RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

CS3532 – Advanced AWS Cloud computing
Project Case Study Report

Deploying a Serverless Architecture for Real-Time S3 Event Notifications using AWS EventBridge, Lambda, and SNS

By

RISHEK J(1RVU23CSE376)
SHIVANAND REDDY(1RVU23CSE431)
RISHIKESH A.R(1RVU23CSE378)

School of Computer Science and Engineering

RV University, Bangalore

2025-2026

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

School of Computer Science and Engineering

CERTIFICATE

Certified that the CS3532 Advanced AWS Cloud computing Project Case Study titled " **Deploying a Serverless Architecture for Real-Time S3 Event Notifications using AWS EventBridge, Lambda, and SNS "**
is carried out by RISHEK J(1RVU23CSE376), RISHEKESH A R (1RVU23CSE378), SHIVANAND REDDY(1RVU23CSE431) who is a bonafide student of the School of Computer Science and Engineering, RV University, Bengaluru, during the year 2025–26. It is certified that all corrections/ suggestions from all the continuous internal evaluations have been incorporated in the project and in this report.

Faculty Guide Program Director

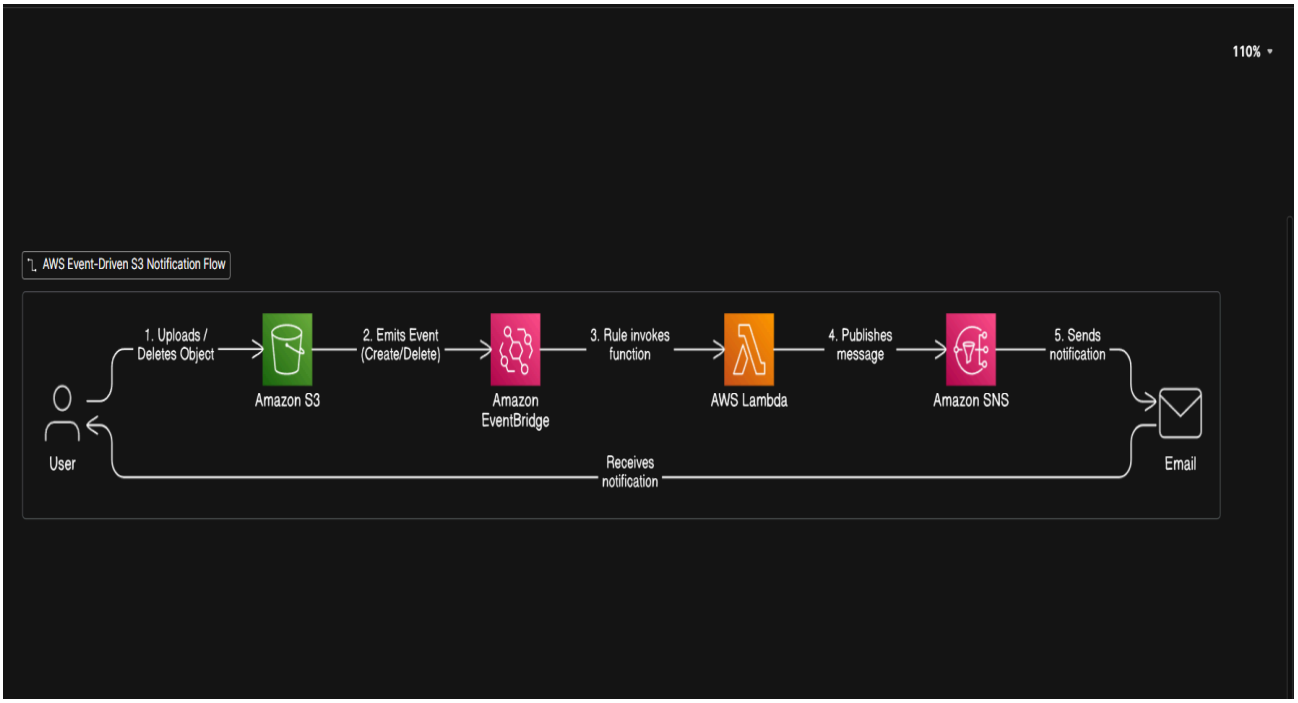
1. Introduction

Cloud computing has transformed the way modern applications are designed and deployed, offering scalable, cost-efficient, and highly available infrastructure services. Among the various cloud architectures, **serverless computing** has emerged as a powerful paradigm that allows developers to build applications without provisioning or managing servers. Instead, the cloud provider handles resource allocation, scaling, and maintenance, enabling developers to focus solely on core application logic.

This project demonstrates a fully **event-driven, serverless architecture** built on Amazon Web Services (AWS) to automate real-time notifications for file-related activities in an Amazon S3 bucket. Whenever a user uploads or deletes an object in a specified bucket, the system instantly detects the event, processes it, and sends a formatted, user-friendly email alert to the subscribed recipient. The workflow leverages AWS EventBridge for event routing, AWS Lambda for compute processing, and Amazon SNS for notification delivery, showcasing an efficient and modular cloud-native design.

The significance of this project lies in its practical demonstration of real-world cloud concepts such as automation, monitoring, event-driven architecture, and serverless workflows. Such systems are widely used in enterprises for audit logging, security monitoring, content management, data pipeline triggers, and operational alerting. By successfully implementing this architecture, the project highlights the importance of serverless automation in modern IT environments, emphasizing its benefits in terms of scalability, cost optimization, reliability, and reduced operational complexity.

2. Architecture



The architecture implemented in this project follows a fully serverless, event-driven workflow designed to provide real-time notifications based on object events in Amazon S3. The structure eliminates the need for traditional server provisioning or maintenance, leveraging AWS-managed services to ensure scalability, reliability, and operational efficiency.

The solution uses **Amazon S3** as the event source, **Amazon EventBridge** as the event routing layer, **AWS Lambda** as the compute engine, and **Amazon SNS** as the communication layer. Each service plays a distinct role in enabling seamless event detection, processing, and alert delivery.

Key Architectural Components

1. **Amazon S3 (Simple Storage Service)**
    - Serves as the storage layer where users upload or delete objects.
    - Configured to emit event notifications (Object Created / Object Deleted) to EventBridge.
    - Acts as the trigger point for the entire workflow.
2. **Amazon EventBridge**
    - Functions as the event bus responsible for receiving events from S3.

- Uses event-pattern rules to filter S3 events and route only relevant ones (create/delete actions) to the Lambda function.

- Decouples services to ensure modularity and ease of scaling.

3. **AWS Lambda**

    - Executes the business logic in response to S3 events forwarded by EventBridge.

    - A Python-based function parses event details, identifies event type, extracts metadata (timestamp, file name, file size), and formats a notification message.

    - Provides serverless compute with pay-per-use efficiency.

4. **Amazon SNS (Simple Notification Service)**

    - Receives processed event messages from Lambda.

    - Publishes email notifications to subscribed recipients, ensuring immediate delivery.

    - Enables easy integration of additional notification channels such as SMS or webhook endpoints.

5. **AWS IAM (Identity & Access Management)**

    - Ensures secure communication among services through least-privilege roles.

    - Grants Lambda permissions to publish messages to SNS.

3. Features / AWS Services Description

Feature	Description
Event-Driven Automation	System automatically reacts to S3 events (file upload/deletion) without manual intervention.
Real-Time Notifications	Email alerts are delivered instantly whenever objects are added or removed from the S3 bucket.
Serverless Architecture	No servers to configure or manage; AWS handles infrastructure provisioning, scaling, and maintenance.
Scalable & Cost-Efficient	Pay only for consumed compute and messaging events, ensuring optimized cloud cost usage.
Secure Role-Based Access	IAM roles and policies ensure secure interaction between AWS services following least-privilege access.
Modular & Extensible Design	Future integrations such as SMS, Slack notifications, or DynamoDB logging can be easily added

AWS Services Used and Their Role

1. Amazon S3 (Simple Storage Service)

Amazon S3 is the primary storage service used to upload, store, and manage objects. In this project, S3 serves as the event source.

Key Features

- Object storage with unlimited scalability
- Built-in event notification capabilities

- High durability (99.999999999% — 11 nines)

Use Case in the Project

- Detects file uploads and deletions
- Generates event messages and forwards them to EventBridge

2. Amazon EventBridge

Amazon EventBridge acts as the event router and filtering layer in the architecture.

Key Features

- Serverless event bus for routing events
- Ability to define pattern-based rules
- Integrates natively with AWS services

Use Case in the Project

- Receives S3 events
- Filters only *Object Created* and *Object Deleted* events from the target bucket
- Triggers Lambda function only when rule conditions match

3. AWS Lambda

AWS Lambda provides event-driven compute without requiring servers.

Key Features

- Fully serverless compute function
- Automatically scales based on events
- Pay only for execution time (per millisecond billing)

Use Case in the Project

- Processes incoming S3 events

- Parses event data and extracts details such as bucket name, object key, timestamp, and size
- Formats a readable email message
- Publishes message to SNS topic

4. Amazon SNS (Simple Notification Service)

Amazon SNS acts as the messaging and notification delivery layer.

Key Features

- Supports multi-protocol delivery (Email, SMS, HTTP, Lambda, etc.)
- High throughput and reliability
- Topic-based message broadcasting

Use Case in the Project

- Receives formatted notification from Lambda
- Delivers email alerts to the subscribed user instantly

5. AWS IAM (Identity and Access Management)

IAM secures access between AWS services through roles and policies.

Key Features

- Centralized security and access control
- Fine-grained permission policies
- Role assignment for service-to-service access

Use Case in the Project

- Lambda execution role created with SNS publish permissions
- Ensures secure, least-privileged access to required AWS services

4. Implementation Details

The implementation of this serverless architecture involved the configuration of four core AWS services: Amazon S3, Amazon SNS, AWS Lambda, and Amazon EventBridge. The following steps outline the process undertaken to build and connect these components into a cohesive workflow.

1.Setting up the S3 Bucket (The Event Source) :

The foundation of the project is an Amazon S3 bucket, which serves as the storage for objects and the source of events.

1. **Bucket Creation:** An S3 bucket named aws-rishek-cp3-project was created in the ap-south-1 (Mumbai) region. S3 bucket names must be globally unique, and choosing a region positions the storage closer to its users for lower latency.
2. **Enabling Event Notifications:** The critical step was to configure the bucket to emit events. This was achieved by navigating to the bucket's **Properties** tab and creating an **Event Notification**.
    - **Event Types:** To fulfill the project requirements, both All object create events and All object delete events were selected. This configuration ensures that S3 generates a notification for any upload (including multipart uploads) and any deletion (including versioned deletes).
    - **Destination:** The destination for these notifications was set to **Amazon EventBridge**. This modern approach allows for flexible and advanced routing, decoupling the S3 bucket from the downstream processing logic. By selecting this, S3 was granted the necessary permissions to send events to the default EventBridge bus in the account.

2. Configuring Amazon SNS (The Notification Channel)

To deliver the final alert, an Amazon SNS topic and an email subscription were established.

1. **Topic Creation:** An SNS Topic named s3ObjectCreatedTopic was created with the **Standard** type. The Standard type was chosen for its high throughput and support for multiple subscription types (Email, SMS, Lambda, etc.), providing flexibility for future enhancements.
2. **Subscription Creation:** An email subscription was added to this topic.
    - **Protocol:** The protocol was set to Email.
    - **Endpoint:** A personal email address was provided as the endpoint.
    - **Confirmation:** Upon creation, AWS sent a confirmation email to the specified address. The link within this email was clicked to confirm the subscription. This is a mandatory security step to ensure the owner of the email address consents to receiving notifications. Without confirmation, the subscription status remains "PendingConfirmation," and no messages are delivered.

3 . Implementing the Processing Logic with AWS Lambda

The core logic of the application resides in an AWS Lambda function, which processes events and formats the notifications.

1. **Function Creation:** A Lambda function named format-s3-notification-email was created.
    - **Runtime:** The **Python 3.9** runtime was selected due to its simplicity, extensive library support (including the Boto3 AWS SDK), and excellent performance for this type of I/O-bound task.
    - **Execution Role:** An IAM role was automatically created alongside the function. This role provides the function with the basic permissions to execute and write logs to Amazon CloudWatch.

2. **Adding Permissions:** The default IAM role was insufficient for the function's needs. It was modified by attaching the AWS managed policy AmazonSNSFullAccess. This policy grants the function the sns:Publish permission, allowing it to send messages to the SNS topic. In a production environment, a more restrictive custom policy would be created to limit publishing rights to only the specific SNS topic ARN.
3. **Developing the Function Code:** The Python code was written to perform several key actions:
    - **Event Parsing:** It inspects the incoming event JSON from EventBridge to determine the detail-type (Object Created or Object Deleted).
    - **Conditional Logic:** It uses an if/elif block to handle the two event types differently, as the "Object Created" event contains object size metadata while the "Object Deleted" event does not. This prevents KeyError exceptions.
    - **Message Formatting:** It dynamically constructs a user-friendly subject and message body, extracting relevant details like the bucket name, object key, and event time. For uploads, it includes a formatted file size (in B, KB, or MB).
    - **SNS Publishing:** It uses the boto3 library to call the publish() API on the SNS client, passing the target topic's ARN, the formatted subject, and the message body.

4. Routing Events with Amazon EventBridge

Amazon EventBridge serves as the central message bus, connecting the event source (S3) to the event processor (Lambda).

1. **Rule Creation:** An EventBridge rule named NotifyOnS3ObjectCreated was created on the default event bus.
2. **Defining the Event Pattern:** The core of the rule is its event pattern, which acts as a filter. The pattern was written in JSON to specifically match the desired events:

```
{
 "source": ["aws.s3"],
 "detail-type": ["Object Created", "Object Deleted"],
 "detail": {
  "bucket": {
   "name": ["aws-rishek-cp3-project"]
  }
 }

}
```

3. This pattern ensures that the rule only triggers for events originating from Amazon S3, for the specific actions of object creation or deletion, and exclusively within the aws-rishek-cp3-project bucket. This precision is vital for preventing the Lambda function from being invoked by irrelevant events, thus controlling costs and ensuring correct behavior.
4. **Configuring the Target:** The target for this rule was set to the **AWS Lambda function** (format-s3-notification-email) created in the previous step. By setting this target, EventBridge was automatically granted the necessary permissions to invoke the Lambda function whenever a matching event is detected.

Through step-by-step configuration of AWS services — S3, EventBridge, Lambda, SNS, and IAM — this project achieved a fully automated notification pipeline capable of processing and notifying users in real-time when file operations occur in the designated S3 bucket. The entire workflow runs without any servers, ensuring scalability, low-cost operation, and near-zero maintenance.

5. Results and Conclusion

5.1 Results

The project successfully achieved its objective of implementing a fully serverless, event-driven AWS architecture capable of sending real-time email notifications when files are uploaded to or deleted from an Amazon S3 bucket. Through the integration of S3, EventBridge, Lambda, and SNS, the system demonstrated seamless communication between cloud components without the need for manual intervention or server management.

The system was tested with multiple file upload and deletion scenarios. Each event triggered the EventBridge rule, invoked the Lambda function for event parsing and message formatting, and delivered a clear, readable email notification via SNS. The notifications included important details such as the bucket name, object key, timestamp, and, in the case of uploads, file size in human-readable format. All events were processed within milliseconds, highlighting the efficiency and scalability of AWS serverless services.

5.2 Conclusion

This project demonstrates the practical application of serverless computing and event-driven design on AWS, showcasing how cloud-native services can be combined to build highly scalable, automated, and cost-efficient workflows. By utilizing Amazon S3 for storage, Amazon EventBridge for event routing, AWS Lambda for compute, and Amazon SNS for notifications, the project highlights the advantages of modern cloud architectures — including zero server maintenance, fine-grained event handling, and pay-as-you-go efficiency.

The system not only provides real-time monitoring and alerting for S3 object activities but also represents a foundation that can be extended for enterprise-level automation scenarios such as audit tracking, security alerting, content processing pipelines, or DevOps monitoring systems. Overall, this project reinforces the value of serverless technologies in modern cloud environments and emphasizes their importance in building scalable and intelligent automation systems.

6. References

Amazon Web Services. (2024). *Amazon S3: Developer guide*. AWS Documentation.

https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html

Amazon Web Services. (2024). *Amazon EventBridge user guide*. AWS Documentation.

https://docs.aws.amazon.com/eventbridge/latest/userguide/what-is-amazon-eventbridge.html

Amazon Web Services. (2024). *AWS Lambda developer guide*. AWS Documentation. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Amazon Web Services. (2024). *Amazon SNS developer guide*. AWS Documentation. https://docs.aws.amazon.com/sns/latest/dg/welcome.html

Amazon Web Services. (2024). *AWS Identity and Access Management (IAM) documentation*. AWS Documentation . https://docs.aws.amazon.com/iam/

Hendrickson, S., Sturdevant, S., Harter, T., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. (2016). *Serverless computation with AWS Lambda*. University of Wisconsin-Madison Technical Report.

Villamizar, M., Garcés, O., Ochoa, L., & Salamanca, L. (2019). *Evaluation of serverless computing architectures for real-time cloud applications*. Journal of Cloud Computing, 8(1), 1–14.

Gannon, D., Barga, R., & Sundaresan, N. (2017). *Cloud-native applications*. IEEE Cloud Computing, 4(5), 16–21.

Hashizume, K., Rosado, D. G., Fernández-Medina, E., & Fernandez, E. B. (2013). *An analysis of security issues for cloud computing*. Journal of Internet Services and Applications, 4(1), 1–13.