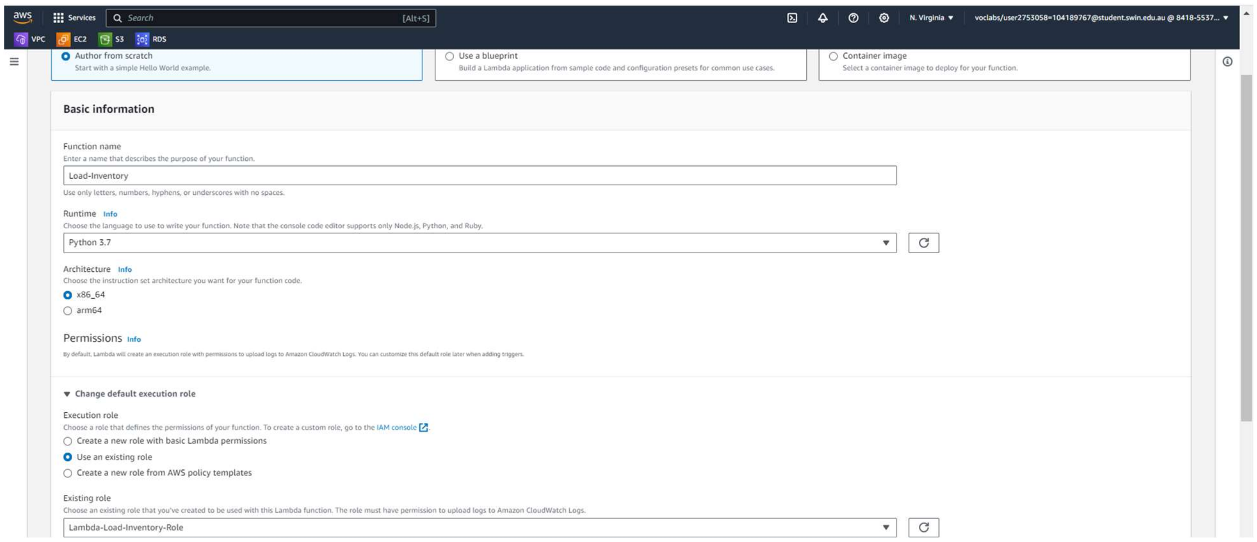
Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda

Name : Pham Do Tien Phong
Student ID: 104189767

Task 1: Creating a Lambda function to load data


Services
Search
[Alt+S]
N. Virginia
voclabs/user2753058=104189767@student.swin.edu.au @ 8418-5537...
VPC
EC2
S3
RDS

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.
Load-Inventory
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 3.7

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

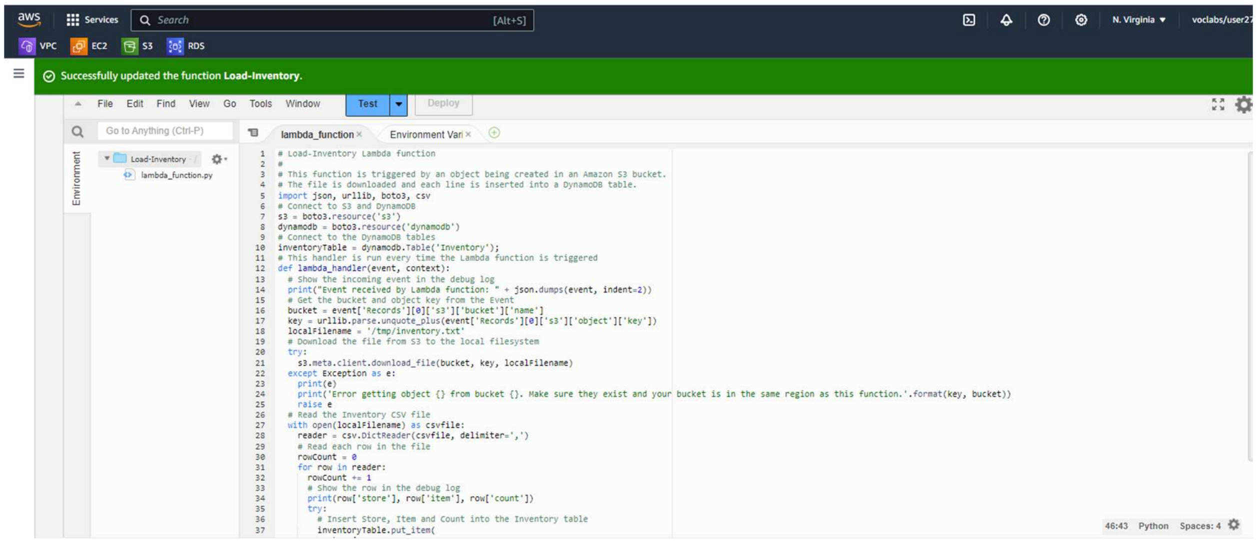
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

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

I create a Lambda function named Load-Inventory , runtime is Python 3.7 as well as using existing role (Lambda-Load-Inventory-Role)

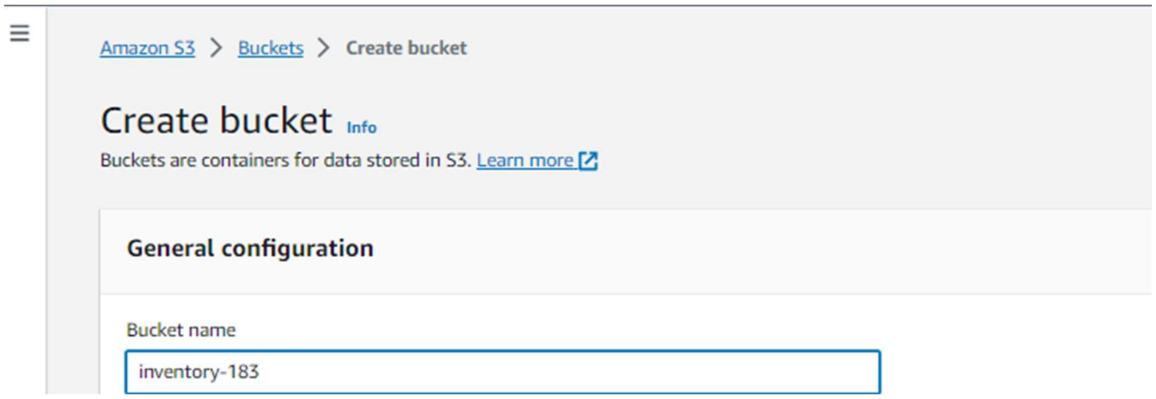


Successfully updated the function Load-Inventory.

```
# Load-Inventory Lambda function
#
# This function is triggered 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 triggered
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(
```

Deploy the code provided

Task 2: Configuring an Amazon S3 event


Amazon S3 > Buckets > Create bucket

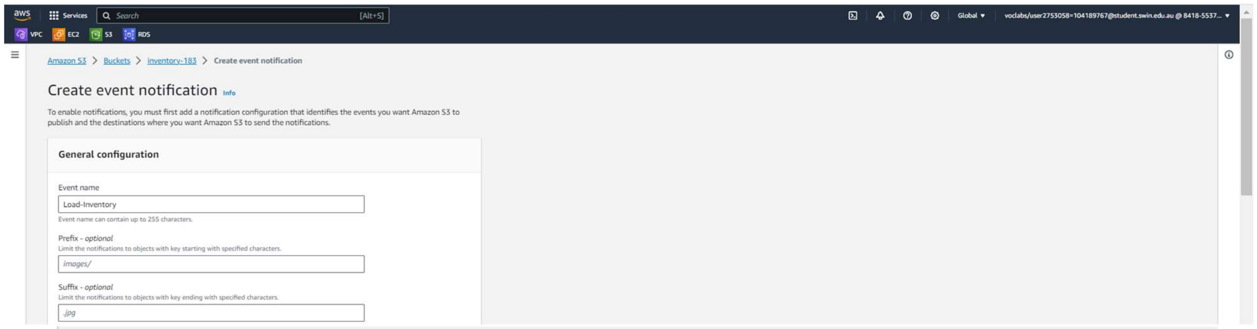
Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name: inventory-183

Creating a bucket has name as inventory-183


aws
Services
Search [Alt+S]
VPC
EC2
S3
RDS
Global
voclabs/user2753058=104189767@student.swin.edu.au @ 8418-5537...

Amazon S3 > Buckets > inventory-183 > Create event notification

Create event notification Info

To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications.

General configuration

Event name: Load-Inventory

Event name can contain up to 255 characters.

Prefix - *optional*

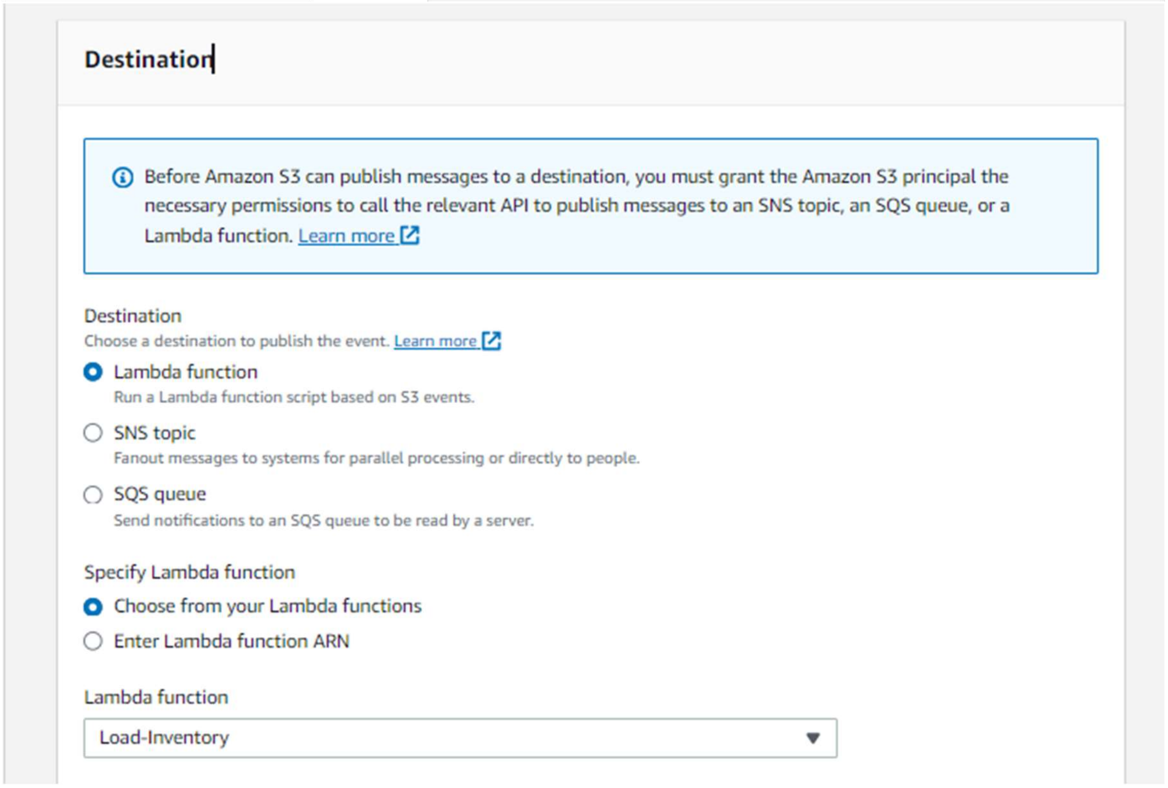
Limit the notifications to objects with key starting with specified characters.

images/

Suffix - *optional*

Limit the notifications to objects with key ending with specified characters.

.jpg


Destination

Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. Learn more

Destination

Choose a destination to publish the event. Learn more

[✓ Lambda function] Run a Lambda function script based on S3 events.

SNS topic — Fanout messages to systems for parallel processing or directly to people.

SQS queue — Send notifications to an SQS queue to be read by a server.

Specify Lambda function

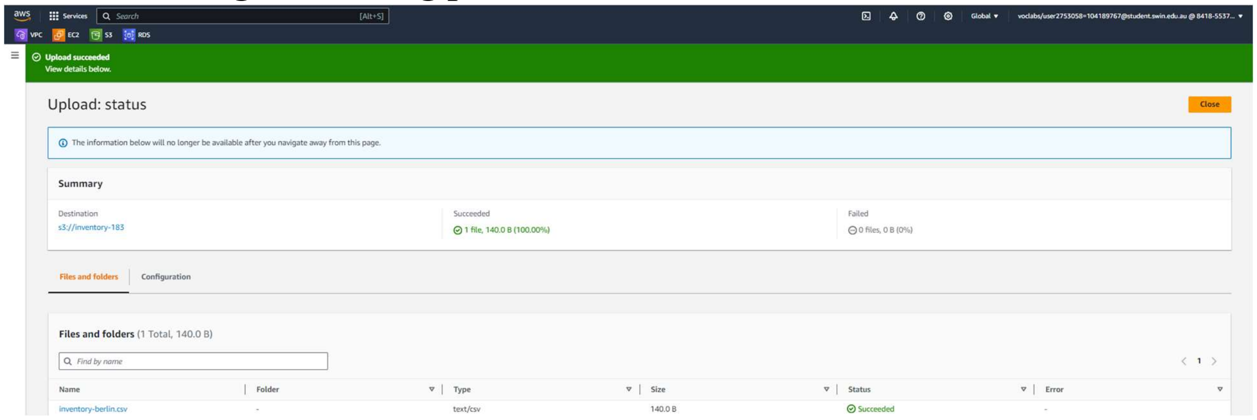
[✓ Choose from your Lambda functions]

Enter Lambda function ARN

Lambda function: Load-Inventory

I create event notification.

Task 3: Testing the loading process



Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

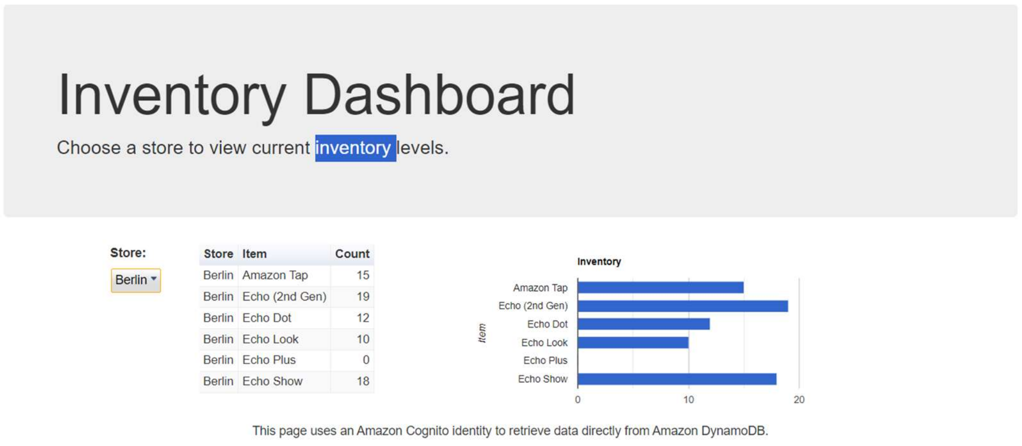
Summary

Destination	Succeeded	Failed
s3://inventory-183	1 file, 140.0 B (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 140.0 B)

Name	Folder	Type	Size	Status	Error
inventory-berlin.csv	-	text/csv	140.0 B	Succeeded	-

I upload the files inventory-berlin.csv



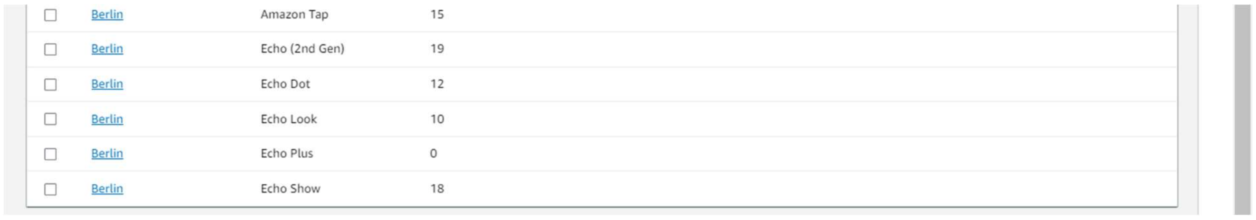
Inventory Dashboard

Choose a store to view current inventory levels.

Store	Item	Count
Berlin	Amazon Tap	15
Berlin	Echo (2nd Gen)	19
Berlin	Echo Dot	12
Berlin	Echo Look	10
Berlin	Echo Plus	0
Berlin	Echo Show	18

This page uses an Amazon Cognito identity to retrieve data directly from Amazon DynamoDB.

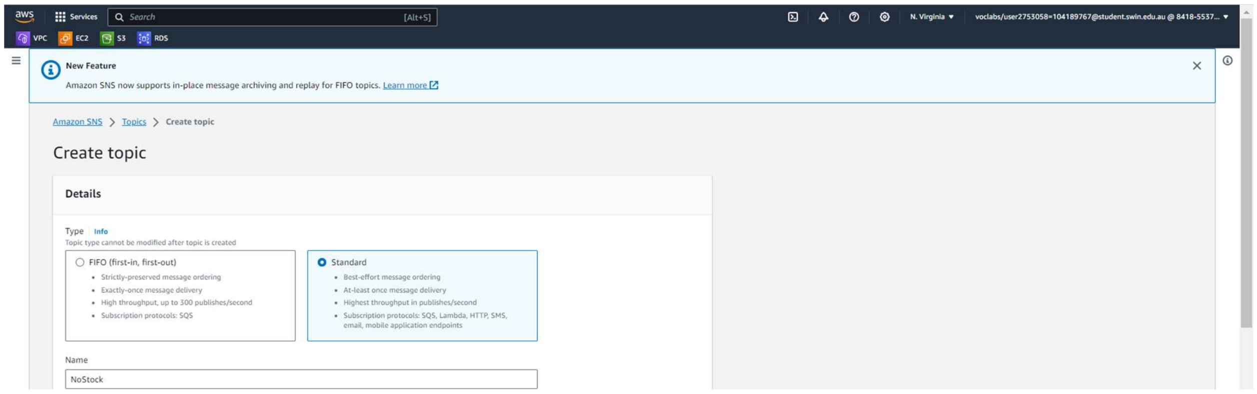
And I can access to inventory Dashboard



☐	Berlin	Amazon Tap	15
☐	Berlin	Echo (2nd Gen)	19
☐	Berlin	Echo Dot	12
☐	Berlin	Echo Look	10
☐	Berlin	Echo Plus	0
☐	Berlin	Echo Show	18

The items of berlin is stored in DynamoDB

Task 4: Configuring notifications



New Feature

Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more

Amazon SNS > Topics > Create topic

Create topic

Details

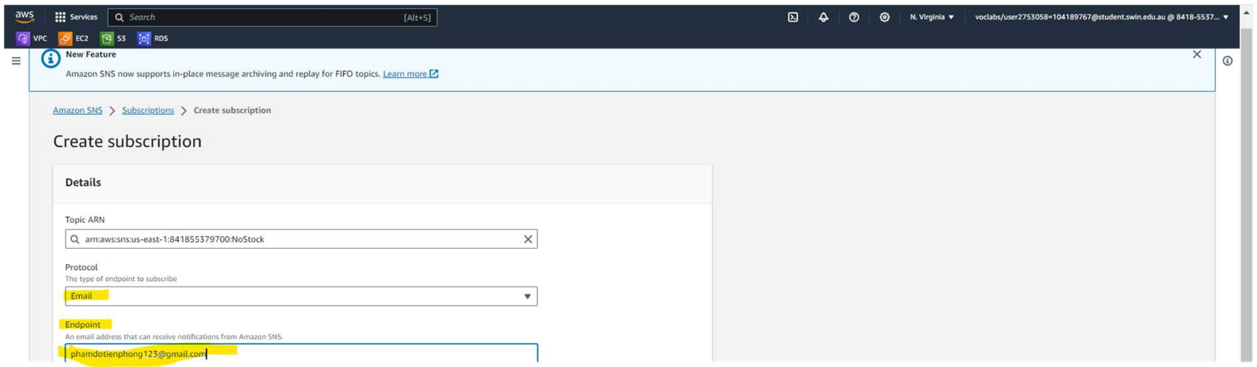
Type Info

Topic type cannot be modified after topic is created

- FIFO (first-in, first-out)
  - Strictly-preserved message ordering
  - Exactly-once message delivery
  - High throughput, up to 300 publishes/second
  - Subscription protocols: SQS
- [✓ Standard]
  - Best-effort message ordering
  - At-least once message delivery
  - Highest throughput in publishes/second
  - Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name: NoStock

I create topic with the Standard type and NoStock name.



New Feature

Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more

Amazon SNS > Subscriptions > Create subscription

Create subscription

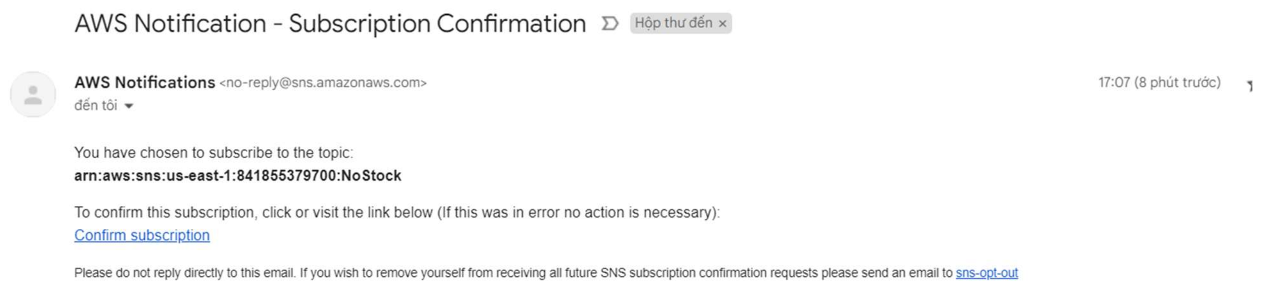
Details

Topic ARN: arn:aws:sns:us-east-1:841855379700:NoStock

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
phamdotienphong123@gmail.com

The next step is that I create subscription with Protocol : Email and endpoint is my Gmail (phamdotienphong123@gmail.com)



AWS Notification - Subscription Confirmation Hộp thư đến ×

AWS Notifications <no-reply@sns.amazonaws.com>
đến tôi
17:07 (8 phút trước)

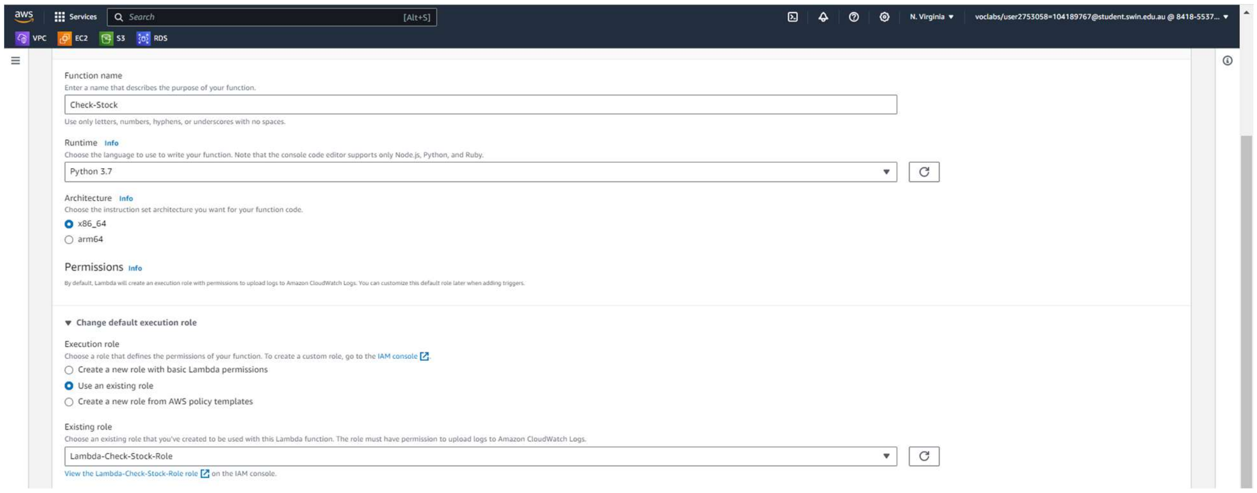
You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:841855379700:NoStock

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

I receive confirmation email

Task 5: Creating a Lambda function to send notifications


Function name
Enter a name that describes the purpose of your function.
Check-Stock
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 3.7

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

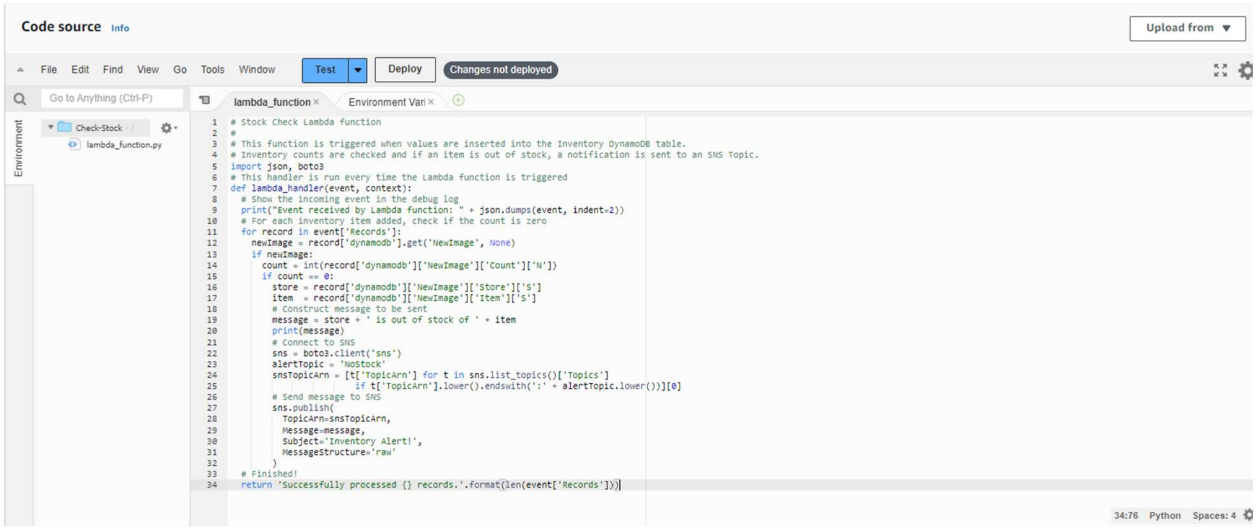
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

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

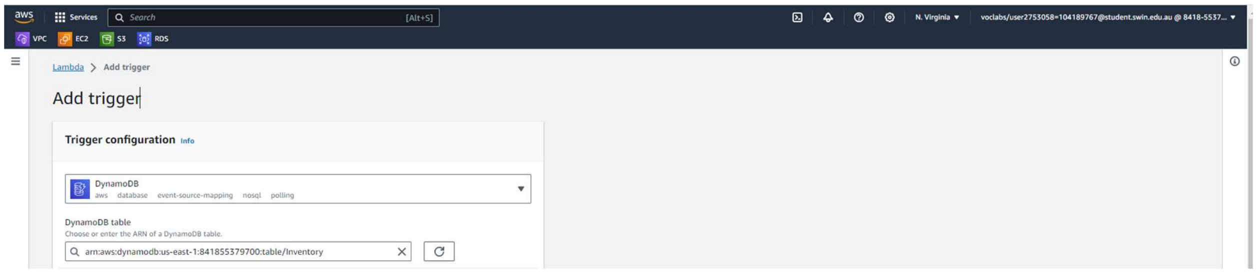
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
Lambda-Check-Stock-Role
View the Lambda-Check-Stock-Role role on the IAM console.

I create Lambda function to send notifications with name : Check-stock , runtime : Python3.7, existing Role : Lambda-Check-Stock-Role



```python
# Stock Check Lambda function
#
# This function is triggered 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 triggered
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']))
```

Deploying the code source



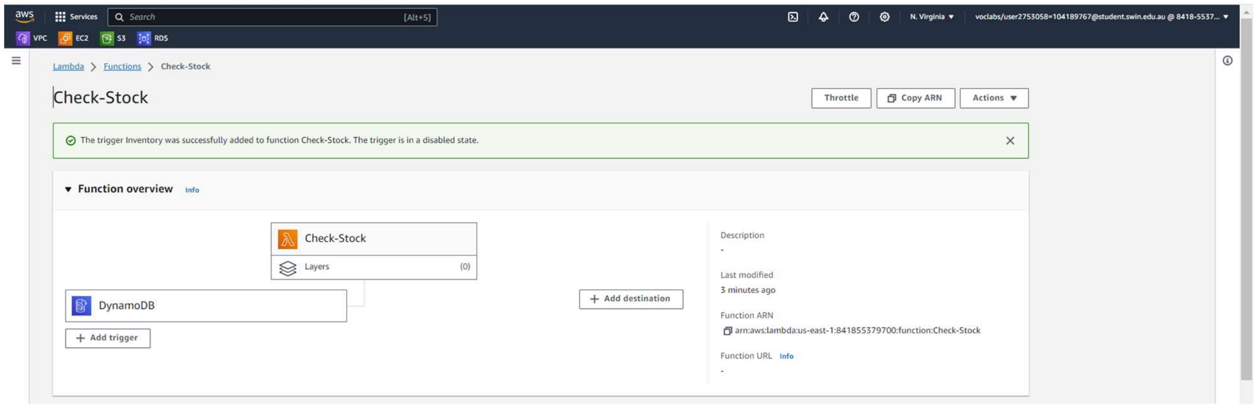
Add trigger

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-1:841855379700:table/Inventory

I add trigger .



Check-Stock

The trigger Inventory was successfully added to function Check-Stock. The trigger is in a disabled state.

Function overview

Description: -

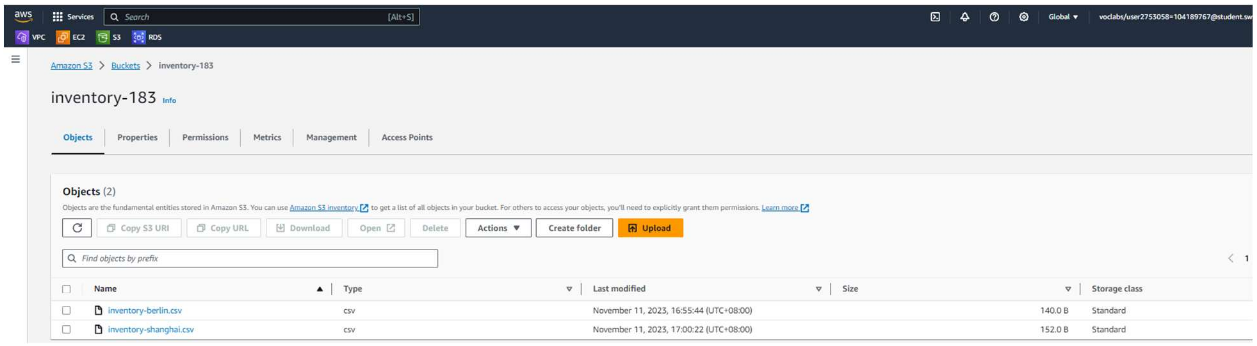
Last modified: 3 minutes ago

Function ARN: arn:aws:lambda:us-east-1:841855379700:function:Check-Stock

Function URL: -

This is my function overview

Task 6: Testing the System



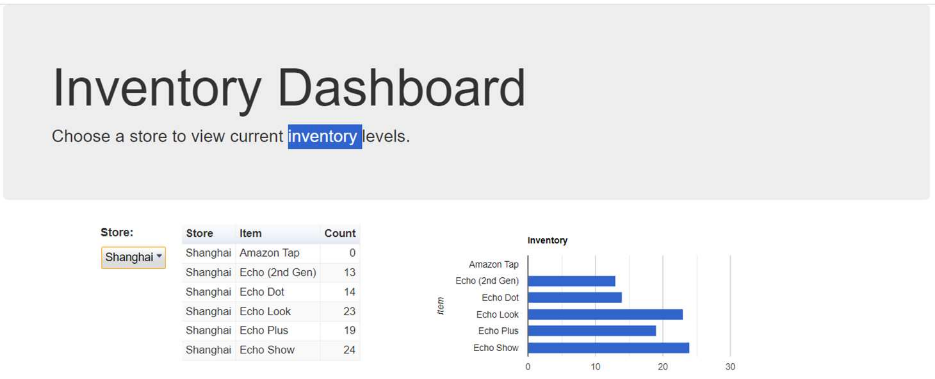
inventory-183

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
inventory-berlin.csv	csv	November 11, 2023, 16:55:44 (UTC+08:00)	140.0 B	Standard
inventory-shanghai.csv	csv	November 11, 2023, 17:00:22 (UTC+08:00)	152.0 B	Standard

I test to upload inventory-shanghai.csv



Inventory Dashboard

Choose a store to view current inventory levels.

Store: Shanghai

Store	Item	Count
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

I can access it through web application