




## Serverless Computing: Advantages, Limitations and Use Cases

Vamsi Krishna Thatikonda    
8921 Satterlee Ave Se, Snoqualmie, WA

### Suggested Citation

Thatikonda, V.K. (2023).  
Serverless Computing: Advantages,  
Limitations and Use Cases.  
*European Journal of Theoretical and  
Applied Sciences*, 1(5), 341-347.  
DOI: [10.59324/ejtas.2023.1\(5\).25](https://doi.org/10.59324/ejtas.2023.1(5).25)

### Abstract:

Serverless computing, also known simply as "serverless," represents a significant transformation in how applications are designed, deployed, and executed. Contrary to traditional methods where developers manage servers, serverless allows them to focus only on coding, trusting cloud providers with infrastructure responsibilities. This paradigm shift results in dynamic resource allocation, where users are billed based on actual resource consumption rather than pre-allocated capacities. The article elucidates serverless computing's

nature, highlighting its core components - Function as a Service (FaaS) and Backend as a Service (BaaS). The benefits of serverless are underscored, including cost efficiency, inherent scalability, rapid development, and reduced operational demands. Yet, it is not without limitations. Concerns such as "cold starts," potential vendor lock-in, restricted customization, and specific security vulnerabilities are discussed. Practical serverless applications include web applications, data processing, IoT backends, chatbots, and ephemeral tasks. In conclusion, while serverless computing heralds a new age in cloud technology, businesses are encouraged to discerningly evaluate its pros and cons, mainly as the landscape evolves. The future is serverless, prompting organizations to determine their readiness for this revolution.

**Keywords:** *Serverless Computing, Function as a Service (FaaS), Backend as a Service (BaaS), Scalability, Cold Starts, and Vendor Lock-in.*

### Introduction

Serverless computing, often called simply "serverless," represents a paradigm shift in how applications are developed, deployed, and executed. Instead of focusing on server management, developers rely on cloud providers to handle the infrastructure, allowing them to concentrate solely on the code (Castro et al., 2019). This computing model dynamically manages the allocation of machine resources, billing users based on the actual number of resources consumed by executions rather than on pre-purchased capacity units (Castro et al., 2019).

Consistent efforts to abstract have marked the evolution of computing paradigms and

simplified development processes. Traditional server-based infrastructure requires developers to manage, maintain, and scale servers, often leading to inefficiencies in resource use and increased operational costs. The transition to serverless is a response to these challenges. It offers a higher level of abstraction, removing the need to interact with the underlying infrastructure directly, expediting development cycles, and reducing overheads (Baldini et al., 2017).

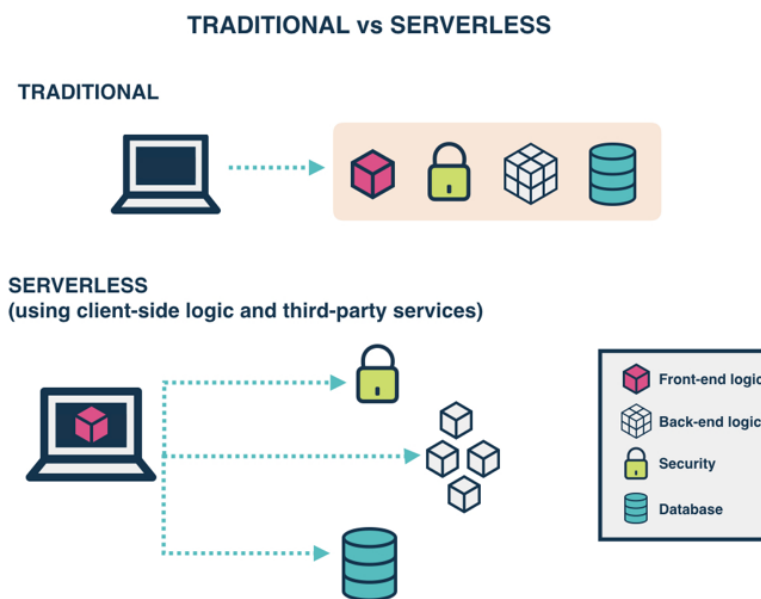
### What is Serverless Computing?

Despite its name, serverless computing does not imply the absence of servers. Instead, it



emphasizes the developer's detachment from server management, allowing them to focus exclusively on the application's functionality (Shafiei, Khonsari, & Mousavi, 2022). This approach relies on cloud service providers to dynamically allocate and execute backend code, billing users based on actual compute processes rather than predetermined server capacity (Shafiei, Khonsari, & Mousavi, 2022).

Traditional computing models necessitated intricate involvement in infrastructure setup, scaling, maintenance, and updates. On the other hand, serverless abstracts these complexities, offering developers an environment where the underlying infrastructure becomes invisible, handled entirely by the cloud provider (Shafiei, Khonsari, & Mousavi, 2022).



**Figure 1. Traditional vs Serverless Computing Schemes**

Two principal components of serverless architectures are Function as a Service (FaaS) and Backend as a Service (BaaS). FaaS involves executing code snippets in response to events, eliminating the need for long-running server processes. Examples include AWS Lambda and Azure Functions (Eismann et al., 2021). Conversely, BaaS provides ready-to-use backend services, such as databases or authentication processes, further simplifying application development (Eismann et al., 2021).

## Advantages of Serverless Computing

### Cost Efficiency

One of the most enticing attributes of serverless computing is its cost efficiency. The pay-as-you-

go model ensures businesses are billed based on actual resource consumption rather than allocated server capacity (Schleier-Smith et al., 2021). This approach eliminates the expenditure tied to idle server time, allowing companies to optimize costs in real-time. Furthermore, the absence of a dedicated infrastructure to manage translates to a significant decrease in associated overheads. Organizations adopting serverless reported a 60% reduction in infrastructure management costs compared to traditional cloud models (Li et al., 2021). Further, the one-time deployment cost of Serverless computing is 68% less than server-based computing, as seen in the following table.

## Scalability

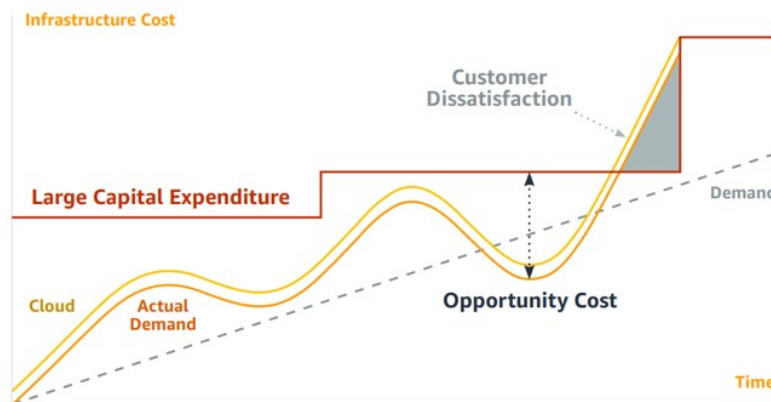
Serverless computing offers inherent scalability. Cloud providers automatically handle scaling based on the application's demand, ensuring seamless operation without manual intervention (Cloudflare, n.a.). This feature is particularly

beneficial for applications that experience fluctuating traffic loads. For instance, e-commerce sites during sale events or startups experiencing rapid user growth can leverage serverless to meet dynamic demands without incurring the costs and challenges of pre-scaling.

**Table 1. Comparison of Deployment Cost**

One-time Development Cost for Server-based (EC2) Compared with Serverless (QWS lambda)			
Development	Server-based	Serverless	Difference
Days to Deploy	~25 days	~8 days	~17 days
One-time Upfront	\$38,300	\$12,300	\$26,000
Monthly Cost	\$640	\$205	\$(435)
			Serverless is 68% cheaper than server-based

**Source:** Arora, Tayal, & Sembhi, (2021)



**Figure 1. Benefits of Cost Savings in Dynamic Scaling**

**Source:** Arora, Tayal, & Sembhi, (2021)

## Improved Development Speed

Serverless architectures speed up software development (Rosenbaum, 2017). Without the need to manage infrastructure, developers can focus solely on writing and refining code, drastically reducing development cycles (Castro et al., 2019). Additionally, with the immediate execution environment offered by serverless platforms, rapid deployment, and iteration become the norm. This quick turnaround means businesses can respond faster to market changes, user feedback, and emerging trends.

## Reduced Operational Overheads

The operational demands in traditional computing models, ranging from server management to software patching, often divert valuable time and resources from core business objectives. By outsourcing these tasks to cloud providers, serverless computing essentially eradicates these overheads (Castro et al., 2019). Deployment becomes a straightforward process, devoid of the complexities tied to server configurations. Moreover, rollback, reverting to previous versions of applications in case of errors or issues, is vastly simplified (Lloyd, 2022). This ensures businesses can maintain continuity in face of challenges.

## Limitations and Concerns of Serverless Computing

While serverless computing offers numerous advantages, it's essential to understand the associated challenges and concerns.

### Cold Starts

One of the most discussed limitations in serverless computing is the phenomenon of cold starts. A cold start occurs when a new instance of a function is initiated, and there's a latency involved before the Function begins executing, primarily because the environment needs to be set up from scratch (Vahidinia, Farahani, & Aliee, 2020). This latency can affect performance, particularly in applications where swift response times are critical. For user-facing applications, the additional latency introduced by cold starts can result in sub-optimal user experiences, potentially impacting user retention and engagement.

### Vendor Lock-in

Adopting serverless architectures often means aligning with a specific provider's toolset, infrastructure, and services. This dependence can lead to vendor lock-in, where migrating to a different platform becomes time-consuming and costly due to incompatible interfaces and configurations (Taibi, Spillner, & Wawruch, 2021). This lock-in can limit flexibility, making it challenging for organizations to adapt to evolving business needs or leverage better offerings in the market.

### Limited Customization and Control

As serverless platforms manage the underlying infrastructure, they often impose certain restrictions on runtime environments, memory limits, and execution durations. This can limit the scope for deep customization, which might be essential for some niche applications. While simplifying many operations, serverless frameworks may not cater to highly specialized use cases, necessitating workarounds or alternative solutions (Baldini et al., 2017).

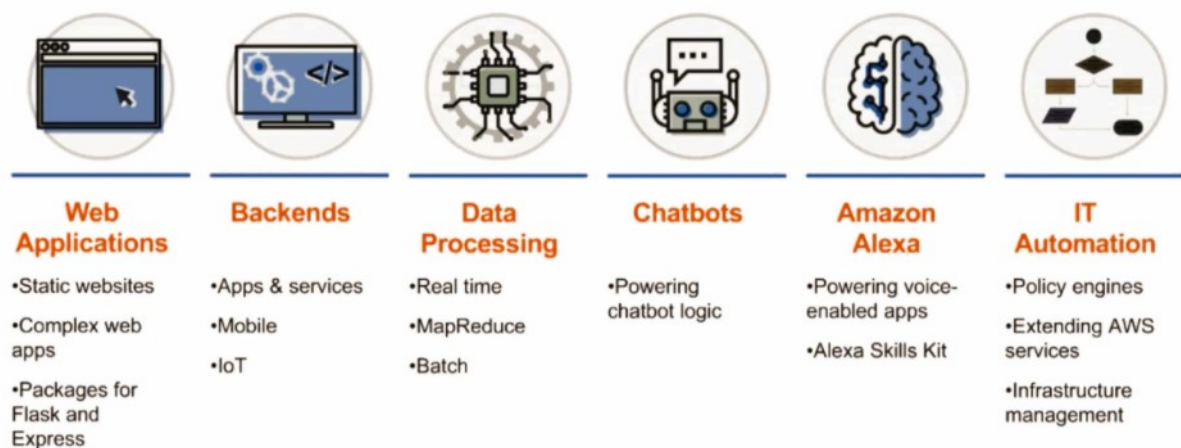


Figure 2. Use Case of Serverless Computing

Source: Adam, J. (2022)

### Security Concerns

With serverless, while the provider primarily handles infrastructure security, the application's

security remains a shared responsibility. This model means developers are responsible for securing their application code, configurations,

and third-party libraries (Marin, Perino, & Di Pietro, 2022). Serverless environments may also introduce unique security vulnerabilities. For instance, the dynamic nature of serverless may expose applications to event-data injections or potential unauthorized function invocations (Marin, Perino, & Di Pietro, 2022). Functions, especially those interacting with other cloud services, can become possible entry points for attackers, making a solid case for rigorous security practices in serverless architectures (O'Meara, & Lennon, 2020).

## Use Cases for Serverless Computing

The use cases for serverless computing are numerous; however, the four prominent use cases are web applications, data processing, IoT Backends, Chatbots and Services, and Ephemeral tasks.

### Web Applications

Serverless computing has proven instrumental in the modern web development. Web applications, particularly those requiring dynamic content generation, can leverage serverless functions to serve content in real-time without server management overhead (Castro et al., 2019). This approach offers scalability, ensuring the application remains responsive during traffic surges. Leveraging serverless, developers can focus solely on front-end development, knowing that the serverless provider optimizes and maintains the backend processes.

### Data Processing

The sheer volume and velocity of data demand efficient processing solutions in today's digital world. Serverless frameworks cater to this need by enabling data transformation, analysis, and processing (Goli et al., 2020). As data streams in, serverless functions are invoked to process and analyze the data instantaneously, making it especially beneficial for applications requiring immediate insights. For instance, serverless computing systems can be applied in the financial sector to fasten data processes while reducing costs (Goli et al., 2020).

## IoT Backend

The Internet of Things (IoT) is characterized by sporadic data bursts from numerous devices. Managing a consistent server setup for such erratic data inflows can be cumbersome. Serverless platforms shine here, seamlessly handling sudden data influxes and ensuring efficient processing and storage without manual intervention (Cicconetti, Conti, & Passarella, 2021).

## Chatbots and AI Services

In customer service and support, AI-powered chatbots have gained immense traction. Serverless provides an ideal environment for these chatbots, ensuring quick responses by eliminating the need for maintaining a complete backend (Lehvä, Mäkitalo, & Mikkonen, 2018). The inherent scalability of serverless ensures that these bots can cater to many users simultaneously, offering consistent performance.

## Ephemeral Tasks

Intermittent Jobs, such as CRON jobs or other scheduled tasks, can be inefficient on traditional servers due to resource wastage during idle times. With serverless, these tasks can be executed on-demand, ensuring optimal resource usage and cost efficiency (Lynn et al., 2017).

## Conclusion

With its dynamic scalability and cost-efficiency, serverless computing marks a transformative shift in cloud technology. While the advantages are compelling, businesses must also weigh the challenges, particularly in customization and potential vendor lock-in. As technological innovations advance, serverless is poised to evolve, potentially addressing its current limitations. The coming years might see a convergence of serverless with other technologies, ushering in a new age of integrated digital solutions. The call is clear for businesses eyeing the future: delve deeper, critically assess, and determine if serverless aligns with your strategic objectives and technological



aspirations. The future is serverless; the question remains, are you ready for it?

## References

- Adam, J. (2022). Serverless architecture for increased Dev Productivity. K&C. Retrieved from <https://kruschecompany.com/serverless-architecture-for-modern-apps-providers-and-caveats/>
- Arora, G., Tayal, A. & Sembhi, R. (2021). Determining the Total Cost of Ownership: Comparing Serverless and Server-based Technologies. Retrieved from [https://www.strategist-hub.com/pdfs/203755\\_AWS\\_MAD\\_TCO\\_eBook\\_2021\\_Final.pdf](https://www.strategist-hub.com/pdfs/203755_AWS_MAD_TCO_eBook_2021_Final.pdf)
- Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Isahagian, V., ... & Suter, P. (2017). Serverless Computing: Current Trends and Open Problems. In: Chaudhary, S., Somani, G., Buyya, R. (eds) *Research Advances in Cloud Computing*. Springer, Singapore. [https://doi.org/10.1007/978-981-10-5026-8\\_1](https://doi.org/10.1007/978-981-10-5026-8_1)
- Castro, P., Ishakian, V., Muthusamy, V., & Slominski, A. (2019). The rise of serverless computing. *Communications of the ACM*, 62(12), 44–54. <https://doi.org/10.1145/3368454>
- Cicconetti, C., Conti, M. & Passarella, A. (2021). A decentralized framework for serverless edge computing in the internet of things. *IEEE Transactions on Network and Service Management*, 18(2) 2166–2180. <https://doi.org/10.1109/tnsm.2020.3023305>
- Cloudflare. (n.a.). Why use serverless computing? Pros and cons of serverless. Retrieved from <https://www.cloudflare.com/learning/serverless/why-use-serverless/>
- Eismann, S., Scheuner, J., Eyk, E., Schwinger, M., Grohmann, J., Abad, C., & Iosup, A. (2021). Serverless applications: Why, when, and how?," *IEEE Software*, 38(1), 32–39. <https://doi.org/10.1109/ms.2020.3023302>
- Goli, A., Hajihassani, O., Khazaei, H., Ardakanian, O., Rashidi, M. & Dauphinee, T. (2020). *Migrating from monolithic to serverless: A fintech case study*. Companion of the ACM/SPEC International Conference on Performance Engineering. <https://doi.org/10.1145/3375555.3384380>
- Lee, H., Satyam, K. & Fox, G. (2018). *Evaluation of production serverless computing environments*. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). <https://doi.org/10.1109/cloud.2018.00062>
- Lehvä, J., Mäkitalo, N., & Mikkonen, T. (2018). Case study: Building a serverless messenger chatbot. *Current Trends in Web Engineering*, 75–86. [https://doi.org/10.1007/978-3-319-74433-9\\_6](https://doi.org/10.1007/978-3-319-74433-9_6)
- Li, Z., Tan, Y., Li, B., Zhang, J. & Wang, X. (2021). A survey of cost optimization in serverless cloud computing. *Journal of Physics: Conference Series*, 1802(3), 032070. <https://doi.org/10.1088/1742-6596/1802/3/032070>
- Lloyd, J. (2022). Containers and Serverless. In *Infrastructure Leader's Guide to Google Cloud: Lead Your Organization's Google Cloud Adoption, migration and modernization journey*. New York: Apress.
- Lynn, T., Rosati, P., Lejeune, A. & Emeakaroha, V. (2017). *A preliminary review of Enterprise Serverless Cloud Computing (Function-as-a-service) platforms*. In 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). <https://doi.org/10.1109/cloudcom.2017.15>
- Marin, E., Perino, D. & Di Pietro, R. (2022). Serverless Computing: A security perspective. *Journal of Cloud Computing*, 11(1). <https://doi.org/10.1186/s13677-022-00347-w>
- O'Meara, W. & Lennon, R.G. (2020). *Serverless Computing Security: Protecting Application logic*. In 2020 31st Irish Signals and Systems Conference (ISSC). <https://doi.org/10.1109/issc49989.2020.9180214>
- Rosenbaum, S. (2017). *Serverless Computing in Azure with .NET*. Packt Publishing Ltd..
- Schleier-Smith, J., Sreekanti, V., Khandelwal, A., Carreira, J., Yadwadkar, N.J., Popa, R.A., Gonzalez, J.E., Stoica, I., & Patterson, D.A.

(2021). What serverless computing is and should become: The next phase of cloud computing. *Communications of the ACM*, 64(5), 76–84. <https://doi.org/10.1145/3406011>

Shafiei, H., Khonsari, A. & Mousavi, P. (2022). Serverless Computing: A Survey of Opportunities, Challenges, and Applications. *ACM Computing Survey*, 54(11s), 239. <https://doi.org/10.1145/3510611><https://doi.org/10.31224/osf.io/u8xth>

Taibi, D., Spillner, J. & Wawruch, K. (2021). Serverless Computing-Where are we now, and where are we heading? *IEEE Software*, 38(1), 25–31. <https://doi.org/10.1109/ms.2020.3028708>

Vahidinia, P., Farahani, B. & Aliee, F.S. (2020). *Cold start in Serverless Computing: Current trends and mitigation strategies*. In 2020 International Conference on Omni-layer Intelligent Systems (COINS).

<https://doi.org/10.1109/coins49042.2020.9191377>

Weston, R. (n.a.). Serverless architectures and continuous delivery. GoCD. Retrieved from <https://www.gocd.org/2017/06/26/serverless-architecture-continuous-delivery/>