Journal of Advances in Developmental Research (IJAIDR)

E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

Architecting Secure and Compliant Enterprise Applications with AWS Serverless

Siva Kumar Mamillapalli

siva.mamill@gmail.com

Abstract

This paper talks about how serverless architectures can revolutionize your IT strategy. This paper guides Cloud, DevOps Engineers and Cloud architects through the essentials of serverless, including the AWS portfolio, and illustrates its impact on time-to-market, agility, and cost savings. Through practical case studies, understand how companies are achieving tangible benefits. Learn how to leverage serverless to build reactive, event-based systems and deploy microservices efficiently and economically.

Keywords: AWS, Serverless Architecture, DevOps, Microservices, Event-Driven Architecture, Cloud Native

1. Introduction

The public cloud has become a cornerstone of modern IT, offering significant cost savings and agility through on-demand resources. Decades of research and practical application have demonstrated that migrating existing application architectures to the cloud lowers the Total Cost of Ownership (TCO) and accelerates time-to-market. However, while cloud migration eliminates the need for physical hardware maintenance, traditional server-based architectures still present challenges in scalability, maintenance, and resource utilization. Companies must still architect peak loads, manage patching and deployment cycles, and wrestle with the inherent inefficiencies of idle servers. Studies indicate average server utilization hovers around a mere 18%, revealing substantial resource waste.

Serverless computing emerges as a transformative solution, addressing these persistent challenges. By abstracting away server management, serverless allows developers to concentrate solely on building and deploying business logic. This paradigm shift leads to faster development cycles, reduced TCO, and a crucial competitive edge through enhanced agility and speed. Developers can focus on innovating and delivering value rather than managing infrastructure. Product owners can achieve faster time-to-market with shorter development, deployment, and testing cycles. The reduction of server management overheads reduces the TCO, which ultimately results in competitive advantages for the companies.

2. Literature Review

2.1 Understanding Serverless Architectures

The benefits of the serverless approaches mentioned above are attractive, but what practical considerations should be taken into account? How does a serverless application differ from its traditional server-based counterpart?

Serverless utilizes managed services where the cloud provider handles tasks like capacity management and software updates, freeing up engineers and architects to concentrate on customer-centric business logic rather than infrastructure maintenance, configuration, operations, and idle capacity. Moreover, serverless encompasses practices and tools that enable agile application development, facilitating faster innovation and adaptability to change.

Serverless applications operate entirely or partially in the public cloud using serverless services, such as AWS's offerings across compute, storage, application integration, orchestration, and databases. This model offers several advantages over traditional server-based designs:

Instead of provisioning, managing, and monitoring infrastructure, the cloud provider handles hardware and platform software, allowing you to deploy your application and its settings directly.

Serverless services inherently incorporate fault tolerance, requiring minimal user configuration for achieving high availability.

Reduced operational overhead enables faster release cycles, feedback incorporation, and quicker time-to-market.

A pay-as-you-go billing model ensures cost efficiency by eliminating over-provisioning and optimizing resource usage on your behalf.

Built-in service integrations simplify application development by reducing the need for extensive configuration, letting you focus on core application development tasks.

2.2 Is serverless the best solution for every scenario?

Nearly all modern applications can be adapted to run efficiently, often more cost-effectively and at scale, on a serverless platform. The decision between serverless and other options doesn’t have to be an either-or situation. Different components of an application can run on servers, in containers, or use serverless architectures within the same stack. However, there are some scenarios where serverless may not be the best option:

- When the goal is to avoid making any changes to the existing application architecture.
- When fine-grained control over the environment is necessary, such as applying specific OS patches or performing low-level network operations.
- For applications requiring ultra-low latency for every incoming request.

- When an on-premises application has not yet been migrated to the public cloud.
- When certain application components exceed the limitations of serverless services, such as if a function runs longer than the AWS Lambda execution timeout or a backend application exceeds the timeout limit of Amazon API Gateway.

2.3 Serverless Usecases

The serverless model is versatile, suitable for a wide range of applications, from startups to large enterprises. Here are some examples:

- **Data Processing**: Serverless simplifies parallelization for tasks like map-reduce, video transcoding, stock trade analysis, and Monte Carlo simulations.
- **Web Applications**: Serverless eliminates servers, scaling applications efficiently from no traffic to peak loads.
- **Batch Processing**: Ideal for multi-step workflows like ETL jobs.
- **IT Automation**: Simplifies tasks like cron jobs and monitoring by removing the need to maintain servers.
- **Mobile Backends**: Enables secure, scalable backends without requiring distributed systems expertise.
- **Media and Log Processing**: Offers parallelism for compute-heavy tasks without complex scaling.
- **IoT Backends**: Simplifies cloud-based systems for device-specific algorithms.
- **Chatbots**: Works well for webhook-based systems like chatbots, activating only when needed.
- **Streaming Data**: Scales based on data flow, supporting high-speed processing of clickstreams or stock trades.
- **Machine Learning Inference**: Hosts models for intermittent inference requests without managing servers.
- **Edge Content Delivery**: Reduces latency by processing events at the internet edge.
- **Edge IoT**: Enables near real-time responses on IoT devices by processing locally.

Serverless apps often use microservices architecture, breaking an app into independent, scalable components. This approach reduces infrastructure costs by scaling only necessary components, ensuring efficient resource use without over-provisioning. Serverless naturally supports microservices, promoting agility and enabling incremental deployments.

2.4 AWS Serverless Capabilities

AWS serverless offerings cover all infrastructure layers, including compute, storage, and orchestration, with tools for authoring, building, deploying, and diagnosing serverless architectures.

A reliable, flexible platform is essential for running serverless applications at scale, from startups to global enterprises, providing end-to-end reliability and scalability for all application components.

AWS's serverless platform offers several key capabilities for large-scale enterprises:

- **Scalable Compute Layer**: AWS Lambda and AWS Fargate provide event-driven compute services, eliminating scaling and management overhead while integrating seamlessly with event sources for automatic scaling.

- **Scalable Storage Layer**: AWS offers fully managed storage like Amazon Aurora Serverless v2, DynamoDB, and S3 that scale with usage, eliminating the need for manual server and storage management.
- **Decoupled Workloads**: Services like Amazon SQS, SNS, EventBridge, and Kinesis help decouple components for independent scaling, allowing innovation and reducing resource waste by charging only for used resources.
- **Orchestration and Workflow Management**: AWS Step Functions provide visual workflow orchestration, managing retries, parallelization, and service integration to simplify complex workflows and enable rapid development.
- **Native Service Integrations**: Services like SQS, SNS, and EventBridge allow seamless communication between decoupled components, minimizing code for integration and simplifying application development.

These services enable efficient, scalable, and flexible serverless architectures, streamlining application development and reducing operational complexity.

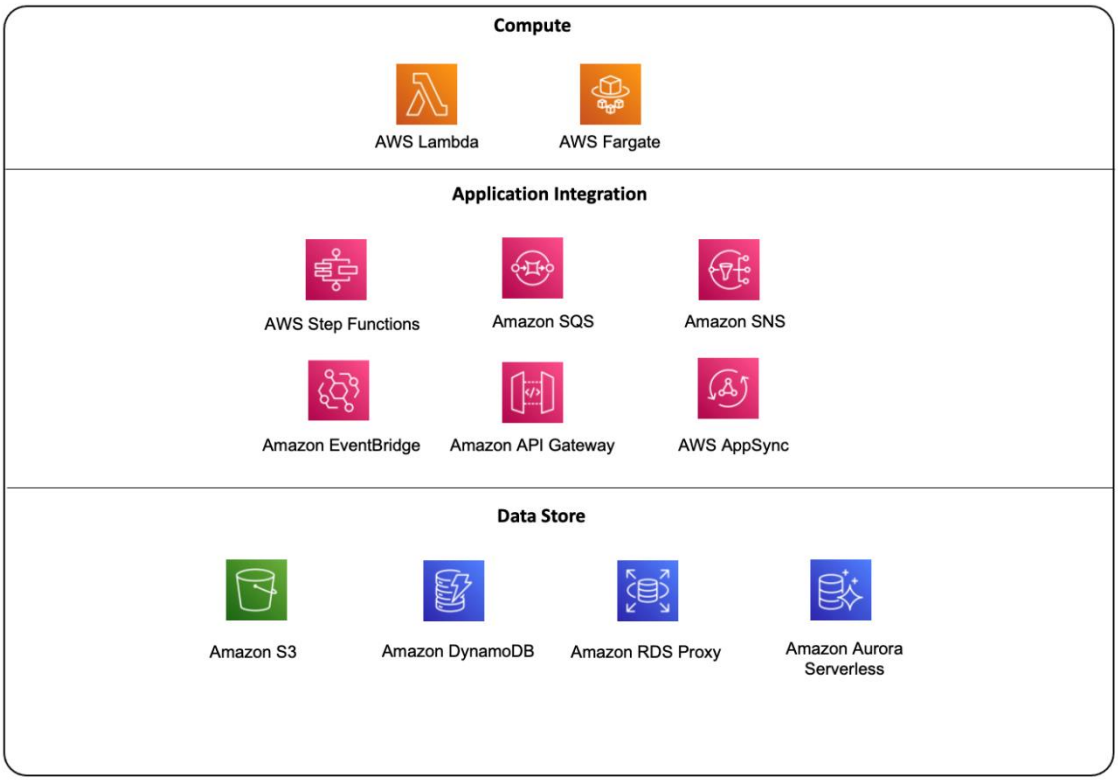

Figure 1: AWS serverless platform components

2.5 AWS Serverless Security

AWS serverless services offer extensive security and access controls, such as support for virtual private networks, role-based and access-based permissions, API-based authentication, and encryption of application elements like environment variables.

These built-in features help developers deploy and publish workloads confidently, reducing time to market. Serverless systems also offer enhanced security and control for several reasons:

- **First-class fleet management and security patching**: Managed serverless services like Lambda, API Gateway, and SQS are continuously monitored, patched quickly, and security-scanned, ensuring timely updates compared to traditional fleets with less stringent patching SLAs.
- **Per-request authentication, access control, and auditing**: Each request between services is authenticated, authorized, and auditable. Amazon API Gateway is integrated with AWS WAF to protect against web exploits, bots, and DDoS attacks. AWS CloudTrail allows companies to audit access and privileges, enhancing security monitoring.

These features eliminate often-overlooked costs, such as security software licenses, staff for server security, and regulatory compliance expenses.

Serverless architectures have smaller blast radii compared to monolithic systems running on virtual machines. With AWS handling server security, customers can focus on applying least privilege access, which reduces potential damage. The decoupled nature of serverless limits the impact of security breaches to fewer services, which helps reduce the financial consequences of attacks.

By adopting serverless, enterprises can reduce or eliminate unnecessary infrastructure costs, repurposing capital and freeing teams to focus on higher-value tasks.

3. Conclusion

Serverless approaches address two major IT management challenges: idle servers and the distraction of managing fleets of servers, which takes focus away from creating customer value.

AWS serverless offerings resolve these issues with a pay-for-value billing model and by removing the need to manage underlying infrastructure. AWS handles infrastructure monitoring, patching, and security, providing built-in fault tolerance with minimal configuration for high availability. This allows developers to focus on business logic, reducing time to market and ensuring enterprises only pay for the resources they use.

Companies are already gaining significant agility and economic benefits from adopting serverless, and enterprises should consider a serverless-first strategy when building cloud-native microservices.

4. References

1. A. Mohamed, "A history of cloud computing," Computer Weekly.com, Apr,2018. [Online]. Available: https://www.computerweekly.com/feature/A-history-of-cloud-computing
2. M. Roberts, "Serverless Architectures," MartinFowler.com, May 22, 2018. [Online]. Available: https://martinfowler.com/articles/serverless.html.
3. Amazon Web Services, "Serverless Computing and Applications," AWS, June 2019. [Online]. Available: https://aws.amazon.com/serverless.

4. A. Patrizio, "One in Five Serverless Apps has a Critical Security Vulnerability," NetworkWorld, Apr 12, 2018. [Online]. Available: https://www.networkworld.com/article/3268415/security/one-in-five-serverless-apps-has-a-critical-security-vulnerability.html.
5. A. Eivy, "Be Wary of the Economics of "Serverless" Cloud Computing," IEEE Cloud Computing, vol. 4, (2), pp. 6-12, 2017.
6. Amazon Web Services, "Build Your First Serverless Web Application," AWS, 2018. [Online]. Available: https://aws.amazon.com/serverless/build-a-web-app.
7. Creating AWS Lambda: http://docs.aws.amazon.com/lambda/latest/dg/get-started-create function.html, Accessed on May 2019
8. T. Lynn et al, "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), 2017.
9. Open Source Collective 501c6 (Non Profit) - APEX Software, "APEX | Serverless Infrastructure.," Github, 2018. [Online]. Available: https://github.com/apex/apex.
10. AWS Serverless reference architectures: https://docs.aws.amazon.com/whitepapers/latest/optimizing-enterprise-economics-with-serverless/reference-architectures.html, Accessed on May 2019
11. Serverless, Inc., "The way cloud should be.," Serverless, 2018. [Online]. Available: https://serverless.com.
12. M. Boyd, "Amazon Debuts Flourish, a Runtime Application Model for Serverless Computing," TheNewStack, May 26, 2016. [Online]. Available: https://thenewstack.io/amazon-debuts-flourish-runtime-application-model-serverless-computing.