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.

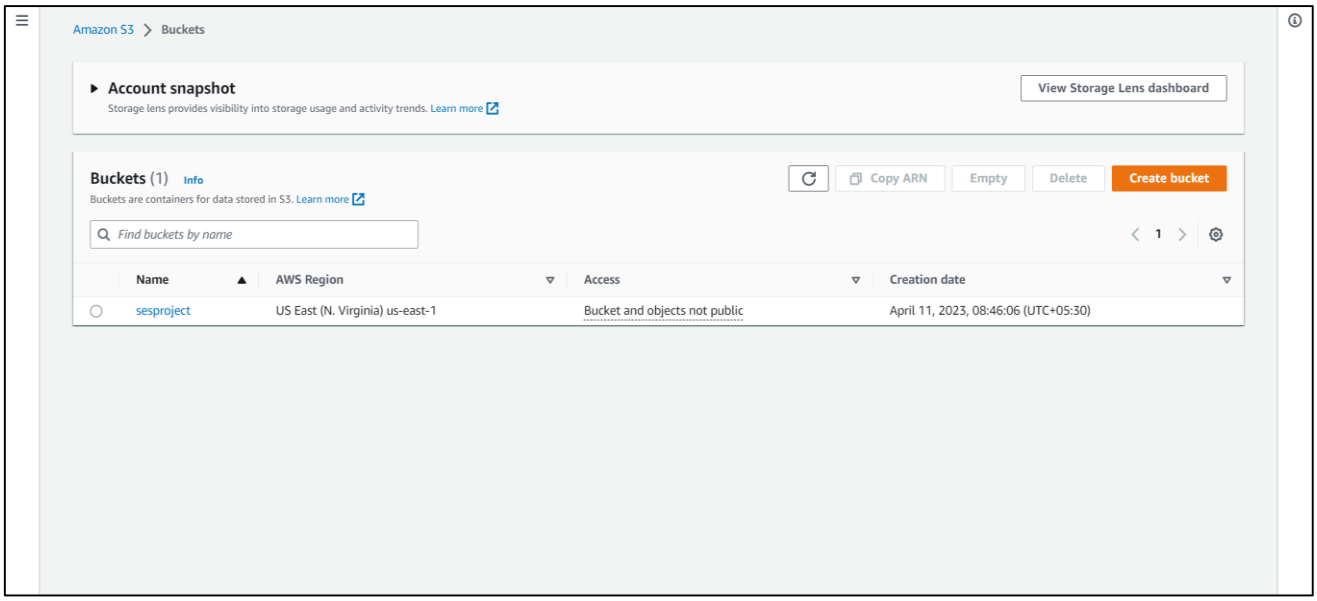
2. Testing: Before deploying the Lambda function to production, it is important to test it thoroughly to ensure that it works as expected. This can be done by creating a test environment that closely mimics the production environment.
3. Deployment automation: It is recommended to automate the deployment 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.

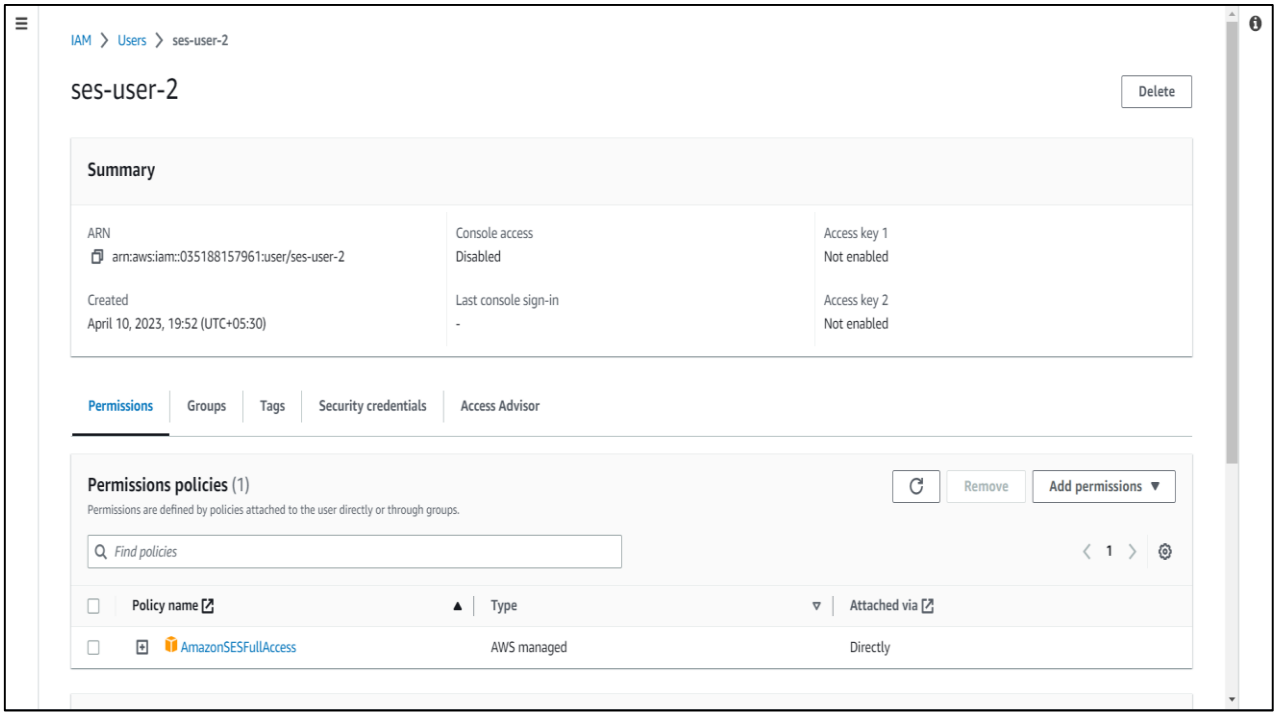
By following best practices and using automation tools, the deployment process can be streamlined and made more efficient.

2.4 Procedure

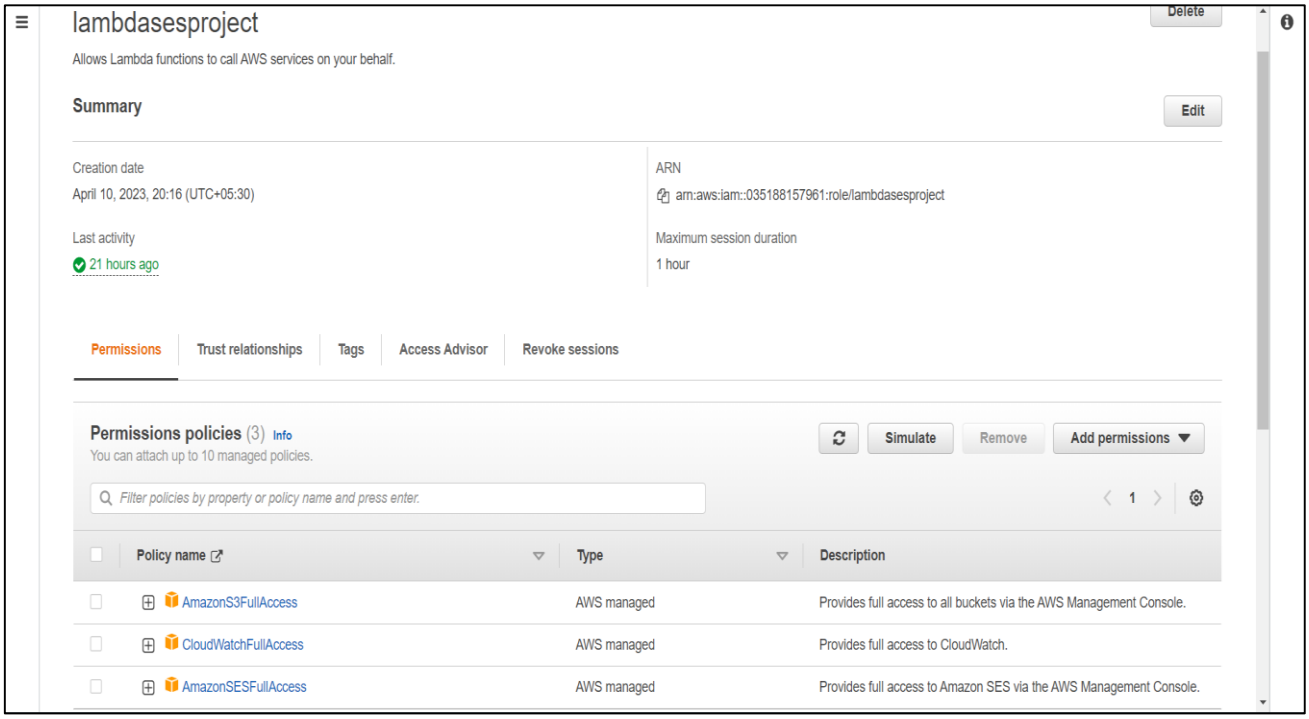
1. Create S3 bucket



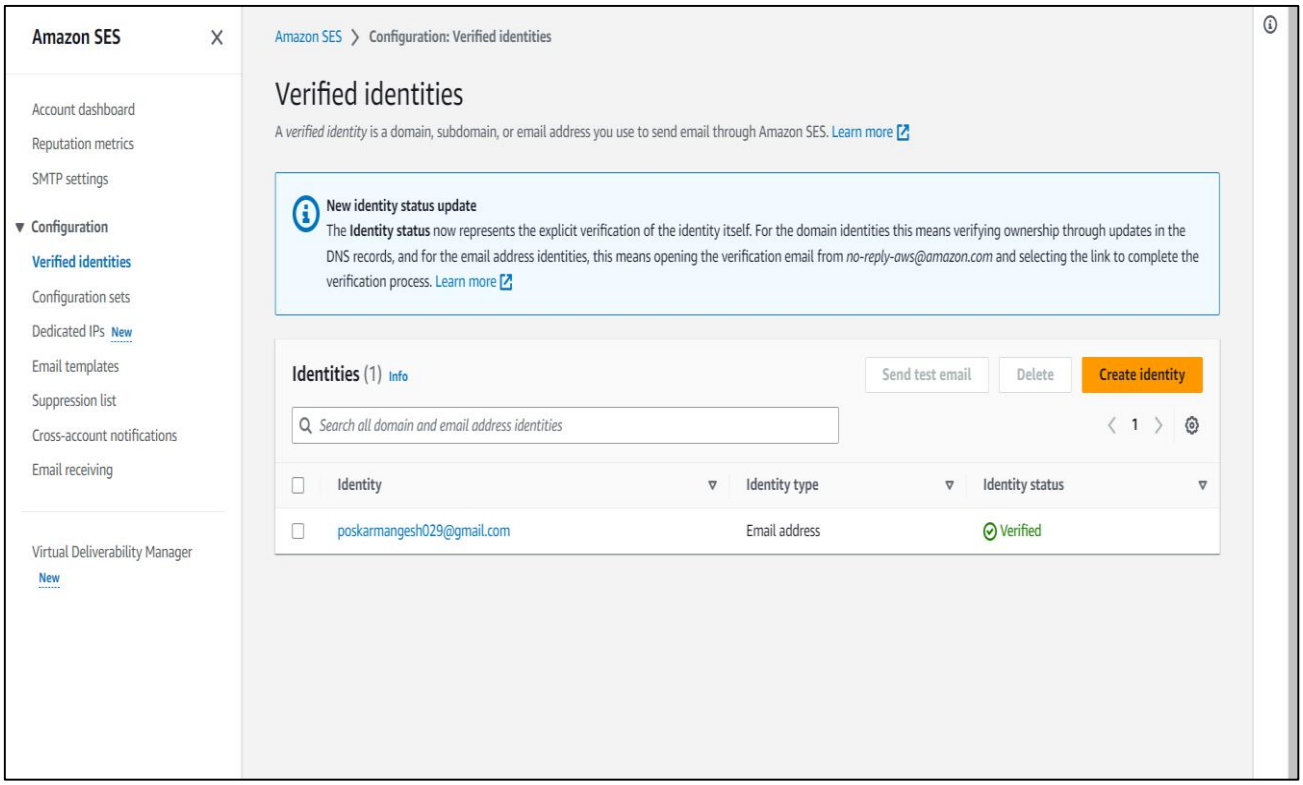
2. Create User and give permission of “AmazonSESFullAccess”



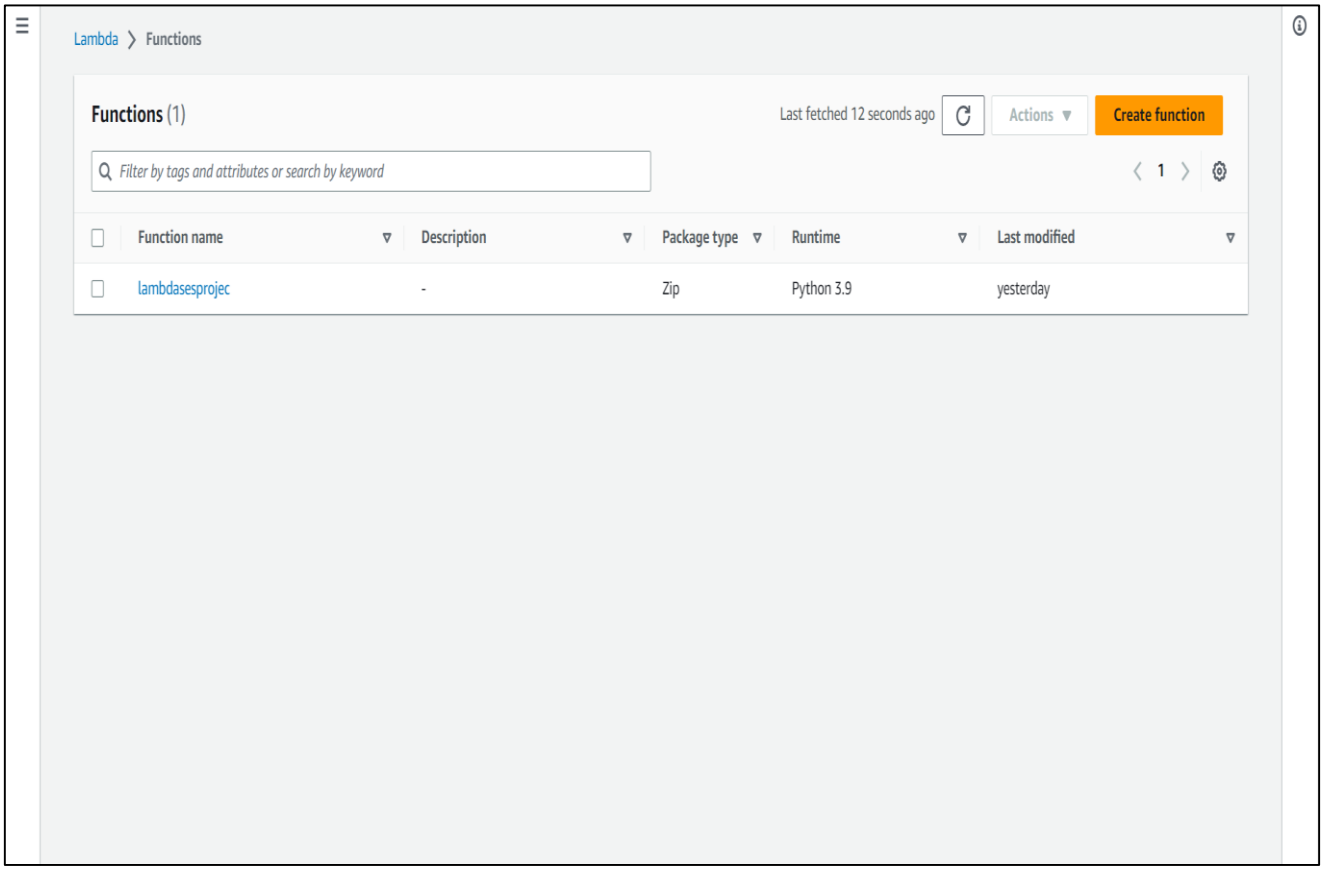
3. Create Role and give permissions of “AmazonS3FullAccess”, “CloudWatchFullAccess” and “AmazonSESFullAccess”



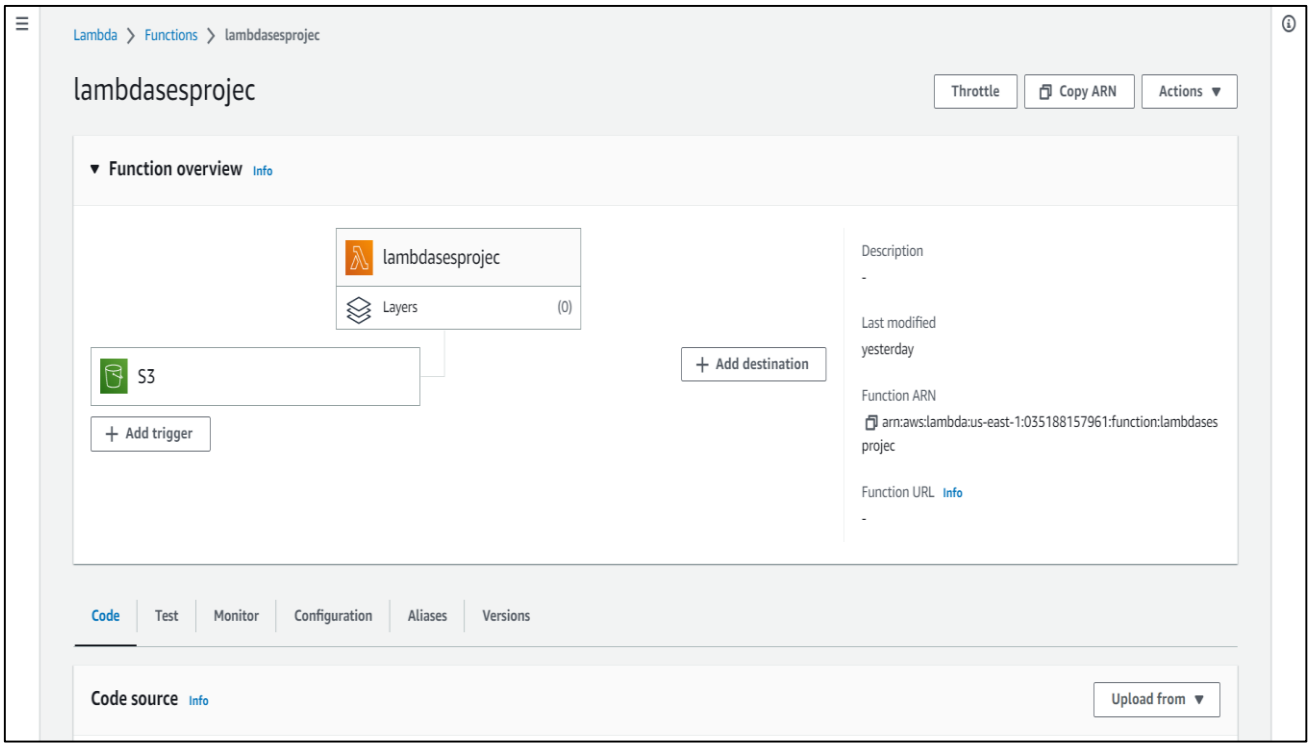
4. Go to Amazon SES and verify email address



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



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



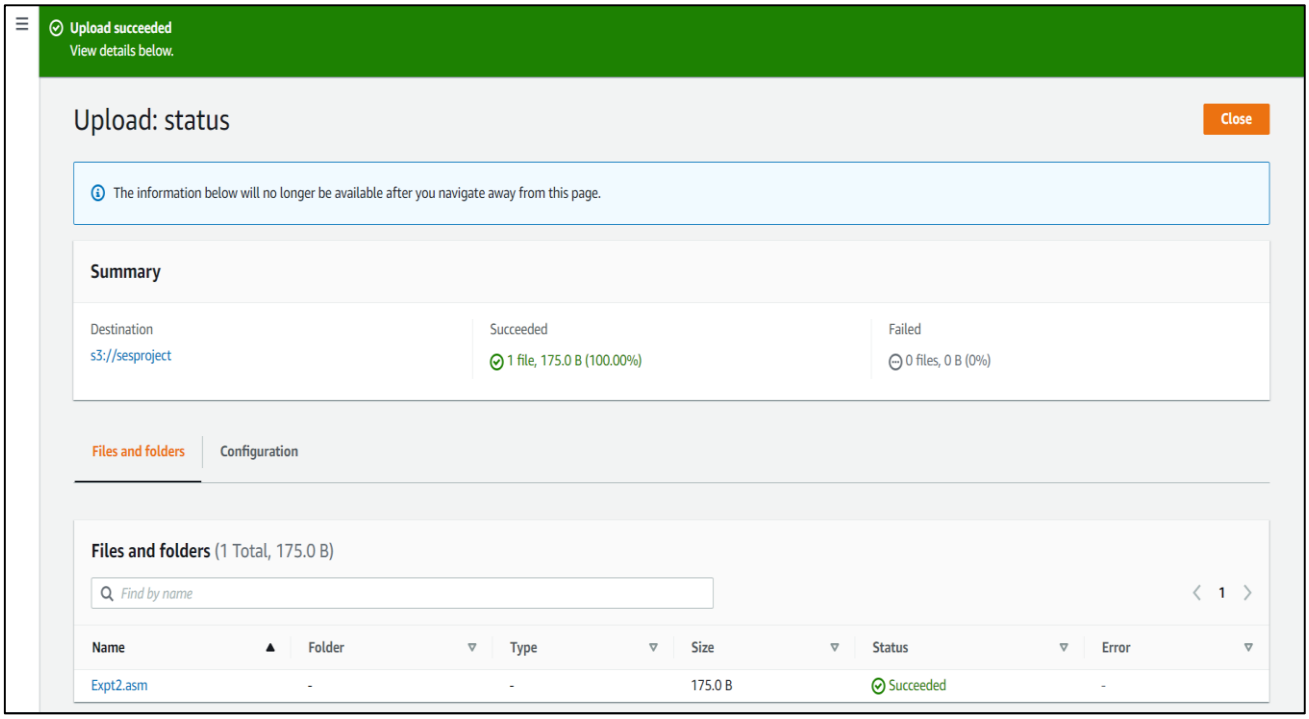
7. Under code section paste the following code and deploy it

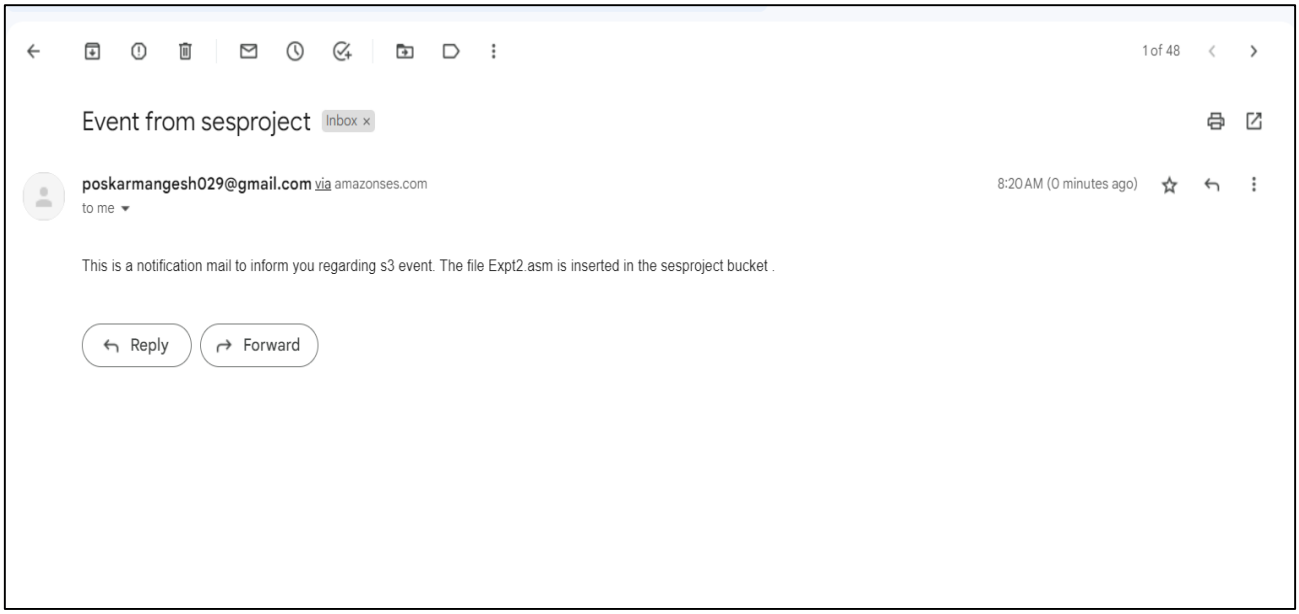
```
import json
import boto3
def lambda_handler(event, context):
    file_name = event['Records'][0]['s3']['object']['key']
    bucketName=event['Records'][0]['s3']['bucket']['name']
    print("Event details : ",event)
    print("File Name : ",file_name)
    print("Bucket Name : ",bucketName)
    subject = '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}}}
```

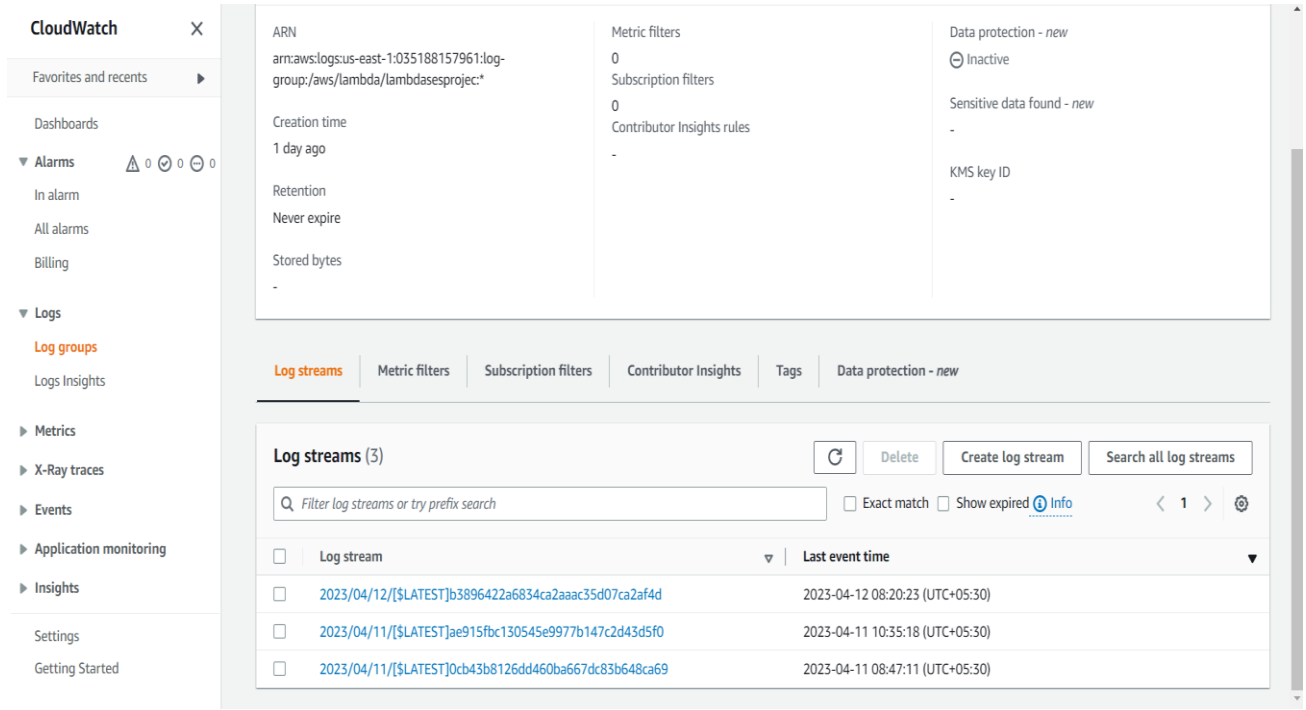
```
    response = client.send_email(Source = "poskarmangesh029@gmail.com",
Destination = {"ToAddresses": ["poskarmangesh029@gmail.com"]}, Message =
message)
    print("The mail is sent successfully")
```

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





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



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