Serverless computing is a key technology that is redefining the way we build, consume and integrate cloud-native applications. CIOs can enable agility, elasticity and cost-effectiveness for next-generation applications by using it for appropriate use cases.

Key Findings
- There is still market confusion on what serverless computing is and its applicability for enterprise use cases.
- Developer productivity, platform elasticity and cost savings are the primary drivers for the adoption of serverless computing.
- Although the tooling to build, deploy, secure and operate serverless environments and applications is improving, the most mature offerings still lead to considerable vendor lock-in.
- Using serverless computing at scale requires organizational maturity and needs significant know-how in the areas of software architecture, security and operations management.

Recommendations
As a CIO building a cloud computing strategy, you should:
- Focus on “greenfield” use cases for serverless functions. Start with event-driven applications that require rapid scalability and are inherently “bursty” to maximize the benefits of the pricing model.
- Encourage a product ownership mindset and DevOps approach for serverless computing, since there should no longer be a line between cloud operations engineers and developers.
- Prioritize serverless security early in the adoption cycle through a combination of process, tooling and culture changes, starting in development and extending into production.
- Consume native services from your cloud provider to fully benefit from a serverless architecture, but run a thorough proof of concept (POC) to validate your assumptions on application scalability, performance and cost of ownership.
Strategic Planning Assumption

More than 50% of global enterprises will have deployed serverless function platform as a service (fPaaS) by 2025, up from less than 20% today.

Analysis

Serverless computing is a relatively new computational model that is quickly gaining adoption in public cloud environments. However, as serverless technology is new and fundamentally disruptive, there is considerable ambiguity about what it is and what it can do. To address this, Gartner has developed the following list of questions and answers to provide CIOs with a comprehensive introduction to serverless technology. With this research, we aim to define “serverless,” dispel myths about it, compare it to other similar technologies, outline the competitive landscape, offer best practices, and demonstrate some use cases that are already generating business and technological dividends.

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 fPaaS. With fPaaS, application code is packaged into fine-grained units called “functions,” with the execution of these functions delivered as a managed service. Essentially, fPaaS abstracts away the runtime environment, enabling developers to focus more on application design and configuration than on infrastructure-related provisioning and management. Thus, from a user perspective, serverless functions allow developers to build and run applications and services without thinking about servers. Examples of serverless fPaaS include Amazon Web Services (AWS) Lambda, Azure Functions, Google Cloud Functions and IBM Cloud Functions.

At its foundational level, serverless functions provide the following key capabilities:

- The platform runs code residing as functions without the need for the user to explicitly provision or manage infrastructure (servers, virtual machines [VMs], containers, etc.).
- The platform automatically provisions and scales the runtime environment, including all the necessary underlying resources (specifically the compute, storage, networking and language execution environment) required to execute many concurrent function instances.
- The platform offers additional capabilities for test and dev and service assurance purposes, such as monitoring, logging, tracing and debugging.

In the past few years, serverless computing as a term has evolved to include much more than fPaaS. Currently, it not only refers to a programming model such as fPaaS, but also to an operational model where all provisioning, scaling, monitoring and configuration of the compute infrastructure are delegated to the platform. Examples of such services include AWS Fargate, Amazon Simple Queue Service (SQS), Amazon Athena, Azure Container Instances (ACI) and Google Cloud Run, to name a few. Hence, fPaaS is no longer the only form of serverless platform services. However, most discussions in this research on serverless computing will focus on fPaaS (aka
serverless functions), which is the most popular manifestation today due to its ability to shape how future applications are built within the enterprise.

What Are the Benefits and Limitations of Serverless fPaaS?

The key benefits of serverless functions are:

- **Operational simplicity** — Most fPaaS leverages containers and VMs in the underlying architecture. However, by obviating the need for infrastructure setup, configuration, provisioning and management, serverless computing architectures have lower operational overheads when compared to those in which developers target the VMs or containers directly.

- **“Built-in” scalability** — In serverless functions, infrastructure scaling is automated and elastic, which makes it very appealing for unpredictable, spiky workloads. Hence, application scalability is most often limited by poor application design rather than any inherent limitation in the underlying infrastructure.

- **Cost-efficiency** — In public cloud-based serverless environments, you only pay for infrastructure resources when the application code is running, which exemplifies the “pay as you go” model of the cloud. Cost is a direct function of application design and code efficiency arising from best practices from the use of fPaaS.

- **Developer productivity and business agility** — Serverless architectures allow developers to focus on what they should be doing — writing code and focusing on application design — and abstracts away most infrastructure aspects. It also enables business agility, where the time to market for new digital projects can be significantly shortened while simultaneously allowing for rapid experimentation.

These benefits must be balanced with the following limitations:

- **Architectural limitations** — While serverless architecture delivers several benefits, it imposes some unexpected limits on the execution environment. Some are inherent, like cold starts (initialization latency). Others are artificial, such as the limits on function runtime, to ensure developers use the platform as intended. The lack of server-side state support (variables that persist in memory from function call to call), while promoting a best practice for modern software architecture, may be less suitable for several workloads than using containers or VMs.

- **Application affinity** — With fPaaS, application logic must be packaged as discrete functions, which are executed when triggered by events. This means existing applications must be significantly refactored to fit this packaging model or new applications need to be written to fit these patterns. This is well-suited to design patterns such as microservices architecture, but could just as easily lead to code sprawl, in which an app becomes a large set of hard-to-manage functions.

- **Skills gap** — Serverless programming, such as with fPaaS, requires a major shift in application architecture skills and best practices. These skills do not exist in abundance in the market, so teams typically pick up the knowledge as part of adopting fPaaS. Early attempts at building serverless applications usually see developers making many mistakes as they gain experience.
with the new model. Although this may eventually lead to success, it may disillusion some early in the adoption phase and delay delivery of projects.

- **Vendor lock-in** — The leading fPaaS implementations are proprietary to a specific cloud provider, where organizations take advantage of native integration and tooling. While this is beneficial for shortening time to market and keeping things simple, if the application has to move from one cloud platform to another, then it will have to be significantly reengineered.

- **Low degree of control** — The managed service model and runtime virtualization of serverless technologies like fPaaS bestow huge benefits, but at the cost of little to no control of the service. The environment is a “black box” that must be used as-is.

How Is Serverless fPaaS Different From Other Virtualization Technologies?

The fundamental differences between VMs, containers and serverless functions are illustrated in Figure 1. The figure shows the architectural layer that each approach virtualizes and how compute components are scaled in those respective environments.

*Figure 1. Comparing VMs, Containers and Serverless Functions*

<table>
<thead>
<tr>
<th>Comparing VMs, Containers, and Serverless Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Evolution:</strong> Hypervisors</td>
</tr>
<tr>
<td>Virtualizes: The hardware</td>
</tr>
<tr>
<td>Unit of Scaling: VMs</td>
</tr>
</tbody>
</table>

VMs, containers and serverless fPaaS will all be relevant for CIOs in the foreseeable future. Table 1 gives advice on the suitability of each across key selection factors:
### Table 1. Decision Matrix for Virtualization Technologies

<table>
<thead>
<tr>
<th>Selection Factors</th>
<th>VMs</th>
<th>Containers</th>
<th>Serverless fPaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Architecture</td>
<td>Well-suited for monolithic three-tier apps.</td>
<td>Provide a high degree of modularity and elasticity for microservice applications as well as for front-end web services</td>
<td>Highly suited for event-driven architectures as long as the application can be decomposed into functions</td>
</tr>
<tr>
<td>Third-Party Independent Software Vendor (ISV) Support</td>
<td>High</td>
<td>Low to Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Hybrid Cloud Architecture</td>
<td>Medium</td>
<td>Medium to High</td>
<td>Low</td>
</tr>
<tr>
<td>Operational Control</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Operational Overhead</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Availability of Skills/Third-Party Service Providers</td>
<td>High</td>
<td>Low to Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

Source: Gartner (April 2020)

---

**What Are the Primary Use Cases for Serverless fPaaS?**

In general, serverless solutions are suitable for projects that have functional components exhibiting at least few of these characteristics:

- Tied to external events, such as receiving telemetry data
- Runs infrequently
- Has scaling requirements that are highly variable or unknown
- Discrete functions are small and ephemeral, often lasting a few seconds
- Can operate in a stateless manner across invocations
- Connects other services together (integration, passing data or routing events between other services)

Typical uses cases that exhibit these characteristics are:

- **Cloud operations** — The most natural use case is to harness fPaaS to execute operational code that manages the cloud environment. Many cloud management objectives can be achieved by creating a serverless function that is triggered when the platform signals an infrastructure event. The function could evaluate the source of the event, and any salient information passed along with the event, then take further action. For example, a function could be triggered whenever an object is placed into a bucket in the object store. It would be notified...
of the object’s handle, and then could investigate the object to act on it, such as looking for uploaded photos to then place into an image catalog, along with a thumbnail.

- **Microservices implementations** — Microservices architecture is a variant of service-oriented architecture (SOA) that emphasizes small, well-defined independent services (do not depend on each other for data or functionality) yet are combined to create an application or suite of applications. fPaaS is a good first choice to investigate for a microservice. However, not all microservices are good fits for fPaaS. Those that must persist data between calls and are called frequently might be better implemented in a container or VM. It is for this reason that many microservices solutions are combinations of services backed by a mix of VMs, containers and serverless functions.

- **IoT platforms** — In these applications, data is being transmitted from the edge into the central cloud. In the simplest case, this is telemetry data from sensors, either sampled at regular intervals or triggered by a physical event. As such, each payload is very small, and the interarrival time distribution of the payloads can be highly variable, unknown and bursty. Serverless functions are a natural ingress point for these data. Each function captures the incoming data, then processes it in some manner, usually resulting in aggregating the data, storing it, or triggering a new event or generating a control signal directly back toward the edge.

- **Service integration** — A key value proposition of the cloud, and of SOA, is to source the best service for a task, regardless of its origin, and knit it into a larger application context. Among the many problems associated with this model, ease of integration is one that fPaaS solves neatly when used with RESTful APIs in combination with an API gateway.

**How Do We Ensure That We Are Organizationally Well-Structured to Take Advantage of Serverless fPaaS?**

There are three aspects of the organization to consider when adopting serverless fPaaS:

- **Application development** — Being “ready” for serverless fPaaS is no different from being ready for the cloud PaaS model in general. However, fPaaS is a cloud service, in which the operations are even farther removed from visibility than VMs and containers. To reap its full advantage, organize to put developers and operators closer together — even on the same team — so that they share close responsibility for the development and operation of a software product throughout its entire life cycle.

- **Security and risk** — The biggest change that security and risk management leaders will have to adjust to is that they no longer own or control the OS, hypervisor, container and application runtime. Securing serverless fPaaS will force information security and risk professionals to focus on the areas that they can control (such as integrity and assurance of the code, access control, network connectivity, and minimizing the attack surface through real-time scanning and heuristic detection).

- **Infrastructure and operations (I&O)** — It is important to remember that serverless technologies like fPaaS do not by themselves obviate all the other forms of infrastructure like containers, VMs, and even physical machines. Most organizations will need all four of these in a changing mix over time. However, it is critical for I&O leaders to rethink IT operations from
infrastructure management to application governance, with an emphasis on monitoring, cost management and ensuring that application SLAs are being met. While the role of I&O teams may be diminished in public cloud fPaaS, they should work closely with developers in aspects of automation and application service management. Although fPaaS can be deployed on-premises, such deployments complicate many of the values of the model such as rapid elasticity and close integration with event sources such as data stores. For this reason, on-premises deployments of fPaaS are both rare and small. However, in those cases where an on-premises deployment is merited, I&O teams have the opportunity to support fPaaS in the role of service provider.

What Serverless fPaaS Solutions Exist Today?

While AWS is an early leader in this category and has the most widely adopted service, most leading public cloud providers offer serverless services. Specifically, for fPaaS, Table 2 lists the key offerings from cloud service providers (CSPs):

Table 2. Public Cloud fPaaS

<table>
<thead>
<tr>
<th>Cloud Provider</th>
<th>Product Name</th>
<th>Popular Languages Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alibaba</td>
<td>Cloud Function Compute</td>
<td>Java, Python, PHP, Node.js</td>
</tr>
<tr>
<td>AWS</td>
<td>Lambda</td>
<td>Java, Go, PowerShell, Node.js, C#, Python and Ruby</td>
</tr>
<tr>
<td>Google Cloud Platform</td>
<td>Cloud Functions</td>
<td>Node.js, Python and Go</td>
</tr>
<tr>
<td>IBM Cloud</td>
<td>IBM Cloud Functions</td>
<td>Node.js, Java, Python, Swift, Go, PHP, .NET core and Ruby</td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td>Azure Functions</td>
<td>Node.js, Java, Python, PowerShell and .NET core</td>
</tr>
<tr>
<td>Oracle Cloud Platform</td>
<td>Oracle Functions</td>
<td>Node.js, Java, Python, Go and Ruby</td>
</tr>
</tbody>
</table>

Source: Gartner (April 2020)

A number of open-source frameworks are evolving for on-premises and hybrid deployments, although their adoption and maturity tend to be nascent today. Table 3 provides a list of key serverless products/projects:
Table 3. Private Serverless Frameworks

<table>
<thead>
<tr>
<th>Project/Product Name</th>
<th>License</th>
<th>Key Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenWhisk</td>
<td>Apache 2.0</td>
<td>IBM, Adobe, Nimbella</td>
</tr>
<tr>
<td>Fission</td>
<td>Apache 2.0</td>
<td>Platform9</td>
</tr>
<tr>
<td>OpenFaaS</td>
<td>MIT license</td>
<td>Independent contributors</td>
</tr>
<tr>
<td>Knative</td>
<td>Apache 2.0</td>
<td>Google, Red Hat, TriggerMesh, IBM, VMware, SAP</td>
</tr>
<tr>
<td>Nuclio</td>
<td>Apache 2.0</td>
<td>Iguazio</td>
</tr>
<tr>
<td>Nimbella</td>
<td>Proprietary — based on OpenWhisk</td>
<td>Nimbella</td>
</tr>
</tbody>
</table>

Source: Gartner (April 2020)

In addition, serverless frameworks are becoming available in edge environments with products such as Cloudflare workers, AWS Lambda@Edge being noteworthy in that category.

What Lessons Can Be Learned From the Design and Operation of Serverless Apps?

The following are some key lessons learned from early adopters:

- Start training on the general cloud infrastructure as a service (IaaS)/PaaS environment, and adopt a DevOps culture to shorten the learning curve and time to adoption of serverless computing.

- Invest time upfront to build a proof of concept (POC) to validate your assumptions about the application design, code, scalability, performance and total cost of ownership.

- To use serverless computing, developers must refactor applications into minimal latency microservices. It is not possible to simply copy and paste old code.

- Keeping functions stateless is important, because this enables the rapid launch of as many copies of the function as needed to scale to the rate of incoming events.

- Do your homework on the security and technical aspects of serverless deployments. Ensure that your existing cloud security posture management (CSPM) tool can provide risk visibility and configuration management fPaaS, as well as scan for vulnerabilities in real-time (see “Security Considerations and Best Practices for Securing Serverless PaaS”).

- For variable workloads, serverless computing can be economical, compared with alternatives, due to its ability to provision and consume infrastructure resources only when they’re needed. Exceptions, such as workloads with heavy invocations, can make API gateway costs high.

- Deploy tooling to gain visibility into functions, and rapidly troubleshoot them to automate the continuous integration (CI)/continuous delivery (CD) pipeline. The tooling around local testing and application debugging is immature, but rapidly evolving.
Create repeatable serverless patterns that can be reused across multiple applications and teams. This broadly distributes best-practices and speeds adoption.

What Is the Future of Serverless fPaaS?

- **Continued improvements to fPaaS** — We expect cloud providers to continuously improve on their offerings in terms of supporting more programming languages, better security, monitoring, application debugging and local testing tools.

- **More serverless back-end-as-a-service from cloud providers** — Expect more cloud platform services — databases, message queues and even container services — to be delivered in a serverless operational model. For example, AWS, Microsoft Azure and Google Cloud have all launched container offerings that minimize operational overhead in a serverless model.

- **Serverless will extend beyond the cloud** — While fPaaS and other back-end services have been primarily delivered as part of a public cloud platform, we are seeing a growing number of open-source projects (as outlined earlier in this research) that seem committed to delivering this in on-premises, edge environments and in a hybrid manner. The extent to which serverless solutions will be adopted outside of the public cloud context remains to be seen.

- **Stateful workloads** — The inability to run stateful workloads (persistent state on the host) in fPaaS is a key limiting factor for enterprise customers. This is an area where we will see more innovation in the future, as well as tooling to enable it.

- **Industry consolidation around startups** — There are several startups in this space that provide products for building, deploying, securing and managing serverless applications. Some of these emerging vendors include Stackery (deployment automation), Lumigo (monitoring), Epsagon (monitoring), TriggerMesh (application integration) and Protego (security). We expect to see more new entrants as well as consolidation in this space in the future.

- **Evolving 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 in their efforts to develop serverless computing solutions on Kubernetes. While Google’s unwillingness to hand over its governance to a more neutral foundation, such as the Cloud Native Computing Foundation (CNCF), remains a sore issue within the community, it does seem to be a promising project with an active community and broad support from other vendors. Another effort to help enable industry-standard platforms for supporting data-intensive applications is the CloudEvents specification for describing event data. The goal of the project, which is hosted by the CNCF, is to enable event interoperability across services, platforms and systems from different vendors and service providers.
“Evolution of Virtualization: VMs, Containers, Serverless — Which to Use When?”

“Top 10 Technologies That Will Drive the Future of Infrastructure and Operations”

“Enhancing Operations Automation With Serverless Computing”

“Top 10 Trends in PaaS and Platform Innovation, 2020”

“Strengthen Your DevOps Capability With Platform Ops”

Evidence

This research is based on more than 100 Gartner client inquiries handled on this topic by the authors. In addition, detailed interviews were conducted with several early adopters of serverless computing on their deployment experiences. Many vendors profiled in this research have briefed the authors over the past two years.


GARTNER HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
USA
+1 203 964 0096

Regional Headquarters
AUSTRALIA
BRAZIL
JAPAN
UNITED KINGDOM

For a complete list of worldwide locations, visit http://www.gartner.com/technology/about.jsp