Gartner

Compute Evolution: VMs, Containers, Serverless — Which to Use When?

Refreshed 22 March 2022, Published 1 June 2021 - ID G00745745 - 13 min read

FOUNDATIONAL This research is reviewed periodically for accuracy.

By Analyst(s): Arun Chandrasekaran

Initiatives: Digital Products and Services

The rise of cloud computing has been a huge catalyst for compute platform innovations. Enterprise architecture and technology innovation leaders, including CTOs, should use this decision framework to align their use cases with the appropriate compute technologies to benefit from these innovations.

Additional Perspectives

- Summary Translation: Compute Evolution: VMs, Containers, Serverless — Which to Use When?
(07 December 2021)

Overview

Key Findings

- The agility, elasticity and automation needs of the next wave of digital applications are fundamentally changing how, where and by whom compute is being consumed and managed, with developer preferences becoming a key deciding factor.

- Technology innovation leaders are keen to harness the potential of cloud-native technologies to build new digital services and modernize legacy applications; however, ecosystem immaturity, organizational friction and lack of skills continue to impede widespread adoption.

- Each computing abstraction offers a set of trade-offs that needs to be carefully evaluated against the organization's application architecture requirements, cloud strategy and DevOps maturity.

- There is growing interest in hybrid deployments, with software vendors and cloud providers increasingly competing to build a hybrid stack to enable workload portability across the different operating environments and computing abstractions.

Recommendations

Enterprise architecture (EA) and technology innovation leaders driving business transformation through technology innovation should:

- Analyze and understand the unique use cases and consumption models for each computing abstraction to ensure strategic alignment.
- Pursue an application-centric approach by aligning the appropriate compute technology based on the application architecture, organizational skills and other decision factors outlined in this research.
- Prioritize platform providers that support compute abstractions across multiple environments (cloud, data center and edge) with a consistent management, governance and support model.
- Assess the emerging trends around containers and serverless, and create an action plan on how and when to respond to them.

Strategic Planning Assumptions

By 2026, more than 90% of global organizations will be running containerized applications in production, which is a significant increase from less than 40% today.

By 2026, more than 50% of global enterprises will have deployed serverless functions as a service (FaaS), up from less than 25% today.

By 2026, 80% of artificial intelligence workloads will use application containers or be built using a serverless programming model necessitating a DevOps culture.

Introduction

As Marc Andreessen wrote, “software is eating the world,” and modern digital businesses are leveraging software to analyze data and build applications to create sustainable differentiation. [1] Enterprise architecture and technology innovation leaders are being tasked with accelerating software velocity, and they are doing so by adopting cloud-native services and new application architectures, such as microservices. The rise of these new application architectures has resulted in innovations at the compute layer, with abstractions such as application containers and serverless functions emerging in the past few years. This has allowed further virtualization of the compute layer beyond server virtualization, allowing it to be more application-oriented, agile, scalable and automated. The evolution and basic differences between these compute technologies is illustrated in Figure 1.

The following section provides an overview on the key compute technologies, their use-case affinity and their evolving consumption models.

Analysis

Analyze and Evaluate the Use Cases and Deployment Models for Each Abstraction

Virtual Machines

What are virtual machines?

Virtual machines (VMs) abstract server hardware capabilities and imitate dedicated hardware for an application. This allows multiple different OSs to share servers in a seamless manner and to provide better hardware utilization. Everything necessary to run an app is contained within the virtual machine. VMs provide benefits in terms of better availability, manageability and security, particularly in comparison to physical servers.

When should we use VMs?

Virtual machines are used when:

- Applications are monolithic, large and static with stringent data persistence needs
- A variety of operating systems and version releases need to be run and managed
- Complete isolation from the host OS is required

What are the different ways to consume VMs?

Server virtualization is a mature market today with several deployment models on-premises and in public clouds. The most common deployment models are:

- **Hypervisor-based virtualization (stand-alone):** Key vendors that provide hypervisor-based virtualization products include VMware, Microsoft, Red Hat and Oracle. The key decision factors that guide the choice of virtualization solutions include degree of automation, resiliency capabilities, ecosystem integration, breadth of hardware support and total cost of ownership (TCO).
- **Hyperconverged and software-defined infrastructure:** Vendors such as VMware and Microsoft offer suites of integrated hyperconverged and disaggregated software capabilities. Vendors such as Nutanix and Huawei have adopted open-source hypervisors and integrated them with their proprietary storage software, management plane and other tooling. Greenfield deployments, server/storage refreshes as well as hypervisor licensing renewals may be triggers to look at hyperconvergence.
- **Public cloud infrastructure as a service (IaaS):** Virtual machines enable rapid deployment of capacity, but public cloud IaaS took that to the next level, enabling self-service capacity acquisition without significant capital expenditure (capex). In addition, the elasticity and automation tools have made this consumption model highly appealing to enterprises.

Containers

What are containers?

Containers virtualize an operating system, allowing multiple instances of an OS user space to share a single OS kernel. A container image includes libraries and other runtime dependencies required to run an application. This packaging approach allows containers to be highly standardized, which enables applications to run in a consistent manner across the software development life cycle and hybrid environments. The emergence of Kubernetes as the de facto standard for scheduling and orchestrating containers and the slew of supporting OSS projects have led to further growth in deployment of containerized apps.

When should we use containers?

Containers are used when:

- Applications are lightweight, are distributed, are ephemeral and need rapid deployment to production.
- Software environments are highly dynamic. They may be agile development, quality assurance environments or used to foster DevOps practices such as continuous integration/continuous delivery (CI/CD).
- Provisioning, scaling and recovery time are critical (measured in seconds, not minutes, like VMs).

What are the different ways to consume containers?

The consumption of containers via a container as a service (CaaS) and PaaS has been growing in the past few years due to the broader market adoption of Kubernetes.

Table 1 describes the most common ways that organizations consume containers and Kubernetes.

Table 1: Container and Kubernetes Consumption Models

(Enlarged table in Appendix)

Deployment Model	Description	Examples
Do-It-Yourself With Upstream Version	A deployment model where users build and manage Kubernetes clusters on-premises and/or off-premises by using upstream open-source projects.	Cloud Native Computing Foundation (CNCF) and other supported community OSS projects
Public Cloud Container Services	Cloud IaaS providers offer a managed service to deploy containers and Kubernetes clusters. In this model, users don't have to build, manage or upgrade Kubernetes clusters themselves.	Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS), Microsoft Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE), IBM Cloud Kubernetes Service, Oracle Container Engine for Kubernetes, and Alibaba Container Service for Kubernetes (ACK)
Hybrid Solutions From Cloud Providers	The cloud providers are also extending these services to customer data centers and edge.	ECS/EKS on AWS Outposts, AKS Engine on Azure Stack, AzureArc, Google Anthos, IBM Cloud Satellite, ACK on Apsara Stack
Serverless Containers	Serverless container services eliminate the server management task and enable better on-demand consumption.	AWS Fargate, Azure Container Instances, GCP Cloud Run
Third-Party Managed Kubernetes Platform	In this model, third-party vendors provide SaaS-based operational management services for container deployments on-premises, in the public cloud or both.	Giant Swarm, Platform9 Managed Kubernetes (PMK), Rafay's Kubernetes Management Cloud (KMC), Red Hat OpenShift Dedicated
Self-Managed Container as a Service (CaaS) Platform	In this model, users build and manage containers and Kubernetes clusters on-premises and/or in a public cloud IaaS by using a packaged and supported solution. The CaaS software combines container runtime and Kubernetes distribution with other management capabilities, such as security, monitoring and storage/networking integration.	D2iQ Konvoy, Mirantis Kubernetes Engine, Rancher, Red Hat OpenShift Kubernetes Engine VMware Tanzu Basic and Standard
Self-Managed Container PaaS	In addition to these capabilities, the PaaS solutions include DevOps toolchains and/or application software.	Red Hat OpenShift Container Platform, VMware Tanzu Advanced

Source: Gartner (July 2021)

Serverless Computing

What is serverless computing?

Serverless computing is a way to build and/or run applications and services without having to manage infrastructure. The most prominent manifestation of serverless computing is serverless functions or FaaS. With FaaS, application code is packaged into fine-grained units called "functions," with the execution of these functions delivered as a managed service. Essentially, FaaS abstracts away the runtime environment, enabling developers to focus more on application design and configuration than on infrastructure-related provisioning and management.

When should we use serverless functions?

Serverless functions can be used for:

- Applications with unpredictable scaling needs that are ephemeral in nature
- Applications where fine-grained operational control of the runtime environment isn't required
- Batch computing tasks like multimedia (image enhancement or video transcoding), event-driven services, automating cloud operations and service integration.

What are the different ways to consume serverless functions?

The primary consumption model for serverless functions is in public cloud IaaS, with most leading public cloud providers offering a managed service. A number of open-source frameworks are evolving for on-premises and hybrid deployments, although their adoption and maturity tend to be low. More recently, there have been content delivery network (CDN) providers offering low latency serverless functions on the edge, integrating it with their edge networks. Table 2 is a list of key serverless products/projects.

Table 2: Key Serverless Products/Projects

(Enlarged table in Appendix)

Deployment Model	Examples
Public Cloud	Alibaba Cloud Function Compute AWS Lambda Azure Functions Google Cloud Functions IBM Cloud Functions Oracle Functions
Private IT	Apache OpenWhisk OpenFaaS Knative [2] Kubeless Platform9 Fission Nimbella
Edge Computing	Lambda@Edge Cloudflare workers Fastly Akamai

Source: Gartner (July 2021)

Pursue an Application-Centric Approach

Given the technical differences and use-case affinity of each of these abstractions, they will continue to coexist for a long period of time. Infrastructure serves the needs of an application. The choice of appropriate computing abstraction should ultimately be defined by the application architecture and by well-defined metrics that analyze the pros and cons of each of the approaches — VMs, containers and serverless FaaS. Table 3 provides a decision framework of factors to consider when choosing a computing abstraction.

Table 3: Decision Framework for Selecting a Computing Abstraction

(Enlarged table in Appendix)

● = High ◐ = Medium ◔ = Low

Selection Factors	Virtual Machines	Containers	Serverless FaaS
Application Architecture	Well-suited for monolithic three-tier applications and porting legacy applications to the cloud.	Well-suited for distributed, cloud-native applications and services. May be used for porting cloud-friendly monolithic applications.	Well-suited for new development of back-end services supporting web and mobile apps, as well as event-driven functions and processes.
Third-Party Independent Software Vendor (ISV) Support	● High degree of support for third-party ISVs. Most commercial off-the-shelf applications can be run on VMs.	◐ Third-party ISV support is nascent, although it continues to grow.	◔ Low degree of support for third-party apps.
Statefulness	● Strong support for data integrity, performance and recoverability with broad integration of storage products.	◐ Support for stateful applications is emerging through container storage interface (CSI), persistent volumes (PVs) and stateful sets.	◔ The primary programming model for serverless is mainly limited to stateless functions.
Hybrid Architecture	◐ While VM-based hybrid architectures can be enabled, they are inherently more complex.	● Standard runtime and orchestration provide a set of common abstractions across private and public cloud environments.	◔ Strong degree of affinity to their native cloud environments.
Operational Control	● High degree of operational control can be exercised over the runtime environment, although it comes with a lot of operational overhead.	◐ High degree of choice and flexibility, although that increases operational overhead and initial setup time.	◔ FaaS provides low degree of choice and flexibility, although that results in faster setup time with low operational overhead.
[illegible]	●	◐	◔

Assess the Future Direction of Containers and Serverless

Open-source communities and commercial vendors are building capabilities to address some of the limitations of these compute abstractions with a view of enhancing their production readiness across a broad set of use cases. These directions need to be viewed as a signpost for the future rather than as a statement on their current maturity. The key emerging trends across containers and serverless are summarized in Table 4 and Table 5, respectively.

Table 4: Containers — Emerging Trends

(Enlarged table in Appendix)

Trend	Highlights
Edge Computing	■ Interest in containers on the edge is driven by its lightweight, service granularity and immutable principles. ■ Available products include AWS IoT Greengrass, Google Anthos, Microsoft Azure IoT Edge, Canonical MicroK8s and Rancher K3s.
Multicloud Management	■ Platform engineering teams are focused on creating a consistent development experience by standardizing on developer workflow tools, DevOps toolchains and Kubernetes runtime environments across hybrid and multicloud. ■ Vendors are addressing this need to expand their K8S deployments across on-premises and public cloud environments as well as build a common management plane. Rancher, Tanzu Mission Control, Red Hat Advanced Cluster Management, Azure Arc, Anthos and AWS are taking steps in this direction.
Third-Party ISV Support	■ High degree of support for containers among open-source communities such as Spark, Kafka, Redis and MongoDB. ■ Marketplaces being built by cloud providers and container platform software vendors such as Red Hat and VMware could be key in bringing more standardization and support from commercial-off-the-shelf vendors.
Convergence With VMs and Bare Metal	■ Projects such as KubeVirt and micro VMs such as Kata Containers and Firecracker promise better manageability and efficiency between VMs and containers in the future. ■ Growing interest in bare-metal deployments is driven by the need for cost-efficiencies and better performance for use cases such as edge and workloads such as big data analytics and AI.
Technology Consolidation	■ Large Kubernetes vendors will consolidate more security, monitoring, CI/CD, GitOps, and serverless functions capabilities onto the core platform as well as embed newer technologies such as service mesh as part of a platform play. ■ This change will inevitably lead to further consolidation in OSS projects as well as startup acquisitions.

Source: Gartner (July 2021)

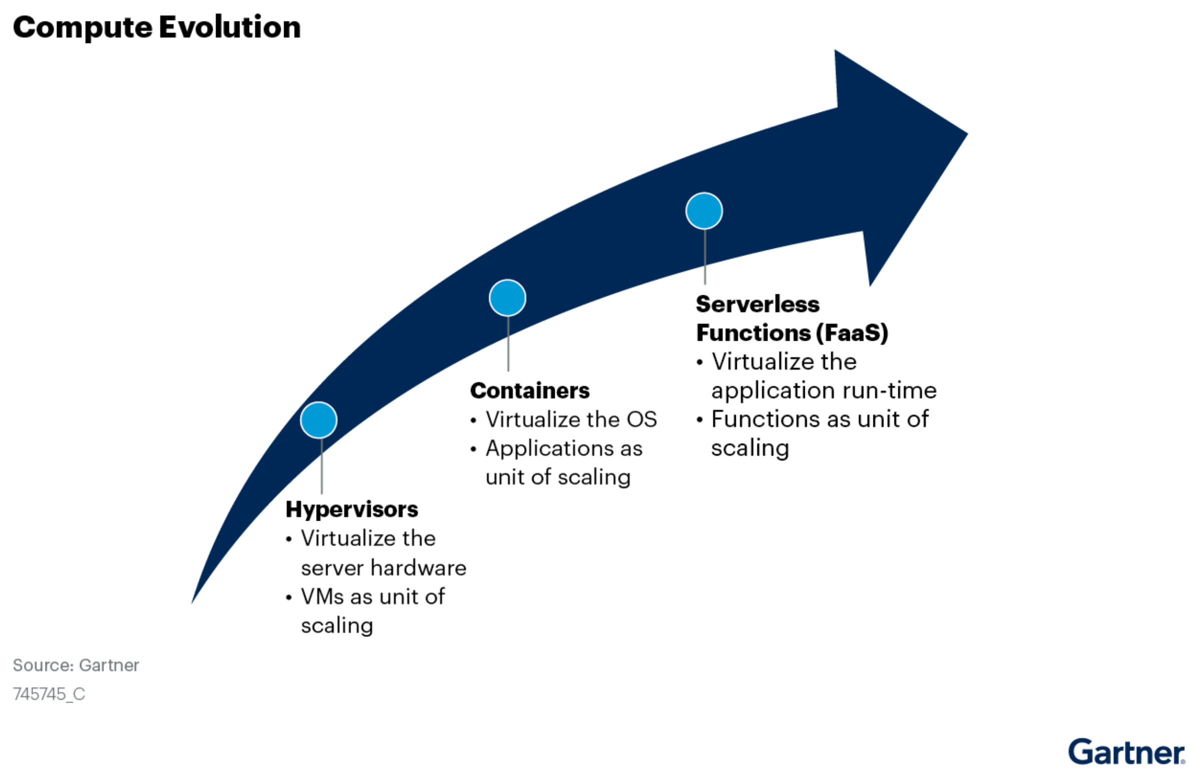
Table 5: Serverless — Emerging Trends

(Enlarged table in Appendix)

Trend	Highlights
Tooling Maturity	■ We expect wider language support and tooling to improve in terms of better security, monitoring and application debugging. ■ Re-creating serverless environments locally is hard, and better local testing, development and well-integrated CI/CD tooling is expected to evolve.
Edge Computing	CDN providers such as CloudFlare, Fastly and Akamai are reinventing the serverless architecture at the edge with an emphasis on performance and better support for stateful applications.
Open Standards	■ A key open-source effort for enabling self-managed fPaaS platforms is Knative, which was originally created by Google and is now being adopted by several other vendors despite some unhappiness about its governance. ■ Another open effort is the CloudEvents specification for describing event data, created by CNCF, to enable event interoperability across services.
Stateful Support	Today, serverless applications are mostly stateless, but in the future, we expect redesign of serverless infrastructure to natively support efficient, consistent and fault-tolerant state management. We are starting to see early evidence of this with vendors such as CloudFlare and Nimbella offering embedded data stores with their serverless FaaS.
WebAssembly as Runtime	While WebAssembly was primarily designed as a binary instruction format to enable the use of compiled code in the browser, its support has been growing on serverless platforms with the development of the WebAssembly System Interface (WASI) standard that promises better portability and security.

Source: Gartner (July 2021)

Figure 1: Compute Evolution



Evidence

This research is based on inquiries with hundreds of clients. In addition, detailed end-user interviews were conducted at AWS re:Invent and CloudNativeCon in 2019 and 2020, and briefings with several vendors represented in this research.

Serverless Architectures, martinFowler.com.

The Cloud Native Computing Foundation (CNCF)

[1] Why Software Is Eating the World, Andreessen Horowitz.

[2] Knative is an open-source project that provides eventing and serving capabilities for enabling FaaS.

Document Revision History

Evolution of Virtualization: VMs, Containers, Serverless — Which to Use When? - 26 September 2019

Recommended by the Author

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

A CIO’s Guide to Serverless Computing

CTO’s Guide to Containers and Kubernetes — Answering the Top 10 FAQs

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

The Future of Cloud in 2025: From Technology to Innovation

Market Guide for Server Virtualization

Top 10 Trends in PaaS and Platform Innovation, 2020

Table 1: Container and Kubernetes Consumption Models

Deployment Model	Description	Examples
Do-It-Yourself With Upstream Version	A deployment model where users build and manage Kubernetes clusters on-premises and/or off-premises by using upstream open-source projects.	Cloud Native Computing Foundation (CNCF) and other supported community OSS projects
Public Cloud Container Services	Cloud IaaS providers offer a managed service to deploy containers and Kubernetes clusters. In this model, users don't have to build, manage or upgrade Kubernetes clusters themselves.	Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS), Microsoft Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE), IBM Cloud Kubernetes Service, Oracle Container Engine for Kubernetes, and Alibaba Container Service for Kubernetes (ACK)
Hybrid Solutions From Cloud Providers	The cloud providers are also extending these services to customer data centers and edge.	ECS/EKS on AWS Outposts, AKS Engine on Azure Stack, AzureArc, Google Anthos, IBM Cloud Satellite, ACK on Apsara Stack
Serverless Containers	Serverless container services eliminate the server management task and enable better on-demand consumption.	AWS Fargate, Azure Container Instances, GCP Cloud Run
Third-Party Managed Kubernetes Platform	In this model, third-party vendors provide SaaS-based operational management services for container deployments on-premises, in the public cloud or both.	Giant Swarm, Platform9 Managed Kubernetes (PMK), Rafay's Kubernetes Management Cloud (KMC), Red Hat OpenShift Dedicated

Self-Managed Container as a Service (CaaS) Platform	In this model, users build and manage containers and Kubernetes clusters on-premises and/or in a public cloud IaaS by using a packaged and supported solution. The CaaS software combines container runtime and Kubernetes distribution with other management capabilities, such as security, monitoring and storage/networking integration.	D2iQ Konvoy, Mirantis Kubernetes Engine, Rancher, Red Hat OpenShift Kubernetes Engine VMware Tanzu Basic and Standard
Self-Managed Container PaaS	In addition to these capabilities, the PaaS solutions include DevOps toolchains and/or application software.	Red Hat OpenShift Container Platform, VMware Tanzu Advanced

Source: Gartner (July 2021)

Table 2: Key Serverless Products/Projects

Deployment Model	Examples
Public Cloud	■ Alibaba Cloud Function Compute ■ AWS Lambda ■ Azure Functions ■ Google Cloud Functions ■ IBM Cloud Functions ■ Oracle Functions
Private IT	■ Apache OpenWhisk ■ OpenFaaS ■ Knative [2] ■ Kubeless ■ Platform9 Fission ■ Nimbella
Edge Computing	■ Lambda@Edge ■ Cloudflare workers

■ Fastly

Source: Gartner (July 2021)

Table 3: Decision Framework for Selecting a Computing Abstraction

● = High ◐ = Medium ◔ = Low

Selection Factors	Virtual Machines	Containers	Serverless FaaS
Application Architecture	Well-suited for monolithic three-tier applications and porting legacy applications to the cloud.	Well-suited for distributed, cloud-native applications and services. May be used for porting cloud-friendly monolithic applications.	Well-suited for new development of back-end services supporting web and mobile apps, as well as event-driven functions and processes.
Third-Party Independent Software Vendor (ISV) Support	● High degree of support for third-party ISVs. Most commercial off-the-shelf applications can be run on VMs.	◐ Third-party ISV support is nascent, although it continues to grow.	◔ Low degree of support for third-party apps.
Statefulness	● Strong support for data integrity, performance and recoverability with broad integration of storage products.	◐ Support for stateful applications is emerging through container storage interface (CSI), persistent volumes (PVs) and stateful sets.	◔ The primary programming model for serverless is mainly limited to stateless functions.
Hybrid Architecture	◐ While VM-based hybrid architectures can be enabled, they are inherently	● Standard runtime and orchestration provide a set of common	◔ Strong degree of affinity to their native cloud environments.

	more complex.	abstractions across private and public cloud environments.	
Operational Control	● High degree of operational control can be exercised over the runtime environment, although it comes with a lot of operational overhead.	◐ High degree of choice and flexibility, although that increases operational overhead and initial setup time.	◔ FaaS provides low degree of choice and flexibility, although that results in faster setup time with low operational overhead.
Operational Overhead	●	◐	◔
Workload Longevity	●	◐	◔
Skills Availability	●	◐	◔

Source: Gartner (July 2021)

Table 4: Containers — Emerging Trends

Trend	Highlights
Edge Computing	■ Interest in containers on the edge is driven by its lightweight, service granularity and immutable principles. ■ Available products include AWS IoT Greengrass, Google Anthos, Microsoft Azure IoT Edge, Canonical MicroK8s and Rancher K3s.
Multicloud Management	■ Platform engineering teams are focused on creating a consistent development experience by standardizing on developer workflow tools, DevOps toolchains and Kubernetes runtime environments across hybrid and multicloud. ■ Vendors are addressing this need to expand their K8S deployments across on-premises and public cloud environments as well as build a common management plane. Rancher, Tanzu Mission Control, Red Hat Advanced Cluster Management, Azure Arc, Anthos and AWS are taking steps in this direction.
Third-Party ISV Support	■ High degree of support for containers among open-source communities such as Spark, Kafka, Redis and MongoDB. ■ Marketplaces being built by cloud providers and container platform software vendors such as Red Hat and VMware could be key in bringing more standardization and support from commercial-off-the-shelf vendors.

Convergence With VMs and Bare Metal	- Projects such as KubeVirt and micro VMs such as Kata Containers and Firecracker promise better manageability and efficiency between VMs and containers in the future. - Growing interest in bare-metal deployments is driven by the need for cost-efficiencies and better performance for use cases such as edge and workloads such as big data analytics and AI.
Technology Consolidation	- Large Kubernetes vendors will consolidate more security, monitoring, CI/CD, GitOps, and serverless functions capabilities onto the core platform as well as embed newer technologies such as service mesh as part of a platform play. - This change will inevitably lead to further consolidation in OSS projects as well as startup acquisitions.

Source: Gartner (July 2021)

Table 5: Serverless — Emerging Trends

Trend	Highlights
Tooling Maturity	■ We expect wider language support and tooling to improve in terms of better security, monitoring and application debugging. ■ Re-creating serverless environments locally is hard, and better local testing, development and well-integrated CI/CD tooling is expected to evolve.
Edge Computing	CDN providers such as CloudFlare, Fastly and Akamai are reinventing the serverless architecture at the edge with an emphasis on performance and better support for stateful applications.
Open Standards	■ A key open-source effort for enabling self-managed fPaaS platforms is Knative, which was originally created by Google and is now being adopted by several other vendors despite some unhappiness about its governance. ■ Another open effort is the CloudEvents specification for describing event data, created by CNCF, to enable event interoperability across services.
Stateful Support	Today, serverless applications are mostly stateless, but in the future, we expect redesign of serverless infrastructure to natively support efficient, consistent and fault-tolerant state management. We are starting to see early evidence of this with vendors such as CloudFlare and Nimbella offering embedded data stores with their serverless FaaS.

WebAssembly as Runtime	While WebAssembly was primarily designed as a binary instruction format to enable the use of compiled code in the browser, its support has been growing on serverless platforms with the development of the WebAssembly System Interface (WASI) standard that promises better portability and security.

Source: Gartner (July 2021)