International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Serverless Application Deployment Using AWS Lambda and API Gateway.

Dr. Rasmi A[1], Dilli Ram Shahu[2], Prakash Timilsina[3], Siddharth Singh[4]

[1]Associate Professor,[2]Final year student,[3]Final year student,[4] Final year student
Department of Information Science Engineering. RR Institute of Technology. Bengaluru

---

Abstract - Serverless computing is transforming the way applications are developed and deployed by abstracting the complexities of server management. This paper presents the design and implementation of a serverless application using AWS Lambda and API Gateway. The application demonstrates how cloud-native features like scalability, reduced operational overhead, and cost efficiency can be achieved through serverless architecture. The methodology focuses on event-driven function deployment, RESTful API exposure, and integration with AWS services such as DynamoDB. The paper includes system architecture, implementation strategies, sequence and activity diagrams, testing results, and a discussion on performance. We conclude with future directions to improve serverless applications in terms of latency and observability.

As enterprises seek to accelerate application development cycles while minimizing infrastructure concerns, serverless computing emerges as a viable solution. This study explores the design, development, and deployment of a serverless application using Amazon Web Services (AWS) Lambda and API Gateway. The solution demonstrates a real-world use case—a task management REST API—integrated with AWS services to offer seamless, on-demand execution of backend logic without server maintenance. Emphasis is placed on architectural design, cloud integration, performance analysis, and deployment best practices.

Key Words: Serverless computing, AWS lambda, Amazon API gateway, Cloud-Native Architecture, Cloud deployment

1.INTRODUCTION

Serverless computing allows developers to build and run applications without managing servers. AWS Lambda, in combination with Amazon API Gateway, provides a robust platform for serverless application development. The purpose of this project is to demonstrate a practical serverless architecture that can process HTTP requests, execute logic via AWS Lambda, and persist data using Amazon DynamoDB.

The traditional server-based architecture often requires provisioning, scaling, and maintaining server instances. Serverless removes this complexity and offers automatic scaling, pay-per-use billing, and enhanced developer productivity.

The evolution of cloud computing has fundamentally changed how applications are developed, deployed, and maintained. Traditional architectures, which relied heavily on provisioning, scaling, and managing servers, often imposed significant operational overhead and infrastructure costs. Serverless computing has emerged as a powerful paradigm, enabling developers to focus solely on writing code while cloud providers handle the underlying infrastructure. Serverless architecture, also known as Function-as-a-Service (FaaS), allows developers to write functions that are executed in response to events. These functions are ephemeral, stateless, and automatically scaled by the cloud provider. In this model, users are only billed for the actual execution time of their code, making it highly cost-effective and resource-efficient. Amazon Web Services (AWS), a leader in cloud technologies, offers AWS Lambda as its FaaS solution. Lambda allows developers to run backend code without provisioning or managing servers. It automatically handles scaling, fault tolerance, and high availability. Amazon API Gateway complements Lambda by acting as a front door for applications, enabling developers to expose Lambda functions as RESTful APIs that can be accessed securely via HTTP(S).

2. METHODOLOGY

The development and deployment of the serverless application followed a structured methodology encompassing design, implementation, integration, and testing phases. The primary goal was to build a lightweight and scalable RESTful API using AWS Lambda and Amazon API Gateway, backed by Amazon DynamoDB for data storage. The application chosen for this implementation was a basic task management system that performs CRUD (Create, Read, Update, Delete) operations on task data.
This methodology ensures a modular, testable, and highly scalable cloud-native application that adheres to serverless best practices. The resulting system architecture not only reduces operational complexity but also demonstrates the viability of serverless models for real-world web applications. The first step involved designing the API endpoints and data model. Each endpoint, such as POST /tasks, GET /tasks/{id}, PUT /tasks/{id}, and DELETE /tasks/{id}.

3. SOFTWARE OVERVIEW

1. The serverless application was developed using Python 3.9, chosen for its simplicity, fast development cycle, and strong support within the AWS ecosystem.
2. The application is hosted on Amazon Web Services (AWS), leveraging a suite of integrated services to build a fully serverless and scalable system.
3. For data persistence, the application uses Amazon DynamoDB, a highly scalable NoSQL database service.
4. Security and permissions are handled using AWS Identity and Access Management (IAM).
5. Development and deployment were carried out using Visual Studio Code as the primary IDE, while AWS Management Console and AWS CLI were used to manage cloud resources.
6. The core service, AWS Lambda, is responsible for executing backend code in response to HTTP requests.

7. To expose the Lambda functions as RESTful API endpoints, Amazon API Gateway was used.

4. SEQUENCE DIAGRAM AND ACTIVITY DIAGRAM

Imagine a serverless web application that allows users to upload images and store them in an Amazon S3 bucket. When an image is uploaded, a Lambda function is triggered to perform image processing and store the processed image in another S3 bucket.

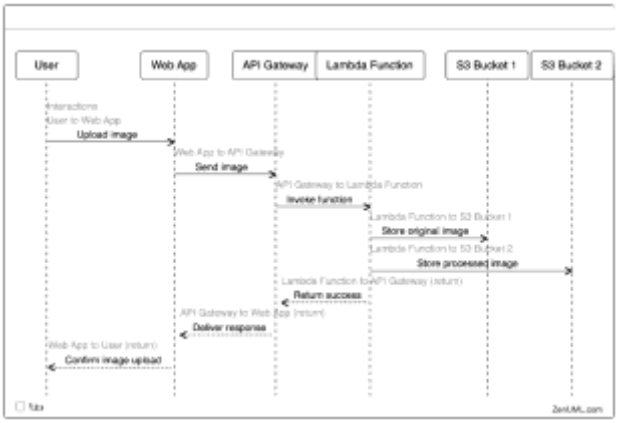

Fig 1. Sequence Diagram

STATE DIAGRAM

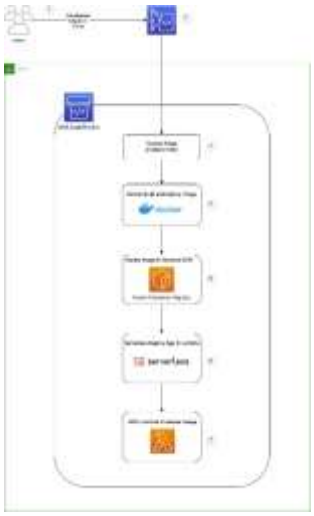

Fig 1.2. Activity Diagram

5. RESULTS AND DISCUSSION

The serverless application was successfully deployed and tested on AWS using Lambda, API Gateway, and DynamoDB. The system met all functional requirements, including the ability to handle CRUD operations through RESTful API endpoints. Each endpoint correctly triggered the corresponding Lambda function, which executed backend logic and interacted with the DynamoDB database to store or retrieve task data. Testing with Postman confirmed that all endpoints responded appropriately with correct status codes and data formatting.

Performance testing showed that the application responded to requests with low latency under moderate load conditions. Most requests completed within 300–500 milliseconds, which is typical for Lambda-backed APIs. However, cold start delays were occasionally observed when invoking a Lambda function after a period of inactivity, especially for the first request. These delays ranged from 200ms to 1 second, depending on the runtime environment and function configuration. While this is an expected limitation of serverless architectures, its impact was minimal for non-real-time use cases.

Scalability was a key highlight. The application automatically scaled to handle concurrent requests without any manual intervention.

6. FUTURE WORK

While the current implementation of the serverless application demonstrates the viability and efficiency of using AWS Lambda and API Gateway, there are several areas that can be enhanced and expanded in future iterations. One key improvement is the integration of authentication and authorization mechanisms, such as AWS Cognito or OAuth 2.0, to secure access to API endpoints and support user-specific data handling. Adding a frontend interface, either as a static website hosted on Amazon S3 or as a React-based single-page application (SPA), would also make the solution more user-friendly and complete.

In terms of functionality, the application can be extended to include scheduled tasks using Amazon EventBridge (formerly CloudWatch Events), enabling time-based execution for automated operations like cleanup routines or reminders. Moreover, implementing detailed logging, tracing, and monitoring using tools like AWS X-Ray can provide deeper insights into performance bottlenecks and help in optimizing cold start times.

7. CONCLUSION

This project successfully demonstrated the deployment of a serverless application using AWS Lambda and API Gateway, showcasing the core advantages of serverless computing such as automatic scaling, reduced operational overhead, and cost efficiency.

Major contributions

This project illustrates the practical deployment of a fully serverless application using AWS Lambda and API Gateway, effectively removing the need for traditional server management and provisioning.

Future Enhancements

Integrate authentication and authorization mechanisms such as AWS Cognito or OAuth 2.0 to secure API endpoints and enable personalized user experiences.

8. ACKNOWLEDGMENT

The satisfaction that accompanies the success in completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made our efforts go in vain. We consider ourselves privileged to express gratitude and respect towards all those who guided us through the completion of this project.

We would like to express our gratitude to **Dr. Mahendra K V**, Principal, RRIT, Bengaluru for providing us congenial environment and surrounding to study.

We would like to express our sincere gratitude to **Dr. Erappa G**, Professor and Head, Department of Information Science and Engineering, RRIT, Bengaluru for giving us the support and encouragement that was necessary for the completion of this project.

We express our deepest gratitude and sincere thanks to our project coordinator**, Dr. Rasmi A**, Associate Professor, Department of Information Science and Engineering, RRIT, Bengaluru for giving us the support and encouragement that was necessary for the completion of this project.

We would like to express our gratitude to our guide **Dr. Rasmi A**, Associate Professor, Department of Information Science and Engineering, RRIT, Bengaluru for giving us the support and encouragement that was necessary for the completion of this project.

We would also like to convey our regards to all faculty members and non-teaching staff of R R Institute of Technology, Bengaluru for constantly motivating and guiding us throughout our journey at RRIT.

Finally, we thank our Parents, Friends and Family members for their co-operation and their guidance in bringing out this project successfully.

9. REFERENCES

[1] **Amazon Web Services, Inc.** (2023). *AWS Lambda Developer Guide*. Retrieved from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[2] **Amazon Web Services, Inc.** (2023). *Amazon API Gateway Developer Guide*. Retrieved from https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

[3] Adzic, G., & Chatley, R. (2017). *Serverless Computing: Economic and Architectural Impact*. IEEE Cloud Computing, 4(5), 16–23. https://doi.org/10.1109/MCC.2017.4121217.

[4] Roberts, M. (2016). *Serverless Architectures*. Retrieved from https://martinfowler.com/articles/serverless.html.

[5] Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C. C., Khandelwal, A., Pu, Q., ... & Stoica, I. (2019). Cloud Programming Simplified: A Berkeley View on Serverless Computing. *arXiv preprint arXiv:1902.03383*. https://arxiv.org/abs/1902.03383.

[6] Shahrad, M., et al. (2019). Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. *Proceedings of the 2020 USENIX Annual Technical Conference*, 205–218. https://www.usenix.org/conference/atc20/presentation/shahrad

[7] Harris, R. (2020). *AWS Certified Solutions Architect Official Study Guide*. Wiley.

[8] Mase, K., & Chen, T. (2019). Building Scalable Applications with AWS Lambda and API Gateway. *International Journal of Computer Applications*, 178(6), 12–17.

[9] Wang, L., Ristenpart, T., & Swift, M. (2018). *Peeking Behind the Curtains of Serverless Platforms*. In Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC '18), 133–146. https://www.usenix.org/conference/atc18/presentation/wang-liang.

[10] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., ... & Suter, P. (2017). *Serverless Computing: Current Trends and Open Problems*. In Research Advances in Cloud Computing (pp. 1–20). Springer. https://doi.org/10.1007/978-3-319-64688-9_1.