

A Mini Project Report
on
Send Email using Lambda and Amazon SES

Submitted in partial fulfilment of the requirements
of the degree of
BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING
by

Rutikesh Rajendra Sawant 65

Niraj Nitin Surve 68

Om Bhushan Tambat 69

Supervisor

Prof. S. S. Tathare



DEPARTMENT OF COMPUTER ENGINEERING

Gharda Institute of Technology

A/P: Lavel, Tal: Khed, Dist: Ratnagiri, 415708

Mumbai University

[2022-2023]

CERTIFICATE

This is to certify that the Mini Project entitled “**Send Email Using Lambda and Amazon SES**” is a bonafide work of **Rutikesh Rajendra Sawant 65, Niraj Nitin Surve 68, Om Bhushan Tambat 69** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of “**Bachelor of Engineering**” in “**Computer Engineering**”.

(Prof. S. S. Tathare)

Supervisor & Mini Project Co-ordinator

(Prof. R. R. Bane)

Head of Computer Department

(Dr. P. B. Patil)

Principal

ABSTRACT

This project report focuses on the implementation of a system that enables sending emails using AWS Lambda and Amazon SES. The project uses a serverless architecture, where Lambda functions are triggered by various events such as API Gateway requests or S3 bucket events.

The report describes the implementation details of the system, including setting up the AWS services, creating the Lambda function, and configuring the Amazon SES service for sending emails. The report also discusses the different use cases for the system, such as sending email notifications for various events or sending mass emails to a large group of recipients.

The project's primary goal is to demonstrate the flexibility and ease of use of AWS Lambda and Amazon SES in building email delivery systems. The report also discusses the advantages of serverless architectures, including cost savings, scalability, and reduced maintenance overhead.

Finally, the project report concludes with a discussion of the future directions for the system, including integrating with other AWS services, enhancing the system's functionality, and improving its performance. Overall, the project report provides a comprehensive overview of using AWS Lambda and Amazon SES to build email delivery systems in a serverless architecture.

ACKNOWLEDGEMENT

In this project, we have made an effort. Nevertheless, it would not have been possible without the generous support and assistance of numerous people and organizations. We would like to express our gratitude towards Gharda Institute of Technology. We would like to extend our sincere thanks to the principal of our institute Dr. P. B. Patil Sir. We would like to express our gratitude towards the HOD of the Computer Department Dr. R. R. Bane Sir for the kind cooperation which helped us in completion of this project. We are highly indebted to our mentor for this project Prof. R.B. Pawar Sir for his kind cooperation, encouragement, guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project. Our thanks and appreciations also go to our classmates in developing the project and people who have willingly helped us out with their abilities.

CONTENT

ABSTRACT	III
ACKNOWLEDGEMENT	IV
1. Introduction	1
1.1 Introduction	1
1.2 Objectives	2
2. Methodology	3
1.1 Methodology	3
1.2 Details of Hardware and Software	3
1.3 Deployment Considerations	3
1.4 Procedure	4
1.5 Result	9
3. Summary and Conclusion	10
4. References	11

Chapter 1

Introduction

1.1 Introduction

In today's digital age, email has become an essential communication medium for businesses and individuals alike. Sending emails to a large number of recipients can be a daunting task, especially if the emails contain personalized content or attachments. AWS Lambda and Amazon SES can be used to build an email delivery system that is cost-effective, scalable, and highly available. In this project report, we describe the implementation details of such a system. In today's fast-paced and digital world, email has become an essential communication tool. Whether it's for personal or business purposes, email enables us to send messages, documents, images, and other files to people across the globe. However, sending emails to a large number of recipients can be a challenging task, especially if the emails contain personalized content or attachments.

This is where AWS Lambda and Amazon SES come in. AWS Lambda is a serverless computing service that allows you to run your code without provisioning or managing servers. It is highly scalable, cost-effective, and can be used to build various types of applications, including email delivery systems. Amazon SES is a cloud-based email sending service that allows you to send and receive emails using your email addresses and domains.

By combining these two AWS services, you can build a powerful and flexible email delivery system that can handle a large volume of emails, personalize content, and handle attachments. This project report describes the implementation details of such a system, including setting up the AWS services, creating the Lambda function, and configuring the Amazon SES service for sending emails.

The primary goal of this project is to demonstrate the flexibility and ease of use of AWS Lambda and Amazon SES in building email delivery systems. The report also discusses the advantages of serverless architectures, including cost savings, scalability, and reduced maintenance overhead. The project report provides a comprehensive overview of using AWS Lambda and Amazon SES to build email delivery systems in a serverless architecture, along with possible future directions for the system.

1.2 Objectives

1.2.1 Objective

The main objective of this project is to demonstrate how to use AWS Lambda and Amazon SES to build a system that can send emails to a large number of recipients. The system should be able to handle personalized content and attachments, be scalable, and have a high degree of availability.

Chapter 2

Methodology

2.1 Methodology

The email delivery system implemented in this project uses a serverless architecture, where Lambda functions are triggered by various events such as API Gateway requests or S3 bucket events. The Lambda function takes the recipient email addresses, message content, and attachments (if any) as input parameters and uses Amazon SES to send the email.

The following are the steps involved in the implementation:

1. Set up an AWS account and create an IAM user with the necessary permissions to access the AWS services.
2. Set up Amazon SES by verifying the email addresses and domains that will be used to send emails.
3. Create an S3 bucket to store the email content and attachments.
4. Create a Lambda function that will handle the email sending.
5. Configure the Lambda function to access the S3 bucket and Amazon SES.
6. Test the Lambda function by triggering it with sample data.

2.2 Details of Hardware and Software

2.2.1 Minimum Hardware Requirements

- Intel Pentium Processor
- 4GB RAM
- 500GB Hard Disk

2.2.2 Minimum Software Requirements

- Windows 7 or above
- Amazon AWS account
- Any internet browser with stable internet

2.3 Deployment considerations

Deploying and updating the Lambda function is an important aspect of the email delivery system. The following are some considerations that should be taken into account when deploying and updating the Lambda function:

1. Versioning: It is important to version the Lambda function to ensure that changes can be rolled back if necessary. This can be done by using the AWS Lambda versioning feature or by creating aliases for different versions of the function.

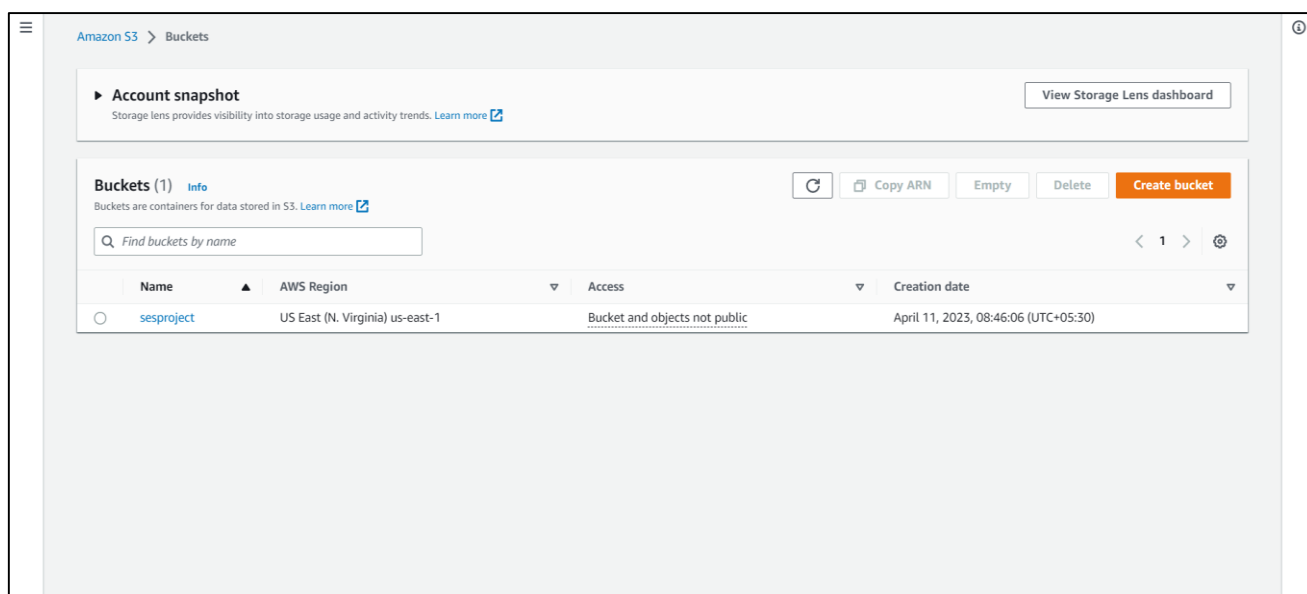
2. **Testing:** Before deploying the Lambda function to production, it is important to test it thoroughly to ensure that it works as expected. This can be done by creating a test environment that closely mimics the production environment.
3. **Deployment automation:** It is recommended to automate the deployment process as much as possible to reduce the risk of errors and ensure consistency across environments. This can be achieved by using tools such as AWS CodeDeploy or AWS CloudFormation.
4. **Monitoring:** Once the Lambda function is deployed, it is important to monitor it for errors and performance issues. This can be done by setting up alerts and dashboards in AWS CloudWatch.
5. **Updating:** When updating the Lambda function, it is important to follow a versioning strategy and ensure that the new version is thoroughly tested before it is deployed to production. It is also important to communicate any changes to stakeholders to minimize the impact of the update.

In addition to the Lambda function, the deployment considerations should also take into account the configuration of Amazon SES. This includes verifying email addresses and domains, setting up email sending limits, and configuring bounce and complaint handling.

By following best practices and using automation tools, the deployment process can be streamlined and made more efficient.

2.4 Procedure

1. Create S3 bucket



2. Create User and give permission of “AmazonSESFullAccess”

The screenshot shows the AWS IAM console page for a user named 'ses-user-2'. The breadcrumb navigation is 'IAM > Users > ses-user-2'. The user's ARN is 'arn:aws:iam::035188157961:user/ses-user-2'. Console access is disabled. Access keys 1 and 2 are not enabled. The user was created on April 10, 2023, at 19:52 (UTC+05:30). The 'Permissions' tab is selected, showing one policy attached: 'AmazonSESFullAccess' (AWS managed, attached directly).

ses-user-2 Delete

Summary

ARN arn:aws:iam::035188157961:user/ses-user-2	Console access Disabled	Access key 1 Not enabled
Created April 10, 2023, 19:52 (UTC+05:30)	Last console sign-in -	Access key 2 Not enabled

Permissions Groups Tags Security credentials Access Advisor

Permissions policies (1) Remove Add permissions

Permissions are defined by policies attached to the user directly or through groups.

<input type="checkbox"/>	Policy name	Type	Attached via
<input type="checkbox"/>	AmazonSESFullAccess	AWS managed	Directly

3. Create Role and give permissions of “AmazonS3FullAccess”, “CloudWatchFullAccess” and “AmazonSESFullAccess”

The screenshot shows the AWS IAM console page for a role named 'lambdasesproject'. The breadcrumb navigation is 'IAM > Roles > lambdasesproject'. The role's ARN is 'arn:aws:iam::035188157961:role/lambdasesproject'. The role was created on April 10, 2023, at 20:16 (UTC+05:30). The last activity was 21 hours ago. The 'Permissions' tab is selected, showing three policies attached: 'AmazonS3FullAccess', 'CloudWatchFullAccess', and 'AmazonSESFullAccess' (all AWS managed).

lambdasesproject Delete

Allows Lambda functions to call AWS services on your behalf.

Summary Edit

Creation date April 10, 2023, 20:16 (UTC+05:30)	ARN arn:aws:iam::035188157961:role/lambdasesproject
Last activity 21 hours ago	Maximum session duration 1 hour

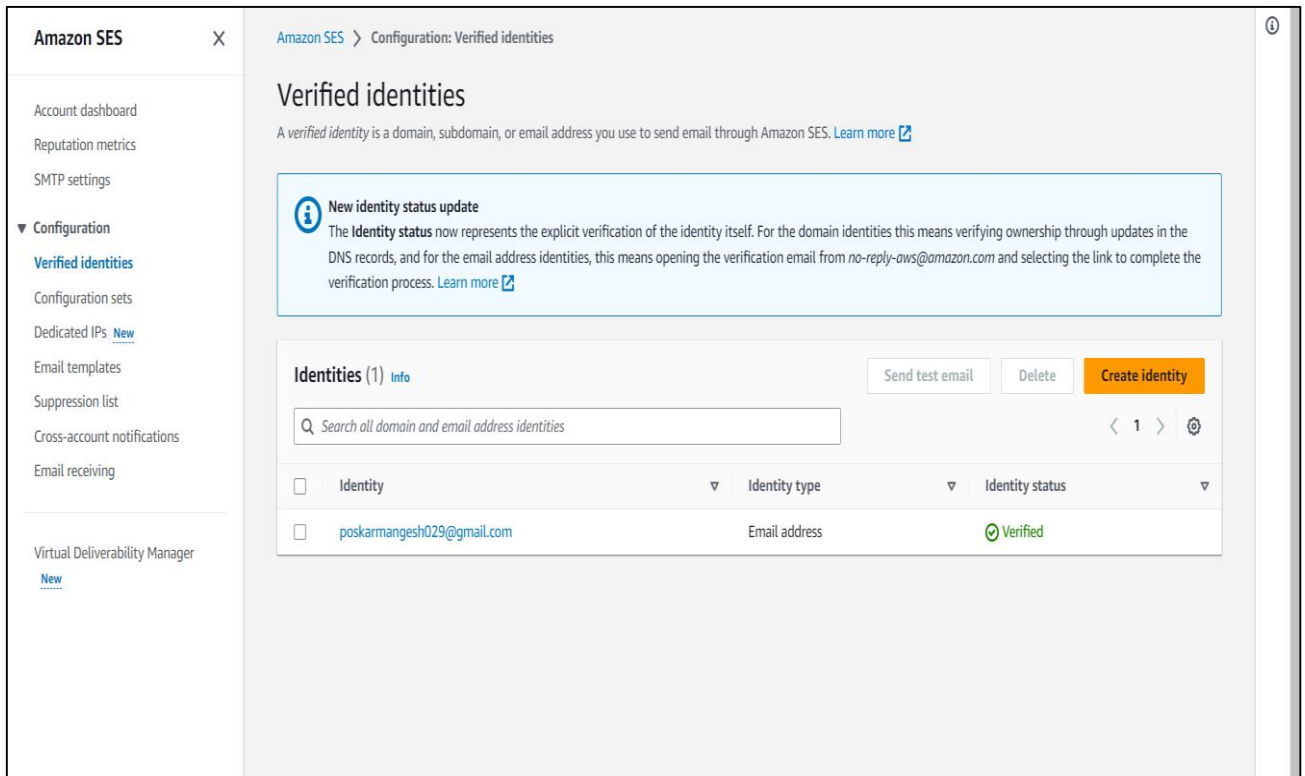
Permissions Trust relationships Tags Access Advisor Revoke sessions

Permissions policies (3) Simulate Remove Add permissions

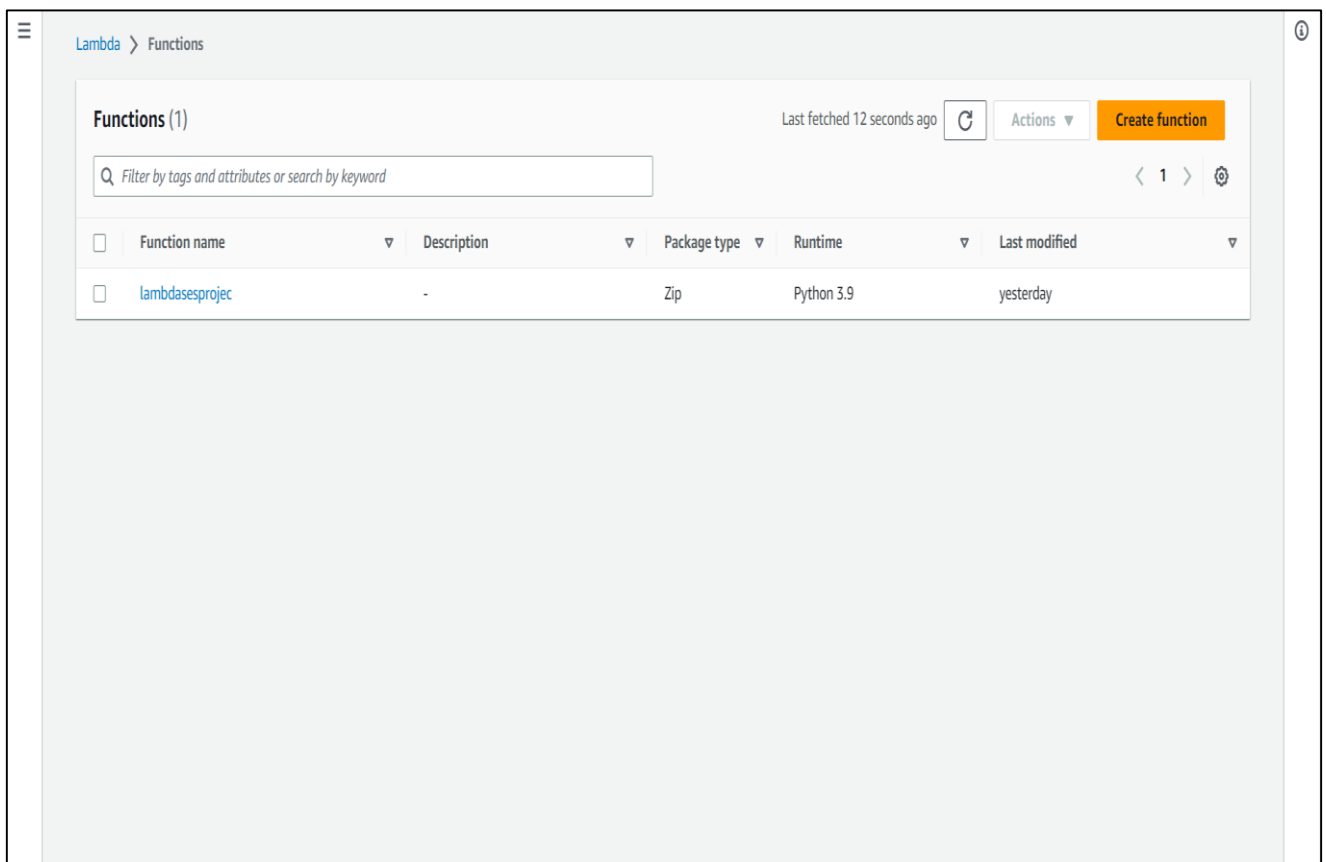
You can attach up to 10 managed policies.

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	AmazonS3FullAccess	AWS managed	Provides full access to all buckets via the AWS Management Console.
<input type="checkbox"/>	CloudWatchFullAccess	AWS managed	Provides full access to CloudWatch.
<input type="checkbox"/>	AmazonSESFullAccess	AWS managed	Provides full access to Amazon SES via the AWS Management Console.

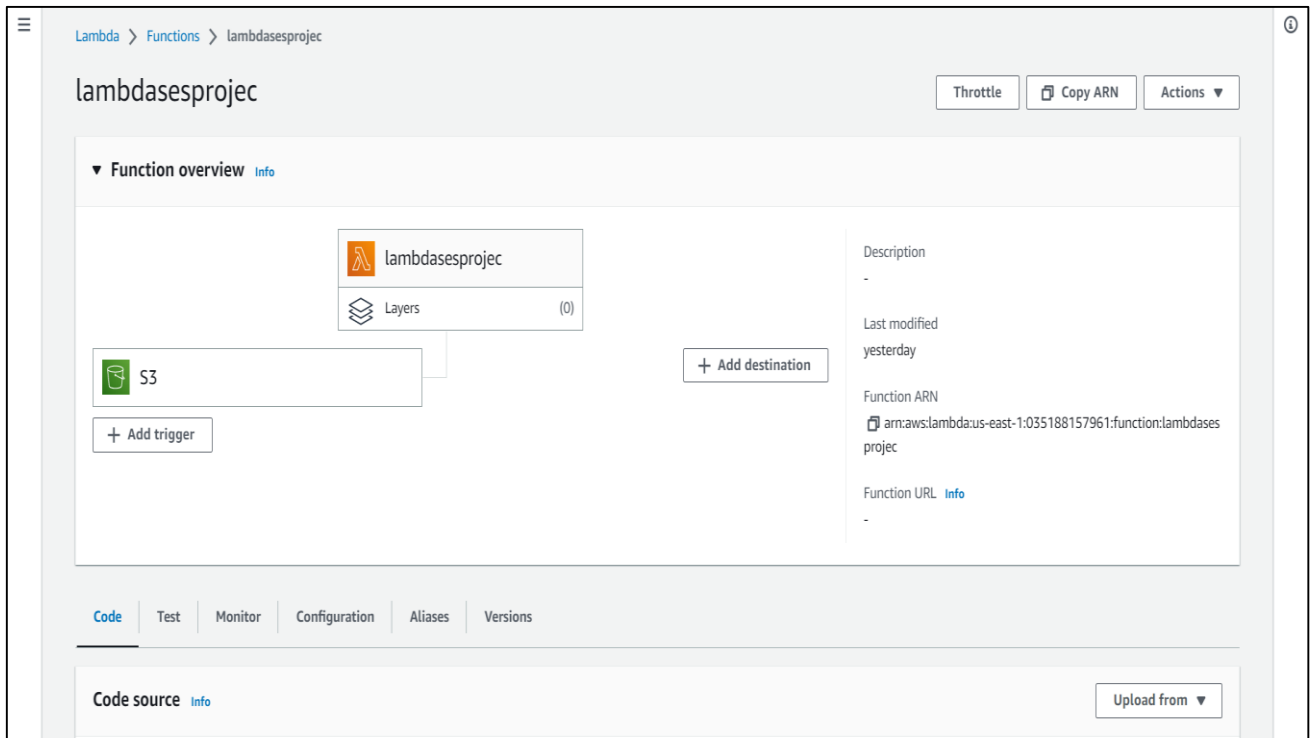
4. Go to Amazon SES and verify email address



5. Then go to Amazon Lambda and create a lambda function



6. On that lambda function add previously created S3 bucket as a trigger



7. Under code section paste the following code and deploy it

```
import json
import boto3

def lambda_handler(event, context):
    file_name = event['Records'][0]['s3']['object']['key']
    bucketName=event['Records'][0]['s3']['bucket']['name']
    print("Event details : ",event)
    print("File Name : ",file_name)
    print("Bucket Name : ",bucketName)
    subject = 'Event from ' + bucketName
    client = boto3.client("ses")
    body = ""
    <br>

    This is a notification mail to inform you regarding s3 event.
    The file {} is inserted in the {} bucket .

    """.format(file_name, bucketName)
    message = {"Subject": {"Data": subject}, "Body": {"Html": {"Data": body}}}
```

```
response = client.send_email(Source = "poskarmangesh029@gmail.com",
Destination = {"ToAddresses": ["poskarmangesh029@gmail.com"]}, Message =
message)

print("The mail is sent successfully")
```

8. Now add any file in S3 bucket you will get notification on your mail about that file

Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

Summary

Destination s3://sesproject	Succeeded ✔ 1 file, 175.0 B (100.00%)	Failed ✖ 0 files, 0 B (0%)
--------------------------------	--	-------------------------------

Files and folders

Configuration

Files and folders (1 Total, 175.0 B)

Find by name

< 1 >

Name	Folder	Type	Size	Status	Error
Expt2.asm	-	-	175.0 B	✔ Succeeded	-

←

📁

🕒

🗑️

✉️

🕒

🔗

📁

🗨️

⋮

1 of 48

Event from sesproject

Inbox x

🖨️

🔗

poskarmangesh029@gmail.com

via amazonses.com

to me

8:20 AM (0 minutes ago)

★

↶

⋮

This is a notification mail to inform you regarding s3 event. The file Expt2.asm is inserted in the sesproject bucket .

↶ Reply

↷ Forward

9. Also under monitor section through cloudwatch metrics we can observe the log

The screenshot displays the AWS CloudWatch console interface. On the left is a navigation sidebar with sections: 'Alarms' (containing 'In alarm', 'All alarms', 'Billing'), 'Logs' (containing 'Log groups', 'Logs Insights'), 'Metrics', 'X-Ray traces', 'Events', 'Application monitoring', and 'Insights'. Below these are 'Settings' and 'Getting Started'. The main content area is titled 'CloudWatch' and shows details for a specific log group. It includes fields for ARN, Creation time (1 day ago), Retention (Never expire), and Stored bytes. To the right, there are sections for 'Metric filters', 'Subscription filters', and 'Data protection - new' (showing 'Inactive' status and 'Sensitive data found - new'). Below these is a tabbed interface with 'Log streams' selected. The 'Log streams (3)' section contains a search bar, filters for 'Exact match' and 'Show expired', and a table of log streams. The table has columns for 'Log stream' and 'Last event time'.

Log stream	Last event time
2023/04/12/[\$LATEST]b3896422a6834ca2aac35d07ca2af4d	2023-04-12 08:20:23 (UTC+05:30)
2023/04/11/[\$LATEST]ae915fbc130545e9977b147c2d43d5f0	2023-04-11 10:35:18 (UTC+05:30)
2023/04/11/[\$LATEST]0cb43b8126dd460ba667dc83b648ca69	2023-04-11 08:47:11 (UTC+05:30)

2.5 Result

The email delivery system implemented using AWS Lambda and Amazon SES is highly scalable, cost-effective, and easy to maintain. The system can handle a large number of recipients and personalized content, and the cost of running the system is minimal compared to traditional email delivery systems.

The Lambda function can be triggered by various events, such as API Gateway requests or S3 bucket events, allowing for a flexible and customizable solution. The system's serverless architecture provides cost savings, scalability, and reduced maintenance overhead.

Chapter 3

Summary and Conclusion

3.1 Summary

The project report describes the implementation details of an email delivery system using AWS Lambda and Amazon SES. The system is designed to handle a large volume of emails, personalized content, and attachments in a cost-effective and scalable manner. The report outlines the steps involved in setting up the system and discusses the advantages of serverless architectures in building email delivery systems. The system is highly flexible, allowing for triggering by various events and can be easily extended to support more use cases. Overall, the project report provides a comprehensive overview of using AWS Lambda and Amazon SES to build email delivery systems in a serverless architecture.

3.2 Conclusion

In conclusion, AWS Lambda and Amazon SES provide a powerful and flexible platform for building email delivery systems. The serverless architecture provides cost savings, scalability, and reduced maintenance overhead. The system implemented in this project can be extended to support more use cases and can be integrated with other AWS services to provide a more comprehensive solution. Overall, the project report provides a comprehensive overview of using AWS Lambda and Amazon SES to build email delivery systems in a serverless architecture. This project can serve as a reference for anyone looking to implement a similar system or learn more about serverless architectures and AWS services.

Chapter 4

References

4.1 Web Sources

1. <https://youtu.be/A58eQ-ZanvA>
2. <https://repost.aws/knowledge-center/lambda-send-email-ses>