

Swinburne University of Technology

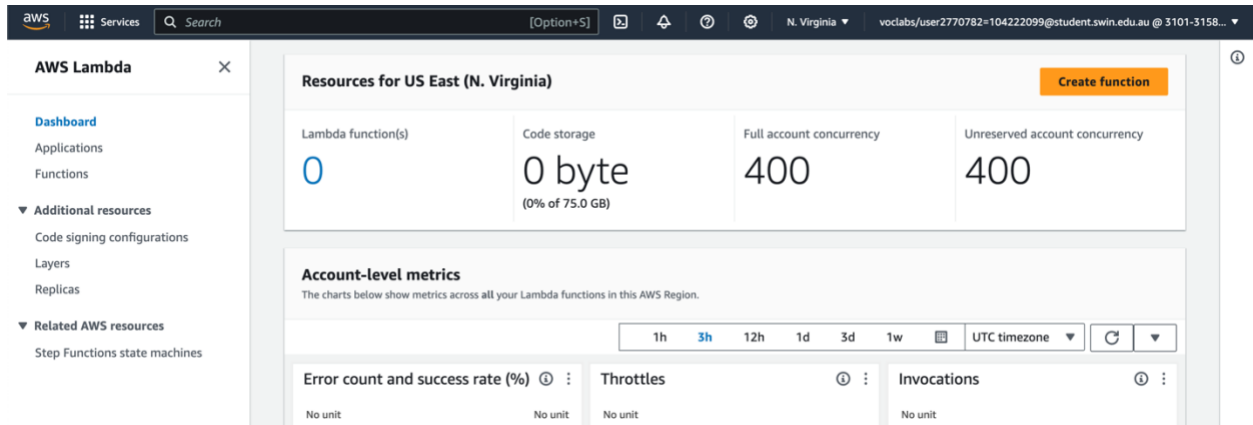
COS20019 Cloud Computing Architecture

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

Saturday 11th October, 2023

Task 1: Creating a Lambda function to load data

On the **AWS Management Console**, on the Services menu, choose **Lambda**.
Choose **Create function**



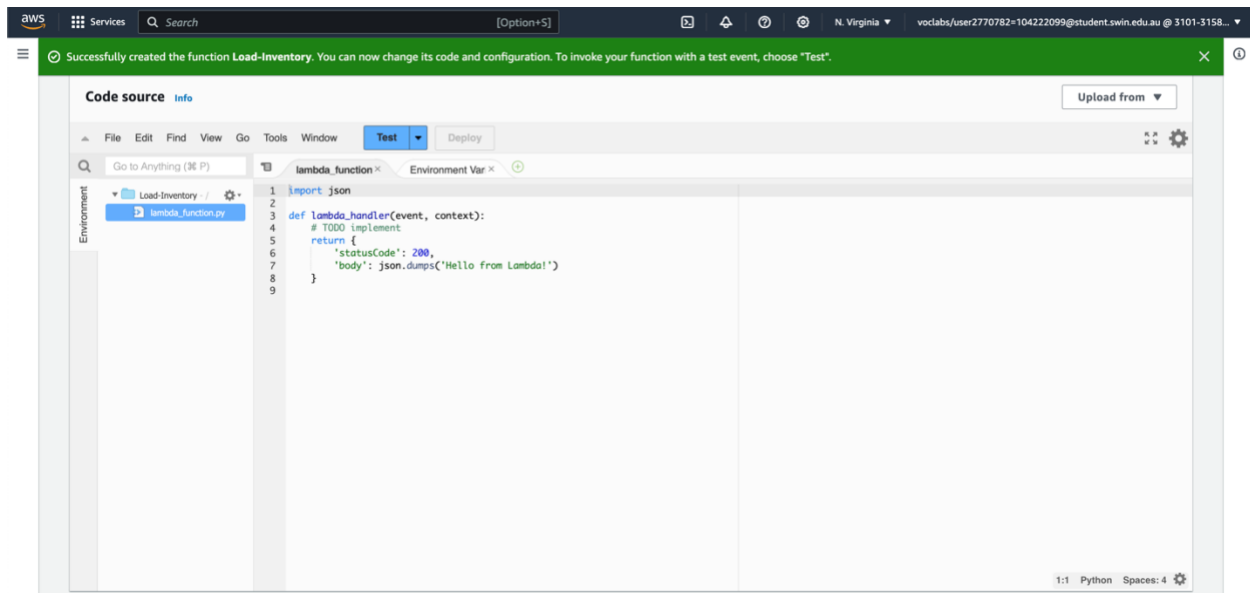
Configure the following settings:

- **Function name:** Load-Inventory
- **Runtime:** *Python 3.7*
- Expand **Choose or create an execution role.**
- **Execution role:** *Use an existing role*
- **Existing role:** *Lambda-Load-Inventory-Role*

The screenshot shows the AWS Lambda console configuration page for a new function. The configuration is as follows:

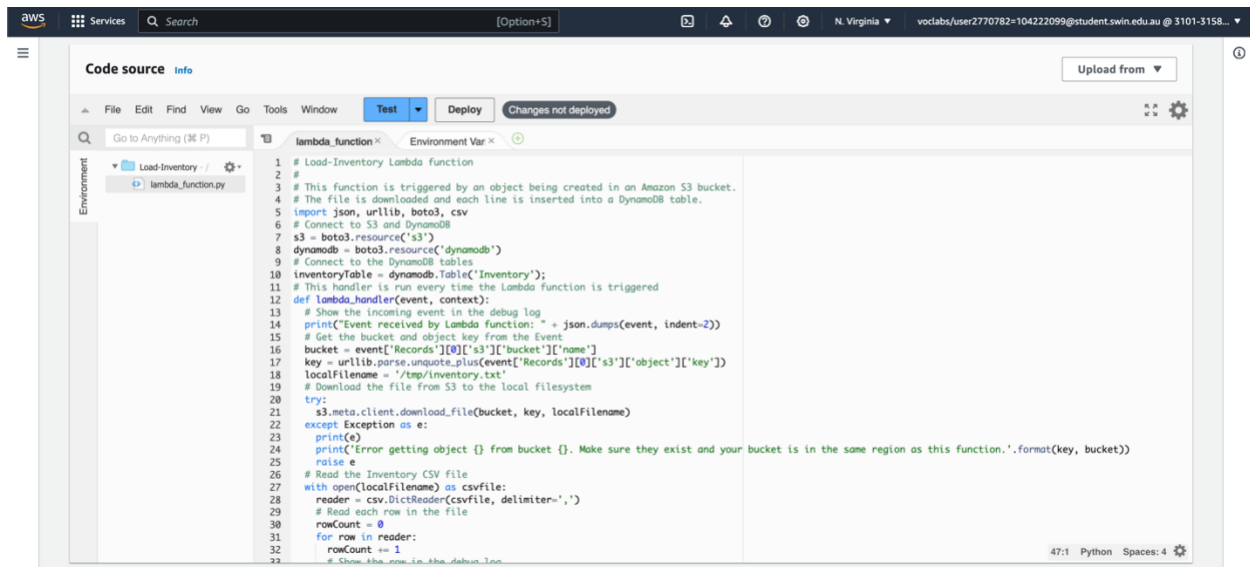
- Function name:** Load-Inventory
- Runtime:** Python 3.7
- Architecture:** x86_64
- Permissions:** Use an existing role
- Existing role:** Lambda-Load-Inventory-Role

Scroll down to the **Code source** section, and in the **Environment** pane, choose `lambda_function.py`.



In the code editor, delete all the code.

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



Choose **Deploy** to save your changes.



Task 2: Configuring an Amazon S3 event

On the **Services** menu, choose **S3**.

Choose **Create bucket**

For **Bucket name** enter: inventory-<number> (Replace with a random number)

The screenshot shows the 'Create bucket' page in the AWS Management Console. The 'General configuration' section is active, showing a bucket name of 'inventory-1402' and the AWS Region set to 'US East (N. Virginia) us-east-1'. Below this, there is a section for 'Object Ownership' with two radio buttons: 'ACLs disabled (recommended)' (which is selected) and 'ACLs enabled'. The 'ACLs disabled' option states that all objects in the bucket are owned by the account and access is specified using only policies. The 'ACLs enabled' option states that objects can be owned by other AWS accounts and access is specified using ACLs. There is also a 'Copy settings from existing bucket - optional' section with a 'Choose bucket' button.

Choose the name of your *inventory*- bucket.

Choose the **Properties** tab.

Scroll down to **Event notifications**.

The screenshot shows the 'Event notifications' page in the AWS Management Console. At the top, there are buttons for 'Edit', 'Delete', and 'Create event notification'. Below this is a table with columns for 'Name', 'Event types', 'Filters', 'Destination type', and 'Destination'. The table is currently empty, with a message stating 'No event notifications' and 'Choose Create event notification to be notified when a specific event occurs.' Below the table, there is a section for 'Amazon EventBridge' with an 'Edit' button. This section includes a description of Amazon EventBridge and a toggle switch for 'Send notifications to Amazon EventBridge for all events in this bucket', which is currently set to 'Off'.

Click **Create event notification** then configure these settings:

- **Name:** Load-Inventory
- **Event types:** *All object create events*

- **Destination:** *Lambda Function*
- **Lambda function:** *Load-Inventory*
- Choose **Save changes**

The screenshot shows the 'General configuration' page in the AWS IAM console. The 'Event name' field is set to 'Load-Inventory'. The 'Prefix - optional' field is set to 'images/'. The 'Suffix - optional' field is set to '.jpg'. Under 'Event types', the 'Object creation' section is expanded, showing 'All object create events' selected. The 'Put' option is also visible.

The screenshot shows the 'Destination' page in the AWS IAM console. A blue box contains a warning: '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](#)'. The 'Destination' section shows 'Lambda function' selected. Under 'Specify Lambda function', 'Choose from your Lambda functions' is selected. The 'Lambda function' dropdown menu shows 'Load-Inventory'.

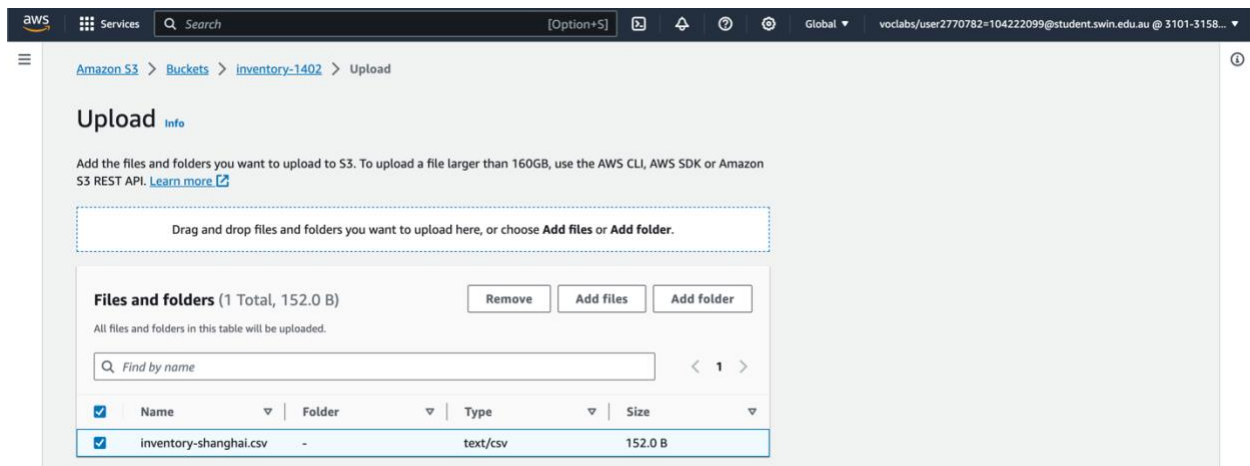
The screenshot shows a green success message at the bottom of the AWS IAM console: 'Successfully created event notification "Load-Inventory". Operation successfully completed.'

Task 3: Testing the loading process

Download the inventory files by opening the context (right-click) menu for these links:

[inventory-berlin.csv](#)
[inventory-calcutta.csv](#)
[inventory-karachi.csv](#)
[inventory-pusan.csv](#)
[inventory-shanghai.csv](#)
[inventory-springfield.csv](#)

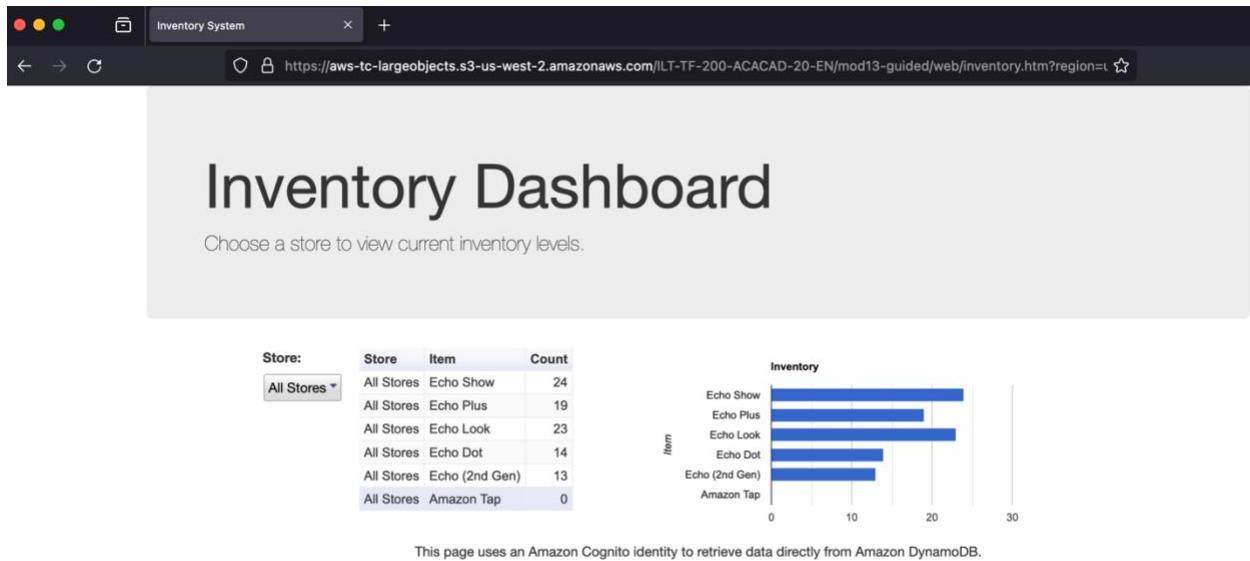
Return to your S3 bucket in the console by choosing the **Objects** tab.
 Choose **Upload** > **Add files**, and select one of the inventory CSV files (You can choose any inventory file) > **Upload**



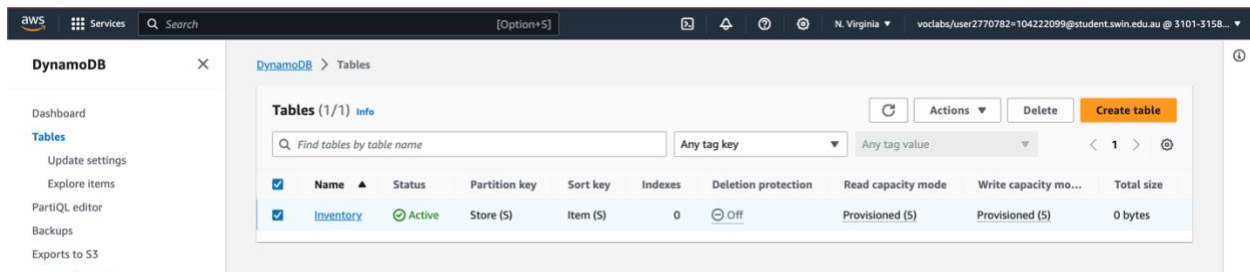
At the top of these instructions, choose the **Details** button; to the right of **AWS**, choose the **Show** button.

SecretKey	1QwWWz+NFMuNg8aEthYEzG+/ofnFPpR+5EzHUgYz
Dashboard	https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/ILT-TF-200-ACACAD-20-EN/mod13-guided/web/inventory.htm?region=us-east-1&poolId=us-east-1:e0147493-fa17-4431-9dcd-b9935b07eaae
IdentityPoolId	us-east-1:e0147493-fa17-4431-9dcd-b9935b07eaae
AccessKey	AKIAUQNKIO5AWDLBMI4Z

From the **Credentials** window, copy the **Dashboard** URL.
 Open a new web browser tab, paste the URL, and press ENTER.



On the **Services** menu, choose **DynamoDB**.
In the left navigation pane, choose **Tables**.
Choose the **Inventory** table.



Choose the **Items** tab.

Inventory

Scan or query items
Expand to query or scan items.

Completed. Read capacity units consumed: 0.5

Items returned (6)

Store (String)	Item (String)	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

Task 4: Configuring notifications

On the **Services** menu, choose **Simple Notification Service**.

Amazon Simple Notification Service
Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

Create topic

Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

MyTopic

Next step

Start with an overview

Pricing

Amazon SNS has no upfront costs. You pay based on

In the **Create topic** box, for **Topic name**, enter: NoStock. Keep **Standard** selected. Choose **Create topic**

aws Services Search [Option+S] N. Virginia voclabs/user2770782=104222099@student.swin.edu.au @ 3101-3158...

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

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - optional [Info](#)

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

My Topic

Maximum 100 characters.

In the lower half of the page, choose **Create subscription** and configure these settings:

- **Protocol:** *Email*
- **Endpoint:** Enter your email address
- Choose **Create subscription**

aws Services Search [Option+S] N. Virginia voclabs/user2770782=104222099@student.swin.edu.au @ 3101-3158...

NoStock Edit Delete Publish message

Details

Name	NoStock	Display name	-
ARN	arn:aws:sns:us-east-1:310131586881:NoStock	Topic owner	310131586881
Type	Standard		

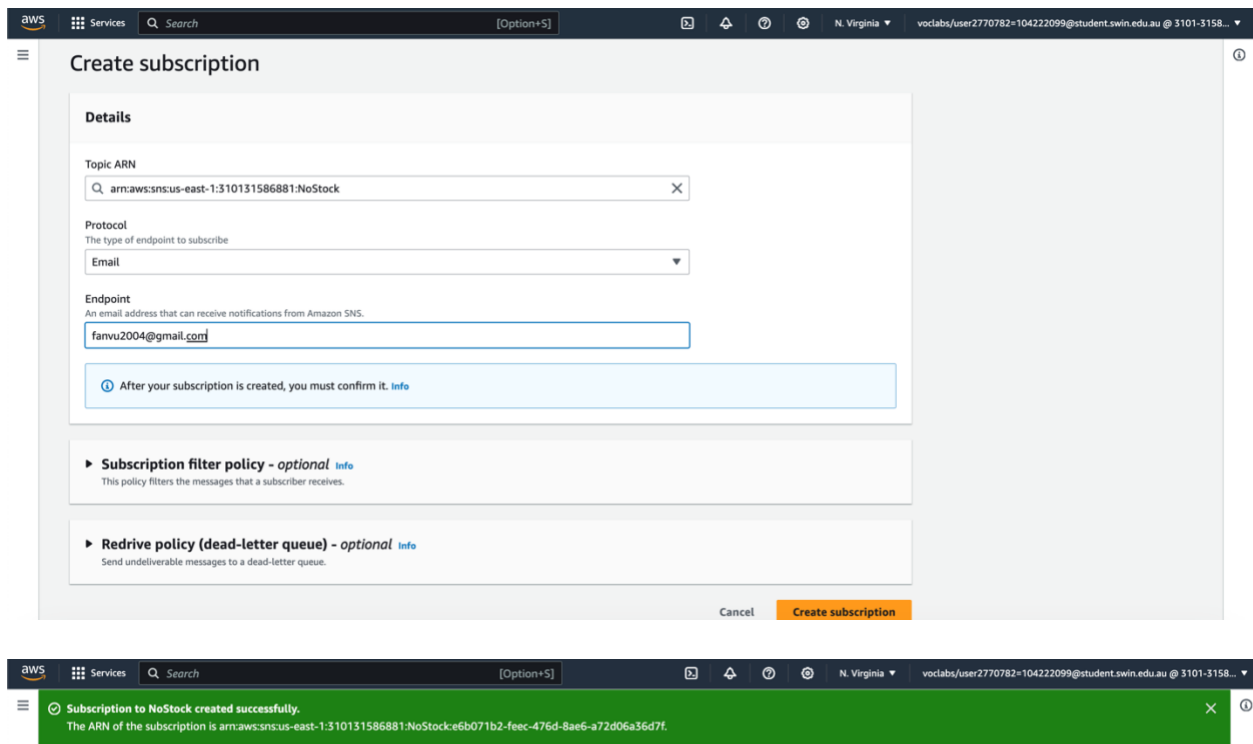
[Subscriptions](#)
[Access policy](#)
[Data protection policy](#)
[Delivery policy \(HTTP/S\)](#)
[Delivery status logging](#)
[Encryption](#)
[Tags](#)
[Integrations](#)

Subscriptions (0) Edit Delete Request confirmation Confirm subscription **Create subscription**

Search

ID	Endpoint	Status	Protocol
No subscriptions found			
You don't have any subscriptions to this topic.			

Create subscription



Task 5: Creating a Lambda function to send notifications

On the **Services** menu, choose **Lambda**.

Choose **Create function** and configure these settings:

- **Function name:** Check-Stock
- **Runtime:** *Python 3.7*
- Expand **Choose or create an execution role**.
- **Execution role:** *Use an existing role*
- **Existing role:** *Lambda-Check-Stock-Role*
- Choose **Create function**

Basic information

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

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.

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 [link](#).
☐ 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.

[View the Lambda-Check-Stock-Role role](#) on the IAM console.

Scroll down to the **Code source** section, and in the **Environment** pane, choose `lambda_function.py`.

In the code editor, delete all the code.

Copy the following code, and in the **Code Source** editor, paste the copied code:

Successfully created the function Check-Stock. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Code source Info

File Edit Find View Go Tools Window **Test** Deploy Changes not deployed

Go to Anything (⌘ P)

Environment

- Check-Stock
 - lambda_function.py

```

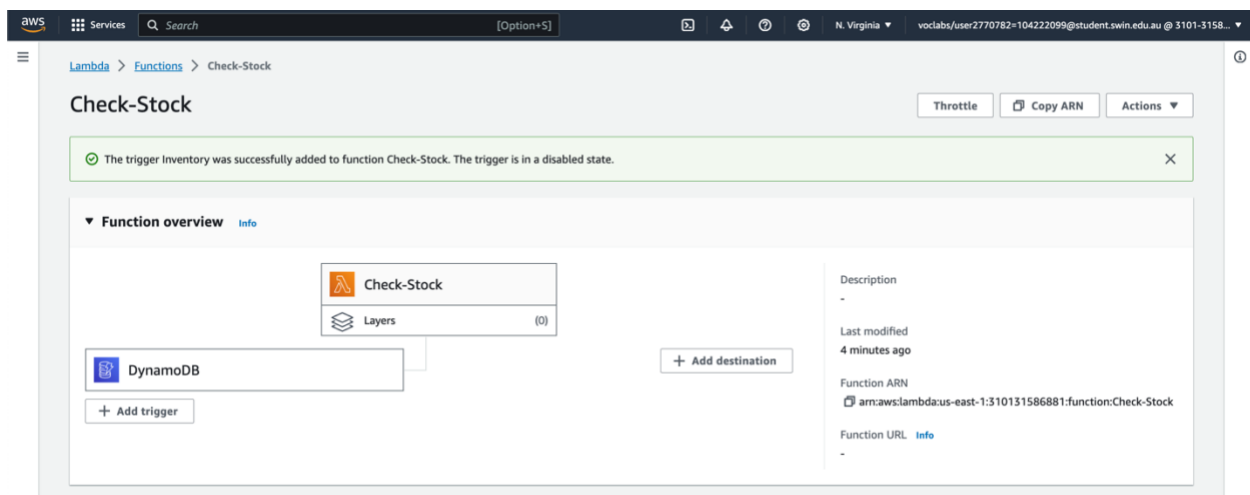
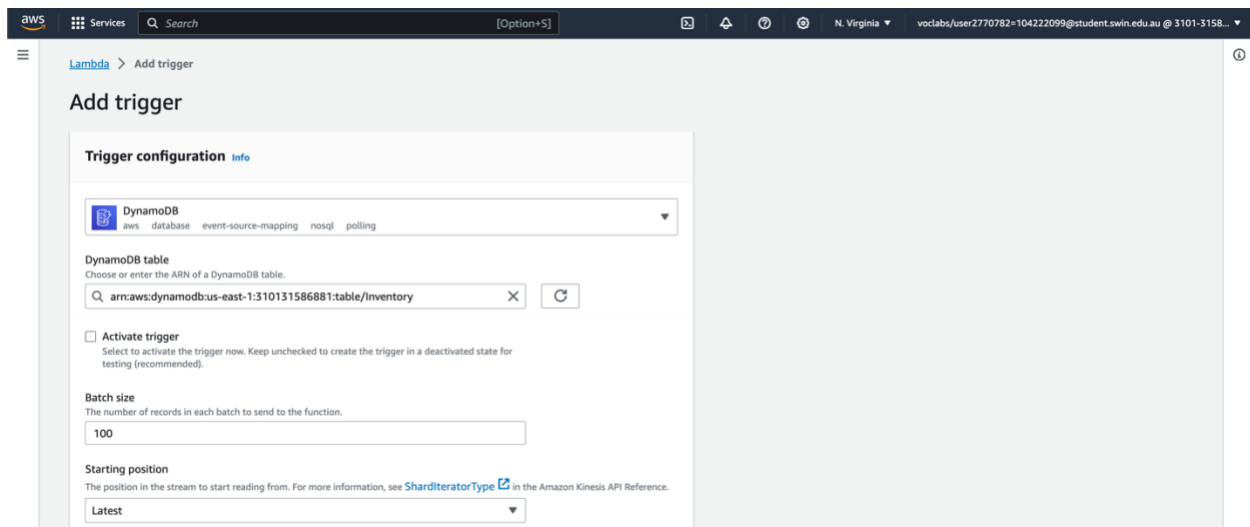
1 # Stock Check Lambda function
2 #
3 # This function is triggered when values are inserted into the Inventory DynamoDB table.
4 # Inventory counts are checked and if an item is out of stock, a notification is sent to an SNS Topic.
5 import json, boto3
6 # This handler is run every time the Lambda function is triggered
7 def lambda_handler(event, context):
8     # Show the incoming event in the debug log
9     print("Event received by Lambda function: " + json.dumps(event, indent=2))
10    # For each inventory item added, check if the count is zero
11    for record in event['Records']:
12        newImage = record['dynamodb'].get('NewImage', None)
13        if newImage:
14            count = int(record['dynamodb']['NewImage']['Count']['N'])
15            if count == 0:
16                store = record['dynamodb']['NewImage']['Store']['S']
17                item = record['dynamodb']['NewImage']['Item']['S']
18                # Construct message to be sent
19                message = store + ' is out of stock of ' + item
20                print(message)
21                # Connect to SNS
22                sns = boto3.client('sns')
23                alertTopic = 'NoStock'
24                snsTopicArn = [t['TopicArn'] for t in sns.list_topics()['Topics']
25                             if t['TopicArn'].lower().endswith(':' + alertTopic.lower())][0]
26                # Send message to SNS
27                sns.publish(
28                    TopicArn=snsTopicArn,
29                    Message=message,
30                    Subject='Inventory Alert'
  
```

Choose **Deploy** to save your code changes

Scroll to the **Designer** section (which is at the top of the page).

Choose **Add trigger** and then configure these settings:

- **Select a trigger:** *DynamoDB*
- **DynamoDB Table:** *Inventory*
- Choose **Add**

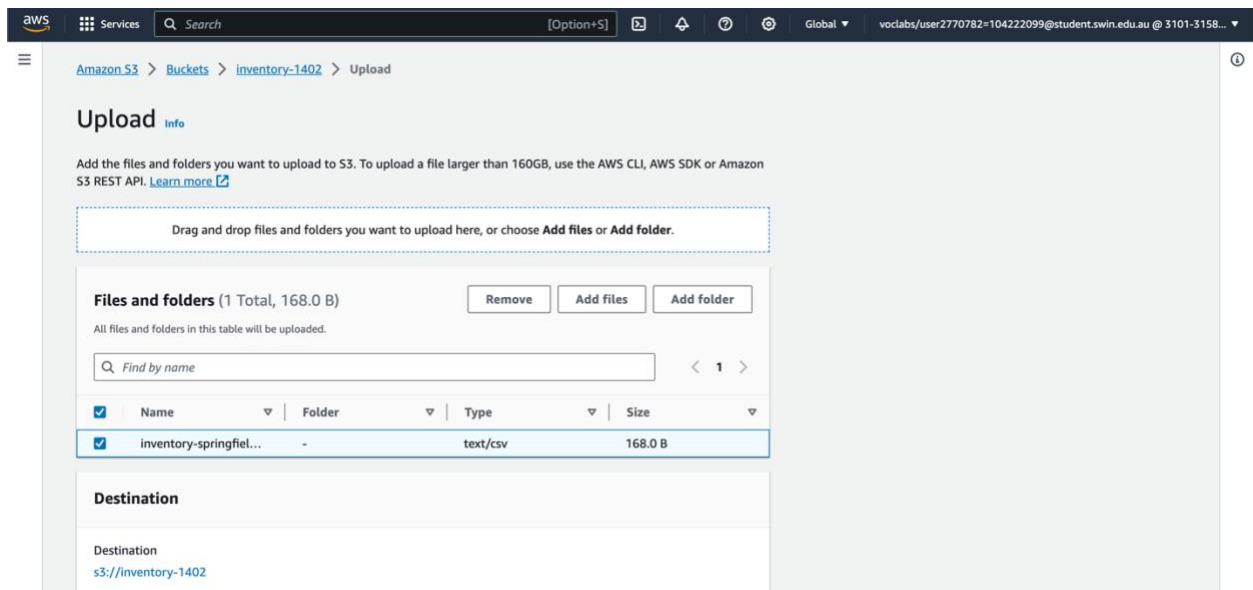


Task 6: Testing the System

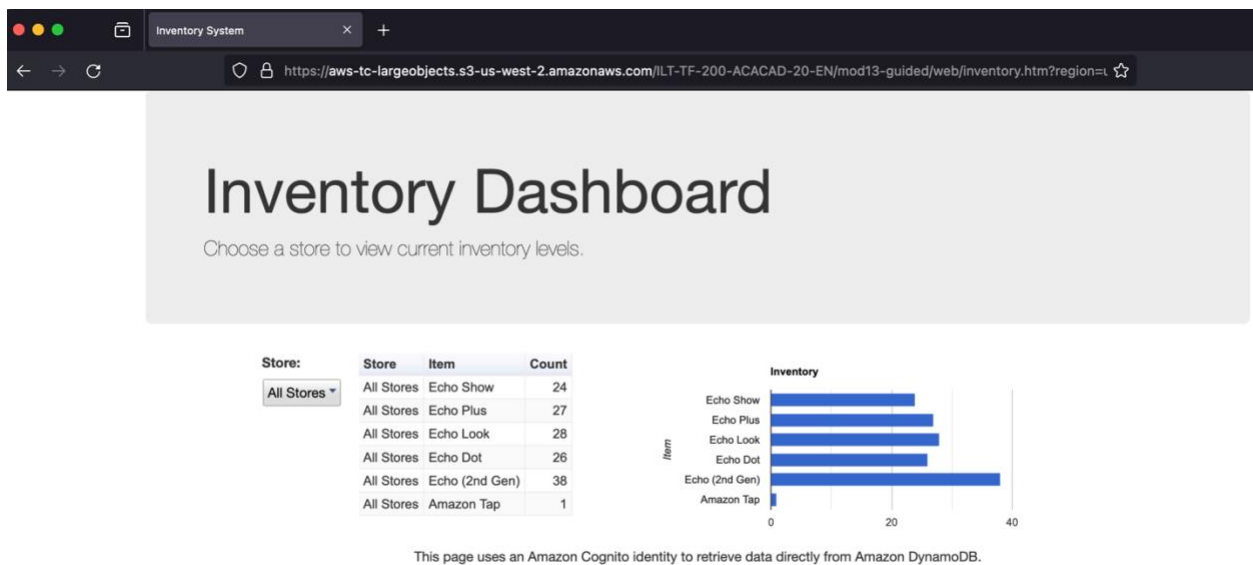
On the **Services** menu, choose **S3**.

Choose the name of your *inventory*- bucket.

Choose **Upload** and upload a different inventory file



Return to the **Inventory System Dashboard** and refresh the page.



Submitting work

SubmitDetailsAWSStart LabEnd Lab2:14InstructionsGradesActions

EN_US

Submitting your work

52. At the top of these instructions, choose **Submit** to record your progress and when prompted, choose **Yes**.

53. If the results don't display after a couple of minutes, return to the top of these instructions and choose **Grades**

Tip: You can submit your work multiple times. After you change your work, choose **Submit** again. Your

bash
eee_W_2451511@runweb101309:~\$

Total score40/40

[Task 1A] Load-Inventory Function Crea

[Task 1B] Load-Inventory Execution Rol

[Task 2A] Bucket Created

[Task 2B] Bucket Event

[Task 3] Database Table Data

ENDLAB.