AWS Lambda Email Sender Documentation

Overview

This AWS Lambda function is designed to send emails with attachments using the AWS Simple Email Service (SES) and the Nodemailer library. The function receives input parameters such as email addresses, subject, and attachments from an HTTP POST request.

Code Structure

The code is divided into several sections:

Dependencies:

- *AWS SDK:* Used for configuring AWS credentials and SES.
- *Nodemailer*: Handles email composition and sending.
- *fs.promises:* Provides file system functions using promises.
- *mime-types:* Helps determine the MIME type of a file based on its extension.

Main Function:

SendEmail: The main function that is triggered when the Lambda is invoked. It parses input from the HTTP request body, configures AWS and SES, checks attachment paths, and sends the email.

```
const AWS = require('aws-sdk');
const nodemailer = require('nodemailer');
const fs = require('fs').promises;
const mime = require('mime-types');

const SendEmail = async (event) => {

  //taking whats coming in body
  const { addresses, subject, attachments } = JSON.parse(event.body);

  //Configuring AWS
  AWS.config.update({
    accessKeyId: 'Your Secret Key',
    secretAccessKey: 'Your Secret Key',
    region: 'us-east-1',
  });

  //Using AWS SES (simple mail service) for sending the email
  const transporter = nodemailer.createTransport({
    SES: new AWS.SES({ apiVersion: '2010-12-01' }),
  });
```

```
  const attachmentPromises = attachments.map(async (attachment) => {
    // Check if the file exists before attempting to read it
    const exists = await fileExists(attachment.path);

    if (!exists) {
      console.error(`File not found: ${attachment.path}`);
      throw new Error(`File not found: ${attachment.path}`);
    }


    const content = await readFile(attachment.path);
             const    contentType    =    mime.lookup(attachment.path)    ||
'application/octet-stream';

    return {
      filename: attachment.filename || 'attachment.txt',
      content: content.toString('base64'),
      encoding: 'base64',
      contentType: contentType,
    };
  });

  const attachmentContents = await Promise.all(attachmentPromises);

  const mailOptions = {
    from: 'rameeshakhan157@gmail.com',
    to: addresses.join(', '),
    subject: subject,
    text: 'Your email body text goes here.',
    attachments: attachmentContents,
  };

  try {
    await transporter.sendMail(mailOptions);

    return {
      statusCode: 200,
      body: JSON.stringify({
        message: 'Email sent successfully!',
        input: event,
      }),
    };
  } catch (error) {
```

```
    console.error('Error sending email:', error);
    return {
      statusCode: 500,
      body: JSON.stringify({
        message: 'Error sending email',
        error: error.message || error,
      }),
    };
  }
};
```

Helper Functions:

fileExists: A helper function to check if a file exists at a given path.

```
  //check if the attachment paths is correct
  const fileExists = async (path) => {
    try {
      await fs.access(path);
      return true;
    } catch (error) {
      return false;
    }
  };
```

readFile: A helper function to read the contents of a file.

```
const readFile = async (path) => {
  try {
    return await fs.readFile(path);
  } catch (error) {
    console.error('Error reading file:', error);
    throw error;
  }
};
```

Usage

Configuration:

Set your AWS credentials and SES region in the AWS SDK configuration.

Replace the sender's email address in the form field of mailOptions with your own email address.

Step 1: Install AWS CLI

- Download AWS CLI:

Visit the official AWS CLI download page and download the installer suitable for your operating system (Windows, macOS, or Linux).

-Install AWS CLI:

Follow the installation instructions for your operating system.

Step 2: Obtain AWS Access Key ID and Secret Access Key

- Sign in to AWS Console:
- Log in to the AWS Management Console using your AWS account credentials.
- Access Security Credentials:
  Navigate to the "Services" dropdown in the top left corner.
  Under the "Security, Identity, & Compliance" section, choose "IAM" (Identity and Access Management).
- Access Keys:
  In the IAM dashboard, select "Users" from the left navigation pane.
- Choose your IAM user.
  In the "Security credentials" tab, find the "Access keys" section.
- Create Access Key:
  If you don't have an access key, create one by selecting "Create access key."
- Note down the "Access key ID" and "Secret access key" that are generated.

Step 3: Configure AWS CLI

- Open Command Prompt or Terminal:
- Open your command-line interface.
- Run Configuration Command:
- Execute the configuration command by typing: aws configure
- Enter your "Access key ID" and "Secret access key" when prompted.
- Set the default region name (e.g., us-east-1).
- Set the default output format (e.g., json).

Step 4: Test Configuration

- Run AWS CLI Command:
  Verify your configuration by running a simple AWS CLI command. For example:
  “aws s3 ls”
  If you see a list of your S3 buckets, your configuration is successful.

—----------

Input Parameters:

The Lambda function expects the input parameters in the HTTP POST request body in JSON format.

Example:

json
Copy code

```
{
    "addresses": ["recipient@example.com"],
    "subject": "Testing Purpose Email",
    "attachments": [
        {
            "path": "path/to/attachment.txt",
            "filename": "custom_filename.txt"
        },
        {
            "path": "path/to/image.png",
            "filename": "image.png"
        }
    ]
}
```

Testing with Postman:

- Open Postman.
- Create a new request or open an existing one.
- Set the request type to POST.
- Enter the URL of your AWS Lambda function endpoint.
- Go to the Body tab, select raw, and set the data type to JSON (application/json).
- Enter your JSON payload in the request body.
- Click the Send button to execute the request.

Error Handling

The function logs errors to the AWS CloudWatch logs and returns an HTTP 500 response with error details if an issue occurs during email sending. For Checking the log at cmd write this command

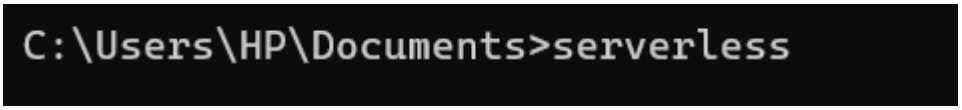
aws logs tail /aws/lambda/<Function-Name>

Deployment

Deploy the Lambda function to your AWS account, and configure the API Gateway or other triggers as needed.

Steps:

1. Start serverless feature in you system by typing the command "**`serverless`**"



2. Select Api Gateway Project to download and setup the starter template

```
C:\Users\HP\Documents>serverless

Creating a new serverless project

? What do you want to make?
  AWS - Node.js - Starter
> AWS - Node.js - HTTP API
  AWS - Node.js - Scheduled Task
  AWS - Node.js - SQS Worker
  AWS - Node.js - Express API
  AWS - Node.js - Express API with DynamoDB
  AWS - Python - Starter
  AWS - Python - HTTP API
  AWS - Python - Scheduled Task
  AWS - Python - SQS Worker
  AWS - Python - Flask API
  AWS - Python - Flask API with DynamoDB
  Other
```

3. Type the project name

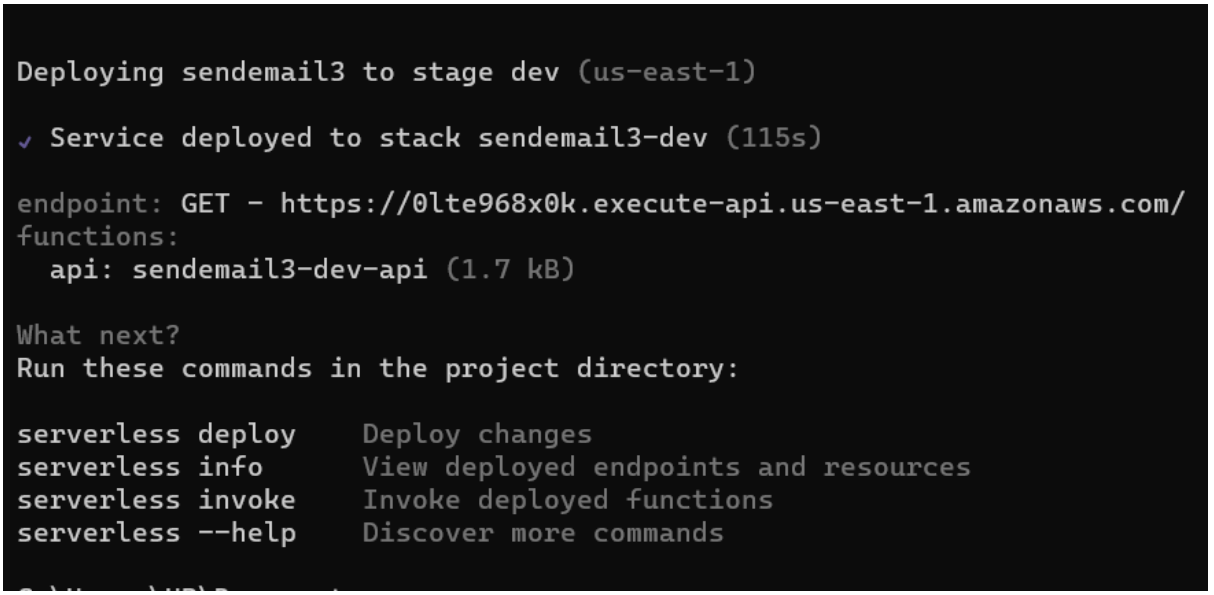
```
C:\Users\HP\Documents>serverless

Creating a new serverless project

? What do you want to make? AWS - Node.js - HTTP API
? What do you want to call this project? SendEmail
```

4. Select the appropriate option

```
? Register or Login to Serverless Framework (Y/n)
```

```
? Do you want to deploy now? (Y/n)
```

5. And Voila , your project would be deployed.



Conclusion

This Lambda function provides a simple and scalable solution for sending emails with attachments through AWS SES. Ensure to configure it appropriately and test thoroughly before deploying it in a production environment.