SERVERLESS VS FUNCTION-AS-A-SERVICE (FAAS): WHAT'S THE DIFFERENCE?

The ever-increasing popularity of cloud computing has garnered a new technological revolution with many cutting-edge technologies. Function as a Service (FaaS) and serverless are two such technologies that approached the forefront due to the popularity of cloud computing. Both these technologies aim to:

- Provide cost-effective cloud platforms
- Eliminate the need for infrastructure management

Sometimes the terms FaaS and serverless are used interchangeably. However, they are two different technologies with some significant differences.

In this article, we will have a look at these technologies and understand their similarities, differences, and how to choose the right technology based on your needs.

What is Serverless?

As the name suggests, serverless is a computing model where infrastructure orchestration is managed by service providers.

The emergence of cloud computing has enabled users to quickly create any service instance, scale up or down, and discard as required, saving CapEx and OpEx while eliminating the need to manage

physical hardware.

However, even with these cloud servers, the management and configuration tasks of the cloud infrastructure were left to the users. This is where serverless comes into play.

Serverless aims to eliminate the management and configuration tasks enabling users to solely focus on the application. This is not limited solely to server instances but extends to other areas, too, such as:

- Serverless databases
- DevOps pipelines
- Kubernetes
- Etc.

Let's consider an example where we need to provision a SQL database within the Azure cloud provider.

In a traditional scenario, we will have to first provision all the underlying resources from networks and security groups to compute instances. Then, we'll need to install and configure the SQL database and continuously manage the infrastructure.

However, with a serverless solution like Azure SQL Database serverless, we can create a SQL server with a few clicks that will automatically scale according to the system load without the need for users to manage any infrastructure.

What is Function as a Service (FaaS)?

Function as a Service is a relatively newer concept that aims to offer developers the freedom to create software functions in a cloud environment easily. In this method, the developers will still create the application logic, yet the code is executed in stateless compute instances that are managed by the cloud provider. FaaS provides an event-driven computing architecture where functions are triggered by a specific event such as message queues, HTTP requests, etc. Some of the FaaS options available through leading cloud providers include:

- Azure Functions
- AWS Lambda
- Google Cloud Functions
- Oracle Cloud Functions

Function as a Service model adheres to a pay-as-you-go model where you have to only pay for the function when it's used.

FaaS allows developers to solely focus on developing the application functionality without having to consider backend infrastructure or server configurations. Instead, you'll simply:

1. Pick the programming language of your choice.
2. Package the function with its software dependencies.
3. Finally, deploy the function.

FaaS vs Serverless: How are they different?

At a high level, both FaaS and serverless refer to a cloud computing platform that eliminates the need for managing infrastructure. However, these two services excel in different other functionalities as well.

Serverless example

For instance, let's consider the scenario of a Kubernetes deployment. In a traditional sense, users will have to provision servers, manage networking, install, and configure Kubernetes cluster software, manage scaling and availability, and finally create the container and deploy the application in the cluster. After that, users need to take care of all of the day-to-day management tasks of the cluster, which is their sole responsibility.

That's a lot of work!

Serverless came along to reduce this workload. Serverless services like AWS Fargate allow users to create an Amazon Elastic Kubernetes Service in a couple of clicks, configure it to suit their needs, and then deploy the application container on the AWS-managed Kubernetes cluster. Because AWS manages everything in this kind of serverless Kubernetes environment, users do not have to:

- Manage any infrastructure
- Worry about scaling and availability

FaaS example

Now, a function as a service platform further abstracts any infrastructure requirement.

Assume that you can simply deploy the web application without provisioning any kind of infrastructure or configurations—just upload the code with dependencies, and you are done! This is what the FaaS services provide: a platform to run functions without worrying about the underlying infrastructure.

These FaaS services are highly useful when creating microservices-based applications. There, we can break down our web application into separate services that run as FaaS functions. Microservices can highly benefit from FaaS as they are targeted towards event-driven architectures. This is not only limited to a single service for a single function, and there might be instances where a single microservice can be a combination of multiple functions, all running on cloud functions communicating through APIs.

The FaaS concept revolves around providing a development platform that can function independently without relying on a larger application or framework.

FaaS vs Serverless: pros & cons

Ok. We've got a good understanding of the differences between each type of service. Now, let's have a look at their advantages and disadvantages.

Serverless advantages

- **No infrastructure configurations**
- **Most options support auto-scaling**, leading to unlimited expansion potential
- **Cost savings** compared to traditional server-based configurations
- **Ease of long-term management** since all updates, security, and optimizations are managed by the service provider

Serverless disadvantages

- **Loss of fine-grained control**, as all the infrastructure is managed by the cloud provider and users have no access to backend infrastructure for custom configurations.
- **Loss of features.** Some serverless applications will lack specialized configurations available to end-users. This is most prevalent when dealing with serverless database applications as some features that are available on normal database deployments will sometimes not be available in the serverless versions.
- **Potentially expensive, depending on the use case.** However, if you are dealing with huge data sets or data requests, it might be cost-effective to have dedicated servers to handle the load.

FaaS advantages

- **Ease of use.** There's no need to write complete applications. You can simply write only the required functional component and deploy it.
- **Reduction in OpEx.** As the FaaS service model is based on a usage-based pricing model, you only have to pay when the function is executed. This can vastly reduce operational expenses.
- **Scalability and efficiency of functions.** Functions allow users to simply scale up or down and provision multiple functions to meet any user demand without making any changes to the application functionality.

FaaS disadvantages

- **Not suitable for complex functionalities** in a single function. Functions are aimed at creating small functions that accomplish a single task.
- **Potential for more tooling.** Managing a large number of functions will require third-party management tools.
- **Added data stores.** The stateless nature of the functions requires separate data stores outside the functions to support stateful services.

How to choose between FaaS or Serverless

Choosing a FaaS or a Serverless solution depends on:

- The user requirements
- The supported functionality

FaaS options, however, can offer more narrow solutions, and can be used to create functions that complement existing applications that live as separate services outside the core application. This

offers users more flexibility when it comes to developing and testing new features.

Additionally, FaaS can be an invaluable asset in microservices-based applications to extend and support application functionalities.

On the other hand, Serverless offers many more options and is not limited to creating functions. It is the best solution if your main goal is to reduce infrastructure management responsibilities while retaining control of the application configurations. However, this comes with additional complexities compared to just deploying a cloud function (FaaS option).

Serverless is also the best option when dealing with large-scale deployments and functional requirements that require multiple technologies rather than simple functions. Unlike FaaS options, serverless options allow users to have a database.

It is not a must to make a strict choice between serverless or FaaS. We always have the option to use both these services in our applications.

For example, FaaS function can power an application component that reads data from a serverless database that pushes data to a serverless message queue or another function via a restful API.

Both Function as a Service and Serverless platforms and services are highly relevant and useful in the current technology landscape. With all the advantages offered by each of these options, it's the user's responsibility to select the right service that can best tackle their requirements.

Related reading

- BMC DevOps Blog
- Serverless Best Practices
- Microservices vs Serverless: What's The Difference?
- AWS Serverless Applications: The Beginner's Guide
- What Is IaC? Infrastructure as Code Explained
- IT Infrastructure Management: An Introduction