WHAT IS SERVERLESS ARCHITECTURE? SERVERLESS ARCHITECTURE EXPLAINED

Many experts believe that serverless computing *is* the future, as mobile and Internet of Things (IoT) applications continue to fuel demand for serverless architecture, coupled with the growing need to integrate cloud applications with mobile and desktop apps.

Serverless architecture refers to the implementation of serverless code to create a number of design patterns that benefit businesses. It forms the foundation of serverless computing.

In this article, we will examine all things serverless architecture, including:

- Types of serverless architecture software
- Existing frameworks of serverless architecture
- Design patterns

Types of Serverless Architecture Software

There are three primary services offered via software developed with serverless architecture.

Function-as-a-Service

In the realm of pre-packaged services, Function-as-a-Service is also sometimes known as Framework-as-a-Service or FaaS, falls in between Software-as-a-Service and Platform-as-a-

Service.

Think of FaaS as a ready-to-implement framework that can be easily tailored to the needs of an enterprise company. For further clarification, SaaS is ready to use out of the box while FaaS is not. However, FaaS does not require the resources to implement that you would need if you were using PaaS.

FaaS can be delivered in customizable templates, for instance, by industry vertical. FaaS uses containers to prime for rapid deployment of applications on all platforms. For instance, developers can stack containers for scalability or write one container for iOS development and another for desktop applications.

Consumers purchase FaaS from third-party vendors who handle server management. They are then charged for actual runtimes instead of pre-allocated units. Companies who use FaaS benefit from improved efficiency as fewer resources are spent on rapid development of applications.

Backend-as-a-Service

Similar to FaaS, Backend-as-a-Service (or BaaS) is another serverless technology. Some will contend that BaaS takes it a step further as a NoOps offering. NoOps essentially refers to infrastructure that has been automated to the point that in-house developers have no hand in its operation.

Either way, here is an easy way to look at BaaS: imagine your enterprise organization is developing a mobile app to connect employees to important information on the go. You might develop the basic application framework in-house, and outsource the functionality. This includes backend processes like access cloud storage, syncing.and social collaboration.

A company's ability to offer backend services that link a mobile application to the cloud is referred to as BaaS.

Database

Database serverless frameworks access and automate your database functions. These are functions that both write and read from a database, as well as provide a response.

Serverless database frameworks offer companies room to grow globally, as multiple applications can be developed by region. Yet, they all run from one location powered by FaaS technology.

All types of architecture benefit from the unique advantages of serverless computing, which can be summed up below:

- Lower operational cost
- Increased efficiency
- Better allocation of resources

Notable Frameworks

As companies prepare for the future, they can expect to see a landscape of technology forever changed by serverless architecture. This is evidenced by the innovative frameworks that are already competing for consumer attention. Notable frameworks include:

AWS Lambda

The most well-known, quintessential serverless framework is AWS Lambda, powered by Amazon. This framework allows developers to reap the benefits of development without the cost of a physical server infrastructure. Additionally, it is one of the first of its kind to gain mainstream popularity.

AWS Lambda is a full administrative solution. Amazon runs your code snippets in response to events and only charges for times when the code is being used. They handle all the backend and administrative functions.

Lambda supports:

- Node.js
- Java
- C#
- Python

Google Cloud Functions

Google Cloud Functions offers serverless computing on Google's open infrastructure. The platform supports Javascript and executes in Node.js. So, it offers a familiar environment for developers to code.

The platform benefits include the following:

- Focus on microservices development
- Allows users to connect cloud services and applications (BaaS)
- Rapid and efficient development environment
- Mobile ready for developers to drop in code

IBM Cloud Functions

IBM Cloud Functions is a FaaS model based on its predecessor Apache OpenWhisk. It's described as a lightweight serverless architecture model that allows developers to rapidly code on-demand, meaning they are only charged for what they use.

Four Design Patterns

One of the big benefits of serverless architecture includes freeing up resources to focus on what the software is supposed to do and not what is needed to make this happen. As a result, frequently used design patterns have emerged.

At the 2017 APIDays Conference, research on AWS Lambda revealed four general design patterns for server architecture that are commonly deployed. These are described below.

Event-Driven Applications

If you do your research on serverless architecture, you'll see the term event-driven a lot. It basically means that runtimes are prompted as a response to events. This simplistic architecture is among the most commonly used.

This form of architecture lends itself to DevOps developers who focus on writing patches that integrate two systems.

Web Apps

In the use case of serverless web apps, processes run to determine who the user is and what information should be served to that user inside the application. In the case of Lambda, processing gets initiated in the API gateway to replace and enhance static content with custom content based on user details stored in a dynamic database.

Mobile and IoT Apps

These cases work similar to web apps in this context. User authentication is deployed to determine who the user is, and custom content is served up based on user profile.

Application Ecosystem

An application ecosystem for serverless applications is one where workflows and applications are created in a serverless environment. These unique ecosystems are powered by functions of both third-party providers and AWS. An ecosystem is created when the functions of a given serverless architecture are integrated with other functions that allow for customization of processes.

The Future is Serverless

Serverless architecture is the way of the future for many types of simple applications. It's an ideal approach for some enterprise businesses because it offers the following benefits:

- Efficient use of resources
- Rapid deployment
- Cost-effective solutions
- Focus on coding
- Familiar programming environment and languages supported
- Increased scalability

For these reasons, it's easy to see why many companies are making the switch. At the same time, it is important to understand that serverless architecture does have drawbacks that need to be considered before making any changes to your business model.

When you use a third party API the following concerns emerge:

- **Vendor control issues:** Developers give up control, especially in the case of BaaS systems.
- **Security concerns:** These arise as a result of leaving security in the hands of the vendor.
- **Contract lock-in:** This occurs when consumers are unable to break contracts with their vendors.
- **Dependent relationship:** Consumers are dependent on vendors for debugging and monitoring.
- **Limitations:** Limits exist on processes to sidestep the pitfalls of overcomplicated architecture.
- **Response Latency:** Because consumers are only billed for runtime, code is powered down between requests. Response times can vary as a result.

Because of the above-mentioned challenges, serverless architecture is often not typically ideal for high-performing, complex application builds.