Chapter 4
Exploring the World of Serverless Computing: Concepts, Benefits, and Challenges

Vishal Goar
Engineering College Bikaner, Bikaner, India

Nagendra Singh Yadav
https://orcid.org/0000-0002-9591-4491
Engineering College Bikaner, Bikaner, India

ABSTRACT

As the landscape of computing continues to evolve, serverless computing emerges as a paradigm shift, offering a transformative approach to application development and deployment. Serverless computing has revolutionized the way businesses build & deploy applications. With its promise of seamless scalability, reduced operational overhead, and cost optimization, serverless systems have become a dominant paradigm in cloud computing. A focal point of this exploration is the examination of the myriad benefits that serverless computing brings to the table. This chapter focuses on the Introduction to Serverless computing, core concepts of serverless computing, Advantages of serverless computing, Challenges in Serverless Computing, Use Cases and Industry Adoption, Best Practices for Serverless Development, and Future Trends in Serverless Computing.

1. INTRODUCTION TO SERVERLESS COMPUTING

Serverless computing is frequently known as Function as a Service which is a cloud computing model where cloud providers automatically manage the underlying infrastructure, enabling developers to aim fully on creation and deployment of code. In this model, applications are segmented into several methods, each of which is run in response to a particular condition or event. Unlike traditional server-based models where developers are required to maintain servers & their scaling, in serverless computing, the cloud service provider is responsible for providing server, scaling, & maintenance.

DOI: 10.4018/979-8-3693-1682-5.ch004

Evolution of Serverless Computing - The concept of serverless computing has evolved over time, with its roots in Platform as a Service & function as a Service offerings. It can be traced through various stages of evolution:

- **PaaS and Early Cloud Services -** The idea of abstracting infrastructure and providing a higher-level environment for application deployment began with PaaS offerings i.e. Google App Engine These platforms automated much of the operational overhead (Nastić et al., 2017).
- **Function as a Service (FaaS) -** FaaS platforms, i.e. AWS Lambda, introduced the concept of serverless by allowing developers to execute each method as an action for events. This represented a shift from managing applications to managing functions.
- **Managed Services -** Cloud providers expanded their serverless offerings, including managed databases, storage, and authentication, reducing the need for developers to manage infrastructure components further.
- **Serverless Frameworks -** The emergence of serverless frameworks, like the Serverless Framework and AWS SAM, helped streamline the development and deployment process, making it easier for developers to adopt serverless technologies.
- **Ecosystem Growth -** The serverless ecosystem has continued to grow, with more providers, tools, and libraries, making it increasingly accessible and versatile for a wide range of use cases.

Key attributes of Serverless Computing - It is characterized by many key features that set it apart from conventional server-based models:

- **Event-Driven -** Serverless functions are triggered by certain events or requests i.e. HTTP requests or custom triggers. The code only runs when there is an event to process, eliminating idle time and reducing costs.
- **Automatic Scaling -** Serverless platforms automatically control the scaling of functions in response to varying workloads (Eismann et al., 2020). Developers are not required to bother about provisioning or management of servers. This is something provided by the cloud providers.
- **Pay-as-You-Go Pricing -** Serverless platforms bill users depending on the real execution time of their methods. This granular billing model means you only bill for the assets utilized during the execution of the method.
- **Stateless -** Serverless methods don't keep details between invocations. Any necessary state must be stored externally i.e. database.
- **Microservices Architecture -** Applications are built as a collection of smaller functions, each with a specific purpose. This microservices approach makes it easier to maintain and scale different parts of the application independently.
- **Managed Infrastructure -** Cloud providers control the principal infrastructure, including server allocations, load balancing, & security, permitting developers to aim at generating code.

Architectural Components - Serverless applications consist of various architectural components, each serving a specific role in the overall system. Some of the essential components include: