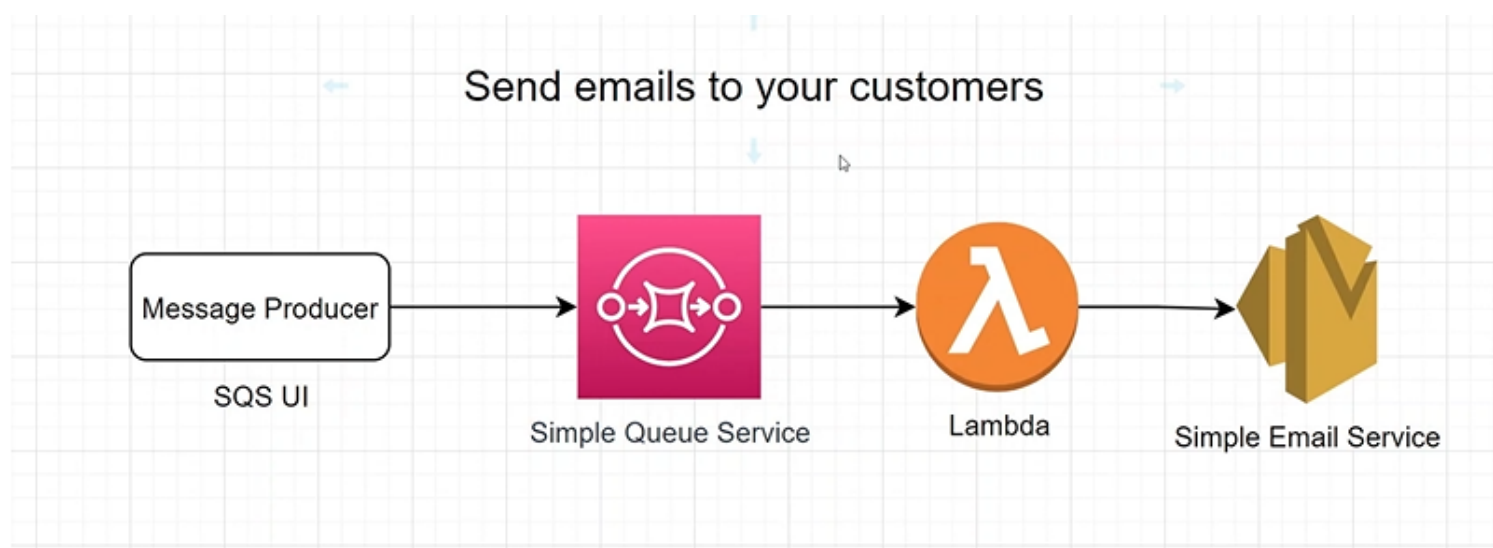


EMAIL setup with SQS, SES, & Lambda

Hi cloud people, I am **Pawan Gambhir** and I'm an AWS Cloud practitioner. In this blog, I'll be walking you through an **email setup** that I recently architected in the AWS cloud.



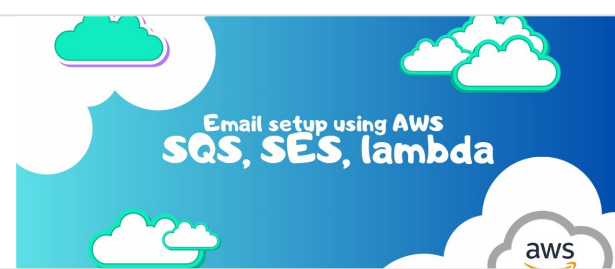
Project Architecture.

Video Guide:

Setup an emailing system using AWS SQS, SES and Lambda!

In this video I've architected an AWS emailing setup as a project by using services like SQS, SES, lambda and IAM roles.

<https://youtu.be/jMSBTydSBKk>



In this project you'll get your hands-on to SQS, SES, Lambda and IAM (roles) services, as the architecture is displayed above. So let's dive into it. 😊

SQS Configuration

1. Select SQS service from AWS dashboard.

Amazon SQS > Queues > Create queue

Create queue

Details

Type
Choose the queue type for your application or cloud infrastructure.

☒ **Standard Info**
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

☐ **FIFO Info**
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

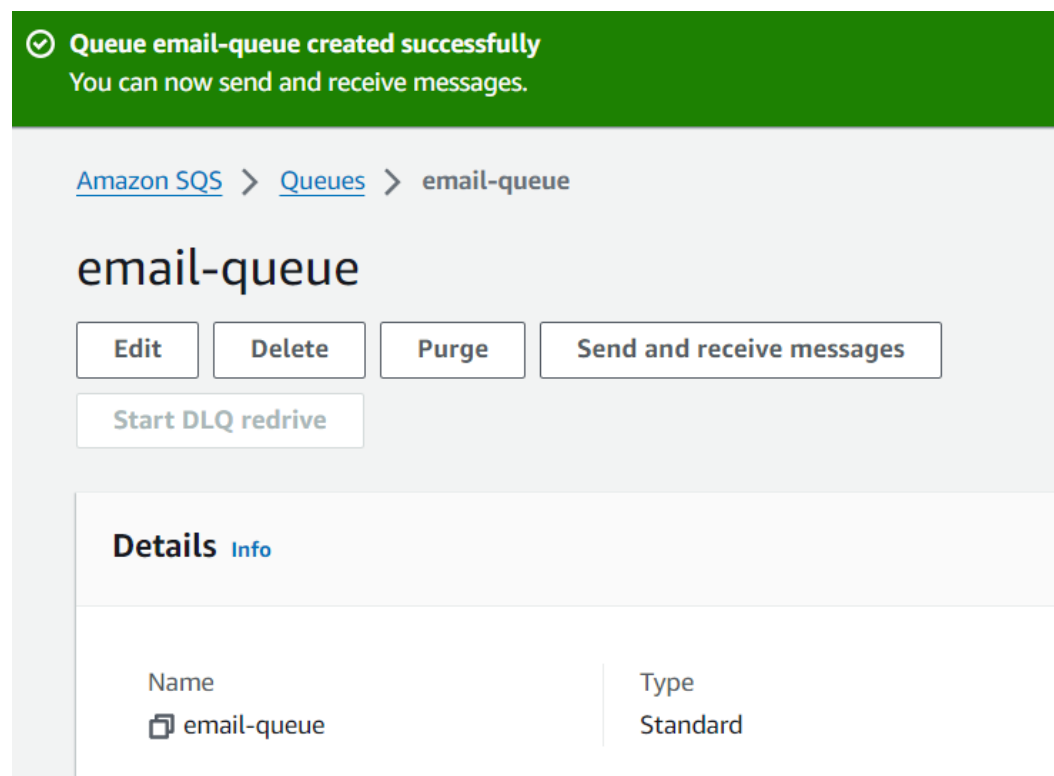
ⓘ You can't change the queue type after you create a queue.

Name

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

2. Create a Queue and select Standard Type.

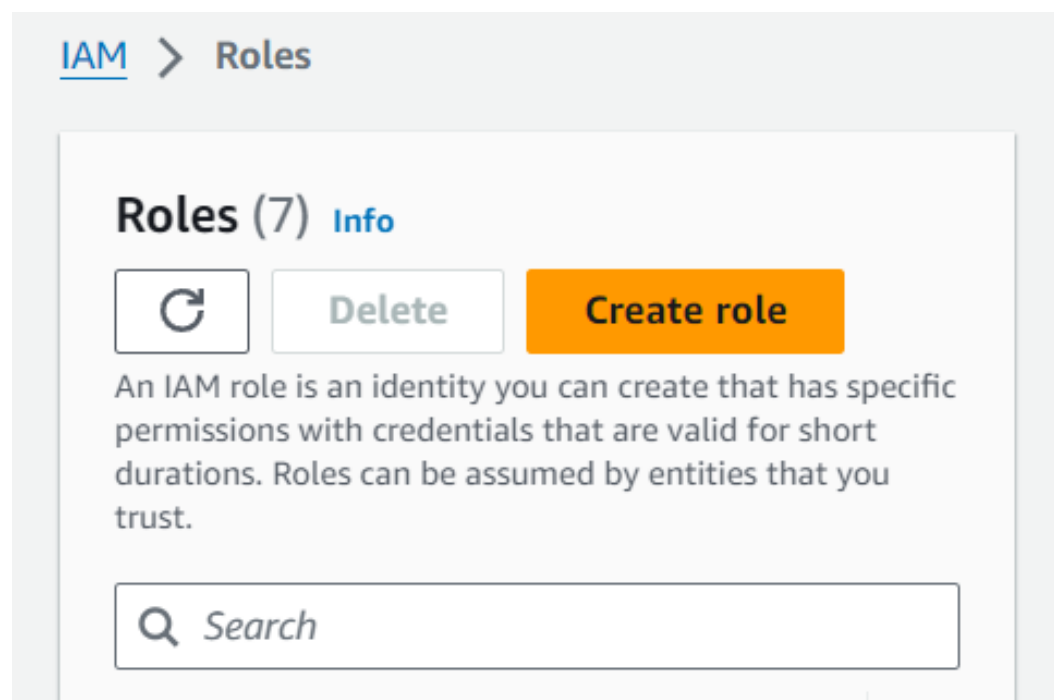
3. Write any name for your queue service. (you can choose any non-existing name)
4. Now, click on 'create' and your queue will be formed. Que



Queue setup completed.

IAM role Configuration

1. Select IAM service from the AWS dashboard.
2. On the left panel, click on roles.
3. Create a new role



4. Select default settings, and in select "Lambda" as service.

Select trusted entity Info

Trusted entity type

☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Lambda ▼

Choose a use case for the specified service.

Use case

☒ Lambda

5. Click next, and choose **“AWSLambdaSQSQueueExecutionRole”** & **“AmazonSESFullAccess”** permissions to give access of the SQS, SES services.

Permissions policies (1/910) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type

All ty...

3 matches

< 1 >

	Policy name	
<input type="checkbox"/>	AmazonSQSFullAccess	
<input type="checkbox"/>	AmazonSQSReadOnlyAccess	
<input checked="" type="checkbox"/>	AWSLambdaSQSQueueExecutionRole	

Permissions policies (2/910) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type

All types

13 matches

< 1 >

⚙

<input type="checkbox"/>	Policy name ↗	Type
<input type="checkbox"/>	AmazonCognitoUnAuthedIdentitiesSes...	AWS managed
<input type="checkbox"/>	AmazonSageMakerServiceCatalogProd...	AWS managed
<input checked="" type="checkbox"/>	AmazonSESEFullAccess	AWS managed
<input type="checkbox"/>	AmazonSESReadOnlyAccess	AWS managed
<input type="checkbox"/>	AwsGlueSessionUserRestrictedNotebo...	AWS managed

6. Finally, write the name for the role you want to give it and click create.

Role details

Role name
Enter a meaningful name to identify this role.

email-setup-project

Maximum 64 characters. Use alphanumeric and '+=, @-_' characters.

Lambda Configuration

1. Select Author from scratch as we'll be putting our own code for this project.
2. Give the Lambda function a name of your choice (try naming services with meaningful names).
3. We'll be using a python code for this project. So, select **Python 3.10** as runtime to get no errors.

Choose one of the following options to create your function.

☒ **Author from scratch**
Start with a simple Hello World example.

☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

email-setup

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.10

4. Click on “execution role” drop menu and select “Use an existing role” as we have created an IAM role just for this function.
5. In the drop down menu of “Existing role” select the IAM role that you created before.
6. Click on create and your Lambda function will be created.

▼ **Change default execution role**

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

email-setup-project

7. Scroll down on the Lambda service page, and paste the Python code in the Code Source.
8. Click on **Deploy** to save your code.

```
import json
import boto3
import os
from botocore.exceptions import ClientError
```

```

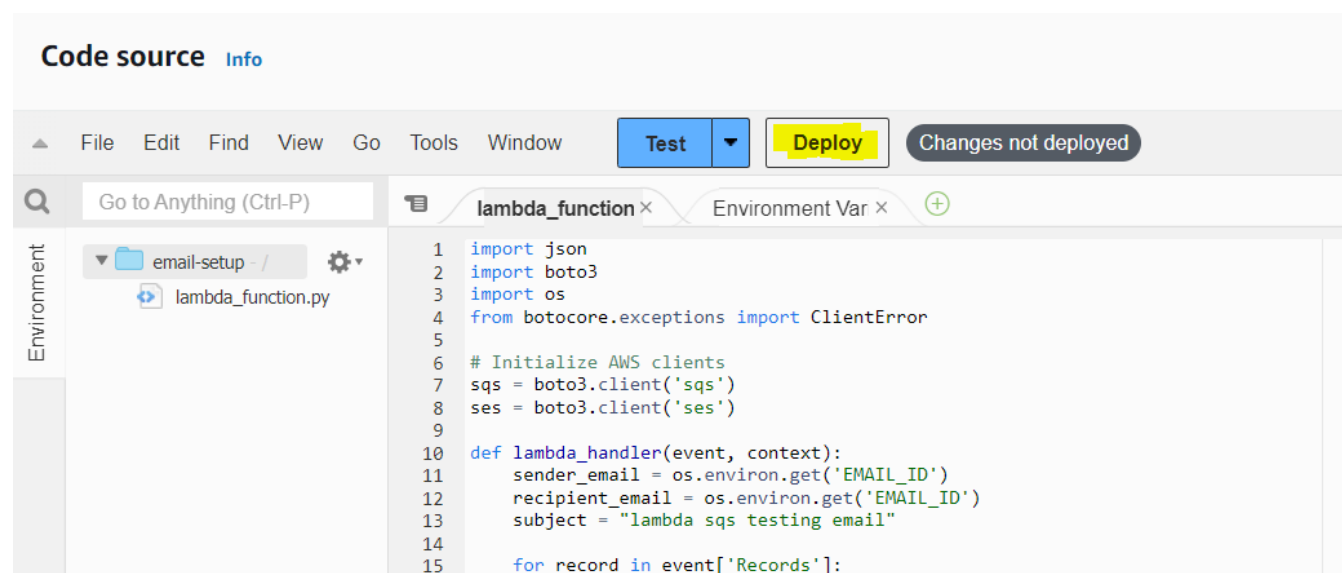
# Initialize AWS clients
sqs = boto3.client('sqs')
ses = boto3.client('ses')

def lambda_handler(event, context):
    sender_email = os.environ.get('EMAIL_ID')
    recipient_email = os.environ.get('EMAIL_ID')
    subject = "lambda sqs testing email"

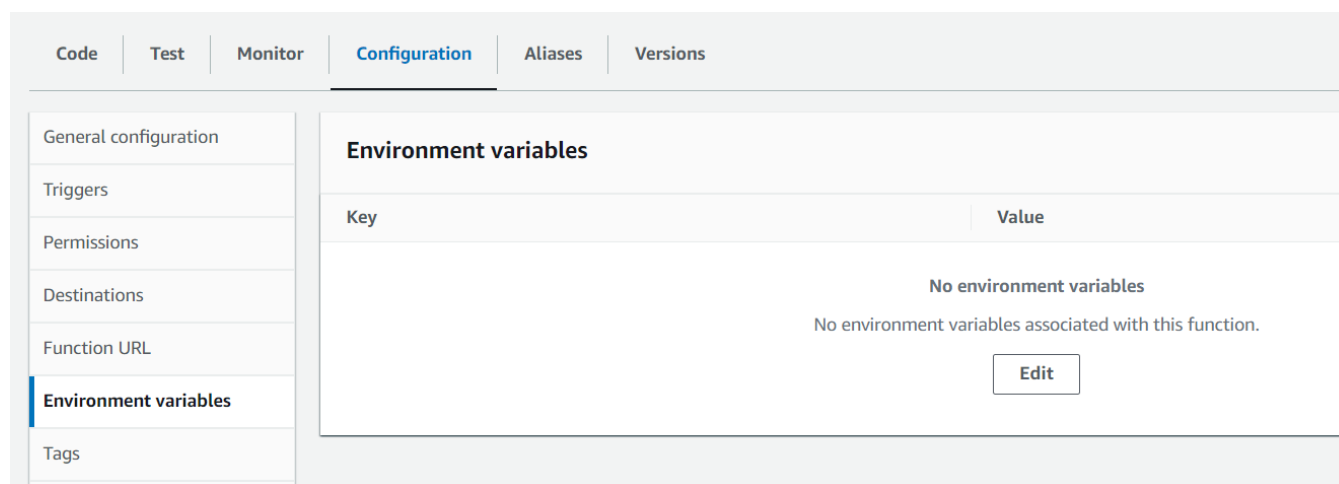
    for record in event['Records']:
        print("test")
        email_body = record["body"]
        print(str(email_body))
        send_email(sender_email, recipient_email, subject, email_body)

def send_email(sender_email, recipient_email, subject, body):
    try:
        response = ses.send_email(
            Source=sender_email,
            Destination={
                'ToAddresses': [recipient_email]
            },
            Message={
                'Subject': {
                    'Data': subject
                },
                'Body': {
                    'Text': {
                        'Data': body
                    }
                }
            }
        )
        print(f"Email sent! Message ID: {response['MessageId']}")
    except ClientError as e:
        print(f"Error sending email: {e.response['Error']['Message']}")

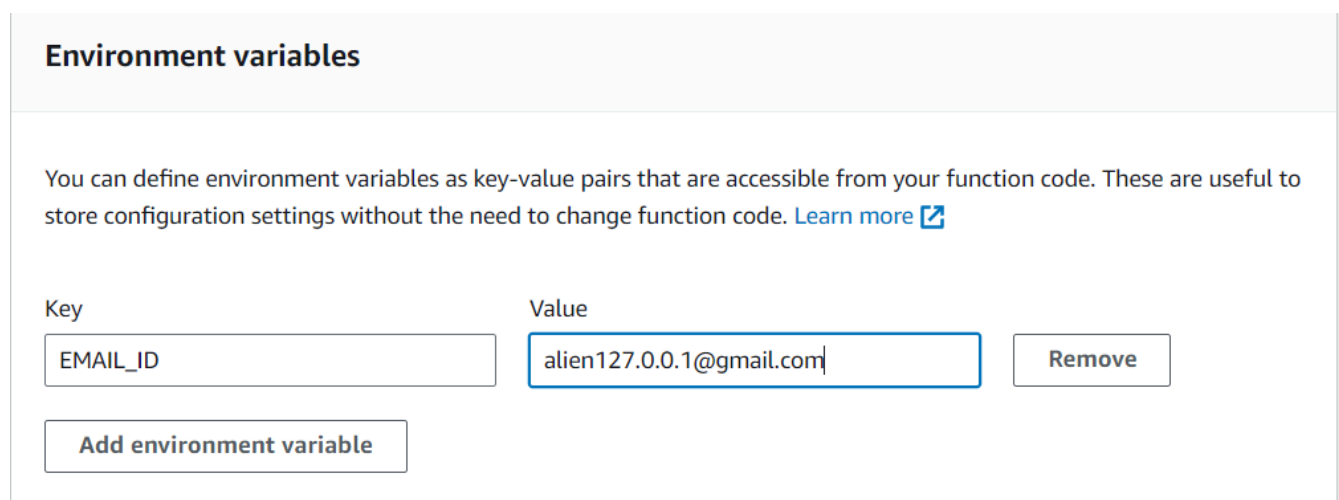
```



9. Now, we have to set the environment variable for the “**EMAIL_ID**” parameter.
10. On top of Code Source, click on **Configuration** and then select **Environment variables** from the left panel.

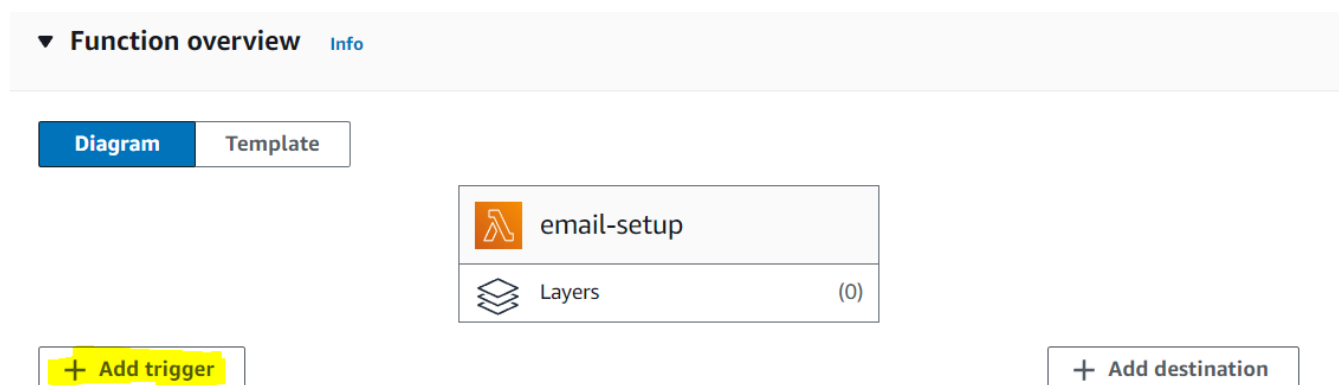


11. Select Edit and write “EMAIL_ID” as **Key** and **your email id** as “**VALUE**” and click create.

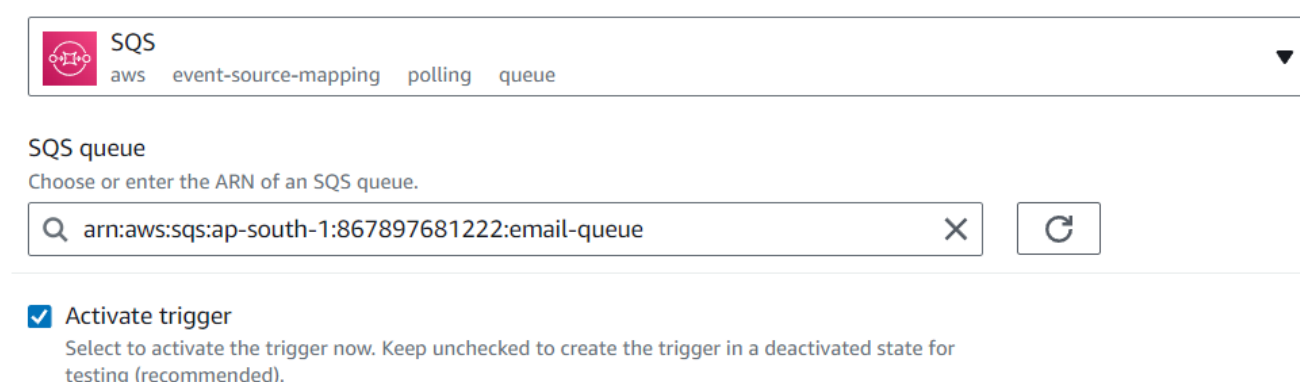


12. Now we are all done with setting up the Lambda function.

13. Click on **Add Trigger** to add the SQS service as a trigger to send emails.

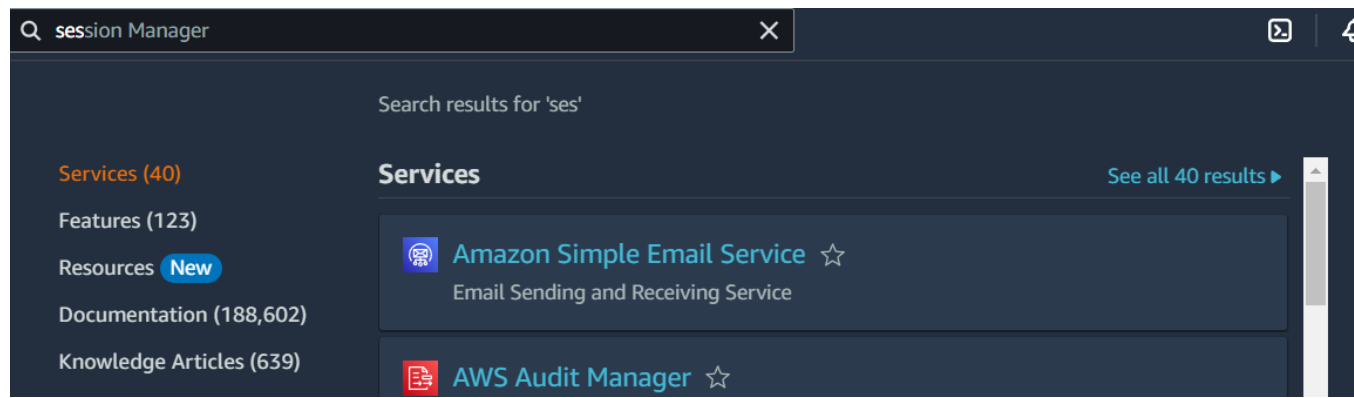


14. In the Trigger menu **search for SQS** and select it. In SQS Queue drop down menu, select the **Queue you created** before.

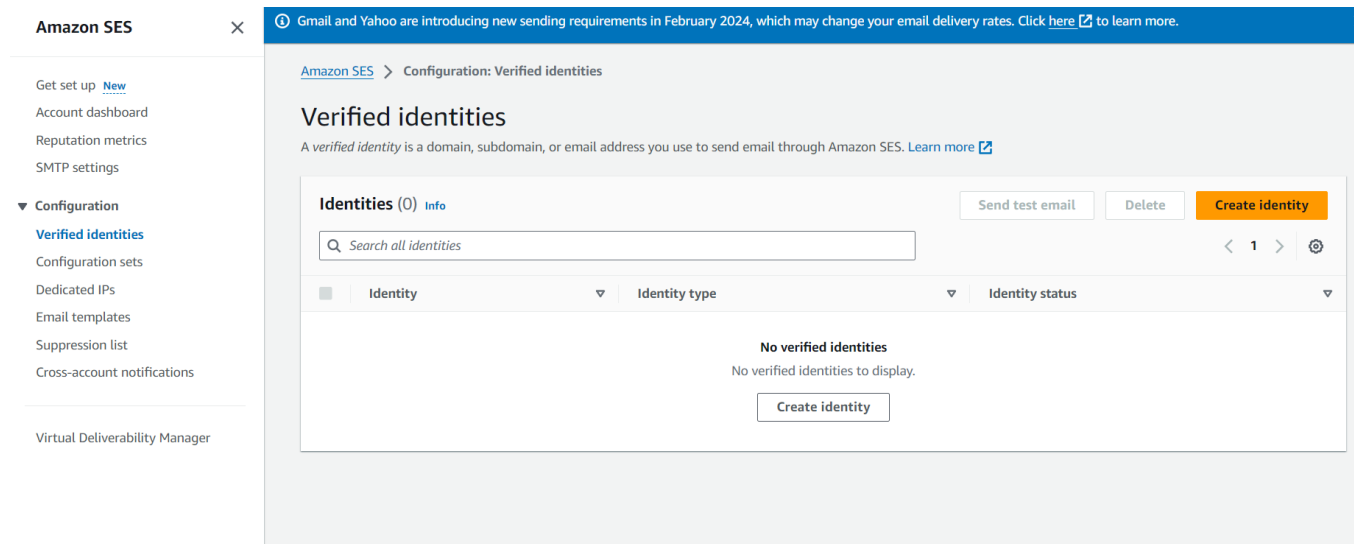


SES Configuration

1. Select Amazon Simple Email Service from AWS dashboard.



2. Now from the left panel, select **Verified identities** to add verify your email for the service.



3. For Identity type we'll choose Email address as we don't have an active domain right now. But if you are using it for your business or you've a domain then you can select Domain.

4. Write your email address in the Email field.

NOTE: THIS EMAIL ID SHOULD BE SAME AS THE ENV VARIABLE THAT YOU SET BEFORE.

5. Click on Create Identity, and you will be be prompted to a verification page.

Identity details
[Info](#)

Identity type

☐ Domain
To verify ownership of a domain, you must have access to its DNS settings to add the necessary records.

☒ Email address
To verify ownership of an email address, you must have access to its inbox to open the verification email.

Email address

Email address can contain up to 320 characters, including plus signs (+), equals signs (=) and underscores (_).

☐ Assign a default configuration set
Enabling this option ensures that the assigned configuration set is applied to messages sent from this identity by default whenever a configuration set isn't specified at the time of sending.

Tags - optional
[Info](#)

You can add one or more tags to help manage and organize your resources, including identities.

No tags associated with the resource.

Add new tag

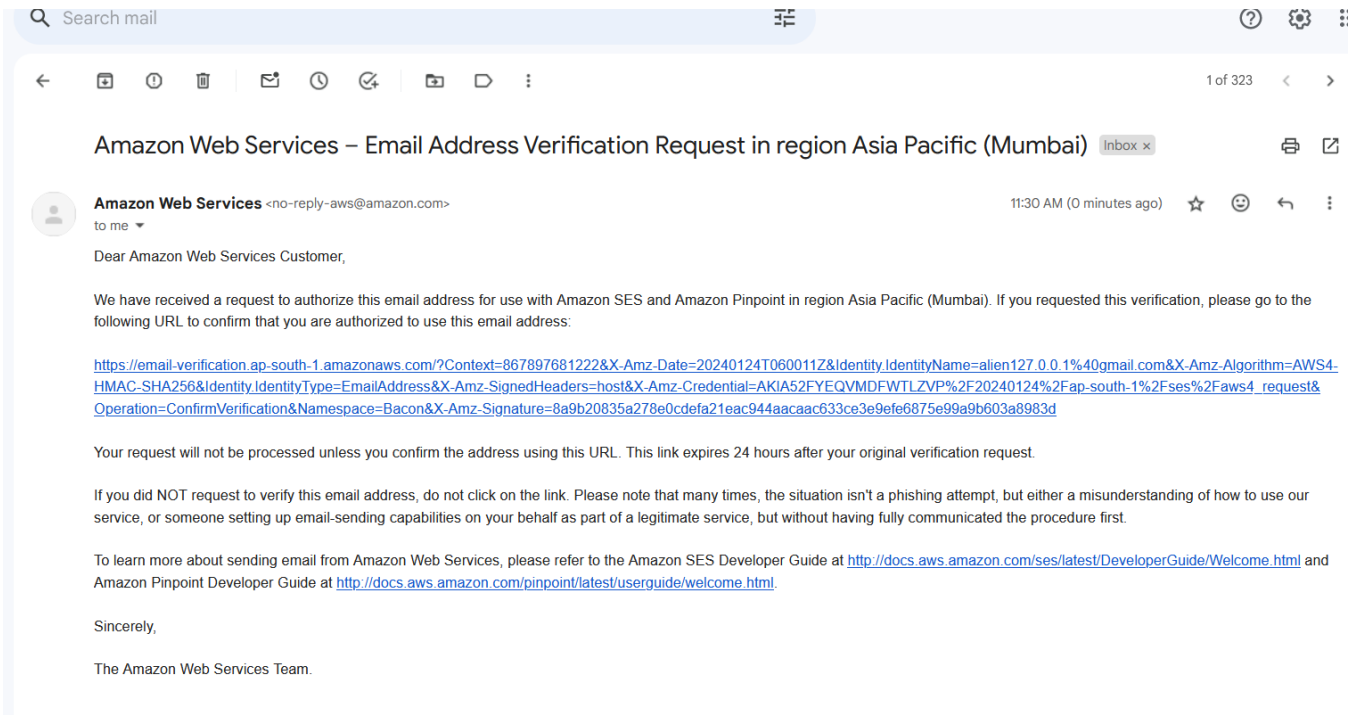
You can add 50 more tags.

Cancel
 Create identity

6. You'll receive a verification email on the Email ID you've put.

NOTE: CHECK YOUR SPAM FOLDER IF YOU DON'T SEE IT IN THE INBOX.

7. Click on the link for the verification and close the tab after you see the Congratulations.



Congratulations!

You have successfully verified an email address. You can now start sending email from this address.

8. Go back to the verification page, refresh it and you'll see Identity status as verified now.

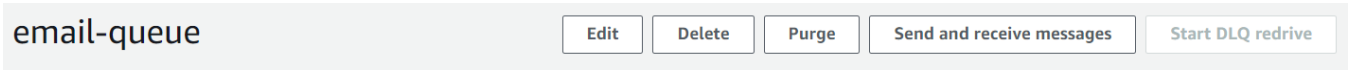
Summary for alien127.0.0.1@q

Identity status

✔ Verified

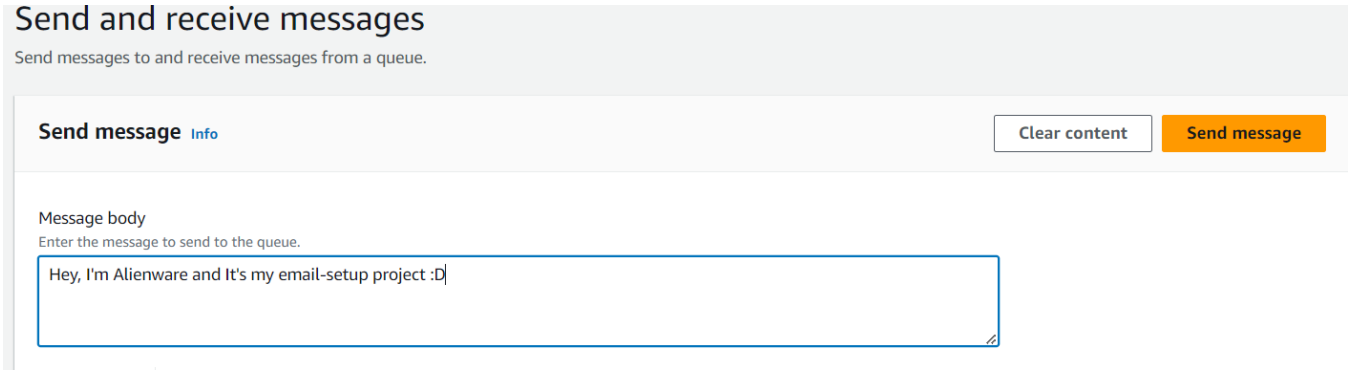
We are done with all the configuration. Now let's test it out!

1. Navigate to SQS service, select the Queue that you created before.

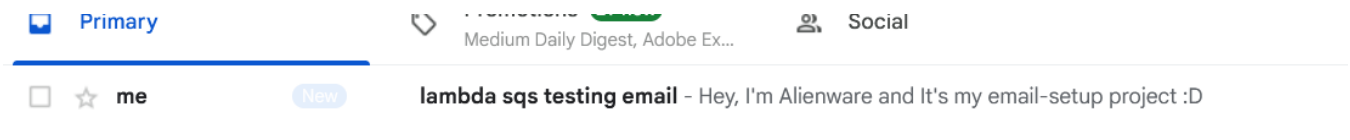


2. Now in the Queue you'll see a **Send and Receive messages** button, click on that.

3. Write the message you want to send in the message body and click on **Send message**.



Open your Gmail account and there you'll see an email sent to you via **AMAZONSES** with the message you wrote in the message body.



lambda sqs testing email Inbox x



alien127.0.0.1@gmail.com via amazonses.com

to me ▼

Hey, I'm Alienware and It's my email-setup project :D

Cleaning Up

Go to all the services that you have used for this project (SQS, SES, Lambda, IAM roles). Select the new entries that you've made and delete them accordingly to don't get any charges.

Lambda > Functions

Functions (1/1)

Last fetched 32 minutes ago

Actions ▲

Create function

<input checked="" type="checkbox"/>	Function name ▼	Description ▼	Package type ▼	Runtime ▼	
<input checked="" type="checkbox"/>	email-setup	-	Zip	Python 3.10	20 minutes ago

Roles (1/8) Info

1 match

<input checked="" type="checkbox"/>	Role name ▲	Trusted entities	Last activity ▼
<input checked="" type="checkbox"/>	email-setup-project	AWS Service: lambda	-

Amazon SES > Configuration: Verified identities > alien127.0.0.1@gmail.com

alien127.0.0.1@gmail.com

Delete

Send test email

Legacy TXT records

Domain verification in Amazon SES is now based on *DomainKeys Identified Mail (DKIM)*, an email authentication standard that receiving mail servers use to validate an email's authenticity. Configuring DKIM in your domain's DNS settings confirms to SES that you're the identity owner, eliminating the need for TXT records.

Queues (1)

	Name ▲	Type ▼	Created ▼	Messages available ▼	Messages in flight ▼	Encryption ▼	
<input checked="" type="radio"/>	email-queue	Standard	2024-01-24T11:00+05:30	0	0	Amazon SQS key (SSE-SQS)	-

THANK YOU CLOUD PEOPLE!!! ☁️💻

If you find this blog helpful, please do like it and subscribe to my page for more such content <3

Connect with me on Linkedin : <https://www.linkedin.com/in/pawanngambhir/>