IMPLEMENTING SERVERLESS ARCHITECTURE ON AWS

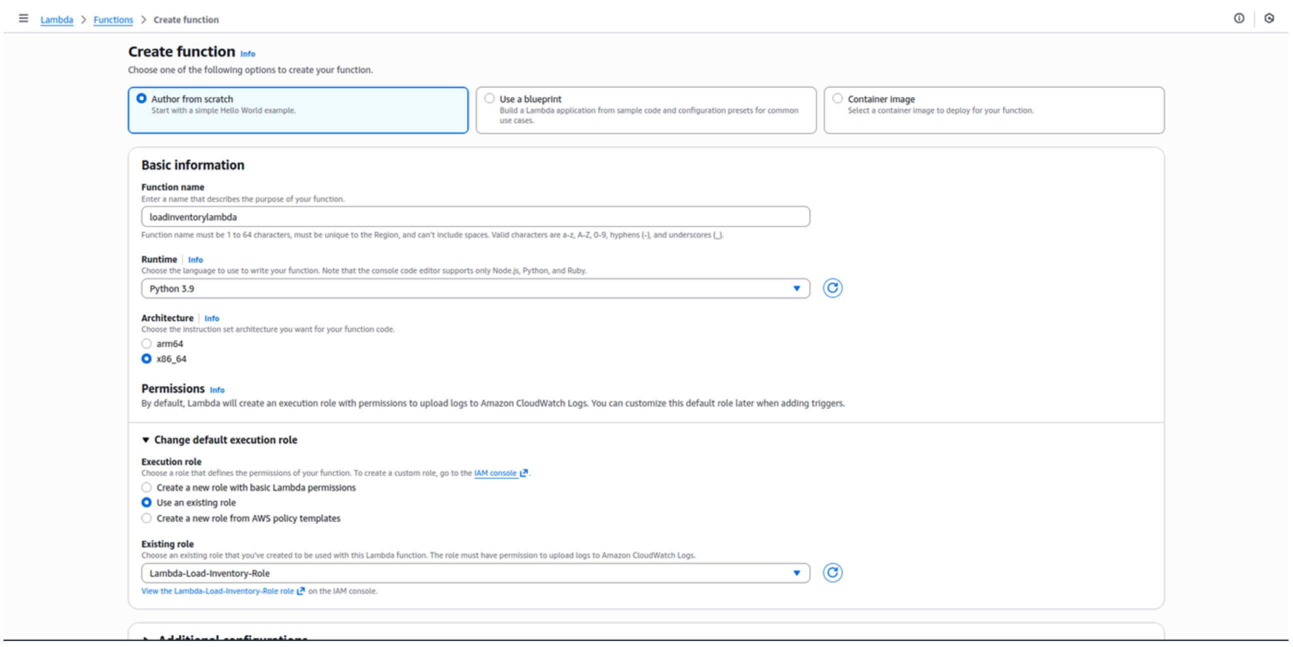
STEP 1: CREATE A LAMBDA FUNCTION

Function name: loadinventorylamda

Runtime: python 3.9

Change execution role: use an exixting role

Choose: lambda_load_inventory_role



Lambda > Functions > Create function

Create function Info

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.
loadinventorylambda
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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 3.9

Architecture Info
Choose the instruction set architecture you want for your function code.
arm64 [✓ x86_64]

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **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.
Lambda-Load-Inventory-Role
View the Lambda-Load-Inventory-Role role on the IAM console.

▸ Additional configurations

In the **Code source** section, in the **Environment** pane, choose **lambda_function.py**.

In the code editor for the **lambda_function.py** file, delete all the default code.

In the **Code source** editor, copy and paste the following code:

```
 Load-Inventory Lambda function

# This function is invoked by an object being created in an Amazon S3 bucket.

# The file Is downloaded and each line is inserted into a DynamoDB table.

import json, urllib, boto3, csv

# Connect to S3 and DynamoDB

s3 = boto3.resource('s3')

dynamodb = boto3.resource('dynamodb')

# Connect to the DynamoDB tables

inventoryTable = dynamodb.Table('Inventory');

# This handler is run every time the Lambda function is invoked

def lambda_handler(event, context):
 # Show the incoming event in the debug log
print("Event received by Lambda function: " + json.dumps(event, indent=2))
# Get the bucket and object key from the Event
bucket = event['Records'][0]['s3']['bucket']['name']
 key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])
localFilename = '/tmp/inventory.txt'
# Download the file from S3 to the local filesystem
try:
s3.meta.client.download_file(bucket, key, localFilename)
except Exception as e:
print(e)
print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key,
bucket))
raise e
# Read the Inventory CSV file
with open(localFilename) as csvfile:
reader = csv.DictReader(csvfile, delimiter=',')
# Read each row in the file
rowCount = 0
for row in reader:
rowCount += 1
# Show the row in the debug log
print(row['store'], row['item'], row['count'])
try:
# Insert Store, Item and Count into the Inventory table
inventoryTable.put_item(
Item={
'Store':  row['store'],
'Item':   row['item'],
'Count':  int(row['count'])})
except Exception as e:
print(e)
print("Unable to insert data into DynamoDB table".format(e))
# Finished!
return "%d counts inserted" % rowCount
```

STEP 2: CREATE S3 BUCKET

Name: myinventory

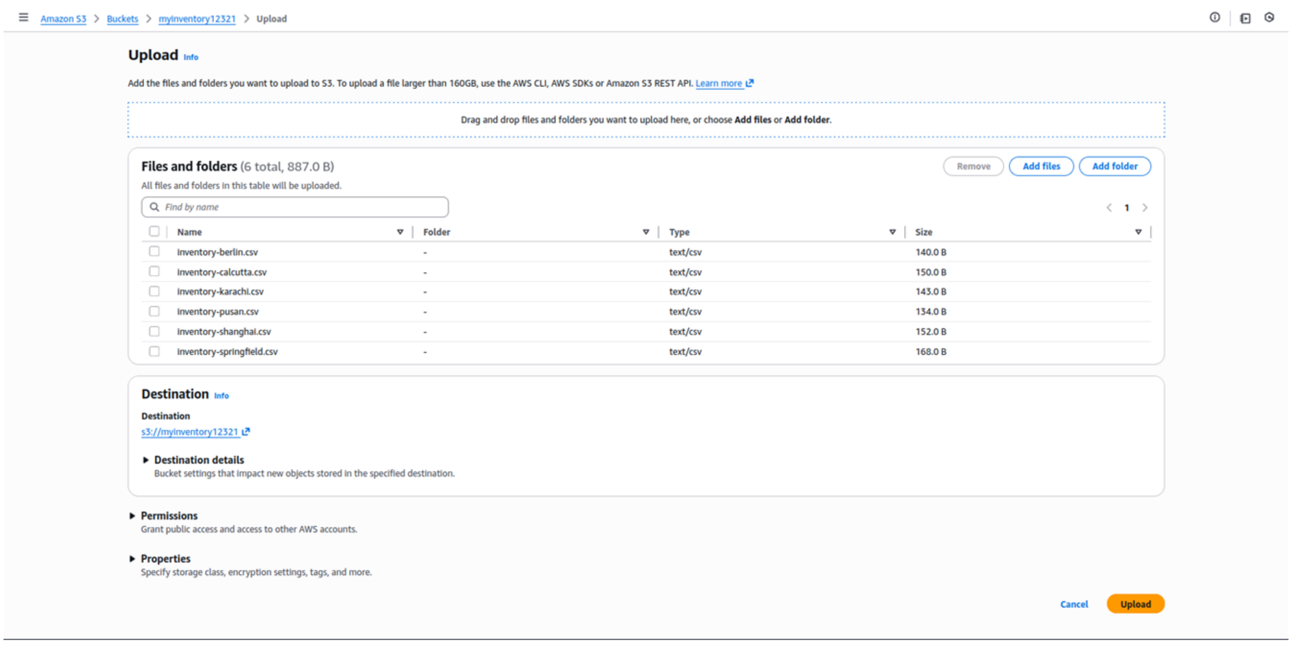
Navigate to Properties→Event notification→Create event notification

Event Name:Inventory load

Event Types:Object creation

Destination:Lambda Function

STEP 3:UPLOAD THE CSV FILES INTO S3



Amazon S3 > Buckets > myinventory12321 > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (6 total, 887.0 B)

All files and folders in this table will be uploaded.

Name	Folder	Type	Size
inventory-berlin.csv	-	text/csv	140.0 B
inventory-calcutta.csv	-	text/csv	150.0 B
inventory-karachi.csv	-	text/csv	143.0 B
inventory-pusan.csv	-	text/csv	134.0 B
inventory-shanghai.csv	-	text/csv	152.0 B
inventory-springfield.csv	-	text/csv	168.0 B

Destination Info

Destination

s3://myinventory12321

Destination details
Bucket settings that impact new objects stored in the specified destination.

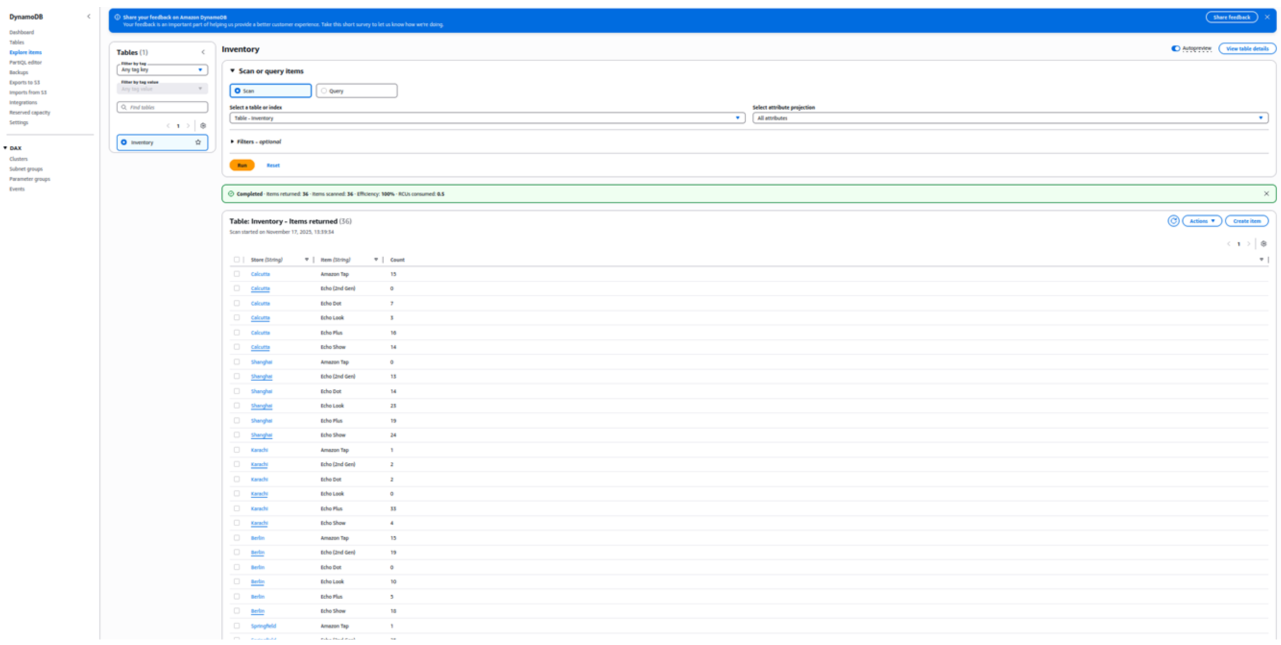
Permissions
Grant public access and access to other AWS accounts.

Properties
Specify storage class, encryption settings, tags, and more.

STEP 4: DYNAMO DB

-Explore the items.

-Items will be displayed.



DynamoDB

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

Inventory

Scan or query items

Select a table or index: Table - Inventory

Select attribute projection: All attributes

Completed · Items returned: 36 · Items scanned: 36 · Efficiency: 100% · RCUs consumed: 0.5

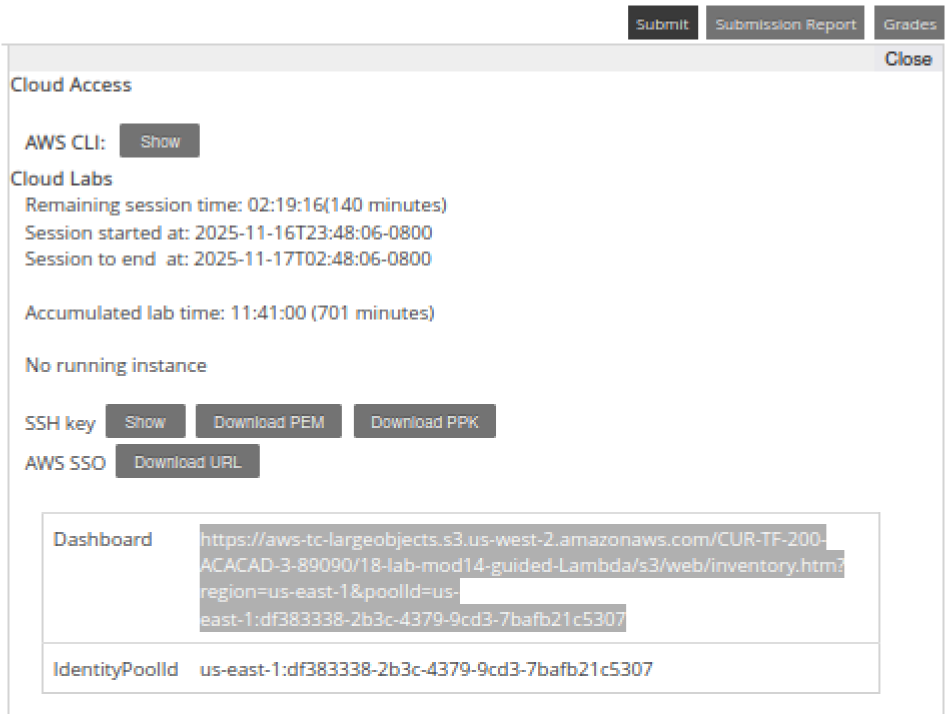
Table: Inventory - Items returned (36)

Scan started on November 17, 2025, 13:39:34

Store (String)	Item (String)	Count
Calcutta	Amazon Tap	15
Calcutta	Echo (2nd Gen)	0
Calcutta	Echo Dot	7
Calcutta	Echo Look	3
Calcutta	Echo Plus	16
Calcutta	Echo Show	14
Shanghai	Amazon Tap	0
Shanghai	Echo (2nd Gen)	13
Shanghai	Echo Dot	14
Shanghai	Echo Look	23
Shanghai	Echo Plus	19
Shanghai	Echo Show	24
Karachi	Amazon Tap	1
Karachi	Echo (2nd Gen)	2
Karachi	Echo Dot	2
Karachi	Echo Look	0
Karachi	Echo Plus	33
Karachi	Echo Show	4
Berlin	Amazon Tap	15
Berlin	Echo (2nd Gen)	19
Berlin	Echo Dot	0
Berlin	Echo Look	10
Berlin	Echo Plus	5
Berlin	Echo Show	18
Springfield	Amazon Tap	1

STEP 5: DASHBOARD

Go to AWS I details



Cloud Access

AWS CLI: Show

Cloud Labs

Remaining session time: 02:19:16(140 minutes)
Session started at: 2025-11-16T23:48:06-0800
Session to end at: 2025-11-17T02:48:06-0800

Accumulated lab time: 11:41:00 (701 minutes)

No running instance

Dashboard	https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD-3-89090/18-lab-mod14-guided-Lambda/s3/web/inventory.htm?region=us-east-1&poolId=us-east-1:df383338-2b3c-4379-9cd3-7bafb21c5307
IdentityPoolId	us-east-1:df383338-2b3c-4379-9cd3-7bafb21c5307

Access the dashboard link.

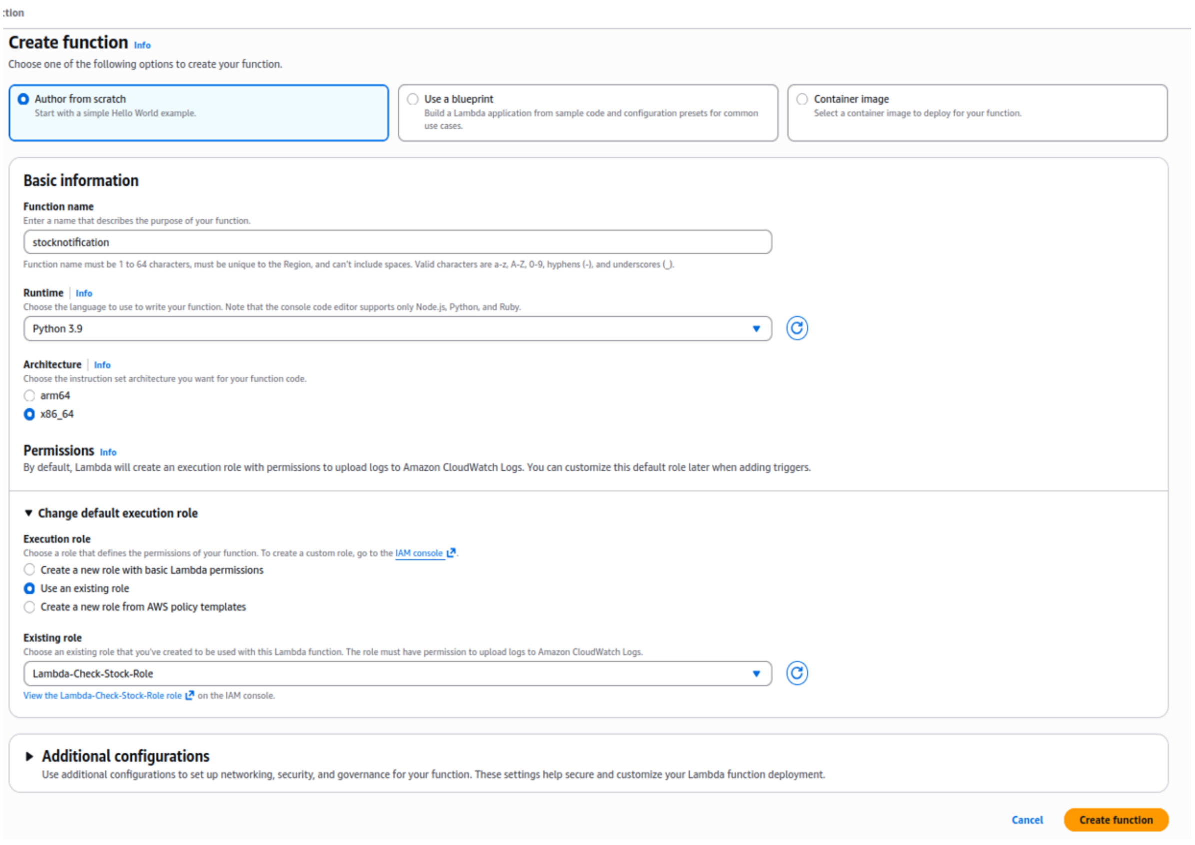
STEP 6: CREATE ANOTHER LAMBDA FUNCTION

Create function: Storenotification

Runtime:python 3.9

use existing role: lambda_check_stock_role

create function



:tion

Create function Info

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.
stocknotification
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

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 3.9

Architecture | Info
Choose the instruction set architecture you want for your function code.
( ) arm64
(●) x86_64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **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

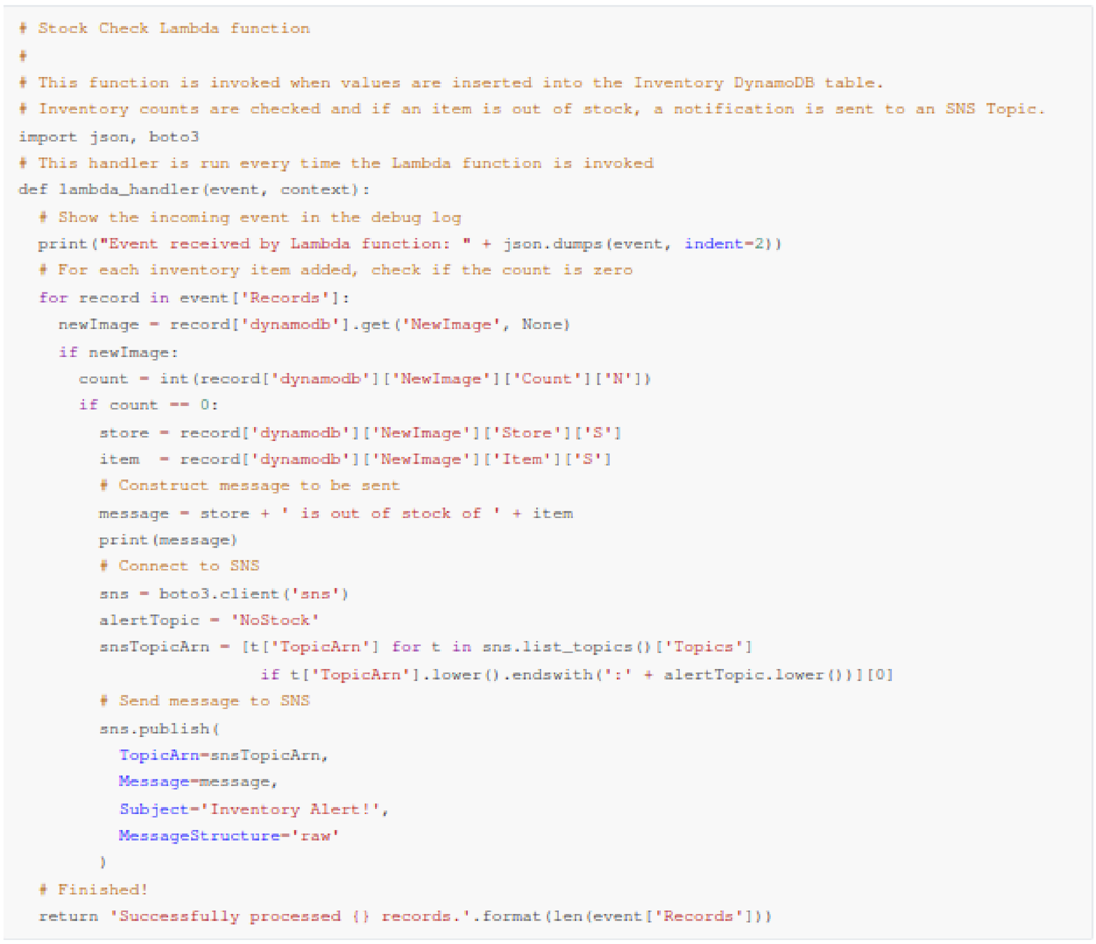
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
Lambda-Check-Stock-Role
View the Lambda-Check-Stock-Role role on the IAM console.

▶ **Additional configurations**
Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

In the **Code source** section, in the **Environment** pane, choose **lambda_function.py**.

In the code editor for the **lambda_function.py** file, delete all the default code.

In the **Code source** editor, copy and paste the following code:



```
# Stock Check Lambda function
#
# This function is invoked when values are inserted into the Inventory DynamoDB table.
# Inventory counts are checked and if an item is out of stock, a notification is sent to an SNS Topic.
import json, boto3
# This handler is run every time the Lambda function is invoked
def lambda_handler(event, context):
  # Show the incoming event in the debug log
  print("Event received by Lambda function: " + json.dumps(event, indent=2))
  # For each inventory item added, check if the count is zero
  for record in event['Records']:
    newImage = record['dynamodb'].get('NewImage', None)
    if newImage:
      count = int(record['dynamodb']['NewImage']['Count']['N'])
      if count == 0:
        store  = record['dynamodb']['NewImage']['Store']['S']
        item   = record['dynamodb']['NewImage']['Item']['S']
        # Construct message to be sent
        message = store + ' is out of stock of ' + item
        print(message)
        # Connect to SNS
        sns = boto3.client('sns')
        alertTopic = 'NoStock'
        snsTopicArn = [t['TopicArn'] for t in sns.list_topics()['Topics']
                        if t['TopicArn'].lower().endswith(':' + alertTopic.lower())][0]
        # Send message to SNS
        sns.publish(
          TopicArn=snsTopicArn,
          Message=message,
          Subject='Inventory Alert!',
          MessageStructure='raw'
        )
  # Finished!
  return 'Successfully processed {} records.'.format(len(event['Records']))
```

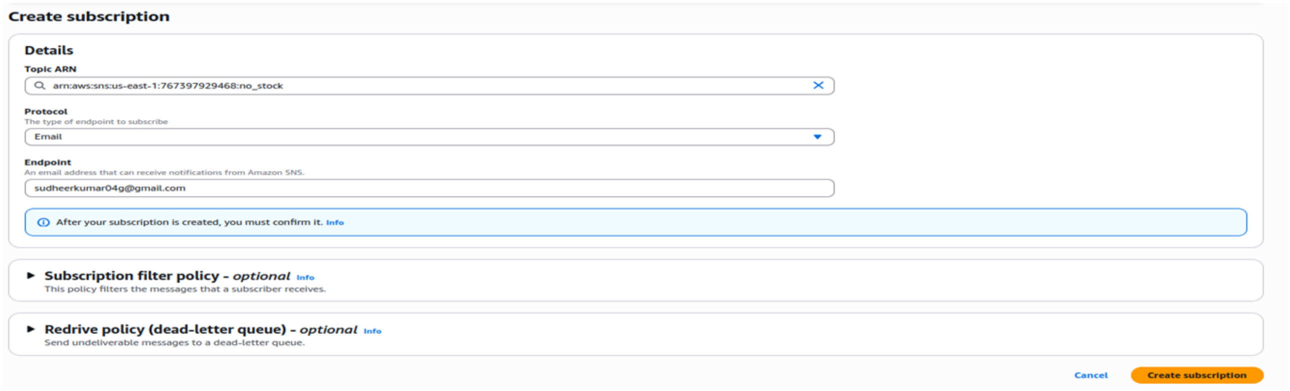
STEP 7: SIMPLE NOTIFICATION SERVICE

Create topic: standard

Name: stock

Create topic.

Create Subscription



Create subscription

Details

Topic ARN

arn:aws:sns:us-east-1:767397929468:no_stock

Protocol
The type of endpoint to subscribe

Email

Endpoint
An email address that can receive notifications from Amazon SNS.

sudheerkumar04g@gmail.com

After your subscription is created, you must confirm it. Info

Subscription filter policy - *optional* Info
This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - *optional* Info
Send undeliverable messages to a dead-letter queue.

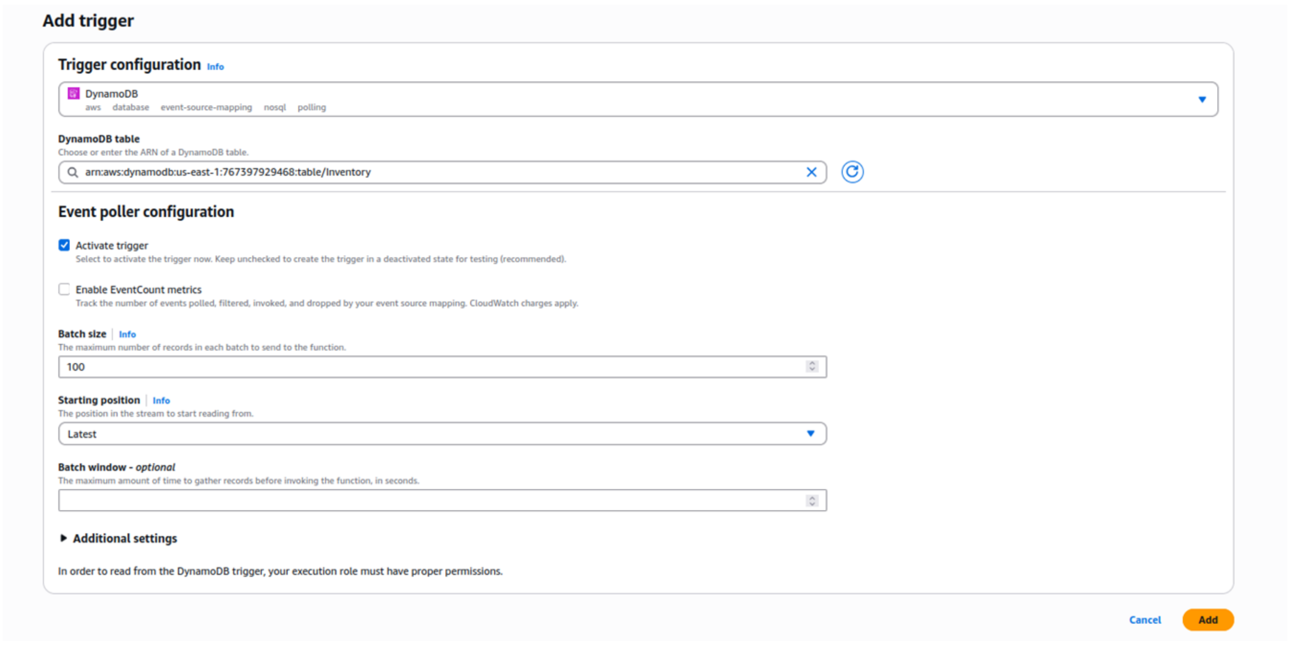
Go mail and confirm subscription.

Step 8:Trigger function

Go to lambda

Add trigger :dynamodb

Dnamo table:inventory



Add trigger

Trigger configuration Info

DynamoDB
aws database event-source-mapping nosql polling

DynamoDB table
Choose or enter the ARN of a DynamoDB table.
arn:aws:dynamodb:us-east-1:767397929468:table/Inventory

Event poller configuration

[✓] Activate trigger
Select to activate the trigger now. Keep unchecked to create the trigger in a deactivated state for testing (recommended).

[ ] Enable EventCount metrics
Track the number of events polled, filtered, invoked, and dropped by your event source mapping. CloudWatch charges apply.

Batch size | Info
The maximum number of records in each batch to send to the function.
100

Starting position | Info
The position in the stream to start reading from.
Latest

Batch window - *optional*
The maximum amount of time to gather records before invoking the function, in seconds.

▸ Additional settings

In order to read from the DynamoDB trigger, your execution role must have proper permissions.

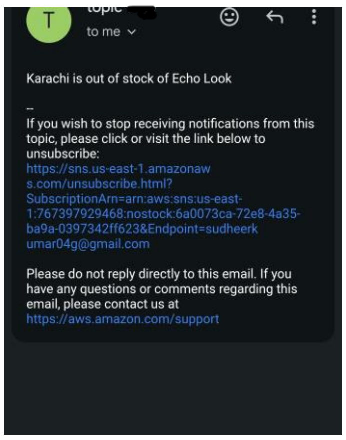
STEP 9: upload files

Go to S3 bucket and upload the csv files.

STEP 10: Notification

Go and check the email.

Email will report should be received.



topic [redacted]
to me

Karachi is out of stock of Echo Look

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:767397929468:nostock:6a0073ca-72e8-4a35-ba9a-0397342ff623&Endpoint=sudheerkumar04g@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support