WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Serverless Computing in Software Engineering

K. Jayasree, Mr. B. Panna Lal

[1,] PG Scholar, Dept. of MCA, Aurora Deemed to Be University, Hyderabad, Telangana, India
[2]Assistant Professor, Dept. of MCA, Aurora Deemed to Be University, Hyderabad, Telangana, India
Email: k.nagjayasree09@gmail.com[1], pannalal@aurora.edu.in[2]

1. ABSTRACT :

Serverless computing is a modern way of building and running software without worrying about managing servers or infrastructure. In traditional systems, developers had to spend a lot of time setting up servers, scaling them, and maintaining them.

With serverless computing, all of that is handled automatically by the cloud provider. Developers can simply focus on writing the code, and the platform takes care of running it whenever needed.

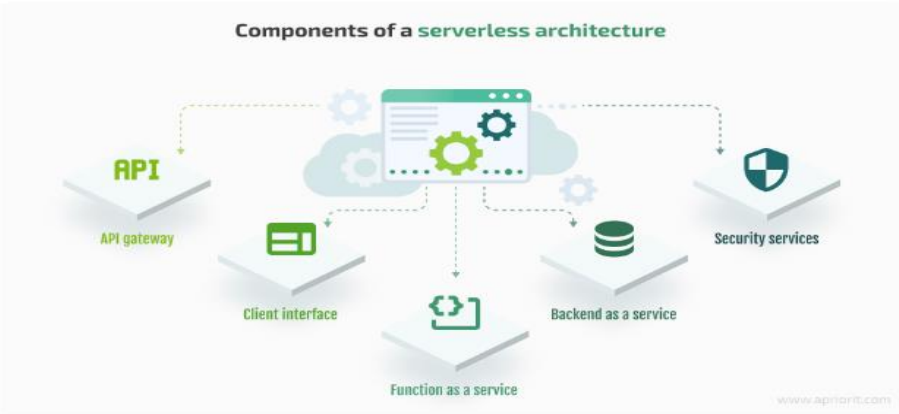
This makes applications more flexible, cost-efficient, and faster to build. Serverless works well for tasks like handling user requests, processing data, or connecting different services together. However, it also comes with challenges such as limited control over the system and potential vendor lock-in.

Overall, serverless computing is becoming an important part of software engineering because it allows teams to innovate quickly while reducing the burden of managing complex infrastructure.

2.Introduction

In today's world, software applications need to run fast, handle many users at once, and still be affordable. Traditionally, this meant developers had to manage servers—setting them up, monitoring them, and scaling them when traffic increased. This often took a lot of time and effort, sometimes more than writing the actual application itself.
Serverless computing changes this by removing the need to directly manage servers. Instead, developers only write the code, and the cloud provider automatically takes care of running it, scaling it up when more users come in, and scaling it down when not much is happening. This makes development simpler, faster, and cost-effective because you only pay for the actual usage, not for idle servers.
Serverless has become popular in modern software engineering because it allows teams to focus more on building features and solving problems for users, rather than worrying about infrastructure. It is especially useful for applications that need flexibility, quick updates, or event-driven tasks like handling user clicks, processing files, or responding to messages.

3. Architecture of Serverless Computing



The architecture of serverless computing is built around the idea that developers focus only on writing code, while the cloud provider manages everything

else behind the scenes. It usually has the following main parts:

3.1 Client (User Side):

This is where requests come from (like a mobile app, web browser, or IoT device). For example, a user clicks a button on a shopping app.

3.2 API Gateway:

Acts as the "front door" of the serverless system. It receives the user's request and passes it to the right function. Example: When a user places an order, the API Gateway routes the request to the "place Order" function.

3.3 Function-as-a-Service (FaaS)

This is the heart of serverless computing. Developers write small, independent pieces of code called *functions*. These functions only run when triggered (event-driven). Example: A "process Payment" function runs only when someone makes a payment.

3.4 Backend Services (Managed by Cloud Provider)

Database, storage, authentication, and messaging services are provided as ready-to-use building blocks. Example: Amazon DynamoDB for databases, Google Cloud Storage for files, or Firebase Authentication for login.

3.5 Event Sources / Triggers

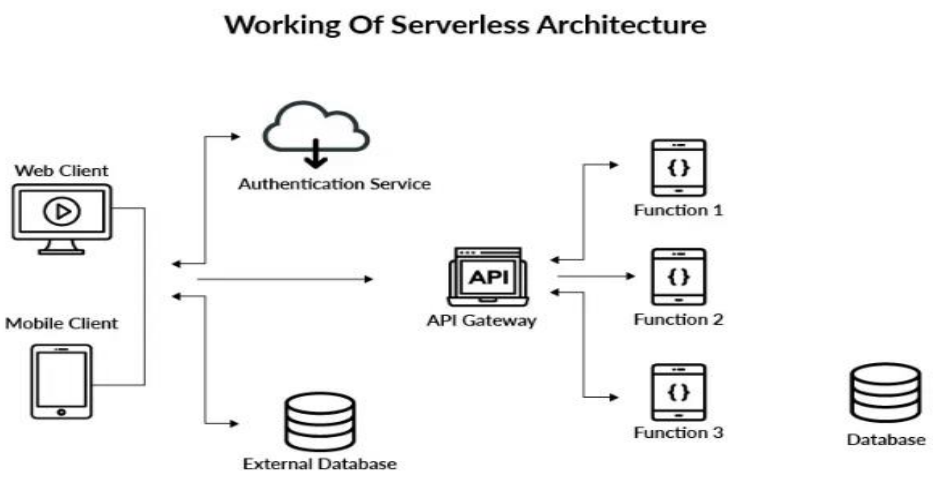
Functions are triggered by events like: A file uploaded to storage. A new database entry. A user clicking a button. Example: Uploading a photo triggers a function to resize the image.

3.6 Monitoring and Scaling (Automatic)

The cloud provider automatically scales functions up (when many requests come) and down (when requests stop). Monitoring tools keep track of performance and errors.

4. Workflow of Serverless computing

Serverless computing works on an event-driven model, where small pieces of code known as functions are executed only when specific events occur, such as a user clicking a button, uploading a file, or making a payment.



When such an action happens, the request is sent to the cloud platform, which automatically triggers the right function to handle that task—for instance, uploading a photo may trigger a function that resizes the image and saves it to a database. Once the function completes its job, it shuts down automatically, ensuring no idle resources are wasted.

This model allows applications to instantly scale up to handle thousands of requests in parallel and scale down when demand decreases, without requiring manual intervention. Since developers are freed from managing servers, operating systems, or scaling strategies, they can focus purely on writing business logic, while organizations benefit from reduced costs by paying only for the exact compute time used.

Overall, serverless computing makes building modern, responsive, and highly scalable applications both efficient and cost-effective, while simplifying operations and accelerating innovation.

5. Advantages and disadvantages

5.1 Advantages of Serverless Computing

- No server management – Developers don't need to set up, maintain, or scale servers; the cloud handles it.
- Cost-effective – You only pay for the actual execution time of functions, not for idle servers.

- Automatic scaling – If 1 or 1,000 requests come in, the cloud automatically adjusts resources.
- Faster development – Developers can focus on writing features instead of worrying about infrastructure.
- Event-driven – Works great for tasks like file uploads, payments, notifications, or data processing.
- High availability – Cloud providers ensure that functions run reliably across multiple servers.

5.2 Disadvantages of Serverless Computing

- Limited control – Since the cloud provider manages everything, developers have less control over the environment.
- Cold start problem – Functions that haven't been used for a while may take a few seconds to start.
- Vendor lock-in – Applications often depend heavily on one cloud provider's tools, making switching difficult.
- Not ideal for long-running tasks – Functions are designed for short jobs, so long or complex processes may not work well.
- Debugging challenges – Since the system is distributed and hidden behind the cloud, testing and troubleshooting can be harder.
- Security concerns – Sharing infrastructure with other users may introduce risks if not managed properly.

6. Serverless v/s other cloud backends

Serverless computing stands apart from other cloud backend models like IaaS and PaaS by offering the least amount of infrastructure management and the highest level of abstraction for developers. In IaaS, developers rent virtual machines and must handle tasks such as server setup, operating system updates, security patches, and scaling, which provides flexibility but requires significant effort.

PaaS simplifies this by offering a managed environment where applications can be deployed without dealing with the underlying infrastructure, though developers still manage the runtime and overall application logic. Serverless goes even further by removing server management entirely—developers only write small, event-driven functions that automatically execute when triggered, with scaling and resource allocation fully handled by the provider.

This makes serverless lightweight, cost-efficient, and highly scalable, enabling rapid innovation, but it also brings trade-offs such as limited control over infrastructure, dependence on a single provider, and unsuitability for long-running or resource-heavy workloads.

7. Conclusion

Serverless computing is transforming the landscape of software engineering by shifting the focus from managing infrastructure to simply writing and deploying code. In this model, developers no longer need to worry about provisioning servers, handling maintenance, or planning for scaling, as all of this is automatically managed by the cloud provider. This brings clear benefits such as faster development cycles, cost efficiency by paying only for the exact compute resources used, and effortless scalability that adjusts to varying workloads in real time.

It is particularly well-suited for event-driven applications like chatbots, real-time notifications, and on-demand data processing. However, serverless also introduces some challenges, including limited control over the underlying infrastructure, vendor lock-in that makes switching providers difficult, and unsuitability for long-running or resource-intensive processes.

Despite these trade-offs, the serverless model has gained strong importance in modern software engineering because it encourages innovation, reduces operational complexity, and enables building applications that are agile, resilient, and capable of adapting quickly to the dynamic needs of today's digital world.

8. REFERENCES

1. https://www.ijert.org/research/a-research-paper-on-serverless-computing-IJERTV11IS090064.pdf
2. https://arxiv.org/pdf/2207.13263
3. https://www.ijfmr.com/papers/2024/6/30737.pdf