iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** XI **Month of publication:** November 2023

DOI: https://doi.org/10.22214/ijraset.2023.56985

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XI Nov 2023- Available at www.ijraset.com

Creating Serverless Web Applications with AWS

Varun Singh[1], Chirag Acharya[2], Anisha jain[3], Karishma Gupta[4], Dr. Khushbu Wajari[5]
Masters of Computer Applications, Ajeenkya D Y Patil University, Pune, India

Abstract: ***Serverless computing has emerged as a groundbreaking innovation, allowing organizations to alleviate the complexities associated with provisioning, scaling, and managing infrastructure. The surge in serverless adoption poses challenges in choosing optimal solutions amid a myriad of proposed designs by experts. This paper categorizes serverless patterns using a multivocal literature review, highlighting benefits and challenges in coordination, aggregation, event management, availability, communication, and validation. To illustrate the practical application of serverless computing, we present a case study involving the use of AWS Lambda, Amazon DynamoDB, and other services to process real-time data streams for a fictional ride-sharing company. The study showcases the advantages of serverless computing, emphasizing its ability to free developers from server-related concerns, enabling them to focus on building scalable and reliable applications. This paper contributes to the understanding of serverless computing's role in the broader context of cloud technologies and its application in real-world scenarios.***

Keywords: Serverless computing, literature review, AWS Lambda, cloud computing, real-time data processing, scalability, infrastructure, cloud services, application architecture.

I. INTRODUCTION

The proliferation of conversational UI-based applications by industry giants such as WhatsApp, Instagram, and Facebook have spurred a surge in interest regarding the integration of relevant technologies to enhance business operations across diverse industry domains. As businesses explore innovative approaches, the latest trend in cloud computing, namely serverless architecture, has gained prominence.

This model, characterized by the cloud provider managing server provision and operations, eliminates the need for users to maintain, monitor, and schedule servers. This paper presents insights from a month-long case study wherein a chat application was developed using Amazon Web Services' (AWS) Lambda framework, a rapidly evolving serverless computing platform.

The study delves into various AWS services and components, aiming to contribute to the understanding of serverless computing's practical applications and benefits. The research review, through a multivocal literature review process, categorizes serverless patterns, offering practitioners valuable insights into their potential advantages and challenges. By examining both peer-reviewed and grey literature, the paper classifies patterns in coordination, aggregation, event management, availability, communication, and validation. The subsequent sections provide a comprehensive exploration of serverless computing, its impact on application development, and its role within the broader landscape of cloud technologies.

II. LITERATURE REVIEW

Security and authentication stand out as pivotal concerns in the realm of serverless computing. This research delves into the diverse techniques offered by AWS, including Single Sign-On sessions and OTP-based authentication, to ascertain their reliability and enhanced security compared to HTTP-based methods. Additionally, the study thoroughly examines the cloud implications associated with serverless architecture, exploring facets like infrastructure elasticity, load balancing, provisioning variation, and considerations related to infrastructure and memory reservation size. A key contribution of this investigation lies in proposing measures to implement abstraction within serverless architecture, facilitating efficient cross-server data management, resource allocation, and isolation, among other critical functionalities.

Within the domain of serverless computing, an extensive exploration of different frameworks tailored to specific real-life application criteria is undertaken. The research also encompasses a comprehensive comparison of various cloud platforms, examining their distinctive features and functionalities. Furthermore, the study delves into interactive applications of Amazon cloud services, critically evaluating the viability of AWS offerings. Notably, the research identifies a substantial paradigm shift towards serverless cloud computing applications, driven by a meticulous infrastructure cost comparison that underscores the economic advantages of deploying web applications on AWS Lambda.

The literature review incorporates seminal works by Roberts and Chapin, providing foundational insights into the serverless space and pinpointing areas requiring refinement, such as vendor lock-in and state management. Lynn et al.'s examination scrutinizes Function as a Service (FaaS) offerings, challenging presumed benefits based on specific use cases. Additional perspectives from Adzic and Chatley focus on production applications successfully transitioning to serverless, emphasizing the cost-saving aspects. Eivy's study compares costs between running applications on virtual machines and serverless, advocating for thorough testing before migration. For microservices adoption, presenting solutions to challenges and emphasizing alternatives like microservices and containers that complement serverless solutions. This comprehensive literature review forms the foundation for a nuanced understanding of the multifaceted landscape of serverless computing, addressing its benefits, challenges, and potential alternatives.

III. METHODOLOGY

In the implementation of the proposed serverless architecture, various AWS components play integral roles, ensuring robust functionality and seamless deployment.

A. *Components of the Architecture*

1) *AWS Amplify:* Utilized as the control centre for full-stack web and mobile application deployments in AWS, AWS Amplify Console encompasses hosting and the Admin UI. Offering a git-based workflow for hosting serverless web apps, Amplify Console supports popular web frameworks and mobile platforms, facilitating continuous deployment. Its Admin UI serves as a visual interface for frontend developers, enabling the creation and management of app backends outside the AWS Management Console. Amplify's support for CI/CD enhances the development and deployment process.
2) *Amazon Cognito:* Providing authentication, authorization, and user management for web and mobile apps, Amazon Cognito offers diverse sign-in options, including third-party providers like Facebook, Amazon, Google, or Apple. It consists of user pools for user directories and identity pools to grant users access to other AWS services, ensuring secure user authentication.
3) *Amazon API Gateway:* As the "front door" for applications to access data or functionality from backend services, Amazon API Gateway is a fully managed service facilitating the creation, publishing, maintenance, monitoring, and securing of APIs. Supporting RESTful APIs and WebSocket APIs, it efficiently handles tasks like traffic management, CORS support, authorization, and access control, ensuring seamless API operation.
4) *AWS Lambda:* Serving as a compute service, AWS Lambda enables running code without the need for provisioning or managing servers. Running code on a high-availability compute infrastructure, Lambda handles server and operating system maintenance, capacity provisioning, automatic scaling, and code monitoring. It supports multiple languages, organizing code into Lambda Functions that scale automatically based on demand, optimizing compute resource utilization.
5) *Amazon DynamoDB:* As a fully managed NoSQL database service, DynamoDB ensures fast and predictable performance with seamless scalability. Offloading administrative burdens associated with operating and scaling a distributed database, DynamoDB provides encryption at rest, on-demand backup capabilities, and easy scalability of throughput capacity, making it suitable for various application data storage needs.
6) *React:* React, a declarative and efficient JavaScript library, is employed for building user interfaces. Utilizing a component-based structure, react facilitates the creation of interactive UIs, ensuring predictability, and ease of debugging.

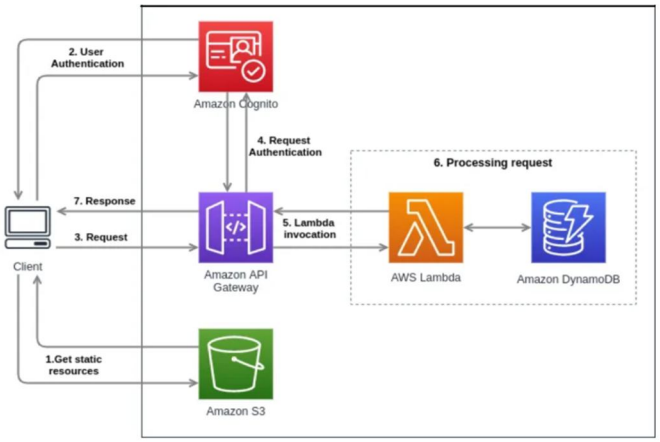


Figure 1. System Structure

B. Working Principle of the Components

1) *DynamoDB Table:* Serving as the persistence layer, a single DynamoDB table stores user information. Configured with a primary key (partition key) using the user ID, the DynamoDB table settings, such as provisioned capacity, are set to ensure optimal performance.
2) *Lambda Functions:* Four Lambda functions, written in Java using the AWS SDK for Java, perform distinct tasks, including interaction with the database and user auto-confirmation upon signup. Deployed using the AWS CLI, these functions execute business logic while automatically scaling based on demand.
3) *Amazon API Gateway:* REST APIs are built at the Amazon API Gateway, integrated with Lambda functions in the backend. Cognito User Pool is configured as the authorizer to ensure only authenticated users access backend resources.
4) *Amazon Cognito User Pool:* Responsible for complete user management and authentication, Amazon Cognito User Pool, coupled with an App client, handles user-related tasks in the front-end React application using the Amazon Cognito SDK for JavaScript.
5) *AWS Amplify:* Upon completion of the front-end and back-end integration, the application is deployed using the AWS Amplify Console. The git repository is connected to Amplify, build settings are configured, and the application is deployed globally via the Amplify console.
6) *React Components:* All required components in React are built and reused, each with its CSS file for styling. Fetch API is employed for making requests, with data stored in component state. React Routing is implemented where necessary, rendering appropriate components for an efficient user interface.

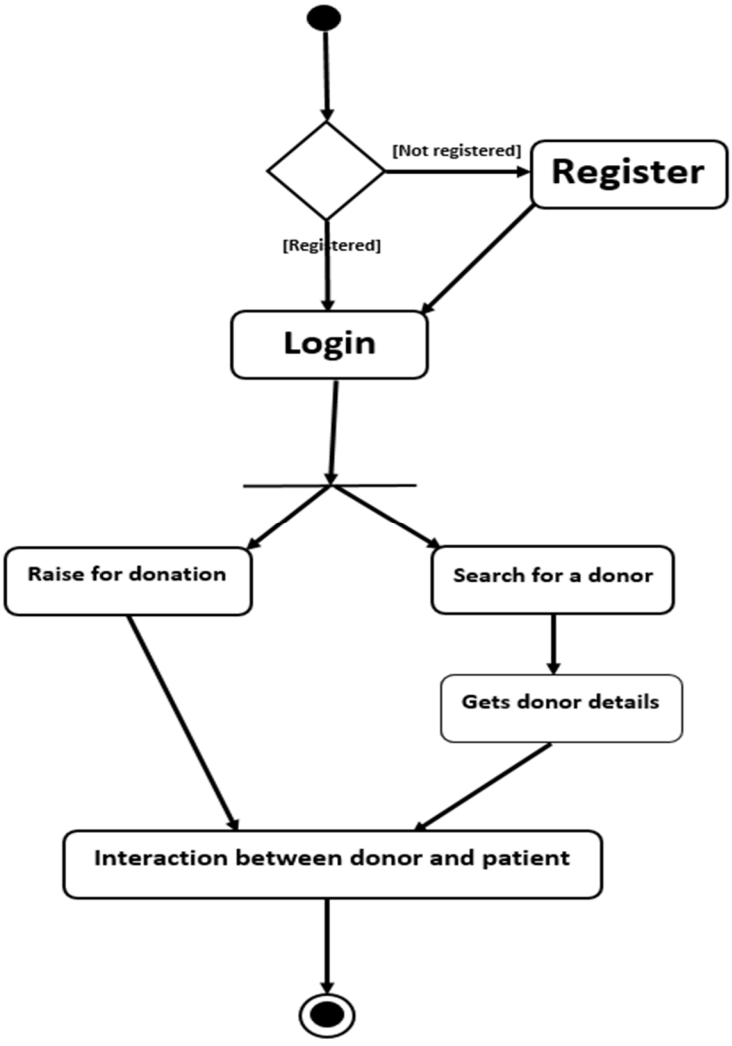


Figure 2. Flow Model

IV. CONCLUSION AND DISCUSSION

Serverless computing has emerged as a transformative paradigm, offering organizations a means to streamline infrastructure management complexities. The increasing adoption of serverless architectures necessitates a careful selection of solutions amidst diverse expert proposals. This research categorizes serverless patterns through a comprehensive literature review, shedding light on the coordination, aggregation, event management, availability, communication, and validation aspects. The practical application of serverless computing is demonstrated through a case study involving AWS Lambda and Amazon DynamoDB, illustrating its capacity to liberate developers from server-related concerns and enable the creation of scalable and reliable applications.

A. Advantages

Serverless computing offers several advantages. Firstly, it eliminates the need for infrastructure provisioning and management, allowing organizations to focus on application development. The cost model based on actual execution rather than pre-purchased capacity provides economic benefits. The architecture's flexibility, demonstrated in the case study, facilitates real-time data processing for applications like ride-sharing. Additionally, serverless computing contributes to enhanced scalability, as evident in the seamless scaling of Lambda functions.

B. Disadvantages

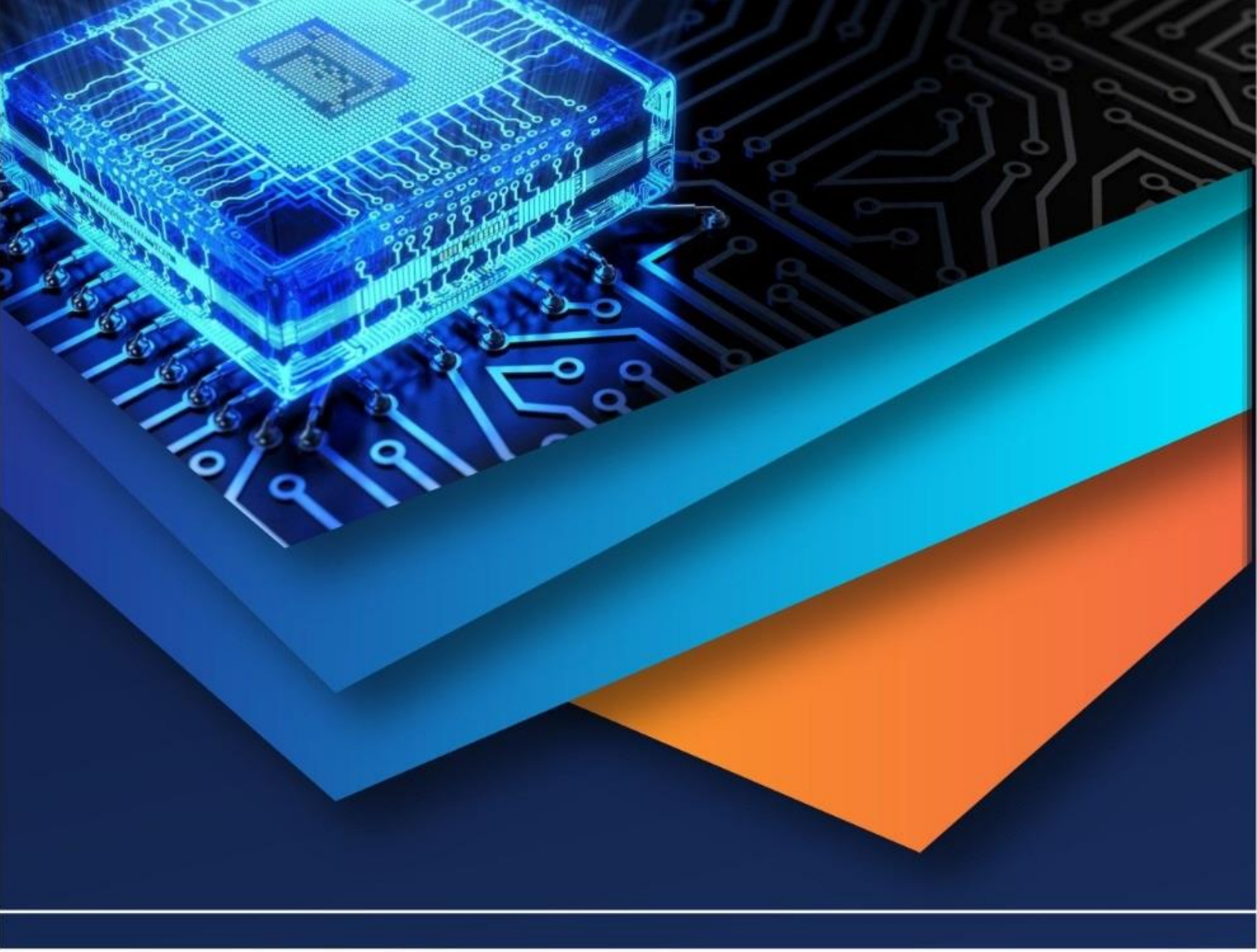
Security and authentication, as highlighted in the literature review, demand careful consideration. While serverless architectures reduce operational burdens, they introduce dependencies on cloud providers, potentially leading to vendor lock-in. State management complexities and concerns regarding the cost-effectiveness of high-throughput applications require attention. Thorough testing, as emphasized in the literature, is crucial to address potential pitfalls before migration. In conclusion, serverless computing represents a pivotal advancement in cloud technologies, and its practical applications, benefits, and challenges have been comprehensively explored. The presented case study and literature review contribute to a nuanced understanding of serverless computing, guiding practitioners in leveraging its advantages while addressing associated challenges.

V. ACKNOWLEDGMENT

We extend our deepest appreciation to our academic mentors Dr. Khushbu Wanjari, for her invaluable insights, expert guidance, and unwavering support throughout the entire research process. Their mentorship has been instrumental in shaping our research endeavours and refining our scholarly pursuits. This research is a collective effort, and you have played a significant role in its realization. Thank you for being an integral part of our academic journey and contributing to the success of this research review paper.

REFERENCES

[1] Sewak and S. Singh, ”Winning in the Era of Serverless Computing and Function as a Service,” 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, 2018, pp. 1-5
[2] Lynn, P. Rosati, A. Lejeune and V. Emeakaroha, ”A Preliminary Review of En terprise Serverless Cloud Computing (Function-as-a-Service) Platforms,” 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, 2017, pp. 162-169.
[3] Swedha and T. Dubey, ”Analysis of Web Authentication Methods Using AmazonWeb Services,” 2018 9th International Conference on Computing, Communicationand Networking Technologies (ICCCNT), CBangalore, 2018, pp. 1-6.
[4] Lloyd, S. Ramesh, S. Chinthalapati, L. Ly and S. Pallickara, ”Serverless Comput ing: An Investigation of Factors Inflfluencing Microservice Performance,” 2018 IEEEInternational Conference on Cloud Engineering, Orlando, FL, 2018, pp. 159-169.
[5] Al-Ali et al., ”Making Serverless Computing More Serverless,” 2018 IEEE 11thInternational Conference on Cloud Computing, San Francisco, CA, 2018, pp. 456-459.
[6] Kritikos and P. Skrzypek, ”A Review of Serverless Frameworks,” 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion, Zurich, 2018, pp. 161-168.
[7] Kotas, T. Naughton and N. Imam, ”A comparison of Amazon Web Services and Microsoft Azure cloud platforms for high performance computing,” 2018 International Conference on Consumer Electronics, Las Vegas, NV, 2018, pp. 1-4.
[8] Yoon, A. Gavrilovska, K. Schwan and J. Donahue, ”Interactive Use of Cloud Services: Amazon SQS and S3,” 2012 12th IEEE/ACM International Symposiumon Cluster, Cloud and Grid Computing, Ottawa, ON, 2012, pp. 523-530.
[9] Villamizar et al., ”Infrastructure Cost Comparison of Running Web Applica tions in the Cloud Using AWS Lambda and Monolithic and Microservice Archi tectures,” 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, 2016, pp. 179-182.
[10] Garca Lpez, M. Snchez-Artigas, G. Pars, D. Barcelona Pons, . Ruiz Ollobarren and D. Arroyo Pinto, ”Comparison of FaaS Orchestration Systems,” 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Zurich, 2018, pp. 148-153.
[11] Narula, A. Jain and Prachi, ”Cloud Computing Security: Amazon Web Service,” 2015 Fifth International Conference on Advanced Computing & Communication Technologies, Haryana, 2015, pp. 501-505.
[12] McGrath and P. R. Brenner, ”Serverless Computing: Design, Implementation, and Performance,” 2017 IEEE 37th International Conference on Distributed Com puting Systems Workshops (ICDCSW), Atlanta, GA, 2017, pp. 405-410.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY