Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2023, 10(5):94-97

Research Article ISSN: 2394 - 658X

Serverless Architecture: Transforming Application Development for the Next Generation

Krishna Mohan Pitchikala

ABSTRACT

In the past, most software development followed monolithic and server-centric architectures, meaning developers had to handle everything from provisioning to managing and scaling servers. This process was often time-consuming and prone to errors, leading to inefficiencies and increased costs for businesses. Serverless architecture was developed to address these issues, making it easier to develop and deploy applications by handling infrastructure tasks such as monitoring and managing servers. This approach is not only cost-effective but also allows applications to scale easily when needed. Many serverless providers have now emerged, offering solutions that take care of tasks like server provisioning, patching, and capacity planning. This allows developers to focus on creating valuable features. Many companies are moving towards serverless architectures because of the advantages and availability they offer. Like any technology, serverless architecture has some drawbacks. However, it has significantly changed how applications are developed by providing a scalable, cost-efficient, and simplified deployment method that overcomes these drawbacks. This paper will explore the concept of serverless architecture, its core principles, benefits and challenges. We will also discuss a case study showing how moving to serverless architecture benefited a company and when it is recommended for a company to adopt serverless architecture.

Key words: Serverless Architecture, Transforming Application Development

WHAT IS SERVERLESS ARCHITECTURE?

Before we begin discussing what serverless architecture is, let us first understand what a server-centric architecture means. In software and IT systems, the term server-centric architecture refers to the traditional method of designing and deploying applications where an organization manages infrastructure such as servers and network resources. Usually, it involves one system in which all parts of an application are closely interconnected and operate on specific machines that are dedicated for this purpose only. This means that, to execute code, there is a need for setting up physical servers (or a cluster of them) which must be maintained afterwards. Keeping these machines working properly entails hiring expensive engineers who will take care of them day by day. The idea behind introducing serverless architecture was to eliminate this overhead and complexity associated with maintaining and managing servers [1, 2].

Serverless architecture, also referred to as Function-as-a-Service (FaaS), abstracts away the underlying infrastructure and server management tasks from developers. In this model, developers focus solely on writing and deploying code functions, while the cloud provider dynamically allocates and manages the compute resources required to execute those functions.

To put it simply, Serverless architecture allows developers to build and run applications without managing servers. Instead of creating and maintaining servers, developers use existing public cloud provider managed services. These services perform functions such as receiving network requests, running code, storing data, and managing files that scale automatically up or down with demand. This method decreases management overheads and is cost effective as companies pay only for what they use.

HOW SERVERLESS ARCHITECTURE WORKS?

Servers allow users to communicate with an application and access its business logic. One of the most popular serverless architectures is Function as a Service (FaaS), where developers write their application code as a set of discrete functions. Each function will perform a specific task when triggered by an event, such as a HTTP request. When a function is invoked, the cloud provider either runs these functions on existing servers or starts new ones when necessary; this happens invisibly to the developer who only writes codes and deploys them.

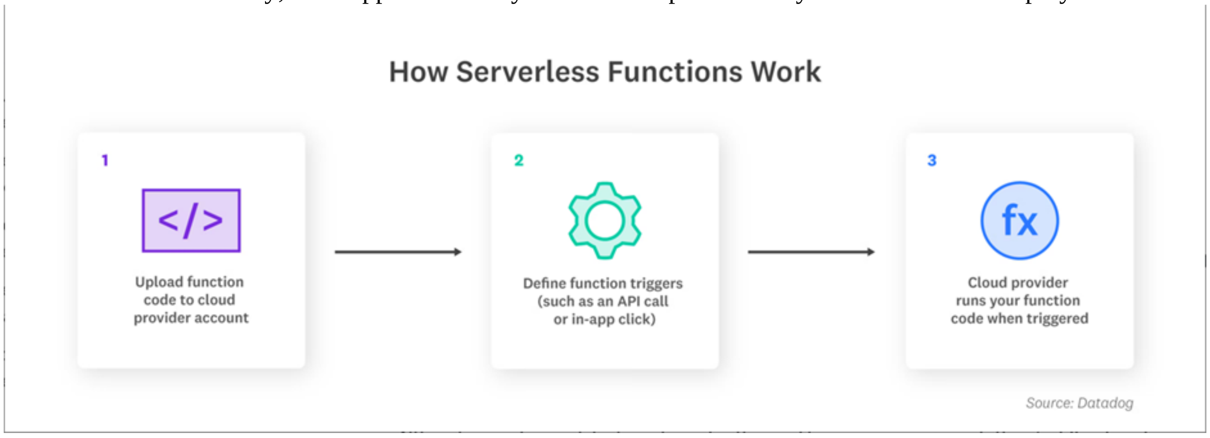
How Serverless Functions Work
1
Upload function code to cloud provider account
2
Define function triggers (such as an API call or in-app click)
3
Cloud provider runs your function code when triggered
Source: Datadog

Figure 1: How Serverless Functions work [6]

While serverless architecture has been around for more than a decade, AWS Lambda was the first FaaS platform to become popular, introduced by Amazon in 2014. Google Cloud Functions and Azure Functions also offer services that are like AWS Lambda, but most developers use AWS Lambda [6].

MICROSERVICES AND SERVERLESS: WHAT'S THE DIFFERENCE?

Microservices and serverless are two different approaches to building applications. Microservices involve breaking down an application into smaller, independent services that work together, allowing for better organization and scalability. Serverless, on the other hand, focuses on running code without managing the underlying servers, with the cloud provider handling the infrastructure. While microservices define the structure of an application, serverless defines how the code is executed. Both can be used together (which is often recommended) or separately, as they address different aspects of application development.

IS DEVOPS STILL NEEDED WITH SERVERLESS?

Yes, DevOps is still needed with serverless. While serverless architecture reduces the need for server management and simplifies deployment, DevOps practices are still essential. DevOps involves collaboration, continuous integration, testing, monitoring, and maintaining the entire application lifecycle, which are all crucial even in a serverless environment. Serverless can complement DevOps by automating infrastructure management, but it doesn't replace the need for DevOps practices.

KEY PRINCIPLES OF SERVERLESS ARCHITECTURE

1. **Event-Driven Execution:** Serverless functions only run when they are needed. They start up in response to specific events like someone visiting a website, updating a database, or sending a message, which means that they don't run constantly, they only execute when something triggers them.
2. **Automatic Scaling:** The cloud automatically adjusts the computing power for your application. When there's a lot of traffic, it adds more resources. When there's less traffic, it reduces resources. This ensures your application runs smoothly and scales appropriately as needed
3. **Pay-Per-Use Billing:** Businesses only pay for the time their functions are running and the resources they use, avoiding the cost of unused server capacity
4. **Stateless Functions:** Each execution of a serverless function has no knowledge of past executions, in other words, it does not remember anything from previous runs but starts afresh every time it is invoked hence making them easy to manage and usable at different parts of an application without depending on past states
5. **Focus on Business Logic:** Developers can write code that solves their business problems directly instead of dealing with managing servers and infrastructure complexity which leads to faster development and more innovation
6. **Security:** While developers need to secure their code, the cloud provider handles the security of the underlying infrastructure, such as applying patches to servers and maintaining overall system security
7. **High Availability and Fault Tolerance:** Serverless architectures are designed to be highly reliable where the provider ensures that your functions are always up and can recover quickly from failures even if there is an issue with the underlying hardware or software

These principles allow developers to create more efficient, scalable, and manageable applications, letting them focus on delivering value instead of managing infrastructure

STRENGTHS OF SERVERLESS ARCHITECTURE

1. **Automated Scaling:** The serverless applications adjust their resources to the current demand automatically. Therefore, it can handle sudden peaks or drops in traffic without any manual intervention. It ensures that the system performs well and remains cost-effective by using resources when necessary. This flexibility keeps high performance during busy periods and cuts down on expenses during periods of low activity.
2. **Optimization of costs:** The organizations can save money by paying only for the resources that they use. This model of payment per use is particularly beneficial to applications with unpredictable workloads and leads to significant savings.
3. **Higher productivity:** Developers can write more code and create new features because they do not need to worry about such routine tasks as setting up servers, applying updates or planning for capacity; thus enabling them focus on adding value through features enhancement and improvement of application.
4. **Improved uptime:** With cloud management services, apps become more reliable hence capable of handling unexpected loads better than ever before leading into less downtimes therefore providing consistent user experience throughout.

Initially, these may appear as minor advantages but in the development of real-world applications, they are crucial. Saving money, automatically scaling and simple maintenance during high traffic are the key benefits of serverless architecture. It is because of these strong points that a lot of businesses opt for serverless technology.

CHALLENGES OF SERVERLESS ARCHITECTURE

1. **Vendor Lock-In:** Serverless architectures are tied to specific cloud providers. This means that the code and infrastructure often rely on the unique features and services of these providers. If a company wants to switch to a different cloud provider, they may face significant challenges because they would need to reconfigure or rewrite serverless functions, APIs, and other integrations to fit the new provider's environment. This dependency on a single provider, known as vendor lock-in, can limit flexibility, making it harder to switch vendors for better deals or to distribute workloads across multiple clouds.
2. **Cold Starts:** When a serverless function is inactive for a while, it experiences a "cold start" when reactivated, leading to slower application performance. This happens because the cloud provider needs to allocate resources and initialize the function before it can run.
3. **Monitoring and Debugging:** Monitoring and debugging serverless applications is challenging because they consist of many small, independent functions spread across different environments. This distributed nature makes it hard to track and diagnose issues. Traditional monitoring tools that work well for cloud-native and container-based development often don't provide enough features or insights for serverless environments.
4. **Execution Times:** Serverless functions have limits on how long they can run (e.g., 15 minutes for AWS Lambda, 10 minutes for Azure Consumption Plan). These limits can be a problem for tasks that need more time and can also be costly.

WHEN TO ADOPT SERVERLESS ARCHITECTURE?

Serverless architecture in software development offers many benefits and efficiencies. However, no single approach, tool, or technology is perfect for all situations. Each has its strengths and weaknesses, and the best choice depends on the specific case. Here are scenarios where serverless architecture works best:

1. **Short-running functions:** Serverless functions, like AWS Lambda, run for a limited time (up to 15 minutes). This is usually enough for most tasks, but some applications may need more time.
2. **Un predictable workloads:** Serverless is ideal for workloads that change unpredictably throughout the day or have seasonal spikes. It can dynamically scale up or down to handle these changes efficiently.
3. **No Vendor lock-in concerns:** Major cloud providers like Azure, AWS, and Google Cloud Platform offer robust serverless platforms. However, these platforms are not compatible with each other, which could be a concern if you want to switch providers.

Case Study: Success Story with Serverless Architecture

Joot, a startup focused on optimizing social media and advertising image engagement through AI, utilized the Serverless Framework to handle web API, machine learning, and image processing workloads. This approach allowed Joot to significantly cut server costs by 70%, streamline development, and reduce devops tasks. They orchestrated complex processing pipelines, managed resources efficiently, and benefited from AWS services like Lambda, S3, and SageMaker. This resulted in rapid deployment, scalable infrastructure, and enhanced client insights, ultimately achieving higher ROI and social engagement [7].

Serverless Cloud Providers:
Several new players have entered the serverless space in recent years, catering to specific needs. However, their services are not as comprehensive as those offered by AWS, Microsoft Azure, or Google Cloud Platform
AWS Lambda: Amazon Web Services (AWS) is the largest cloud computing provider, offering a vast range of tools and services. AWS Lambda stands out with its extensive and well-documented support. It is trusted by high-profile customers like Netflix, Samsung and many more.
Azure Functions: Microsoft Azure is rapidly expanding its capabilities and market share, challenging AWS. While it may not match AWS feature-for-feature, Azure excels in .NET and TypeScript integration. It offers excellent documentation and community support.
Google Cloud Functions: Google Cloud Functions competes with AWS Lambda and Azure Functions, offering similar features. Google has invested heavily in clear, user-friendly documentation.

CONCLUSION

Serverless architecture is a major shift in how applications are created. It helps lower operational costs, improves scalability, speeds up time to market, and can be more cost-effective. When planning to migrate legacy apps or start new projects, considering serverless computing is important. To do this effectively, you need to understand the benefits, know how to implement it properly, and be aware of potential challenges like vendor lock-in, monitoring issues, security concerns, and cold starts, so you can address them correctly. Despite these challenges, the future of serverless technology looks bright. As cloud providers continue to improve their serverless offerings and more businesses adopt this model, we will see a significant transformation in how applications are developed, deployed, and scaled, preparing them for the next generation of digital experiences.

REFERENCES

[1]. https://www.oracle.com/cloud/cloud-native/functions/what-is-serverless/
[2]. https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith
[3]. https://www.doc.ic.ac.uk/~rbc/papers/fse-serverless-17.pdf
[4]. https://kruschecompany.com/serverless-architecture-for-modern-apps-providers-and-caveats/#Does_Serverless_make_DevOps_redundant
[5]. https://www.serverless.com/blog/serverless-architecture
[6]. https://www.datadoghq.com/knowledge-center/serverless-architecture/
[7]. https://www.serverless.com/case-studies/joot
[8]. https://www.semanticscholar.org/reader/6058e777ff668552a01141f110c03e8a32fa7349
[9]. https://www.researchgate.net/publication/337429660_The_rise_of_serverless_computing
[10]. https://aws.amazon.com/serverless/
[11]. https://azure.microsoft.com/en-us/solutions/serverless/
[12]. https://cloud.google.com/serverless