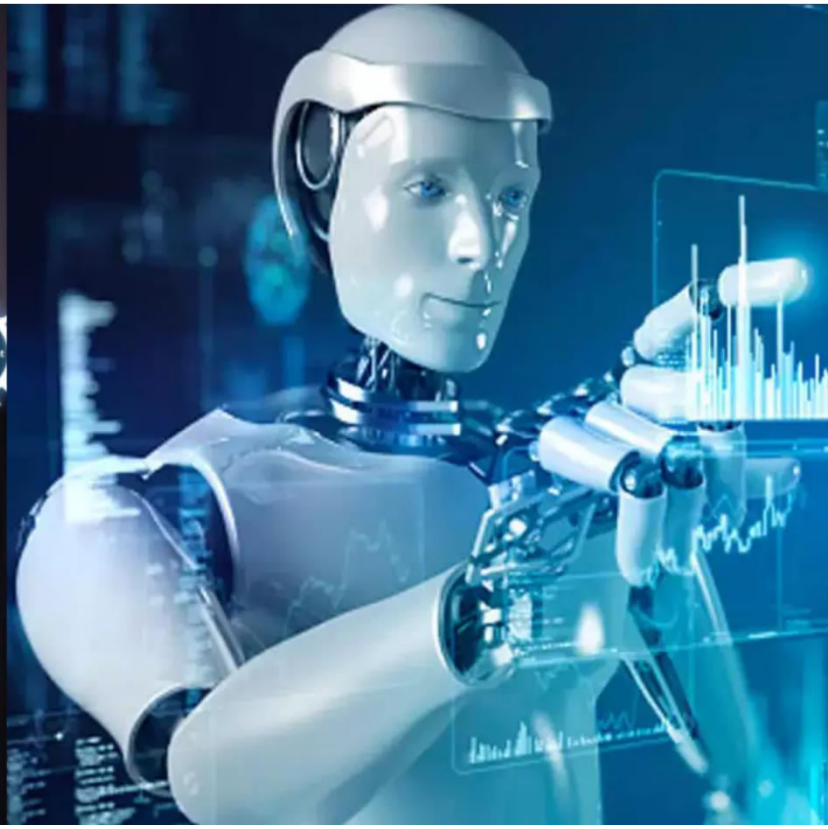
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 5, May 2025

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405300|

Build a Serverless EC2 Monitor & Control Dashboard with AWS Lambda, API Gateway, and S3

Sudipta Mukherjee, Tuhin Garai, Yashika Choudhary, Upasana Rath, Dr. Pushpa J.

P.G Student, Department of Computer Science and IT, Jain Deemed-to-be University, Bengaluru, India

P.G Student, Department of Computer Science and IT, Jain Deemed-to-be University, Bengaluru, India

P.G Student, Department of Computer Science and IT, Jain Deemed-to-be University, Bengaluru, India

P.G Student, Department of Computer Science and IT, Jain Deemed-to-be University, Bengaluru, India

Assistant Professor, Department of Computer Science and IT, Jain Deemed-to-be University, Bengaluru, India

ABSTRACT: This project presents a serverless EC2 Monitoring and Control Dashboard built using AWS services like Lambda, API Gateway, DynamoDB, S3, and SES. The solution automatically monitors EC2 instances for inactivity and stops them after a configurable idle period, also it helps in maintaining cloud cost efficiency. The user interface, hosted on Amazon S3, allows manual control and real-time status monitoring of instances. Designed with simplicity and cost efficiency in mind, this project demonstrates the practical application of serverless architecture and automation using core AWS technologies.

KEYWORDS: EC2 Monitoring, Real-time, Cloud Cost Efficiency, Manual Control

I. INTRODUCTION

Cloud computing has become the backbone of modern application deployment, and Amazon EC2 (Elastic Compute Cloud) is widely used for hosting scalable and on-demand virtual machines. However, one common pitfall among developers and cloud users — especially beginners — is forgetting to stop EC2 instances after use. This leads to unwanted billing and wastes resources. Addressing this issue, the Serverless EC2 Monitoring and Control Dashboard was developed as a lightweight, automated solution to manage idle EC2 instances efficiently. The project utilizes completely serverless architecture, leveraging a combination of Amazon S3, API Gateway, AWS Lambda, DynamoDB, IAM, and SES. The frontend, built with standard web technologies (HTML, CSS, JavaScript), is hosted on Amazon S3 for high availability and low cost. API Gateway acts as the interface between the frontend and the backend logic, invoking Lambda functions that handle EC2 monitoring and instance stopping based on defined thresholds. Lambda functions form the core of the application, with one function dedicated to checking EC2 instance activity and another responsible for shutting down instances that remain idle for more than a set period — defaulted to 5 minutes. We can use EventBridge service of AWS to automate the monitoring. Additionally, Amazon DynamoDB stores metadata such as instance IDs, timestamps, and usage statistics to help track patterns and maintain system state across function invocations. Security and access control are managed via IAM roles and policies, ensuring that Lambda functions have just the right permissions to interact with EC2, SES, and DynamoDB services. To enhance transparency and real-time alerts, Amazon SES (Simple Email Service) is used to send notifications when an idle instance is detected and is about to be stopped. This project is not only a cost-saving tool but also a learning platform that demonstrates real-world integration of multiple AWS services in a serverless architecture. For students and cloud enthusiasts, it offers hands-on experience with key AWS concepts such as Lambda triggers, RESTful APIs, and automated cloud resource management — all without the need to manage servers or deploy complex infrastructure. In summary, the Serverless EC2 Monitoring Dashboard is a smart, efficient, and educational project that aligns with modern DevOps practices. It simplifies cloud management, improves cost control, and illustrates how cloud-native technologies can be used to solve practical problems in a scalable and maintainable way.

II. ARCHITECTURE AND COMPONENT JUSTIFICATION

The Serverless EC2 Monitor & Control Dashboard exemplifies the power of AWS's cloud-native services, orchestrated to deliver a cost-efficient, scalable, and automated solution for managing EC2 instances. By leveraging a fully

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405300|

serverless architecture, the system eliminates traditional infrastructure overhead, enhances operational efficiency, and aligns with modern DevOps practices. The following subsections explore the critical roles of serverless architecture and each AWS service—Lambda, API Gateway, S3, DynamoDB, SES, and IAM—in achieving these objectives, highlighting their technical advantages and practical implications for cloud resource management.

A. Why Serverless Architecture?

A serverless approach eliminates the need to provision, manage, and maintain physical or virtual servers. AWS services like Lambda, DynamoDB, and API Gateway allow you to build applications that automatically scale, offer high availability, and only charge when used.

If traditional (non-serverless) architecture were used, it would require maintaining EC2 instances or container environments 24/7—leading to higher costs, increased complexity, and greater security risks.

- **What Happens If We Don't Use Serverless Architecture?**

Opting for a non-serverless approach, like using EC2 instances or managed containers (ECS or EKS), would be a costly headache. Running servers 24/7 for APIs or databases means paying even when idle, unlike our pay-per-use serverless setup. You'd need to patch operating systems, configure load balancers, and monitor resource usage, adding hours of work. Security risks grow with exposed servers requiring constant updates and firewall tweaks. For our lightweight app, which checks EC2 instances periodically, this overhead is unnecessary. Non-serverless systems also struggle with scaling—over-provisioning wastes money, while under-provisioning slows performance. Our serverless design dodges these issues, saving costs and effort.

B. AWS Lambda – Backend Logic Execution

Lambda handles core backend operations such as retrieving EC2 instance statuses, identifying idle instances, and stopping them. It runs code only when triggered (e.g., via API Gateway or scheduled events), eliminating idle costs. This makes it ideal for event-driven workloads. Lambda also seamlessly integrates with other AWS services, reducing development complexity.[1]

C. Amazon API Gateway – Secure and Scalable Interface

API Gateway connects the frontend to Lambda via RESTful endpoints, managing authentication with IAM, throttling, and CloudWatch monitoring. It ensures secure, scalable API access without server management, supporting low-latency communication and handling varying traffic, unlike EC2-based APIs requiring manual scaling.[2]

D. Amazon S3 – Static Web Hosting

The dashboard frontend (HTML, CSS, JavaScript) is hosted on S3, offering low-latency access and near-infinite scalability. S3's cost-effectiveness and integration with other AWS services make it an ideal choice for hosting static web content, compared to setting up a web server on EC2.[3]

E. Amazon DynamoDB – Efficient State Management

DynamoDB stores metadata like instance status and usage logs. As a serverless NoSQL database, it offers rapid read/write performance, automatic scaling, and high availability. Unlike traditional relational databases, DynamoDB eliminates administrative tasks and suits the lightweight nature of monitoring data.[4]

- **Why DynamoDB Over Other AWS Database Services?**

When picking a database, we had options like RDS, Aurora, or ElastiCache, but DynamoDB was the clear winner for our serverless setup. RDS and Aurora, being relational databases, need managed servers, which means dealing with provisioning, backups, and scaling—exactly what we wanted to avoid. These databases are great for complex queries but overkill for our simple key-value data, like instance IDs and usage logs. Plus, their always-on nature racks up costs, unlike DynamoDB's pay-per-use model, which aligns with our cost-saving goal. ElastiCache, while fast for caching, is meant for temporary data, not persistent storage, and requires server management. DynamoDB's serverless nature, auto-scaling, and fast read/write performance fit our lightweight, event-driven app like a charm. It handles bursts of monitoring data without fuss, and TTL keeps storage costs in check by auto-deleting old logs. For a small-scale project like ours, DynamoDB's simplicity and integration with Lambda made it the practical, pocket-friendly choice over other AWS databases.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405300|

F. Amazon SES – Real-time Notifications

SES is used to send automated email alerts when EC2 instances are idle and about to be stopped. It ensures users stay informed without the complexity of configuring SMTP servers or third-party services.[5]

G. AWS IAM – Secure Access Control

IAM is essential for managing roles and permissions between services. It ensures each component operates with only the necessary access privileges, following the principle of least privilege. This is critical in serverless systems where many services communicate programmatically.[6]

III. WORKING PRINCIPLE

The Serverless EC2 Monitor & Control Dashboard leverages AWS services to deliver a cost-effective, scalable, and automated solution for managing EC2 instances. Users interact with a responsive web interface, built with HTML5, CSS3, and JavaScript, hosted on Amazon S3 as a static website for high availability and low cost. S3 bucket policies secure the frontend, which displays real-time instance statuses and enables control actions like stopping instances. When users query instance states or issue commands, the frontend sends HTTP requests to Amazon API Gateway, which exposes RESTful endpoints secured by IAM authentication and throttling. API Gateway routes requests to AWS Lambda functions, the core of the backend logic. One Lambda function, triggered every five minutes by CloudWatch Events, monitors EC2 metrics like CPU utilization via the AWS SDK, identifying idle instances based on thresholds (e.g., below 5% for 15 minutes). Another function handles user-initiated actions, such as stopping instances, with optimized execution settings. IAM roles enforce least-privilege access, granting Lambda permissions for EC2, DynamoDB, and SES operations. Amazon DynamoDB, a serverless NoSQL database, stores instance metadata, including IDs and usage logs, with a schema using instance IDs as partition keys and TTL for cost-efficient data retention. If an idle instance is detected, Lambda triggers Amazon SES to send email alerts with details like instance ID and idleness duration, ensuring user awareness. The serverless architecture eliminates server management, reducing costs and complexity while scaling automatically. CloudWatch Metrics and Logs enhance monitoring and debugging. Optimized for automation, security, and performance, this system minimizes EC2 costs and serves as an educational tool, demonstrating AWS service integration for cloud-native applications.

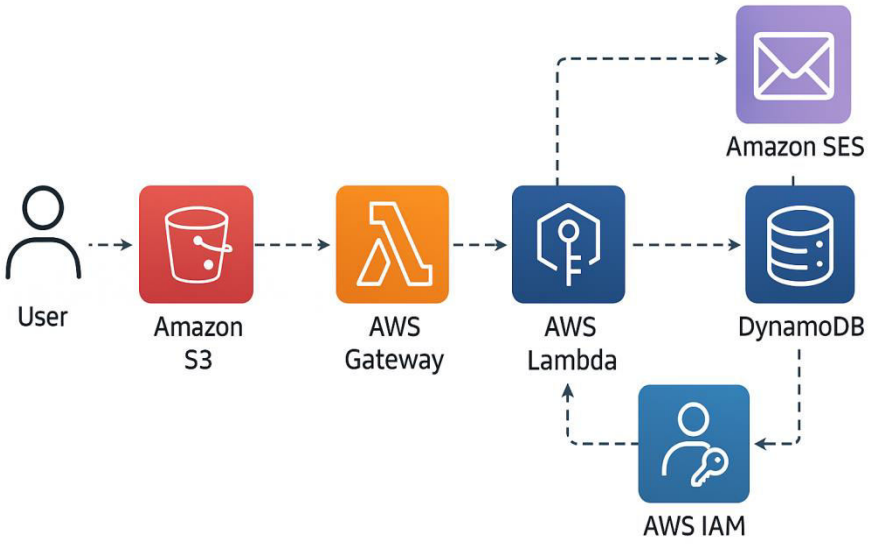


Figure: AWS Serverless Workflow Diagram

IV. CONCLUSION

This project showcases a smart and cost-effective way to monitor and control AWS EC2 instances using a fully serverless architecture. By combining Lambda, API Gateway, DynamoDB, S3, and SES, we built a scalable and

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405300|

automated dashboard that reduces manual effort, improves cloud efficiency, and enables real-time control and alerts—all without managing any servers. It's a practical example of how serverless technologies can simplify infrastructure management and optimize cloud resources.

REFERENCES

1. Serverless Architectures with AWS Lambda: https://d1.awsstatic.com/whitepapers/serverless-architectures-with-aws-lambda.pdf
2. Amazon API Gateway Developer Guide:
3. https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html
4. Amazon S3 User Guide: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
5. Amazon DynamoDB Developer Guide:
6. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
7. Amazon SES Developer Guide: https://docs.aws.amazon.com/ses/latest/dg/Welcome.html
8. AWS IAM User Guide: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details