IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

Serverless Architecture of WordPress on AWS

Dr. N. Nanthini M.E., Ph.D. [1], Uday Kiran L[2], K. Balaji Barath Kumar[3]

Assistant Professor, Sathyabama Institute of Science and Technology Chennai, India[1]

Department of CSE-CS, Sathyabama Institute of Science and Technology, Chennai, India[2]

Department of CSE-CS, Sathyabama Institute of Science and Technology, Chennai, India [3]

Abstract: Serverless architecture provides a modern, scalable, and cost-effective solution for hosting WordPress websites, eliminating the complexities associated with traditional server-based setups. By leveraging AWS services such as AWS Lambda for executing backend logic, Amazon API Gateway for routing HTTP requests, Amazon S3 for hosting static assets, and Amazon RDS for managing the relational database, this architecture supports dynamic scalability, high availability, and operational efficiency. This paper delves into the design and implementation of deploying WordPress in a serverless environment, offering insights into the technical workflow and integration of serverless components. The study also examines the benefits of automatic scaling, reduced infrastructure costs, and minimized maintenance efforts while addressing challenges such as cold starts, integration complexities, and latency issues. Through a detailed analysis, the paper highlights the potential of serverless architecture in transforming traditional WordPress deployments into highly efficient and resilient systems, offering significant value for developers, businesses, and end-users.

Keywords: Serverless Architecture, WordPress, AWS Lambda, API Gateway, Amazon S3, Amazon RDS, Scalability, Cost Optimization

I. INTRODUCTION

WordPress is one of the most widely used content management systems, powering a significant portion of websites globally. Its popularity stems from its flexibility, ease of use, and extensive plugin ecosystem. However, traditional hosting solutions, such as shared hosting or dedicated servers, often struggle to efficiently manage sudden surges in traffic. These traffic spikes can lead to performance bottlenecks, downtime, and high operational costs, which negatively impact user experience and business outcomes. Addressing these limitations requires a hosting solution that can dynamically scale, optimize resource utilization, and reduce operational overhead.

Serverless architecture offers a transformative approach to hosting WordPress by eliminating the need for traditional servers and enabling on-demand resource allocation. In serverless environments, resources are automatically provisioned based on workload, and users are charged only for the compute power and storage they actually use. This "pay-as-you-go" model not only improves cost efficiency but also simplifies infrastructure management by abstracting away tasks such as server provisioning, scaling, and maintenance.

Amazon Web Services (AWS) provides a robust ecosystem of serverless services that can be integrated to create a highly efficient and scalable WordPress deployment. AWS Lambda handles dynamic execution of backend logic, while API Gateway facilitates seamless HTTP request routing. Amazon S3 serves as a cost- effective and durable storage solution for static assets, and Amazon RDS ensures reliable management of relational databases. Together, these services provide a foundation for deploying WordPress in a serverless architecture.

This paper presents a comprehensive study of the serverless architecture for WordPress on AWS. It explores the design considerations, implementation steps, and integration of key AWS components, highlighting how these technologies enable dynamic scalability, cost savings, and operational efficiency. Additionally, the paper discusses the challenges inherent in serverless WordPress deployments, such as managing cold starts, ensuring database connectivity, and maintaining compatibility with WordPress's core functionality. By addressing these aspects, the study aims to provide a valuable resource for developers and organizations seeking to adopt serverless architecture for their WordPress projects.

The remainder of this paper is organized as follows: Section II reviews related work and existing approaches to hosting WordPress. Section III details the proposed serverless architecture, including its components and workflow. Section IV discusses the implementation process, highlighting best practices and optimization strategies. Section V evaluates the architecture's performance, cost, and scalability based on real-world use cases. Section VI identifies potential challenges and limitations. Finally, Section VII concludes with recommendations and future directions for leveraging serverless architecture in WordPress hosting.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

II. LITERATURE SURVEY

The adoption of serverless architecture has transformed the way web applications, including WordPress sites, are deployed and managed on the cloud. This literature survey explores various works and resources that provide insights into serverless computing, scalable database management, and best practices for implementing WordPress in a serverless environment.

1. **Amazon Aurora Serverless Documentation** Amazon Aurora Serverless offers a dynamic and scalable solution for database management by automatically adjusting capacity based on workload demands. This documentation highlights its benefits, such as high availability, pay-as-you-go pricing, and seamless integration with other AWS services. Aurora Serverless is particularly well-suited for serverless WordPress architectures, ensuring reliable and efficient database operations.

2. **WordPress on AWS**
The AWS tutorial for deploying WordPress using Amazon RDS provides step-by-step guidance on setting up WordPress in a traditional AWS environment. It focuses on integrating Amazon RDS for database management, which forms the foundation for transitioning WordPress to a serverless setup by leveraging services like AWS Lambda and API Gateway.

3. **Serverless WordPress Solutions**
A blog post on serverless WordPress implementation explores innovative approaches such as using tools like WP2Static to convert WordPress sites into static HTML. This reduces reliance on backend servers and enhances performance by serving pre-rendered pages, making it a viable option for serverless hosting scenarios.

4. **Static WordPress Documentation**
The WP2Static documentation discusses the process of converting WordPress sites into static HTML, which can be hosted on services like Amazon S3 and delivered via CloudFront. This approach minimizes the need for dynamic server-side processing, aligning well with the principles of serverless architecture.

5. **AWS Serverless Application Model (SAM)** The AWS Serverless Application Model (SAM) provides a framework for building and managing serverless applications. It simplifies the deployment of AWS Lambda functions, API Gateway configurations, and other serverless components, making it an essential tool for implementing a serverless WordPress setup.

6. **Eassa, A. M. (2025)**
In the article *"Optimizing Web Application Development: A Proposed Architecture Integrating Headless CMS and Serverless Computing"*, Eassa explores the integration of serverless computing with headless CMS solutions. The work emphasizes scalability, cost-efficiency, and flexibility, providing a strong foundation for understanding how serverless architecture can be applied to WordPress as a CMS.

7. **Zanon, D. (2017)**
"Building Serverless Web Applications" by Zanon provides a comprehensive guide to designing and implementing serverless architectures. It covers key concepts, best practices, and real-world examples, offering valuable insights into building scalable and resilient web applications, including WordPress sites.

8. **Wittig, A., & Wittig, M. (2023)**
The book *"Amazon Web Services in Action"* offers an in-depth guide to AWS services, including serverless computing and cloud infrastructure design. It provides practical examples and strategies for leveraging AWS tools like Lambda, S3, and CloudFront to create scalable applications, making it highly relevant for serverless WordPress hosting.

9. **Juncosa Palahí, M. (2022)**
In the thesis *"Platform for Deploying a Highly Available, Secure, and Scalable Web Hosting Architecture to the AWS Cloud with Terraform"*, Palahí presents a robust solution for deploying scalable web applications on AWS. The work highlights the use of Infrastructure as Code (IaC) tools like Terraform to automate deployments, which can be applied to serverless WordPress implementations for enhanced efficiency.

10. **Wadia, Y., Udell, R., Chan, L., & Gupta, U. (2019)**
The book *"Implementing AWS: Design, Build, and Manage Your Infrastructure"* provides a detailed guide to designing scalable and fault- tolerant cloud environments using AWS. It emphasizes security, scalability, and reliability, offering practical insights for transitioning WordPress to a serverless architecture while ensuring best practices are followed.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

III. SUMMARY

The reviewed literature and resources collectively highlight the growing adoption of serverless architecture for web applications, including WordPress. From scalable database management with Amazon Aurora Serverless to leveraging tools like WP2Static and AWS SAM, these works demonstrate the potential of serverless solutions to enhance scalability, reduce costs, and simplify operations. While challenges such as adapting WordPress's stateful nature to a stateless serverless environment remain, the combination of theoretical insights and practical guidelines provided in these references establishes a solid foundation for implementing a serverless WordPress architecture.

IV. SYSTEM OVERVIEW

A serverless WordPress architecture leverages various AWS services to optimize scalability, efficiency, and cost-effectiveness. The system is designed to decouple WordPress components, allowing each to be handled by the most appropriate AWS service. This modular approach ensures dynamic scalability, improved performance, and simplified management. The key components of the architecture are as follows:

A. Application Deployment

The initial deployment of WordPress typically occurs on Amazon Lightsail or EC2 instances to serve as a staging or migration point. These traditional hosting options provide a familiar environment for developers to configure WordPress before transitioning to a serverless setup.

In the serverless architecture, AWS Lambda handles backend processes, executing WordPress's dynamic functionalities such as PHP scripts for handling user requests, processing forms, or generating dynamic content. By using Lambda, compute resources are allocated on-demand, ensuring scalability and cost- efficiency. Lambda's event-driven nature allows it to automatically respond to HTTP requests and other triggers, making it ideal for managing WordPress workloads during high-traffic periods.

API Gateway acts as the primary interface between the frontend and backend components. It routes HTTP requests from users to AWS Lambda, enabling seamless communication between WordPress's dynamic content generation and the user-facing frontend. API Gateway also facilitates secure and scalable interactions by managing request throttling, caching, and authentication.

B. Media and File Storage

Media files, themes, and plugins are integral to WordPress functionality. These static assets are stored in Amazon S3, which offers highly scalable, secure, and durable storage. By offloading media storage to S3, the architecture minimizes the load on backend components and ensures quick access to assets.

Amazon S3 also supports versioning, lifecycle policies, and cross-region replication, enabling effective management of WordPress assets. By integrating S3 with WordPress using plugins or custom configurations, developers can streamline the storage and retrieval of media files directly from the WordPress dashboard.

C. Database Management

WordPress's relational database, which stores user data, posts, pages, and configurations, is managed by Amazon RDS or Amazon Aurora. These managed database services ensure compatibility with MySQL, a requirement for WordPress, while offering features such as automated backups, multi-AZ deployment for high availability, and read replicas for scaling read-heavy workloads.

Amazon Aurora, in particular, provides a highly performant and scalable database solution, with built-in support for MySQL and PostgreSQL. Its serverless configuration can dynamically scale database capacity based on usage, making it an excellent choice for handling traffic fluctuations.

D. Content Delivery

To enhance performance and reduce latency, Amazon CloudFront is integrated as a Content Delivery Network (CDN). CloudFront caches static assets such as images, CSS, JavaScript, and HTML pages across a global network of edge locations, ensuring quick delivery of content to users regardless of their geographic location.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

CloudFront also supports secure communication through HTTPS, integrates with AWS Shield for DDoS protection, and offers fine-grained cache control. By offloading content delivery to CloudFront, the architecture reduces the load on backend components and provides a seamless experience for end-users.

V. ADVANTAGES

The adoption of a serverless architecture for hosting WordPress sites offers numerous benefits compared to traditional server-based approaches. By leveraging AWS's suite of serverless services, organizations can achieve improved scalability, cost efficiency, reduced operational complexity, and enhanced global accessibility. Below, we expand on the key advantages:

A. Scalability

One of the standout features of serverless architecture is its ability to automatically scale resources to match varying workloads. Services like **AWS Lambda**, **Elastic Container Service (ECS)**, and **Fargate** dynamically adjust compute power in response to traffic demands.

- **Dynamic Scaling**: As the traffic to the WordPress site fluctuates, Lambda functions and other serverless resources scale up during high-demand periods and scale down when traffic subsides, ensuring that the application remains responsive under all conditions.
- **No Pre-Provisioning Required**: Unlike traditional hosting, where servers must be pre- provisioned for peak traffic (often leading to over-provisioning), serverless architecture eliminates the need for capacity planning.
- **Improved User Experience**: By maintaining responsiveness during traffic spikes, serverless architecture ensures uninterrupted performance and minimizes downtime, even during high- traffic events such as product launches or viral content spikes.

B. Cost Efficiency

The serverless pay-as-you-go pricing model provides a significant advantage for cost optimization:

- **Usage-Based Billing**: Organizations are billed only for the compute time and resources consumed. For example, AWS Lambda charges per execution and duration, while Amazon S3 and CloudFront charge based on data storage and transfer. This eliminates costs associated with idle server capacity, which is a common inefficiency in traditional hosting.
- **Reduced Infrastructure Costs**: The need for expensive server maintenance and hardware upgrades is eliminated, as AWS manages the underlying infrastructure.
- **Cost Transparency**: By utilizing AWS cost management tools, organizations can track and optimize spending across various serverless services, ensuring that they stay within budget.

C. Reduced Management Overhead

AWS's serverless ecosystem significantly reduces the operational burden on development and IT teams:

- **Infrastructure Management**: AWS services automatically handle tasks such as server provisioning, patching, and scaling. This allows teams to focus their efforts on developing and optimizing the WordPress application rather than managing infrastructure.
- **Built-In Monitoring**: Tools like **Amazon CloudWatch** provide real-time monitoring and logging, enabling teams to quickly identify and address performance issues without requiring additional monitoring solutions.
- **Simplified Deployment**: With services like AWS Lambda and API Gateway, deploying updates becomes streamlined, reducing the time and complexity associated with traditional deployment pipelines.

D. Global Availability

The integration of **Amazon CloudFront** as a Content Delivery Network (CDN) ensures that WordPress sites are globally accessible with optimal performance:

- **Edge Locations**: CloudFront's global network of edge locations ensures that static assets are cached closer to end-users, reducing latency and improving load times.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

- **Low-Latency Access**: Visitors from different geographic regions experience consistently fast page loads, enhancing user satisfaction and engagement.
- **Enhanced Reliability**: CloudFront improves the availability of WordPress sites by distributing traffic across multiple locations, ensuring that localized issues at a single data center do not impact global access.
- **DDoS Protection**: CloudFront integrates with **AWS Shield** to provide protection against Distributed Denial-of-Service (DDoS) attacks, ensuring secure and reliable delivery of content to users.

Additional Benefits

1. **Improved Security**: Serverless services like AWS Lambda and S3 operate within isolated environments, reducing the attack surface and enhancing the security posture of WordPress deployments.
2. **Flexibility and Customization**: The modular nature of serverless architecture allows developers to integrate custom solutions and third-party APIs seamlessly, enabling tailored functionalities for WordPress sites.
3. **Faster Time-to-Market**: The simplified management and deployment workflows of serverless services enable faster development cycles, allowing organizations to launch new features and updates more quickly.

VI. CHALLENGES

While serverless architecture provides significant advantages for hosting WordPress, it introduces unique challenges that require careful consideration and planning. These challenges stem primarily from WordPress's traditional design, which assumes a stateful server environment, and the inherent characteristics of serverless systems. This section outlines the key challenges and potential approaches to address them.

A. Stateless Design

Serverless functions, such as AWS Lambda, operate in a stateless environment, meaning they do not retain any data or session information between executions. However, WordPress's core functionalities, such as session management, caching, and file uploads, are designed for a stateful server setup.

- **Session Management**: WordPress relies on PHP sessions to manage user login states, shopping cart data, and other session-based features. In a stateless serverless architecture, external solutions like **Amazon ElastiCache (Redis)** or **DynamoDB** can be used to handle session data.
- **Cache Management**: Transient data and caching, typically stored on the server in traditional setups, need to be shifted to external services like Redis or Memcached to ensure persistent and shared access across function invocations.
- **Adaptation of Plugins**: Some WordPress plugins may not be compatible with a stateless environment and may require modifications or alternative solutions to work seamlessly.

B. Database Compatibility

WordPress is tightly coupled with MySQL databases, and integrating it with managed databases like Amazon RDS or Amazon Aurora requires addressing several compatibility and performance issues:

- **Latency**: Stateless serverless functions like AWS Lambda must establish a connection with the database on each invocation, potentially introducing latency. Connection pooling solutions, such as **RDS Proxy**, can mitigate this issue by maintaining persistent connections to the database.
- **Scaling Databases**: While serverless functions can scale rapidly, databases must also scale to handle increased traffic. Amazon Aurora Serverless provides auto-scaling capabilities, but care must be taken to ensure minimal downtime during scaling operations.
- **Data Consistency**: High-concurrency scenarios in a serverless environment can lead to challenges in maintaining data consistency. Implementing proper transaction handling and isolation levels is crucial to avoid conflicts.

C. File Storage

WordPress relies heavily on a local file system for handling media uploads, themes, and plugin installations. In a serverless environment, these operations must be offloaded to persistent storage solutions such as Amazon S3:

- **Dynamic File Handling**: WordPress's native file handling needs to be modified to work with S3, typically through plugins like **WP Offload Media** or custom scripts. These ensure that uploaded files are directly stored in S3 and referenced via URLs.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

- **Versioning and Access Control**: Amazon S3 offers versioning to track changes to files and fine-grained access control to secure stored assets, but these need to be configured carefully to align with WordPress's requirements.
- **Latency and Bandwidth**: While S3 provides scalable storage, retrieving files during high- traffic periods can introduce latency. Integrating S3 with **Amazon CloudFront** helps reduce retrieval times by caching assets closer to users.

D. Cold Starts

AWS Lambda functions experience cold starts when a new instance of a function is invoked after a period of inactivity. This delay, though typically short, can impact performance, especially for high-traffic WordPress sites:

- **Mitigation Techniques**: Cold start latency can be mitigated by ensuring functions remain warm. Solutions include periodic invocations of the Lambda function (e.g., using **AWS CloudWatch Events**) or leveraging **Provisioned Concurrency**, which keeps a predefined number of function instances warm and ready to handle requests.
- **Impact on User Experience**: High-traffic applications may experience performance degradation during cold starts, affecting user experience. Careful monitoring and proactive scaling can help address this challenge.

E. Scaling Management

Although serverless architecture inherently supports scaling, ensuring that all components of the WordPress ecosystem scale seamlessly is critical for maintaining performance:

- **Database Scaling**: As the number of Lambda function invocations increases, the database must scale to accommodate concurrent connections. Using **RDS Proxy** or migrating to **Aurora Serverless** can ensure that database capacity aligns with workload demands.
- **Storage Scaling**: Media and file storage on S3 can handle massive amounts of data, but careful bucket organization and lifecycle management are necessary to optimize performance and costs.
- **Balanced Resource Allocation**: While serverless compute scales automatically, other services, such as CloudFront, API Gateway, and S3, must also be configured appropriately to avoid bottlenecks in the architecture.

Additional Challenges

1. **Plugin and Theme Compatibility**: Many WordPress plugins and themes are designed for traditional server environments. Testing and customization are often required to ensure compatibility with serverless systems.
2. **Monitoring and Debugging**: Serverless applications introduce complexity in monitoring and debugging due to their distributed nature. AWS tools like **CloudWatch**, **X-Ray**, and third-party solutions like **Datadog** are essential for gaining insights into performance and troubleshooting issues.
3. **Security Management**: Ensuring secure communication between components, managing permissions (IAM roles), and protecting data (e.g., encrypting data at rest and in transit) are critical in a serverless environment.

VII. PROPOSED ARCHITECTURE

The proposed architecture leverages AWS’s serverless services to create a scalable, efficient, and cost-effective hosting solution for WordPress. This architecture decouples traditional WordPress components and integrates them into a robust serverless ecosystem, ensuring optimal performance, dynamic scalability, and minimal operational overhead. Below is an expanded overview of the architecture's core components and additional considerations.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

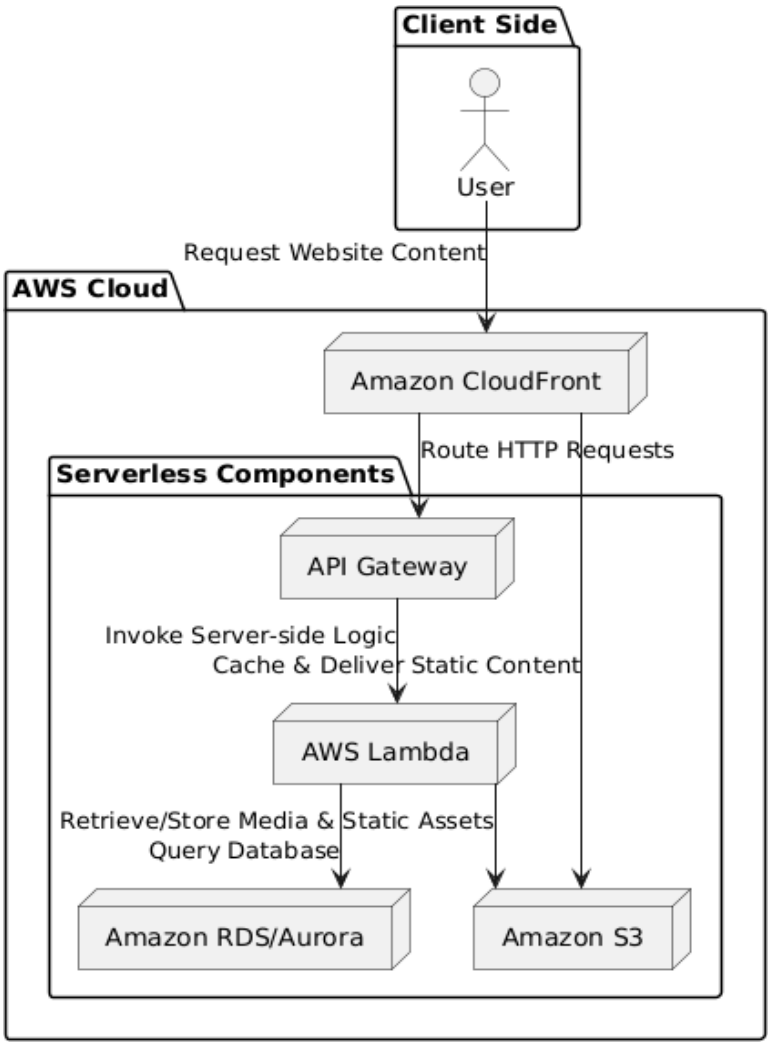

Fig 1 - Architecture Diagram

VIII. CONCLUSION

Serverless architecture on AWS offers a transformative approach to hosting WordPress, providing a scalable, efficient, and cost-effective alternative to traditional hosting solutions. By leveraging managed services such as AWS Lambda, API Gateway, Amazon S3, Amazon RDS/Aurora, and CloudFront, this architecture eliminates the need for manual server management, significantly reduces operational overhead, and dynamically adapts to traffic demands.

Key Takeaways

- **Dynamic Scalability**: The serverless model ensures resources are automatically scaled to match workload requirements, providing consistent performance even during high- traffic periods.
- **Cost Optimization**: The pay-per-use model minimizes expenses by charging only for resources consumed, making it particularly advantageous for applications with fluctuating traffic patterns.
- **Operational Efficiency**: AWS services handle tasks like scaling, patching, and monitoring, allowing development teams to focus on innovation and content management.

Challenges Addressed

While the transition to serverless hosting introduced challenges such as adapting to a stateless design, managing database connections, and mitigating cold starts, these were effectively addressed through AWS features like RDS Proxy, Provisioned Concurrency, and external storage solutions.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 3, March 2025

DOI: 10.17148/IARJSET.2025.12307

Future Directions

To further enhance serverless WordPress deployments, future work could explore:

1. **Hybrid Models**: Combining serverless and traditional hosting to support use cases that require stateful operations or persistent connections.
2. **Plugin Adaptation**: Developing tools or guidelines to streamline the migration of WordPress plugins and themes to serverless environments.
3. **Performance Enhancements**: Investigating edge computing solutions and alternative runtime optimizations to further reduce latency and improve user experience.

REFERENCES

[1]. Amazon Aurora Serverless Documentation: Guide on using Aurora Serverless for scalable database management.
[2]. WordPress on AWS:AWS tutorial for deploying WordPress on AWS using Amazon RDS.
[3]. Serverless WordPress Solutions: Blog post discussing the implementation of WordPress in a serverless environment WP2Static.
[4]. Documentation: Information on converting WordPress sites to static HTML which can be useful for a serverless setup.
[5]. AWS Serverless Application Model (SAM):Provides guidance on building and managing serverless applications using AWS SAM.
[6]. Eassa, A. M. (2025). Optimizing Web Application Development: A Proposed Architecture Integrating Headless CMS and Serverless Computing. *IJCI. International Journal of Computers and Information*, *12*(1), 103-119.
[7]. Zanon, D. (2017). *Building Serverless Web Applications*. Packt Publishing Ltd.
[8]. Wittig, A., & Wittig, M. (2023). *Amazon Web Services in Action: An in-depth guide to AWS*. Simon and Schuster.
[9]. Juncosa Palahí, M. (2022). *Platform for deploying a highly available, secure and scalable web hosting architecture to the AWS cloud with Terraform* (Bachelor's thesis, Universitat Politècnica de Catalunya).
[10]. Wadia, Y., Udell, R., Chan, L., & Gupta, U. (2019). *Implementing AWS: Design, Build, and Manage your Infrastructure: Leverage AWS features to build highly secure, fault-tolerant, and scalable cloud environments*. Packt Publishing Ltd.