Serverless Computing: An Evolution from Servers to Functions.

Manasha Saqib
(E-mail: manashasaqib@gmail.com)

Abstract— Cloud computing has empowered organizations to concentrate more on their core services and products and less on their IT framework. Today, a wide range of organizations use cloud services without worrying about any hidden problems related to infrastructure. The new intake version of Serverless Computing has emerged. In standard cloud computing, dynamically allotted pay-per-use resources replace the dedicated hardware, for example, virtual servers. These resources are usually referred to as "pay-per-use," as they are charged based on allocation rather than actual use, which may lead a customer to pay more than necessary. Resources are generally not charged or allocated in serverless computing until a function is called. In addition to Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS), Serverless computing or Function-as-a-Service is the next level of abstraction capable of changing the way many organizations devour cloud services. Serverless Computing "allows designers to concentrate solely on composing code without managing servers"- basically make the procedure “serverless” to the client. The unit of scale in Serverless technology is the function and abstracts the runtime of the language. The potential benefits of serverless computing -easier operational management, faster innovation, increased development, and testing complexity, reduced cost of maintenance and infrastructure, increased scalability and higher performance - appear to outweigh the risks for several organizations.

Keywords— Serverless Computing, Infrastructure-as-a-Service, Platform-as-a-Service, Function-as-a-Service, Cloud computing.

I. INTRODUCTION

The term ‘Serverless’ refers to a new generation of Platform-as-a-Service (PaaS) offerings by major cloud service providers. For the deployment of cloud applications, Serverless Computing has emerged as a new and compelling paradigm, mostly because of the recent shift of enterprise application architectures to containers and micro-services [1]. The word ‘serverless’ doesn’t mean ‘No Servers’. Rather, servers are a fundamental part of this concept; but, it is the responsibility of cloud service provider to provide an ephemeral compute service and to handle the complexity of managing individual servers that will execute a piece of code on-request triggered through requests and events.

Serverless is an event-driven computer model in which the underlying infrastructure (includes physical and virtual hosts, machines, containers, and operating systems) is abstracted from the developer and the service consumer. Applications run in stateless containers based on event triggering. The application logic is encapsulated in functions which runs on containers in the infrastructure of the cloud provider. As the load of the application increases, more functions are executed proportionately, the cloud service provider takes care of the scaling of the underlying infrastructure.

II. EVOLUTION

The origin of Serverless goes back to the early 2000s when cloud computing vendors began providing IT organizations with open-source software and infrastructure as hosting services in private or hybrid clouds. The framework has matured following the standardization of cloud-computing models, namely, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The serverless cloud is an evolving step in making full use of the cloud-from Infrastructure-as-a-Service (IaaS) to Platform-as-a-Service (PaaS) to Function-as-a-Service (FaaS) as shown below in Table 1 [2]. While IaaS abstracts the underlying infrastructure to provide ready-to-use virtual machines and PaaS abstracts the entire middleware layer and operating system to develop the application, FaaS goes one step further in abstracting up the entire runtime of programming to provide options for the easy deployment and execution of a piece of code without worrying about its deployment.

TABLE I. EVOLUTION OF SERVERLESS COMPUTING

	IaaS	*PaaS*	*FaaS*
Unit of deployment	Operating System	Applications	Functions
Provides	Virtual machines packaged with operating systems	Application development platform	Execute code (with business logic) on-demand
Abstracts	Physical hardware	Operating system & middleware	Programming runtime

Cloud providers have also begun to provide hardware and software assets as per-use service-based models, leading to significant costs reductions compared to IT organizations existing expenditure. Due to competitive and upcoming services from Microsoft to Google to IBM, Serverless, originally a market limited to AWS's Lambda service, has expanded dramatically. In the last few years, leading cloud providers such as Amazon, Microsoft, Google, and IBM have launched serverless services. In 2014, Amazon Web Services released an innovative service called AWS Lambda that offered a new way to deploy code to the cloud for the web, mobile or IOT applications. Instead of deploying the application's entire codebase at once, Lambda allows the application to deploy its function individually, to their own containers. Microsoft announced its response to AWS Lambda in 2015 with the launch of "Azure Function". Google released "Cloud Function" in February 2016 as its own Serverless service. IBM also released its OpenWhisk serverless service.

III.SERVERLESS COMPONENTS

The three key components of serverless computing stack are:

- API Gateway.
- Function as a Service (FaaS).
- Backend as a Service (BaaS)

A. API Gateway

The API Gateway acts as the communication layer between the front-end and the Function as a Service layer. It maps the endpoints of the REST API with the respective business logic functions. There is no need to deploy and manage load balancers n this model with servers out of the equation.

B. Function as a Service (FaaS)

Function as a Service (FaaS) is a type of cloud computing service that allows customers to develop, run, and manage application functionality without the complexity of building and maintaining the infrastructure. The core of the "Serverless" architecture is to build an application with this model.

FaaS functions do not need to be coded to a specific framework and are regular language and environment applications. [3] All aspects are fully automatic, elastic, and managed by the provider from execution to scaling.

The key components of Function as a Service are:

- Functions
- Events
- Resources

A **Function** is an independent deployment unit like a micro-service. The responsibilities of a function may encompass:

- Saving a user in the database.
- Processing a file.
- Performing a scheduled task.

Anything that triggers the performance of a function is considered to be an **Event**. It could be:

- Call for an HTTP endpoint registration service.
- Upload of files to the file server.
- The message published in the message queue.

A resource refers to the components used by the Functions. For example:

- Database services (e.g. for saving data from Users/Posts/Comments).
- File System services (e.g. for saving files or images).
- Message Queue services (e.g. for publishing messages).

C. Backend as a Service (BaaS)

This is essentially a distributed No SQL database based on the cloud that essentially removes overheads for database management.

IV. SERVERLESS ARCHITECTURE

In today's world, cloud services are used by all sorts of companies without worrying about any underlying infrastructure problems [4]. This new consumption model has evolved into a serverless architecture. Customers can re-imagine their next-generation products from ideation to production by adopting serverless architectures without waiting or worrying about infrastructure. In Serverless architecture, the design is completely dependent on third-party services where the code runs in ephemeral containers using FaaS calling BaaS for data storage needs. Serverless architectures have gained momentum through the creation of scalable and cost-effective applications.

In a serverless architecture, the "User" service would be divided into more granular functions. In Figure 1, [5] each API endpoint corresponds to a specific function and file. If the client initiates a "create user" request, the entire codebase of the "User" service must not run; instead, only create user.js will run. Containers do not need to be pre-supplied, as standalone functions only consume resources when needed, and users are only charged for their functions actual runtime. This granularity also facilitates parallel development work, since functions can be independently tested and deployed.

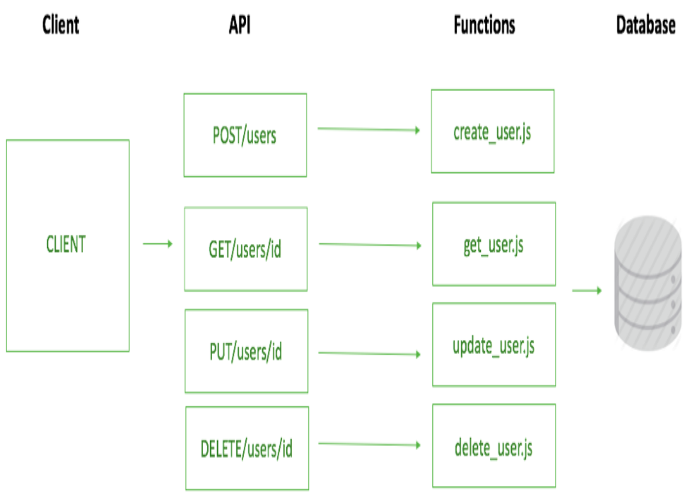

Fig 1. Serverless Architecture

V.SERVERLESS CHARACTERISTICS

A number of characteristics help to distinguish the different serverless platforms. When selecting a platform, developers should be aware of these properties.

a) Cost: Usage is usually measured and users only pay for the time and resources used to run serverless functions. One of the key differentiators of a serverless platform is the ability to scale to zero instances. The measured resources such as memory or CPU, and the price model, such as off-peak discounts, vary from provider to provider [6].

b) Performance and limits: The runtime resource requirements of serverless code are limited in a variety of ways, including the number of simultaneous requests, the maximum available memory and CPU resources for a function call. Some limits can be increased when users need to grow, such as the threshold for simultaneous requests, while others are inherent in platforms, such as the maximum memory.

c) Programming Languages: Serverless services support a wide range of programming languages including Python, JavaScript, Go, Java, C# and Swift. Most platforms support more than one language of programming.

d) Composability: Platforms can invoke one serverless function from another, but some platforms provide higher level mechanisms to compose these functions and facilitate the construction of more complex serverless applications.

e) Deployment: Serverless architectures are built to improve the productivity of developers and inherently agile building, testing and release cycles. With the serverless approach, one can perform as many test runs as possible without worrying whether infrastructure is ready or components in the solution are available for deployment.

f) Security and accounting: Serverless platforms are multitenant and must isolate the performance of functions among users and provide detailed accounting so that it will be easy for users to understand how much they have to pay.

g) Monitoring and Debugging: Each platform supports basic debugging using print statements recorded in the logs of execution. Additional capabilities can be provided to help developers trace errors, bottlenecks, and understand function execution circumstance better.

VI.BENEFITS OF SERVERLESS ARCHITECTURE

Serverless applications allow developers to concentrate time and energy on the core product at hand, instead of worrying about server management and operation or runtime, whether in the cloud or on premises. This can significantly reduce overhead costs, and at the same time lead the development of better, higher-quality and scalable products.

a) Easier Operational Management: The serverless platform makes operational management easy and cost-effective by automatically scaling tools like FaaS [7]. The clear division of applications and support infrastructure leads to higher quality products.

b) No need for Server Management: Serverless architecture allows to focus only on the code that is important for the application and does not have to worry about the provision and maintenance of servers. There are also no requirements for software installation, maintenance or management.

c) Reduced time to market: Ideas can be transformed into reality in minutes or hours by using a serverless architecture. Serverless architecture also allow tight deadlines to be met by running multiple versions of code.

d) Flexibility of Scaling: Serverless applications enable easy scaling, as they can be automatically scaled or switched to throughput, memory and other consumption units. This is in contrast to complexity of the traditional applications associated with scaling units of individual servers.

e) Built-in Default Capabilities: Another advantage of serverless applications is that, unlike traditional applications that require one to architect for these capabilities, the availability and fault tolerance capabilities are by default built into the system.

f) Zero Idle Capacity Low Costs: Like cloud services, serverless is a new way to offload IT overhead. A serverless architecture eliminates the responsibility for the management of servers, databases, and even application logic, reducing the costs of setup and maintenance. One has to pay only for the time code runs and the operating costs are reduced. Cloud management costs are reduced by serverless architecture.

g) Step Ahead of PaaS: Even if PaaS is set up to allow auto-scale application. It cannot be done at an individual request level. However, the provider uses the serverless framework to find a server where the code is to run and scale up when necessary. As soon as the execution ends. The containers used to perform these functions are decommissioned.

VI. TRADE-OFFS

a) Third-Party Vendor Locking: Another trade-off for the use of serverless architecture is dependence on third-party vendors and services, resulting in a lack of control over problems such as system downtime, unexpected limits, changes in costs, loss of functionality, etc.

b) The Hidden Costs: While hidden costs can lead to reduced operating system, development costs and complexity of the application are increased. Projects that small to medium-sized teams had developed in the past now require many small teams and significant complexity in product and team management.

c) Increased Development and Testing Complexity: The Serverless architecture combines the complexity of dividing a single application into a service and function fabric, which increases significantly with the number and variety of services. Similarly, it is difficult to carry out integration tests and load tests for a serverless platform, since it depends on externally provided systems.

VII. CONCLUSION

Serverless Computing is an evolution of the trend towards higher levels of abstraction in cloud programming models, which is currently exemplified by the Function-as - a-Service (FaaS) model, in which developers write small snippets of stateless code and allow the platform to manage the complexities of performing the function in a flaw-tolerant way. The serverless paradigm can eventually lead to new types of programming models, languages and platform architectures. Adopting serverless can bring many benefits but depending on the use case, the road to serverless can be challenging and like any new technology innovation, serverless architectures will evolve into an obvious standard.

REFERENCES

[1] NGINX Announces Results of 2016 Future of Application Development and Delivery Survey. https://www.nginx.com/press/nginx-announces-results-of2016-future-of-application-development-and-delivery-survey.

[2] Ma, J. (2016, September 22). Serverless Architectures: The Evolution of Cloud Computing - DZone Cloud. https://dzone.com/articles/serverless-architectures-the-evolution-of-cloud-co.

[3] Serverless Architecture: Evolution of a New Paradigm. (2017 December12) https://www.globallogic.com/gl_new