Automated Mass Email Notification System using AWS Lambda, SES & EventBridge

Introduction :

The Automated Email Notification System is a cloud-based solution built using AWS **Lambda**, Amazon **SES** (Simple Email Service), and Amazon **EventBridge** to automatically send scheduled and event-based email notifications without using any servers.

Amazon SES Setup:

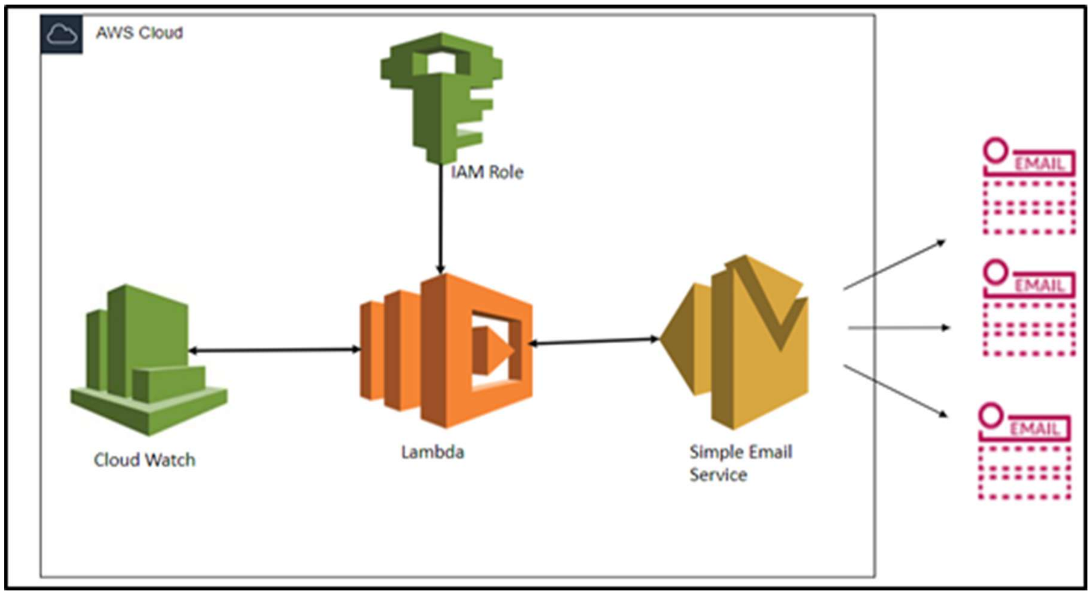
Verify Identities: Verify the email addresses or domains you will use as the sender in SES. This is a crucial step for deliverability and to avoid "Email address not verified" errors.

AWS Lambda :

Create a Lambda Function: Develop a Lambda function (e.g., in Python, Node.js) that will handle the email sending logic.

IAM Role :

Create an **IAM** role for the Lambda function and attach a policy that grants it permissions to interact with SES (e.g., ses:SendEmail, ses:SendRawEmail).

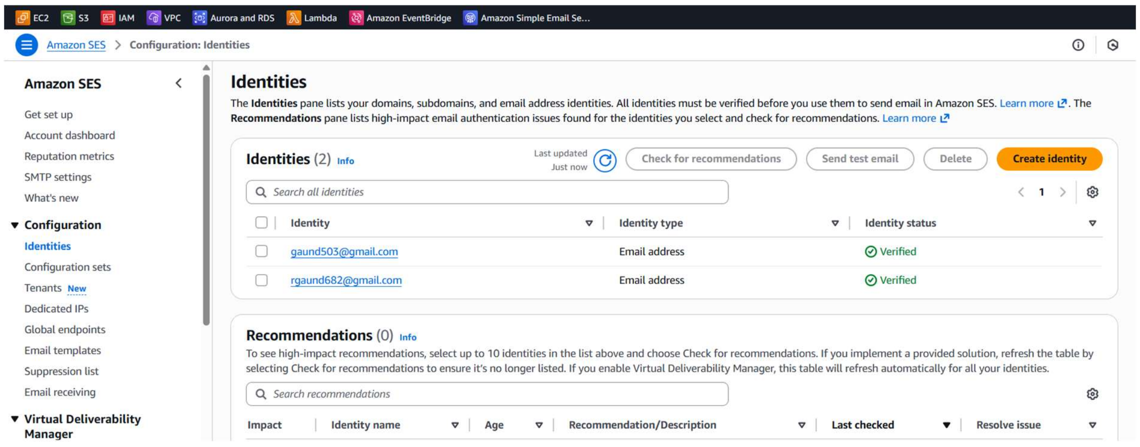


Objective :

- To design a serverless automated email notification system using AWS cloud services.
- To use AWS Lambda for executing email sending logic without maintaining servers.
- To use Amazon SES for sending secure and reliable HTML-formatted emails.
- To configure EventBridge to schedule emails automatically at predefined times.

Step 1 : Create SES (Simple Email Services) Setup

- AWS Console → **SES** → Identities
- Verified **Identities** : Sender & Receiver Email-ID
- Enter Email Address → Create → Open Gmail and Click
- Successful Verify



Identities

The **Identities** pane lists your domains, subdomains, and email address identities. All identities must be verified before you use them to send email in Amazon SES. Learn more. The **Recommendations** pane lists high-impact email authentication issues found for the identities you select and check for recommendations. Learn more

Identity	Identity type	Identity status
gaund503@gmail.com	Email address	✓ Verified
rgaund682@gmail.com	Email address	✓ Verified

Recommendations (0) Info

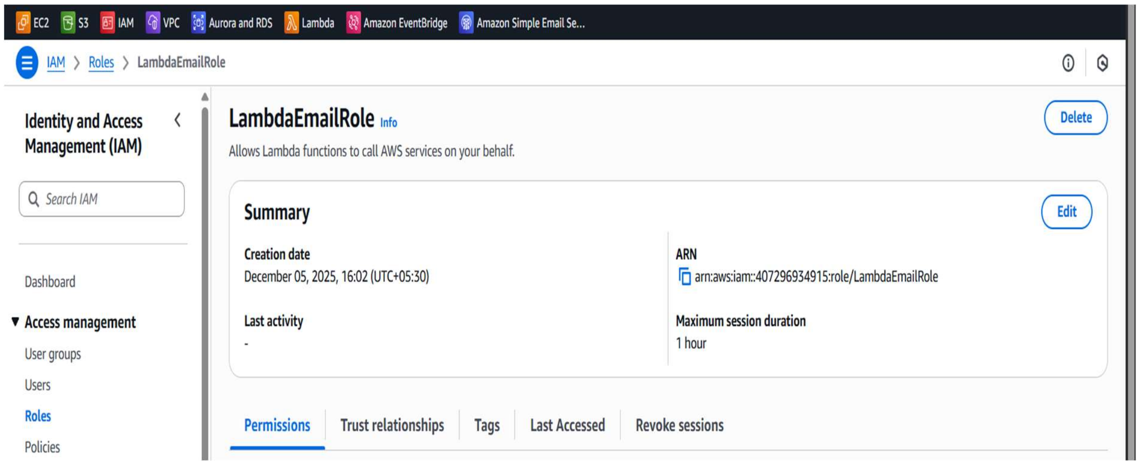
To see high-impact recommendations, select up to 10 identities in the list above and choose Check for recommendations. If you implement a provided solution, refresh the table by selecting Check for recommendations to ensure it's no longer listed. If you enable Virtual Deliverability Manager, this table will refresh automatically for all your identities.

Impact	Identity name	Age	Recommendation/Description	Last checked	Resolve issue

SES enable to send Email

Step 2 : Create IAM role and assign policies

- AWS → IAM → Create role
- Trusted Entity : AWS service
- Use Case : Lambda → Next
- Attach Policies : **AmazonSESFullAccess & CloudWatchFullAccess**
- Role Name : LambdaEmailRole (you can enter own name)
- Create Role



LambdaEmailRole Info

Allows Lambda functions to call AWS services on your behalf.

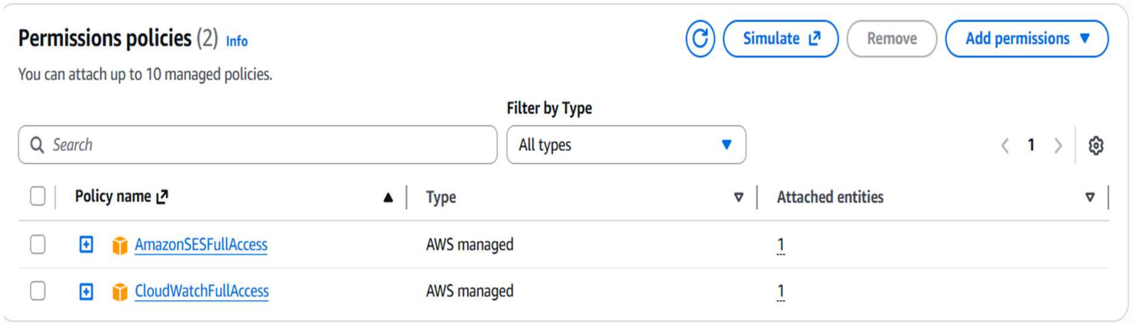
Summary

Creation date: December 05, 2025, 16:02 (UTC+05:30)

ARN: arn:aws:iam::407296934915:role/LambdaEmailRole

Last activity: -

Maximum session duration: 1 hour



Permissions policies (2) Info

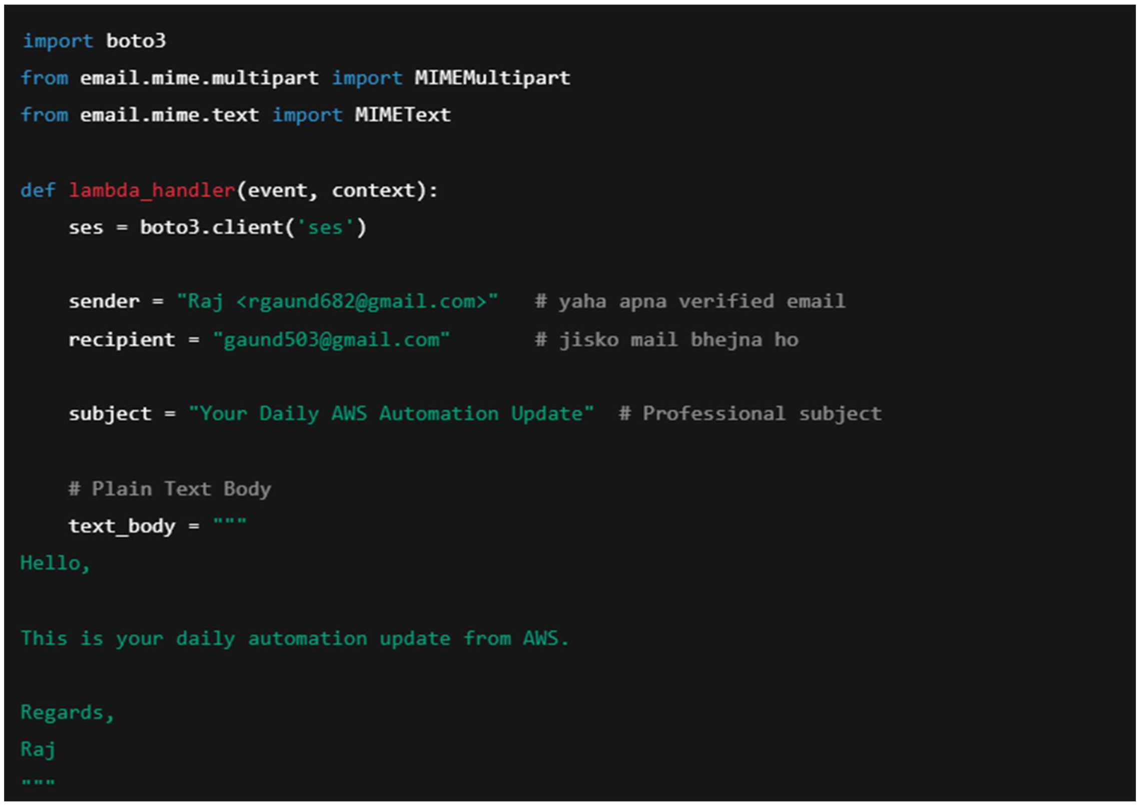
You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonSESFullAccess	AWS managed	1
CloudWatchFullAccess	AWS managed	1

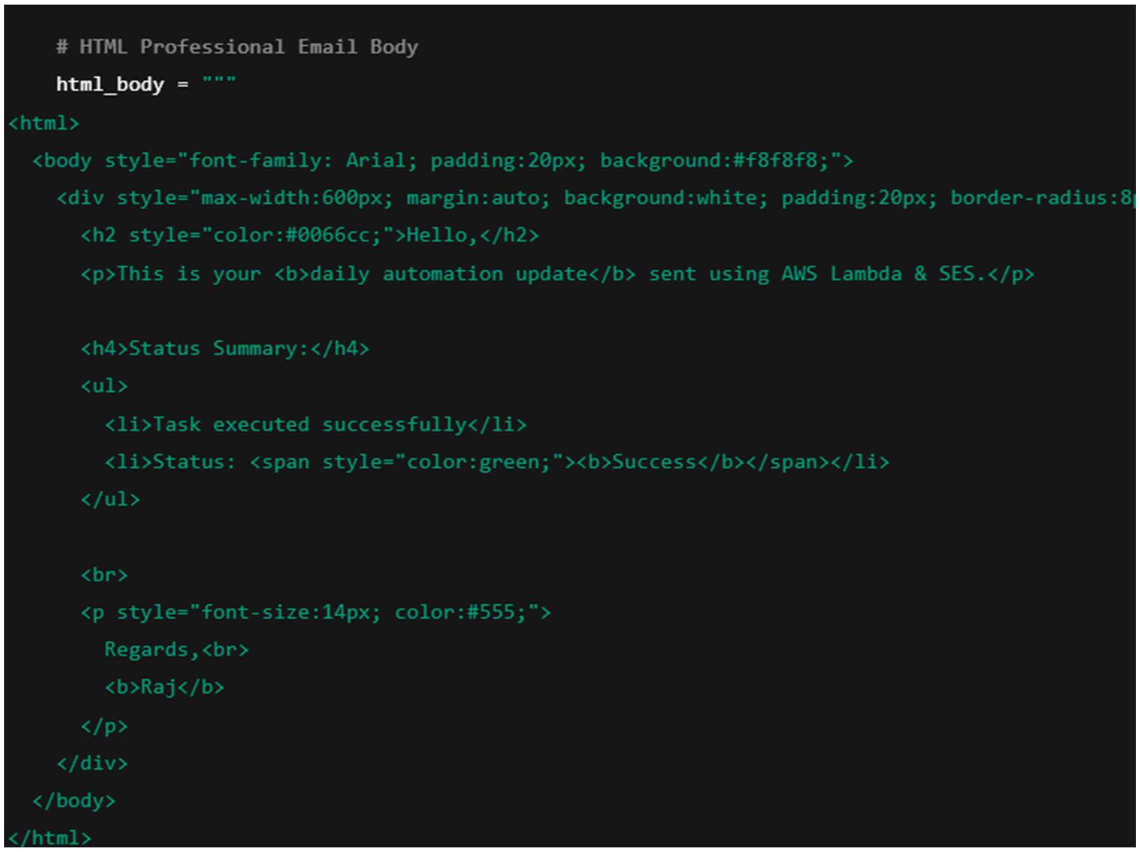
IAM role ready and give permission to Lambda to enable mailing function

Step 3 : Create Lambda Function (Python 3.10)

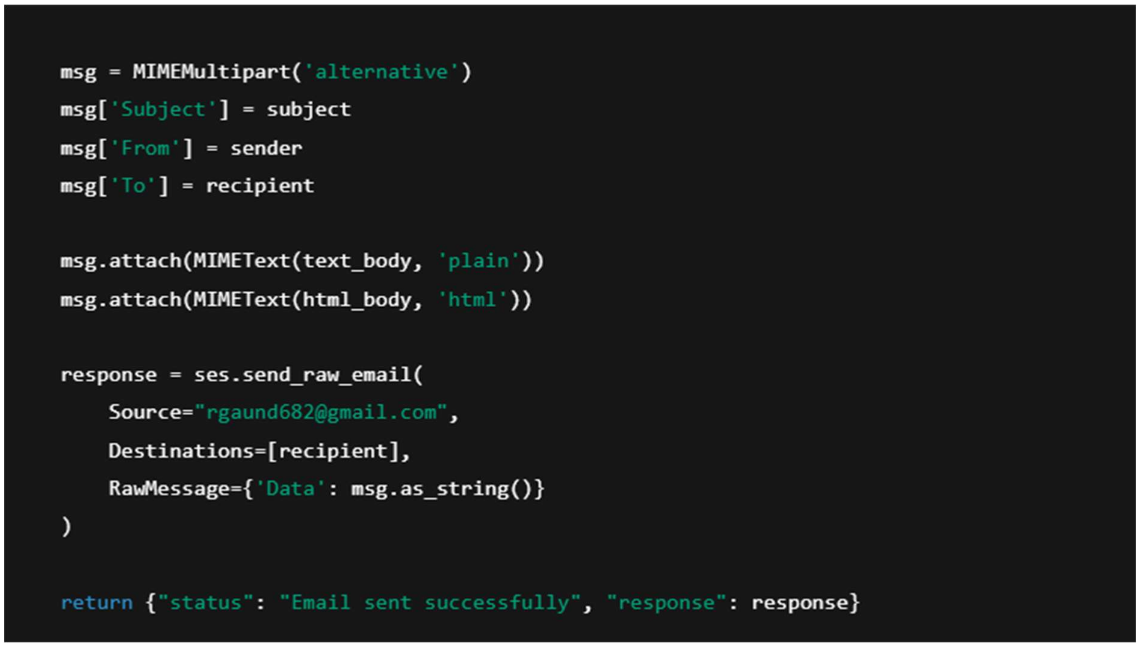
- AWS → Lambda → Author From Scratch
- Name : SendEmailFunction
- Runtime : (**Pyhton 3.10**)
- Change Default → use existing role : **LambdaEmailRole**
- Create Function → Click on lambda Function
- Enter Code → Deploy
- Test → create new → Enter Test Name (**Emailtest**)
- Save and again Test



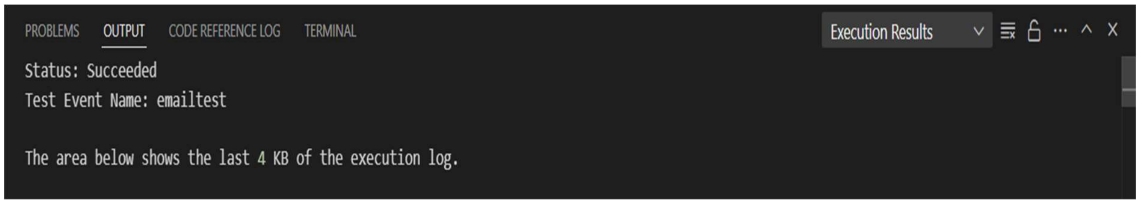
```
import boto3
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

def lambda_handler(event, context):
    ses = boto3.client('ses')

    sender = "Raj <rgaund682@gmail.com>"   # yaha apna verified email
    recipient = "gaund503@gmail.com"       # jisko mail bhejna ho

    subject = "Your Daily AWS Automation Update"  # Professional subject

    # Plain Text Body
    text_body = """
Hello,

This is your daily automation update from AWS.

Regards,
Raj
"""
```



```
    # HTML Professional Email Body
    html_body = """
<html>
  <body style="font-family: Arial; padding:20px; background:#f8f8f8;">
    <div style="max-width:600px; margin:auto; background:white; padding:20px; border-radius:8
      <h2 style="color:#0066cc;">Hello,</h2>
      <p>This is your <b>daily automation update</b> sent using AWS Lambda & SES.</p>

      <h4>Status Summary:</h4>
      <ul>
        <li>Task executed successfully</li>
        <li>Status: <span style="color:green;"><b>Success</b></span></li>
      </ul>

      <br>
      <p style="font-size:14px; color:#555;">
        Regards,<br>
        <b>Raj</b>
      </p>
    </div>
  </body>
</html>
```



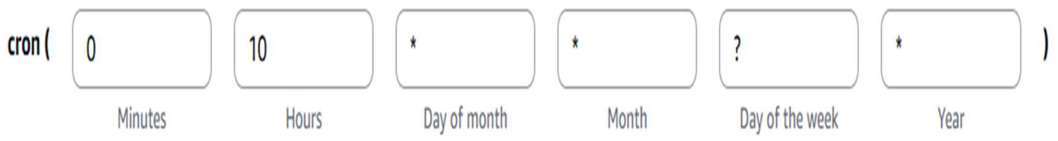
```
    msg = MIMEMultipart('alternative')
    msg['Subject'] = subject
    msg['From'] = sender
    msg['To'] = recipient

    msg.attach(MIMEText(text_body, 'plain'))
    msg.attach(MIMEText(html_body, 'html'))

    response = ses.send_raw_email(
        Source="rgaund682@gmail.com",
        Destinations=[recipient],
        RawMessage={'Data': msg.as_string()}
    )

    return {"status": "Email sent successfully", "response": response}
```



```
Status: Succeeded
Test Event Name: emailtest

The area below shows the last 4 KB of the execution log.
```

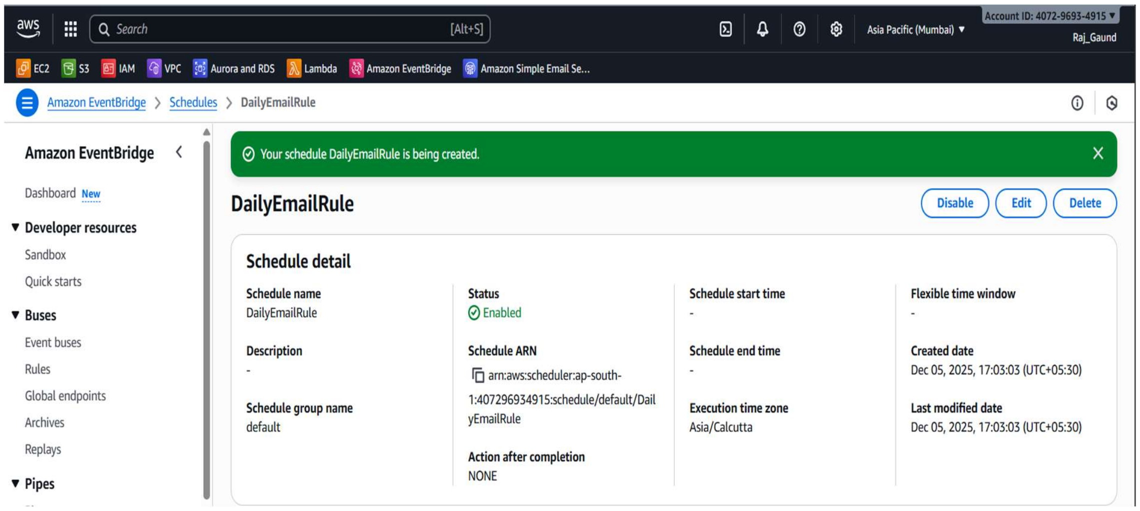
Step 4 : Daily Automatic Email Scheduling Setup using EventBridge

- AWS → EventBridge → EventBridge Schedule
- Create Schedule
- Role Name : Daily email rule (you can use own)
- Schedule Pattern : Recurring Schedule



cron (	0	10	*	*	?	*	)
	Minutes	Hours	Day of month	Month	Day of the week	Year	

- Flexible Time Window : OFF
- Setting Target : AWS Lambda Invoke
- Lambda function : use existing function (SendEmailFunction)
- Create Schedule



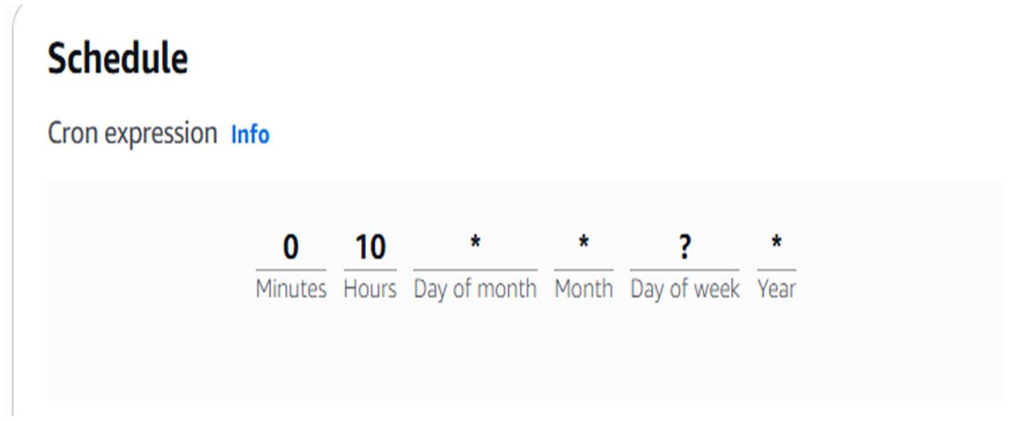
Your schedule DailyEmailRule is being created.

DailyEmailRule

Schedule detail

Schedule name DailyEmailRule	Status Enabled	Schedule start time -	Flexible time window -
Description -	Schedule ARN arn:aws:scheduler:ap-south-1:407296934915:schedule/default/DailyEmailRule	Schedule end time -	Created date Dec 05, 2025, 17:03:03 (UTC+05:30)
Schedule group name default		Execution time zone Asia/Calcutta	Last modified date Dec 05, 2025, 17:03:03 (UTC+05:30)
	Action after completion NONE		

Event Schedule was created



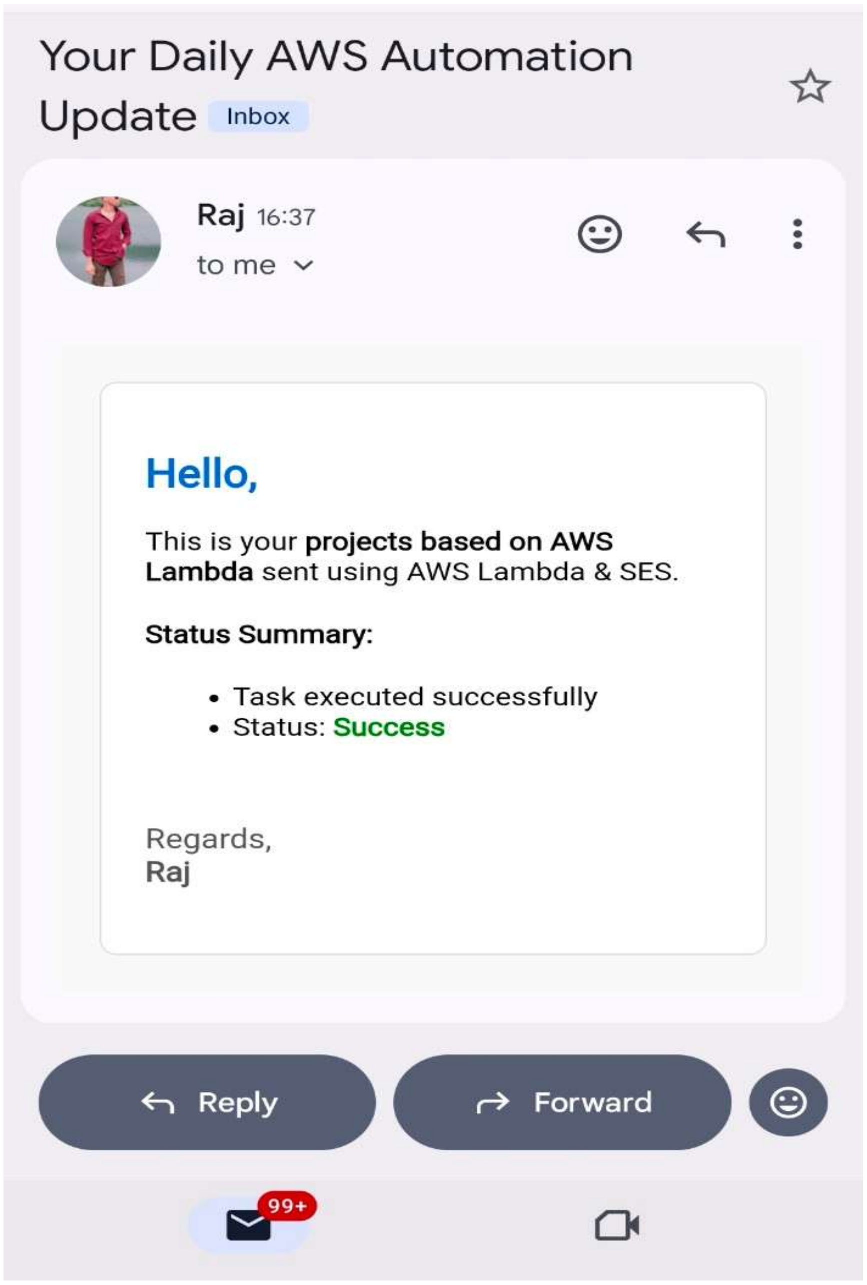
Schedule

Cron expression Info

Minutes	Hours	Day of month	Month	Day of week	Year
0	10	*	*	?	*

Scheduled Time was Defined

Final Result : Automatic Email was send at Scheduled time



Your Daily AWS Automation Update Inbox

Raj 16:37
to me

Hello,

This is your **projects based on AWS Lambda** sent using AWS Lambda & SES.

Status Summary:

- Task executed successfully
- Status: **Success**

Regards,
Raj

PROJECT SUMMARY

This project demonstrates how to automate email sending using Amazon Web Services (AWS). The system uses AWS Lambda to execute email-sending logic, Amazon SES to securely deliver emails, and Amazon EventBridge to schedule automatic triggers. Whenever the EventBridge rule runs at a specified time (e.g., 10:00 AM daily), it invokes the Lambda function, which sends a professionally formatted email to the receiver.

This solution is completely **serverless**, meaning it does not require maintaining servers or infrastructure. It is **scalable, cost-effective, reliable**, and suitable for real-world automation such as OTP delivery, daily reports, reminders, alerts, and notifications.

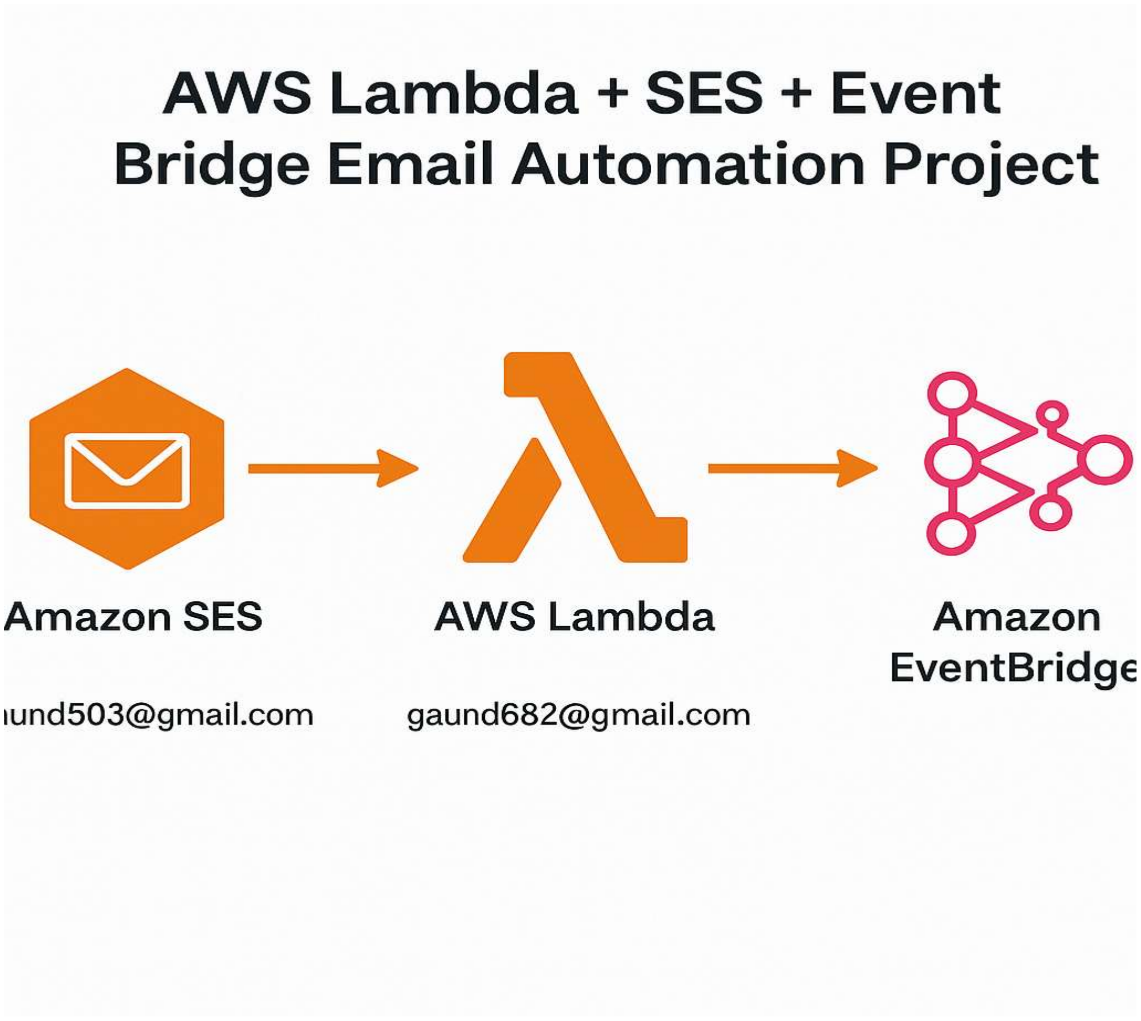
PURPOSE OF THE PROJECT

- To automate the process of sending emails without manual work.
- To create a serverless email automation system using AWS cloud services.
- To use Amazon SES for secure, fast, and professional email delivery.
- To use EventBridge to schedule tasks at fixed times (e.g., daily at 10AM).
- To understand how cloud automation reduces human effort and increases efficiency.

Software & Hardware Requirements

1. Python
2. Aws Account
3. Internet
4. Desktop (eg : 2 Gb laptop)

Architecture Diagram of Project



Conclusion :

This project successfully demonstrates a serverless and automated email notification system using AWS Lambda, Amazon SES, and EventBridge. By integrating these cloud services, the system sends scheduled emails reliably without requiring any manual effort or server management. The solution is cost-efficient, scalable, and suitable for real-world use cases such as reminders, alerts, reports, and OTP delivery.

Project Made By : Raj Gaund