful consideration of the migration path to new post-quantum-era cryptography.
- Blockchain adoption may run into challenges such as the lack of success and scope definition, misaligned expectations, determining viable use cases, and developing future-proof architecture.
- Blockchains can be disruptive by changing the business models and governance structures of enterprises, based on autonomous and peer-to-peer transactions using asset tokenization.

User Recommendations

- Evaluate blockchain relevance to the current (and future) enterprise business model. Decide on a realistic progressive adoption approach based on priorities.
- Develop a blockchain-agnostic architecture strategy for decentralized application use cases to address security and scalability requirements.
- Ensure the architecture enables business innovation in the short term and a migration pathway to more suitable blockchain technologies in the long term.
- Assess your candidate blockchain technologies using the normalized functional model provided in Guidance for Blockchain Assessments.
- Ensure that your preferred blockchain platform has modernization pathways to support future Web3 requirements where it is applicable.
- Minimize any proprietary blockchain infrastructure technology and/or vendor lock-in by adopting open-source infrastructure technology with multiple independent implementations.
- Ensure support for future migrations to other winning blockchain technologies in the long term.

Sample Vendors

Algorand; Avalanche.org; ethereum.org; Hedera; Hyperledger Foundation; Quorum; R3; Solana Foundation; VeChain Foundation; VMware

Gartner Recommended Reading

Guidance for Blockchain Assessments

Guidance for Blockchain Solution Adoption

Climbing the Slope

Machine Learning

Analysis By: Shubhangi Vashisth, Peter Krensky

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Machine learning (ML) is an AI discipline that solves business problems by utilizing statistical models to extract knowledge and patterns from data. The three major approaches that relate to the types of observation provided are supervised learning, where observations contain input/output pairs (also known as “labeled data”); unsupervised learning (where labels are omitted); and reinforcement learning (where evaluations are given of how good or bad a situation is).

Why This Is Important

Over the last few years, ML has gained a lot of traction and is entering mainstream adoption because it helps organizations to make better decisions at scale with the data they have. ML aims to eliminate traditional trial-and-error approaches based on static analysis of data, which are often inaccurate and unreliable, by generalizing knowledge from data.

Business Impact

ML drives improvements and new solutions to business problems across a vast array of business, consumer and social scenarios, such as:

- Credit approval automation
- Price optimization
- Customer engagement
- Supply chain optimization
- Predictive maintenance
- Fraud detection

ML impacts can be explicit or implicit. Explicit impacts result from ML initiatives. Implicit impacts result from products and solutions that you use without realizing they incorporate ML.

Drivers

- Augmentation and automation (of parts) of the ML development process has improved productivity of data scientists and enabled citizen data scientists to make ML pervasive across the enterprise.
- Availability of quality, labeled data is driving ML adoption at enterprises.
- Pretrained ML models are increasingly available through cloud service APIs, often focused on specific domains or industries.
- ML education is becoming a standard at many academic institutions, fueling the supply of talent in this space.
- Active research in the area of ML in different industries and domains is driving applicability far and wide.
- Newer learning techniques — such as zero- or few-shot learning — are emerging, reducing the need to have high volumes of quality training data for ML initiatives, thus lowering the barrier to entry.
- New frontiers are being explored, including federated/collaborative, generative adversarial, transfer, adaptive and self-supervised learning — all aiming to broaden ML adoption.

Obstacles

- Conventional engineering approaches are unable to handle the growing volumes of data, advancements in compute infrastructure and associated complexities.
- ML is not the only popular AI initiative to emerge in the last few years. Organizations also rely on other AI techniques, such as rule-based engines, optimization techniques and physical models, to achieve decision augmentation or automation.
- Organizations still struggle to take their ML models into production. MLOps continues to be a hot trend and organizations look to specialized vendors and service providers for support in their journeys of better operationalizing ML models.

- Application of ML is often oversimplified as just model development. Several dependencies that are overlooked — such as data quality, security, legal compliance, ethical and fair use of data, and serving infrastructure — have to be considered in ML initiatives.

User Recommendations

- Assemble a (virtual) team that prioritizes ML use cases, and establish a governance process to progress the most valuable use cases through to production.
- Utilize packaged applications that fit your use-case requirements to derive superb cost-time-risk trade-offs and significantly lower the skills barrier.
- Explicitly manage MLOps and ModelOps for deploying, integrating, monitoring and scaling analytical, ML and AI models.
- Adjust your data management and information governance strategies to enable your ML team. Data is your unique competitive differentiator, and adequate data quality — such as the representativeness of historical data for current market conditions — is critical for the success of ML.

Sample Vendors

Amazon; ClearML; Databricks; Dataiku; Domino Data Lab; Google; H2O.ai; KNIME; Microsoft; MindsDB

Gartner Recommended Reading

Market Guide for Multipersona Data Science and Machine Learning Platforms

Market Guide for DSML Engineering Platforms

How to Improve the Performance of AI Projects

Infographic: Common Layers of Data Science and Machine Learning Activity

Use Gartner's MLOps Framework to Operationalize Machine Learning Projects

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

Event Stream Processing

Analysis By: W. Roy Schulte, Pieter den Hamer

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Event stream processing (ESP) is computing that is performed on streaming data (sequences of event objects) for the purpose of stream analytics or stream data integration. ESP is typically applied to data as it arrives (data "in motion"). It enables situation awareness and near-real-time responses to threats and opportunities as they emerge, or it stores data streams for use in subsequent applications.

Why This Is Important

ESP enables continuous intelligence and real-time aspects of digital business. ESP's data-in-motion architecture is a radical alternative to the conventional data-at-rest approaches that have historically dominated computing. ESP platforms have progressed from niche innovation to proven technology, and now reach into the early majority of users. ESP will reach the Plateau of Productivity in less than two years and eventually be adopted by multiple departments within every large company.

Business Impact

ESP transformed financial markets and became essential to telecommunications networks, smart electrical grids, and some IoT, supply chain, fleet management and other transportation operations. However, most of the growth in ESP during the next 10 years will come from areas where it is already established, especially IoT and customer engagement. Stream analytics from ESP platforms provide situation awareness through dashboards and alerts, and detect anomalies and other significant patterns.

Drivers

Six factors are driving ESP growth:

- Organizations have access to ever-increasing amounts of low-cost streaming data from sensors, machines, smartphones, corporate websites, transactional applications, social computing platforms, news and weather feeds, and other data brokers. Many new AI and other analytical applications need this streaming data to satisfy business requirements for situation awareness and faster, more-accurate decisions.
- The wide use of Apache Kafka and similar streaming messaging systems is reducing the cost and complexity of ingesting, storing and using streaming data.

- Conventional data engineering pipelines take hours or days to prepare data for use in BI and analytics, causing delays that are unacceptable for some purposes. Therefore, an increasing number of data engineering pipelines are being reimplemented as real-time data flows (continuous ETL) in ESP platform products or stream data integration tools with embedded ESP. These real-time data flows filter, aggregate, enrich, and perform pattern detection and other transformations on streaming data as it arrives.
- ESP products have become widely available, in part because open-source ESP technology has made it less expensive for more vendors to offer ESP. More than 30 ESP platforms or cloud ESP services are available. All software megavendors offer at least one ESP product, and numerous small-to-midsize specialists also compete in this market. Cloud ESP platforms have lowered the cost of entry.
- Vendors are embedding ESP platforms into a wide variety of other software products, including industrial IoT platforms, stream data integration tools, unified real-time platforms (aka continuous intelligence platforms), insider threat detection tools and AI operations platforms.
- Vendors are adding highly productive development tools that enable faster ESP application development. Power users can build some kinds of ESP applications via low-code techniques and off-the-shelf templates.

Obstacles

- ESP platforms are overkill for many applications that process low volumes of streaming data (i.e., under 1,000 events per second), or that do not require fast response times (i.e., less than a minute). Conventional BI and analytics tools with data-at-rest architectures are appropriate for most stream analytics with these less-demanding requirements.
- Many architects and software engineers are still unfamiliar with the design techniques that enable ESP on data in motion. They are more familiar with processing data at rest in databases and other data stores, so they use those techniques by default unless business requirements force them to use ESP.
- Some streaming applications are better-implemented on unified real-time platforms that process both data in motion and data at rest. Some unified platforms use embedded open-source ESP platform products, while others get their ESP capabilities from custom internal code.

User Recommendations

- Use ESP platforms when conventional data-at-rest architectures cannot process high-volume streams fast enough to meet business requirements.
- Acquire ESP functionality through a SaaS offering, an IoT platform or an off-the-shelf application that has embedded ESP logic if a product that targets specific business requirements is available.
- Use vendor-supported closed-source platforms or open-core ESP products that mix open-source with closed-source extensions for applications that need enterprise-level support. Use free, community-supported, open-source ESP products if developers are familiar with open-source software, and license fees are more important than staff costs.
- Use ESP platforms or stream data integration tools to ingest, filter, enrich, transform and store event streams in a file or database for later use.
- Choose a unified real-time platform with embedded ESP capabilities over a plain ESP platform if the application uses both data at rest and data in motion.

Sample Vendors

Confluent; EsperTech; Google; Hazelcast; IBM; Microsoft; Oracle; SAS; Software AG; TIBCO Software

Gartner Recommended Reading

Market Guide for Event Stream Processing

5 Essential Practices for Real-Time Analytics

Create an Optimal IoT Architecture Using 5 Common Design Patterns

Adopt Stream Data Integration to Meet Your Real-Time Data Integration and Analytics Requirements

Open-Source NLP Toolkits

Analysis By: Adrian Lee

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Open-source natural language processing (NLP) toolkits enable end users and commercial companies to review, modify or design the source code and models for their own purposes, mostly free of charge. Open-source NLP toolkits address the problems of analyzing and processing large, unstructured natural language corpora by means of structured libraries to facilitate the support for applications that require NLP tools and tasks, such as text analytics or virtual assistants.

Why This Is Important

Open-source NLP toolkits are widely adopted and expected to grow in usage by strong market interest in generative AI outputs. Current releases enable more advanced NLP tasks that incorporate large language models to provide value by optimizing business processes and operations. Applicable use cases are in business intelligence (text analytics), customer service (improving customer satisfaction and increasing engagement) and employee support (productivity applications and knowledge base management).

Business Impact

Open-source NLP toolkits enable:

- Capabilities for summarization, translation, Q&A or search in use cases for text and sentiment analysis.
- Rapid prototyping of conversational agents with virtual assistants (VAs) as front ends for leveraging NLP.
- Combination with NLP toolkits and solutions as a test environment to advance NLP accuracy and outputs.
- Enterprises to gain control over the application to develop customized solutions.
- Conversational AI platform and other vendors with NLP elements to develop their solutions without costly ML investments.

Drivers

- Continued growth in the adoption of NLP capabilities and heightened interest in generative AI outputs is lowering the barrier to entry for enterprises building solutions with open-source NLP toolkits to integrate into their digital products.

- Enterprises need to customize their integrations of NLT with their applications and domain-specific use cases.

- There is a lack of availability of customization of custom-made and packaged NLT integrated development environment (IDE) solutions without high professional services fees.

- NLP toolkits are a mature NLP foundation with broad user communities that are widely accepted by enterprises and conversational AI providers to enable the most common modes of linguistic analysis to help machines understand the text.

- Core features of open-source NLP toolkits position them to be suitable, lower-cost platforms for prototyping and piloting conversational agents or natural language applications and appeals to new or smaller organizations.

- A significant proportion of open-source NLP toolkits come with some pretrained intent models, multiple languages and test suites to accelerate the project delivery for end users.

Obstacles

- NLP toolkits are often focused toward experimental research and projects combining commercial solutions. Prototypes may not make it into production or be scaled up for commercial use.
- Enterprises that build their own products leveraging NLP toolkits may find it challenging to keep pace with the rapid evolution of NLP as it requires custom data and in-house AI and ML skills that require mid-to-long-term funding.
- IT leaders typically consider managed service providers that fulfill domain-specific needs and prebuilt intent models to accelerate deployment.
- Limitations can exist where end users do not augment toolkits with suitable ML algorithms and language model optimization to improve the performance of the NLP output.
- Using NLP toolkits is strictly iterative and requires customization and integrations with enterprise applications to deliver business benefits.
- Differences exist between conversational AI providers and open-source platforms in ontologies, taxonomies and domain specificity to drive contextual natural language understanding.

User Recommendations

- Implement open-source NLP toolkits to build an internal NLT stack by starting with a suitable toolkit to match existing IT infrastructure or to incorporate into a sellable digital product.
- Possess skilled in-house resources of data scientists and AI technology engineers for NLP accuracy and model fine-tuning.
- Evaluate a long-term and more costly initial IT investment by using open-source NLP toolkits in order to benefit subsequently from lower total costs of ownership.

Sample Vendors

Amazon; Baidu; Google; Hugging Face; NLTK Project; OpenAI; PyTorch; Rasa Technologies; spaCy

Gartner Recommended Reading

Cool Vendors in Conversational and Natural Language Technology

Selecting Conversational AI Solutions for Chatbot and Virtual Assistant Initiatives

Emerging Tech Roundup: ChatGPT Hype Fuels Urgency for Advancing Conversational AI and Generative AI

Entering the Plateau

Full Life Cycle API Management

Analysis By: Shameen Pillai

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Early mainstream

Definition:

Full life cycle API management involves the planning, design, implementation, testing, publication, operation, consumption, versioning and retirement of APIs. API management tools help creating API ecosystems, publishing and operating APIs, and collecting analytics for monitoring and business value reporting. These capabilities are typically packaged as a combination of a developer portal, API gateway, API design, development and testing tools as well as policy management and analytics.

Why This Is Important

APIs are widely used as the primary choice to connect systems, applications and devices; integrate business partners, and build modular and composable software architectures. The use of APIs as digital products — monetized directly or indirectly — is also on the rise. Digital transformation initiatives have accelerated the need for creation, management, operations and security of APIs. They have also made full life cycle API management an essential foundational capability for every organization.

Business Impact

Full life cycle API management provides the framework and tools necessary to manage and govern APIs that are foundational elements of multiexperience applications, composable architectures and key enablers of digital transformations. It enables the creation of API products, which may be directly or indirectly monetized, while its security features serve to protect organizations from the business risk related to API breaches.

Drivers

- Organizations are facing an explosion of APIs, stemming from the need to connect systems, applications, devices and other businesses. The use of APIs in internal, external, B2B, private and public sharing of data is driving up the need to manage and govern APIs using full life cycle API management.

- APIs that package data, services and insights are increasingly being treated as products that are monetized and enable platform business models. Full life cycle API management provides the tooling to treat APIs as products.

- Digital transformation drives an increased use of APIs, which in turn increases the demand for full life cycle API management.

- Organizations looking for growth acceleration and business resilience are adopting API-based architectures to eliminate or modernize their legacy and monolithic applications.

- Developer mind share for APIs is growing. Newer approaches to event-based APIs, design innovations and modeling approaches — such as GraphQL — are driving interest in full life cycle API management, leading to more experimentation and growth.

- Cloud adoption and cloud-native architectural approaches to computing (including serverless computing) are increasing the use of APIs in software engineering architectures, especially in the context of microservices, service mesh and serverless.

- Greater awareness of and increasing significance to API security are driving organizations to take charge of discovering and managing APIs. This often starts with operational management of existing APIs.

- Regulated, industry-specific initiatives in the financial, insurance and healthcare industries continue to provide a steady demand. However, use of APIs is spreading across most industries (especially retail, technology and telecom, manufacturing), increasing the demand.

- Commoditization and widespread availability of API gateways as part of cloud services, security solutions and other bundled software applications are increasing the need for distributed API management that involves multiple gateways, including those from multiple vendors.

- Rising influence of generative AI and large language models in software engineering is likely to increase and reshape the need for APIs.

Obstacles

- A lack of commitment to adequate organizational governance processes hinders the adoption of full life cycle API management. This can be due to a lack of skills or know-how, or due to putting too much focus on bureaucratic approaches rather than federated and automated governance approaches.

- A lack of strategic focus on business value (quantifiable business growth or operational efficiency) and too much focus on technical use cases can disengage business users and sponsors. This is particularly apparent in cases where API programs fail to deliver the promised return on investment.
- Traditional, single-gateway approaches to API management no longer fit well with modern, distributed API management approaches.
- A partial or full set of embedded API management capabilities provided by vendors in other markets — such as application development, integration platforms, security solutions and B2B offerings — can partially meet the use cases for API management and shrink the market opportunities.

User Recommendations

- Use full life cycle API management to power your API strategy and address both technical and business requirements for APIs. Select offerings that can address both well beyond the first year.
- Treat APIs as products managed by API product managers in a federated API platform team. Adopt business metrics for API products.
- Choose a functionally broad API management solution that supports modern API trends, including microservices, multigateway and multicloud architectures. Ensure that the chosen solution covers the entire API life cycle.
- Use full life cycle API management to enable governance of all APIs, including third-party (private or public) APIs that you consume.
- Ask full life cycle API management vendors about their support for automation, especially for API validation, testing and DevOps integration. Also ask about support for low-footprint and third-party API gateways.
- Full life cycle API management solutions are suitable for implementing a single-vendor strategy as opposed to a best-fit tooling approach for different API life stages. Ensure the choice aligns with your organization’s goals.

Sample Vendors

Axway; Google; IBM; Kong; Microsoft; Salesforce (MuleSoft); Software AG

Gartner Recommended Reading

Magic Quadrant for Full Life Cycle API Management

The Evolving Role of the API Product Manager in Digital Product Management

How to Use KPIs to Measure the Business Value of APIs

API Security: What You Need to Do to Protect Your APIs

Reference Model for API Management Solutions

Software Composition Analysis

Analysis By: Dale Gardner

Benefit Rating: Transformational

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Software composition analysis (SCA) products are specialized application security tools that detect open-source software (OSS) and third-party components known to have security vulnerabilities, and identify potentially adverse licensing and supply chain risks. It is an essential element in strategies to ensure an organization's software supply chain includes secure and trusted components and, therefore, that the strategy aids in secure application development and assembly.

Why This Is Important

SCA promotes the use of OSS in application development by identifying known vulnerabilities, ensuring components are properly licensed, and advancing trust in the software supply chain. It is an essential activity, given the ubiquity of OSS in applications and the potential for significant risk.

Business Impact

- SCA must be considered a foundational element of application security testing to identify known vulnerabilities and potential supply chain risks in open-source packages and other artifacts.
- Its primary users are application development and security teams, tasked with ensuring the integrity of open-source components (and, less frequently, commercial software) in development.
- License assessments — once the primary use case for SCA — remain an important function for legal and sourcing groups.

Drivers

- SCA has advanced to mainstream usage due to increased usage of open source, and through its role in supporting software supply chain security. Supply chain risks — including the potential for malicious code, poorly supported packages, and other attacks — have emerged as a primary driver for SCA. Organizations are placing an increased focus on addressing these risks in response to high-profile incidents and expanding global regulatory and market mandates.

- An underlying driver for the growing importance of SCA is the ubiquitous use of OSS in application development. The prevalence of OSS, both as individual packages and within container images, has led to SCA becoming a key control for continuous integration/continuous delivery (CI/CD) pipelines and containerized environments.

- Repeated instances of high-impact vulnerabilities in OSS have become pervasive because of the underlying components' broad use across organizations. This, along with ever-present supply chain attacks, demonstrates the need to better understand the risks posed by OSS and commercial software packages.

- New requirements around the ability to address supply chain risks, and to enhance developer productivity when remediating issues, have prompted organizations to revisit their SCA tooling. More effective SCA tools now provide guidance on a preferred update version — balancing stability, remediation of flaws and potential adverse impacts on the functionality of existing code. In some cases, tools have begun to incorporate assessments of operational risk, in an effort to help prevent supply chain attacks. There are also varied levels of support for software bill of materials (SBOM) analysis and generation — another rapidly emerging requirement.

- Meeting compliance requirements and ethical and legal standards is a growing concern for organizations, and SCA technology can help ensure developers are meeting these requirements. Tools help reduce the likelihood of unwanted or unapproved code that creates risks to an organization's intellectual property.

Obstacles

- Tools can report a large number of issues, creating friction with developers. The ability to confirm usage of affected code or its exploitability has become a required feature, but is not universally available.
- Although SCA tools are often seen as a complete solution to the prevention of supply chain attacks, in practice, they offer only a partial defense. With few exceptions, they don't test code to identify new issues, nor do they automatically remediate issues. To establish a more complete solution, users will need to augment SCA itself, and consider other approaches to address tasks such as protecting the build process and artifacts.
- Commercial software libraries are often outside the scope of SCA, but pose many of the same risks, leaving a protection gap.
- Legal and procurement teams are traditional users of SCA tools, although products are increasingly optimized for use by development teams. Organizations must evaluate the ability of tools to support multiple use cases when evaluating solutions.

User Recommendations

- Implement SCA technologies as a key ingredient of every software security program. SCA tools can be acquired from open-source projects, as stand-alone solutions or as a component of a dedicated AST tool.
- Ensure legal experts analyze SCA license warnings to address the legal issues stemming from IP ownership or license obligations.
- Proactively use SCA tools on a regular basis to audit OSS repositories to ensure the software used by the enterprise meets organizational standards. Consider establishing a trusted repository containing approved code.
- Incorporate SCA tools within DevSecOps workflows, where scanning can be automated as part of the rapid development processes to identify issues and track usage of components.
- Use SCA tools in conjunction with a formal corporate IP strategy that has established clear responsibility across the company.
- Favor SCA tools with strong supply chain risk management support, including the ability to consume and generate SBOMs.

Sample Vendors

Checkmarx; GitHub; Mend.io; Phylum; Revenera; ReversingLabs; Snyk; Sonatype; Synopsys; Veracode

Gartner Recommended Reading

Market Guide for Software Composition Analysis

Critical Capabilities for Application Security Testing

Magic Quadrant for Application Security Testing

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

Appendixes

See the previous Hype Cycle: Hype Cycle for Open-Source Software, 2022

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)

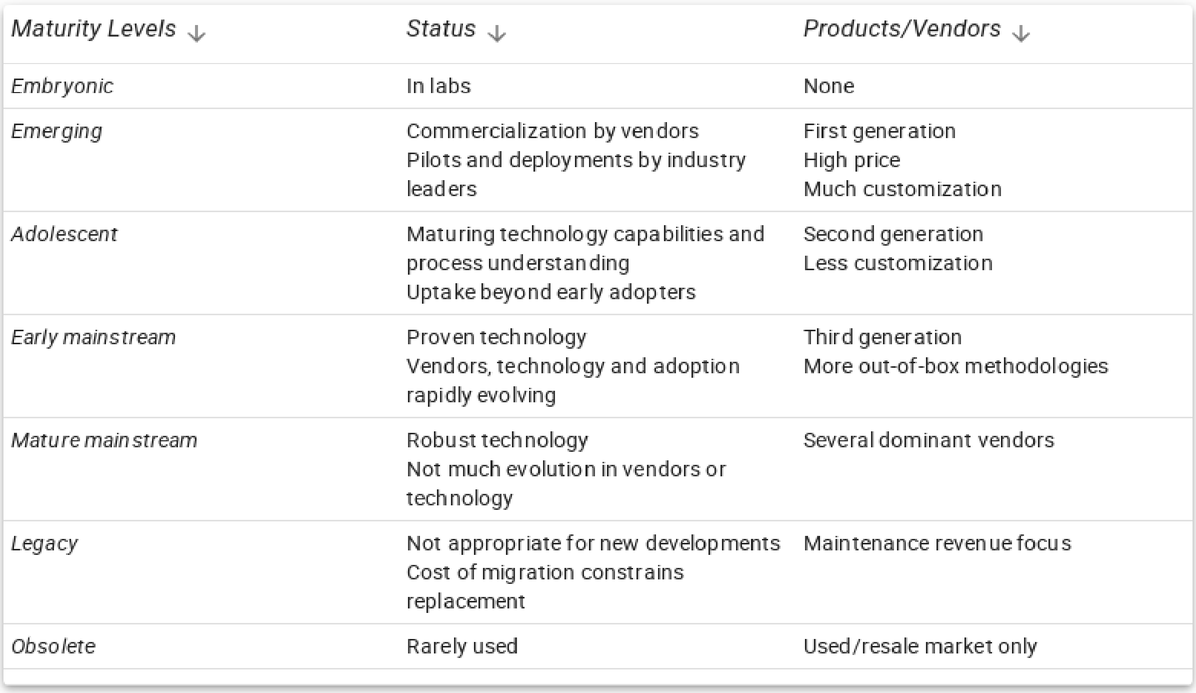
Table 3: Benefit Ratings

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)

Evidence

[1] About the GNU Operating System — GNU History.

[2] Government's Role in Promoting Open Source Software, Center for Strategic and International Studies.

[3] Octoverse 2022: The State of Open Source Software, GitHub.

[4] 2023 Open Source Security and Risk Analysis (OSSRA) report, Synopsys.

Document Revision History

Hype Cycle for Open-Source Software, 2022 - 20 July 2022

Hype Cycle for Open-Source Software, 2021 - 26 July 2021

Hype Cycle for Open-Source Software, 2020 - 13 July 2020

Hype Cycle for Open-Source Software, 2019 - 6 August 2019

Hype Cycle for Open-Source Software, 2018 - 18 October 2018

Hype Cycle for Open-Source Software, 2017 - 12 October 2017

Hype Cycle for Open-Source Software, 2016 - 11 July 2016

Hype Cycle for Open-Source Software, 2014 - 30 July 2014

Recommended by the Authors

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

A CTO's Guide to Open-Source Software: Answering the Top 10 FAQs

3 Steps for Assessing an Open-Source Software Project

Toolkit: Gartner's Open-Source Software Assessment Framework — 1.0

Tool: OSS Governance Policy Template

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

Four Steps to Adopt Open-Source Software as Part of Your Test Automation Stack

How to Evaluate Open-Source Integration Software

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Table 1: Priority Matrix for Open-Source Software, 2023

Benefit	*Years to Mainstream Adoption*			
↓	***Less Than 2 Years*** ↓	***2 - 5 Years*** ↓	***5 - 10 Years*** ↓	***More Than 10 Years*** ↓
Transformational	Event Stream Processing Software Composition Analysis	AI-Augmented Software Engineering Layer 1 Blockchains Machine Learning Platform Engineering	Responsible AI Web 2.5 Web3 WebAssembly (Wasm)	
High	Full Life Cycle API Management Open-Source NLP Toolkits	Cloud Event Brokers Container Management Internal Developer Portal Open-Source Program Office Software Bill of Materials	Cloud Portability Open-Source Software Assessment Practices	
Moderate	Cloud-Native Architecture		Event-Driven APIs Innersource	
Low		Service Mesh		

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)