

Experiment : 7

Reg No- 097

Title : Creating a lambda function in AWS to email daily reports

Aim : Automate Sending Emails at a Specific Time with AWS Lambda, CloudWatch and SES

Pre-requisites : AWS Console, Amazon SES, Amazon Lambda, Amazon CloudWatch.

Procedure : We are going automate sending email to a person or a group of people. AWS **Cloudwatch** is used to setup a schedule to trigger AWS **Lambda** function and then its going to use AWS **SES (Simple Email Service)** to send out emails to people.

Steps:

1. Go to AWS SES (Simple email service), click on “Create Identity”. Use email address as a type and type the email address.

The screenshot shows the 'Create identity' page in the AWS SES console. The 'Identity type' section has 'Email address' selected. The 'Email address' field contains 'akil.ukh.786@gmail.com'. A note below says 'Email address can contain up to 320 characters, including plus signs (+), equals signs (=) and underscores (_.)'. There is a checkbox for 'Assign a default configuration set' which is unchecked. The 'Tags - optional' section is present but empty.

2. Verify the email address that reviewed an email from aws to tell you to verify that.

The screenshot shows the 'Verified identities' page in the AWS SES console. It lists two identities: 'akil.ukh.786@gmail.com' and 'aa8345@smriti.edu.in', both of which are marked as 'Verified'. A note at the top explains the new identity status update, stating that explicit verification is now required for domain identities through DNS updates and for email address identities through verification emails. The 'Identities' table shows the identities with their respective details and status.

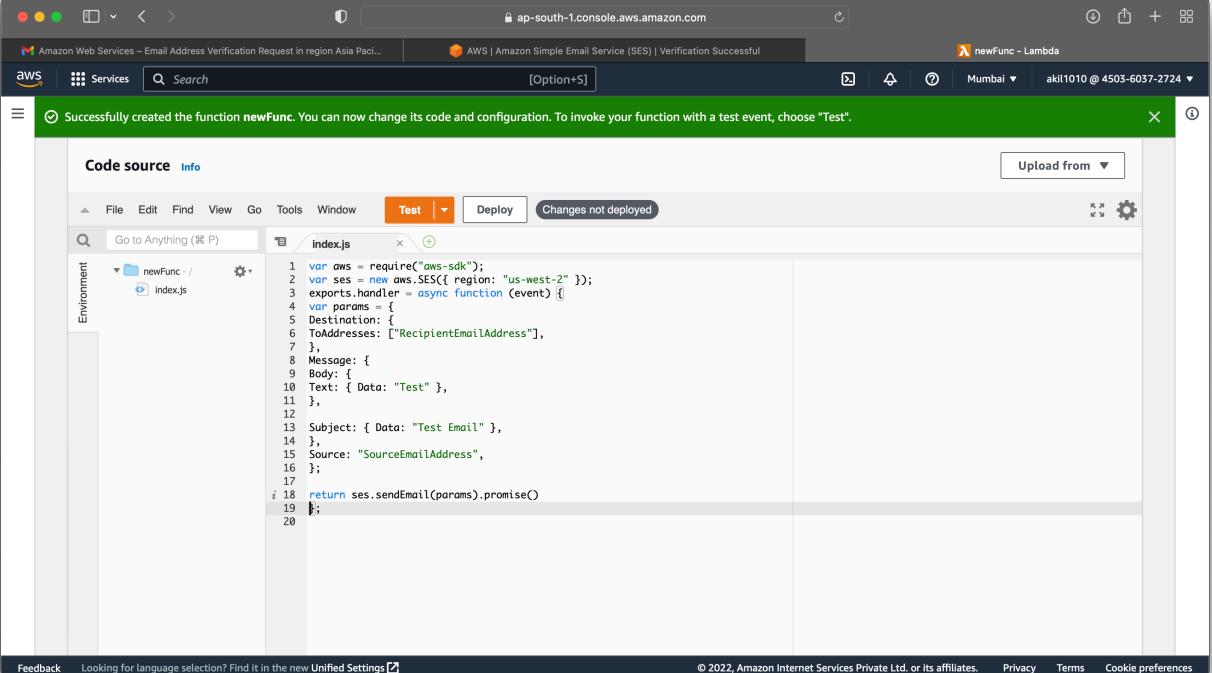
3. Create two identities (email address).
One for sending emails and another for receiving.
4. Create an IAM role.
Give Use case as lambda and give full access to cloudwatch, SES.

The screenshot shows the AWS IAM Management Console interface. The top navigation bar includes 'Services' and 'Search' buttons. The main content area is titled 'Add permissions' and shows the 'Permissions policies' section. A search bar at the top of this section contains the text '*ses*'. Below the search bar is a table with columns for 'Policy name', 'Type', and 'Description'. One row in the table is selected, showing 'AmazonSESFullAccess' under 'Policy name', 'AWS m...' under 'Type', and a detailed description under 'Description'. Other policies listed include 'AmazonSESSendEmail', 'AWSVendorInsightsAssessorFullAccess', 'AwsGlueSessionUserRestrictedNotebookPolicy', 'AmazonSageMakerServiceCatalogProductsFirehoseServ...', and 'AWSOpsWorksRegisterCLI_OnPremises'.

5. Go to Lambda Service, create a lambda function.
Give name, runtime as NodeJS, execution role as created IAM role previously.

The screenshot shows the AWS Lambda service interface. The top navigation bar includes 'Services' and 'Search' buttons. The main content area shows a success message: 'Successfully created the function newFunc. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'. Below this message, the 'newFunc' function is listed in the 'Functions' section. The 'Function overview' tab is selected, showing the function name 'newFunc', a 'Layers' section (0 layers), and a 'Description' field (empty). The 'Last modified' field shows '6 seconds ago'. The 'Function ARN' field contains the ARN: arn:aws:lambda:ap-south-1:450360372724:function:newFunc. The 'Function URL' field is empty. At the bottom of the page, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is currently selected.

6. Use this template for the code:



The screenshot shows the AWS Lambda function editor for a function named 'newFunc'. The code source tab is selected, displaying the 'index.js' file. The code is a template for sending an email using the AWS SES SDK. It includes imports for 'aws-sdk', creates a new SES client for the 'us-west-2' region, defines an asynchronous handler function, and constructs a 'params' object for the 'sendEmail' method. The 'ToAddresses' field is set to a placeholder 'RecipientEmailAddress'. The 'Message' field contains a 'Body' with the text 'Test'. The 'Subject' field has a placeholder 'Data: "Test Email"'. The 'Source' field is set to 'SourceEmailAddress'. The code concludes with a call to 'ses.sendEmail(params).promise()'. A success message at the top of the editor states: 'Successfully created the function newFunc. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'

```
var aws = require("aws-sdk");
var ses = new aws.SES({ region: "us-west-2" });
exports.handler = async function (event) {
  var params = {
    Destination: {
      ToAddresses: ["RecipientEmailAddress"],
    },
    Message: {
      Body: {
        Text: { Data: "Test" },
      },
      Subject: { Data: "Test Email" },
    },
    Source: "SourceEmailAddress",
  };

  return ses.sendEmail(params).promise()
};
```

```
var aws = require("aws-sdk");
var ses = new aws.SES({ region: "us-west-2" });
exports.handler = async function (event) {
  var params = {
    Destination: {
      ToAddresses: ["RecipientEmailAddress"],
    },
    Message: {
      Body: {
        Text: { Data: "Test" },
      },
      Subject: { Data: "Test Email" },
    },
    Source: "SourceEmailAddress",
  };

  return ses.sendEmail(params).promise()
};
```

The screenshot shows the AWS Lambda function configuration page for 'newFunc'. A green success message at the top states: 'Successfully updated the function newFunc.' Below this, the 'Code source' tab is selected, showing the 'index.js' file content:

```
function handler(event) {
  const response = {
    "ResponseMetadata": {
      "RequestId": "db024844-72c9-4b76-92df-7c650d1825ba"
    },
    "MessageId": "0109018470c5a3b2-5fb47f32-zead-4e27-b17c-9247282221da-000000"
  };
  return response;
}
```

The 'Execution result' section shows a successful execution with the following details:

- Status: Succeeded
- Max memory used: 81 MB
- Time: 891.99 ms

Below the code editor, there are tabs for 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test', and 'Deploy'. The 'Test' tab is currently active.

7. Click on Deploy and then TEST, you wil receive the message in your mentioned emails.

The screenshot shows the AWS Lambda function configuration page for 'newFunc'. A green success message at the top states: 'Successfully created the function newFunc. You can now change its code and configuration. To invoke your function with a test event, choose "Test".' Below this, the 'Function overview' section is visible, showing the function name 'newFunc' and a 'Layers' section with '(0)'.

The right side of the screen displays the function's details:

- Description: -
- Last modified: 6 seconds ago
- Function ARN: arn:aws:lambda:ap-south-1:450360372724:function:newFunc
- Function URL: [Info](#)

At the bottom, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is selected.

8. For scheduled daily report, go to AWS Cloudwatch , navigate to rule section (now called as eventBridge).
9. Create rule- give name, ruletype- schedule, use cron expression for schedule pattern .
For e.g. : 15 19 * * ? *

Cron expression
15 19 * * ? *

Next 10 trigger date(s) UTC

Target Name	Type	Arn	Input	Role
newFunc	Lambda function	arn:aws:lambda:ap-south-1:450360372724:function:newFunc	Matched event	-

10. Select Targets as lambda function, and use the above defined function.

Permissions
Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure the proper permissions.

Target 1

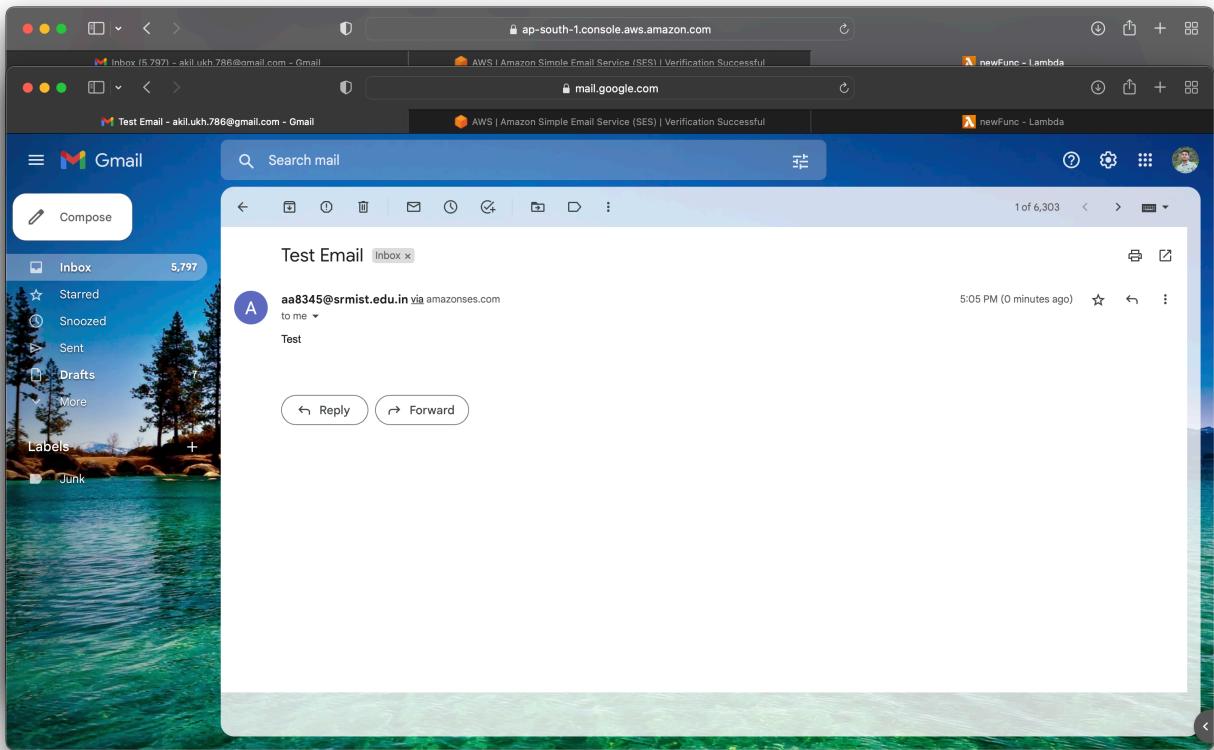
Target types
Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.
 EventBridge event bus
 EventBridge API destination
 AWS service

Select a target Info
Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule)

Lambda function
newFunc

Add another target Cancel Previous Next

11.Go to monitoring in Lambda service, click on View logs in cloudWatch and check your mail inbox .



Result:

Hence, the lambda function is created and also implemented using SES, CloudWatch to schedule daily reports.