ACA Module 13 Guide Lab Implementing a Serverless Architecture with AWS Lambda

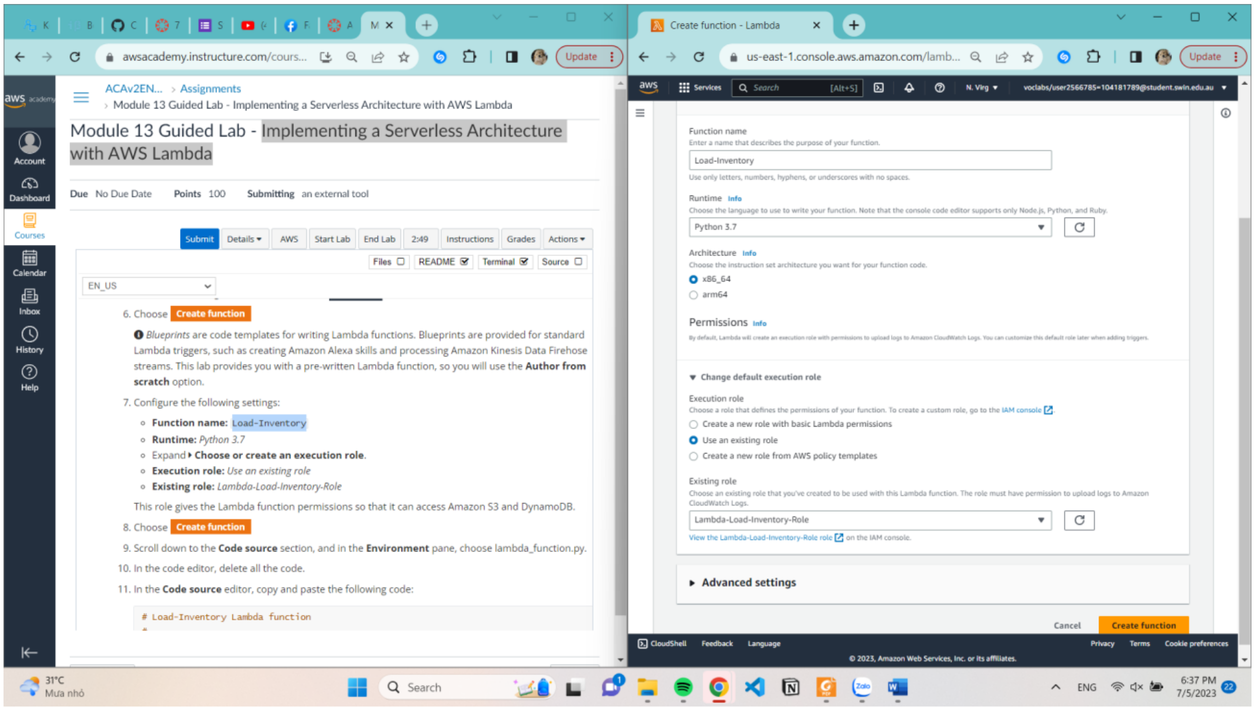
COS 20019- Cloud Computing Architecture

Nguyen Manh Dung

20/5/2023

Below screenshotted is all the step I need to finish ACA Module 13 (a screenshot also include requirements)

Task 1: Creating a Lambda function to load data



ACAv2EN... › Assignments › Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda

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

Due No Due Date **Points** 100 **Submitting** an external tool

6. Choose **Create function**

   Blueprints are code templates for writing Lambda functions. Blueprints are provided for standard Lambda triggers, such as creating Amazon Alexa skills and processing Amazon Kinesis Data Firehose streams. This lab provides you with a pre-written Lambda function, so you will use the **Author from scratch** option.

7. 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*

   This role gives the Lambda function permissions so that it can access Amazon S3 and DynamoDB.

8. Choose **Create function**
9. Scroll down to the **Code source** section, and in the **Environment** pane, choose lambda_function.py.
10. In the code editor, delete all the code.
11. In the **Code source** editor, copy and paste the following code:

```
# Load-Inventory Lambda function
#
```

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

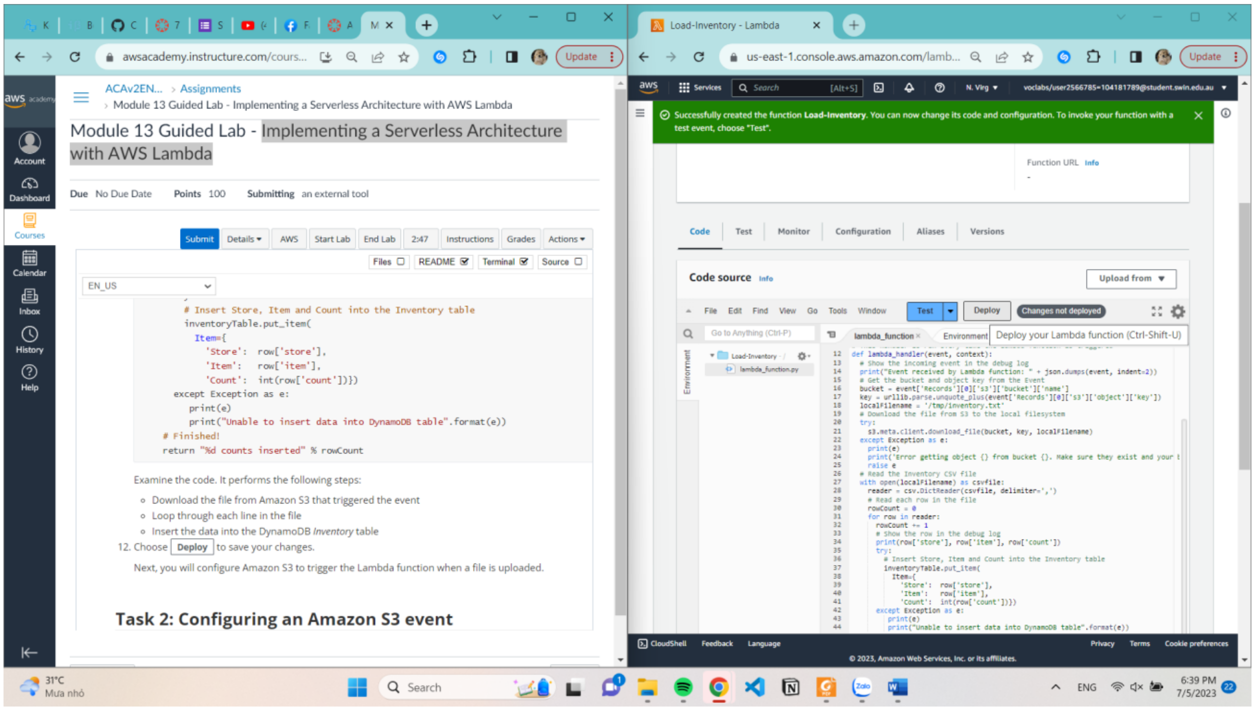
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
○ Create a new role with basic Lambda permissions
● Use an existing role
○ Create a new role from AWS policy templates

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

▸ Advanced settings

Cancel | Create function

© 2023, Amazon Web Services, Inc. or its affiliates.



ACAv2EN... › Assignments › Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda

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

Due No Due Date **Points** 100 **Submitting** an external tool

```
      # 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
```

Examine the code. It performs the following steps:

- Download the file from Amazon S3 that triggered the event
- Loop through each line in the file
- Insert the data into the DynamoDB *Inventory* table

12. Choose **Deploy** to save your changes.

Next, you will configure Amazon S3 to trigger the Lambda function when a file is uploaded.

Task 2: Configuring an Amazon S3 event

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

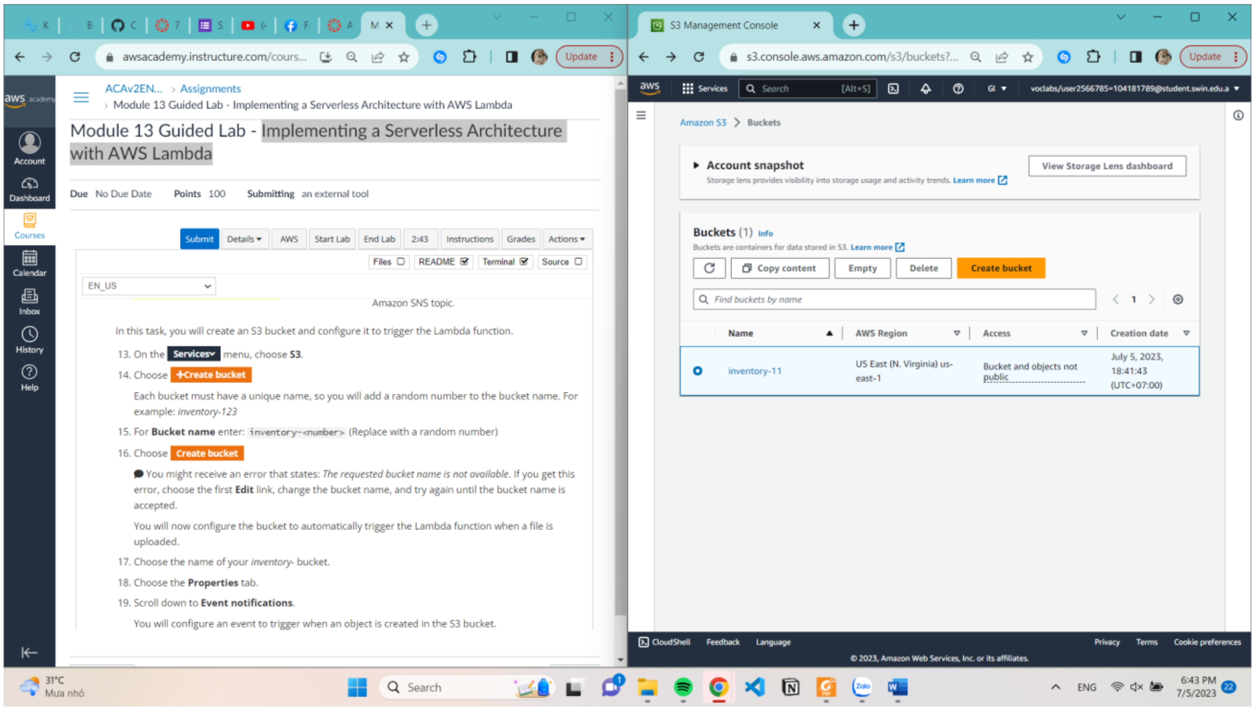
Function URL Info
-

Code source Info

```
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 t
    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))
```

© 2023, Amazon Web Services, Inc. or its affiliates.

Task 2: Configuring an Amazon S3 event



> Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda
>
> Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda
>
> Due No Due Date Points 100 Submitting an external tool
>
> Amazon SNS topic.
>
> In this task, you will create an S3 bucket and configure it to trigger the Lambda function.
>
> 13. On the **Services** menu, choose **S3**.
> 14. Choose **+Create bucket**
>
>   Each bucket must have a unique name, so you will add a random number to the bucket name. For example: *inventory-123*
> 15. For **Bucket name** enter: `inventory-<number>` (Replace with a random number)
> 16. Choose **Create bucket**
>
>   You might receive an error that states: *The requested bucket name is not available*. If you get this error, choose the first **Edit** link, change the bucket name, and try again until the bucket name is accepted.
>
>   You will now configure the bucket to automatically trigger the Lambda function when a file is uploaded.
> 17. Choose the name of your *inventory-* bucket.
> 18. Choose the **Properties** tab.
> 19. Scroll down to **Event notifications**.
>
>   You will configure an event to trigger when an object is created in the S3 bucket.

Account snapshot

Storage lens provides visibility into storage usage and activity trends. Learn more

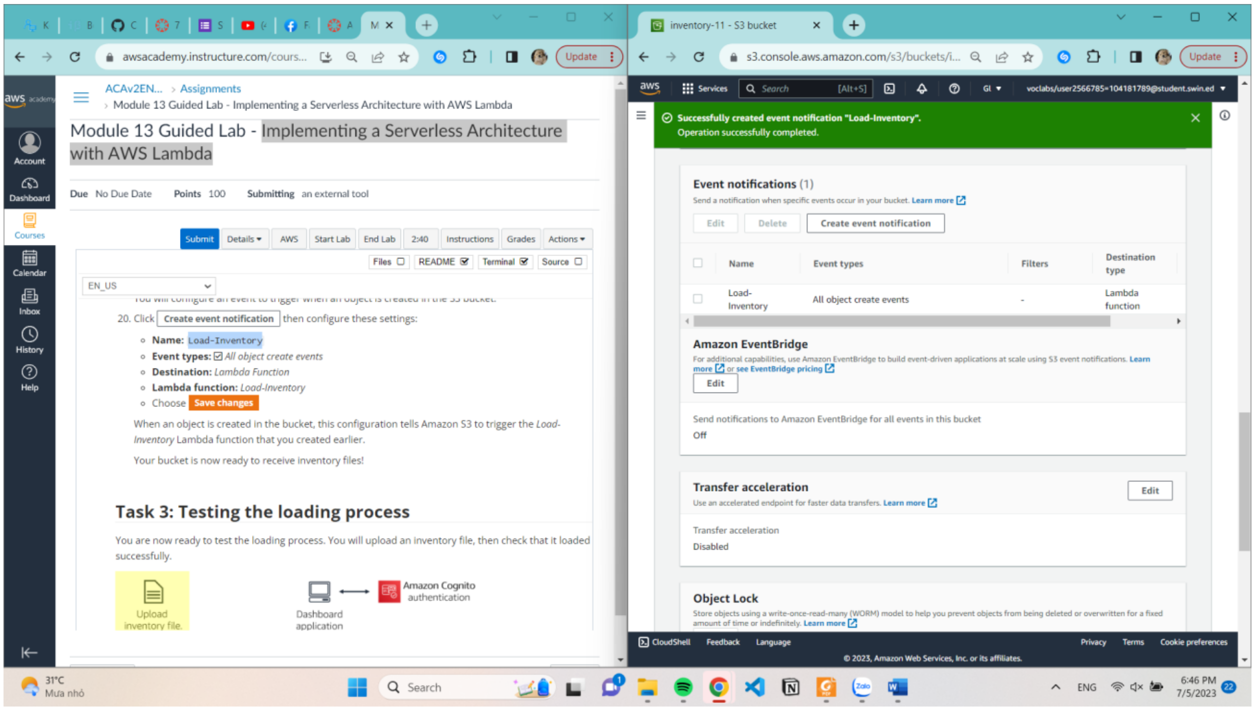
Buckets (1) Info

Buckets are containers for data stored in S3. Learn more

	Name	AWS Region	Access	Creation date
●	inventory-11	US East (N. Virginia) us-east-1	Bucket and objects not public	July 5, 2023, 18:41:43 (UTC+07:00)

© 2023, Amazon Web Services, Inc. or its affiliates.

Successful create event notifications



> Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda
>
> Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda
>
> Due No Due Date Points 100 Submitting an external tool
>
> You will configure an event to trigger when an object is created in the S3 bucket.
>
> 20. 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**
>
>   When an object is created in the bucket, this configuration tells Amazon S3 to trigger the *Load-Inventory* Lambda function that you created earlier.
>
>   Your bucket is now ready to receive inventory files!
>
> **Task 3: Testing the loading process**
>
> You are now ready to test the loading process. You will upload an inventory file, then check that it loaded successfully.

Successfully created event notification "Load-Inventory".
Operation successfully completed.

Event notifications (1)

Send a notification when specific events occur in your bucket. Learn more

☐	Name	Event types	Filters	Destination type
☐	Load-Inventory	All object create events	-	Lambda function

Amazon EventBridge

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

Edit

Send notifications to Amazon EventBridge for all events in this bucket
Off

Transfer acceleration

Use an accelerated endpoint for faster data transfers. Learn more

Edit

Transfer acceleration
Disabled

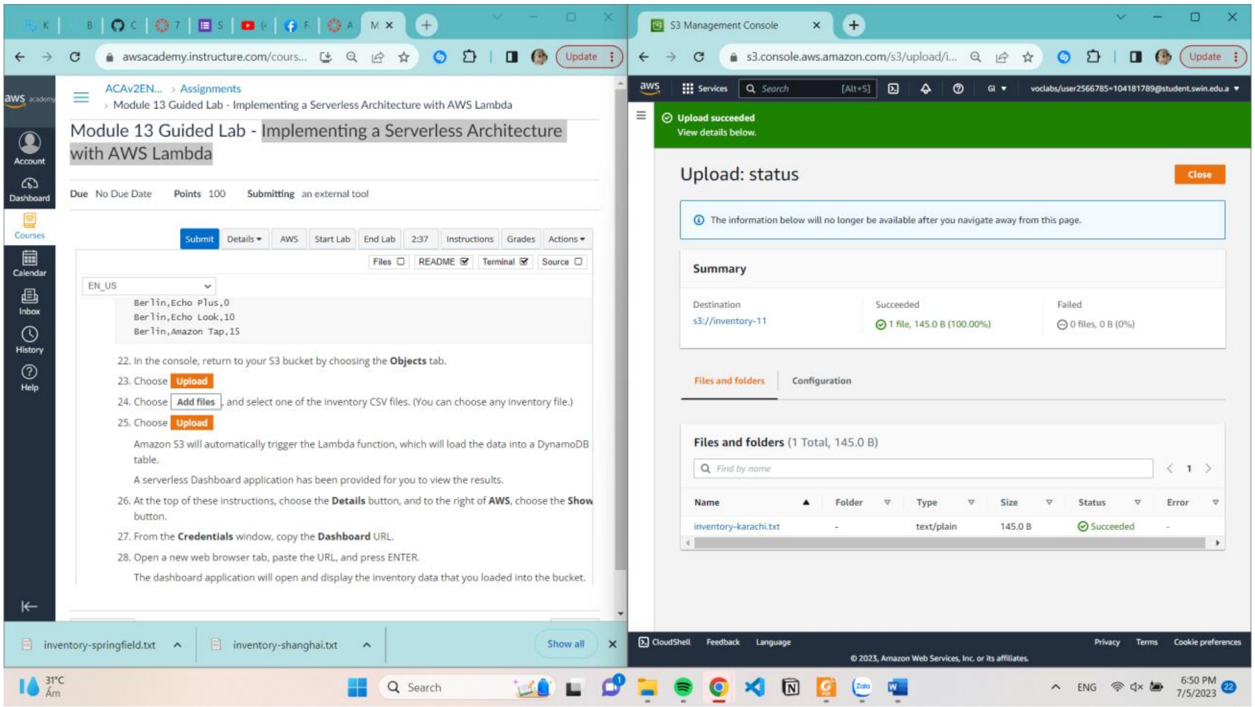
Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Learn more

© 2023, Amazon Web Services, Inc. or its affiliates.

Task 3: Testing the loading process

Upload inventory file



awsacademy.instructure.com/cours...

ACAv2EN... › Assignments › Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda

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

Due No Due Date **Points** 100 **Submitting** an external tool

Submit | Details ▾ | AWS | Start Lab | End Lab | 2:37 | Instructions | Grades | Actions ▾

Files ☐ README ☑ Terminal ☑ Source ☐

EN_US

```
Berlin,Echo Plus,0
Berlin,Echo Look,10
Berlin,Amazon Tap,15
```

22. In the console, return to your S3 bucket by choosing the **Objects** tab.
23. Choose **Upload**
24. Choose **Add files**, and select one of the inventory CSV files. (You can choose any inventory file.)
25. Choose **Upload**

    Amazon S3 will automatically trigger the Lambda function, which will load the data into a DynamoDB table.

    A serverless Dashboard application has been provided for you to view the results.
26. At the top of these instructions, choose the **Details** button, and to the right of **AWS**, choose the **Show** button.
27. From the **Credentials** window, copy the **Dashboard** URL.
28. Open a new web browser tab, paste the URL, and press ENTER.

    The dashboard application will open and display the inventory data that you loaded into the bucket.

inventory-springfield.txt | inventory-shanghai.txt | Show all

S3 Management Console

s3.console.aws.amazon.com/s3/upload/i...

voclabs/user2566785=104181789@student.swin.edu.a

Upload succeeded
View details below.

Upload: status

Close

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

Summary

Destination	Succeeded	Failed
s3://inventory-11	1 file, 145.0 B (100.00%)	0 files, 0 B (0%)

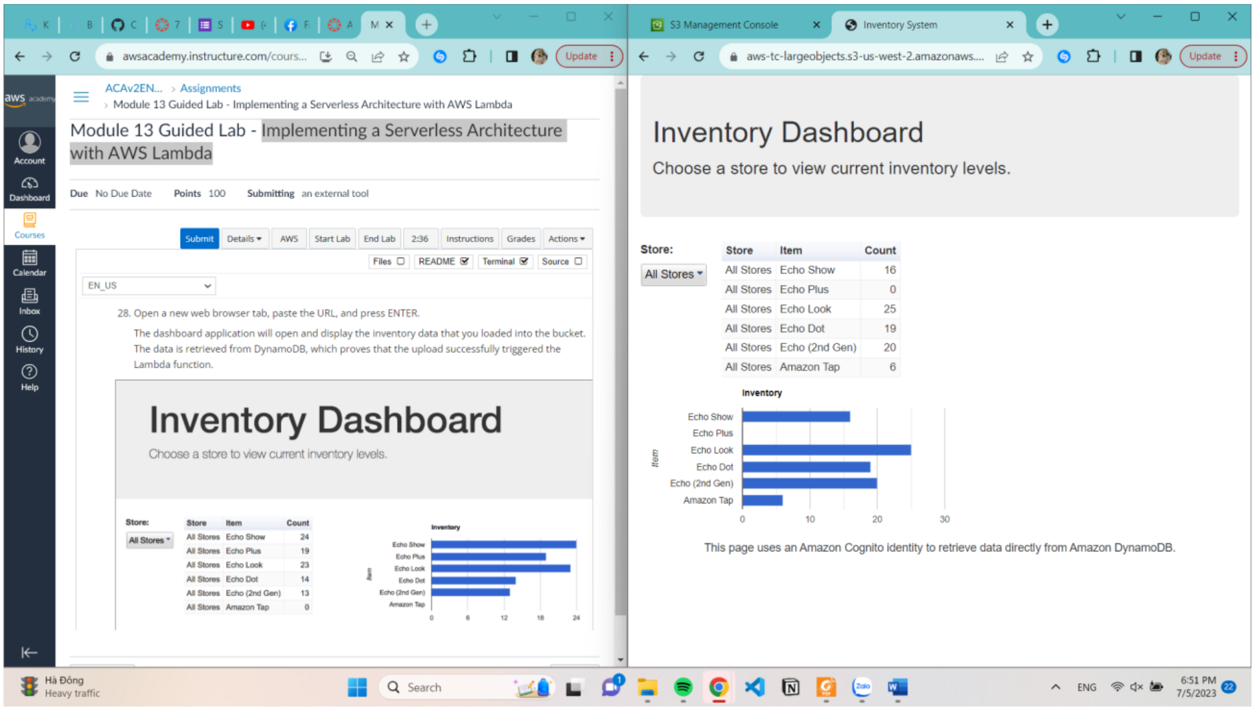
Files and folders | Configuration

Files and folders (1 Total, 145.0 B)

Name	Folder	Type	Size	Status	Error
inventory-karachi.txt	-	text/plain	145.0 B	✓ Succeeded	-

© 2023, Amazon Web Services, Inc. or its affiliates.

Inventory Dashboard



awsacademy.instructure.com/cours...

ACAv2EN... › Assignments › Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda

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

Due No Due Date **Points** 100 **Submitting** an external tool

Submit | Details ▾ | AWS | Start Lab | End Lab | 2:36 | Instructions | Grades | Actions ▾

Files ☐ README ☑ Terminal ☑ Source ☐

EN_US

28. Open a new web browser tab, paste the URL, and press ENTER.

    The dashboard application will open and display the inventory data that you loaded into the bucket. The data is retrieved from DynamoDB, which proves that the upload successfully triggered the Lambda function.

Inventory Dashboard

Choose a store to view current inventory levels.

Store: All Stores ▾

Store	Item	Count
All Stores	Echo Show	24
All Stores	Echo Plus	19
All Stores	Echo Look	23
All Stores	Echo Dot	14
All Stores	Echo (2nd Gen)	13
All Stores	Amazon Tap	0

S3 Management Console | Inventory System

aws-tc-largeobjects.s3-us-west-2.amazonaws....

Inventory Dashboard

Choose a store to view current inventory levels.

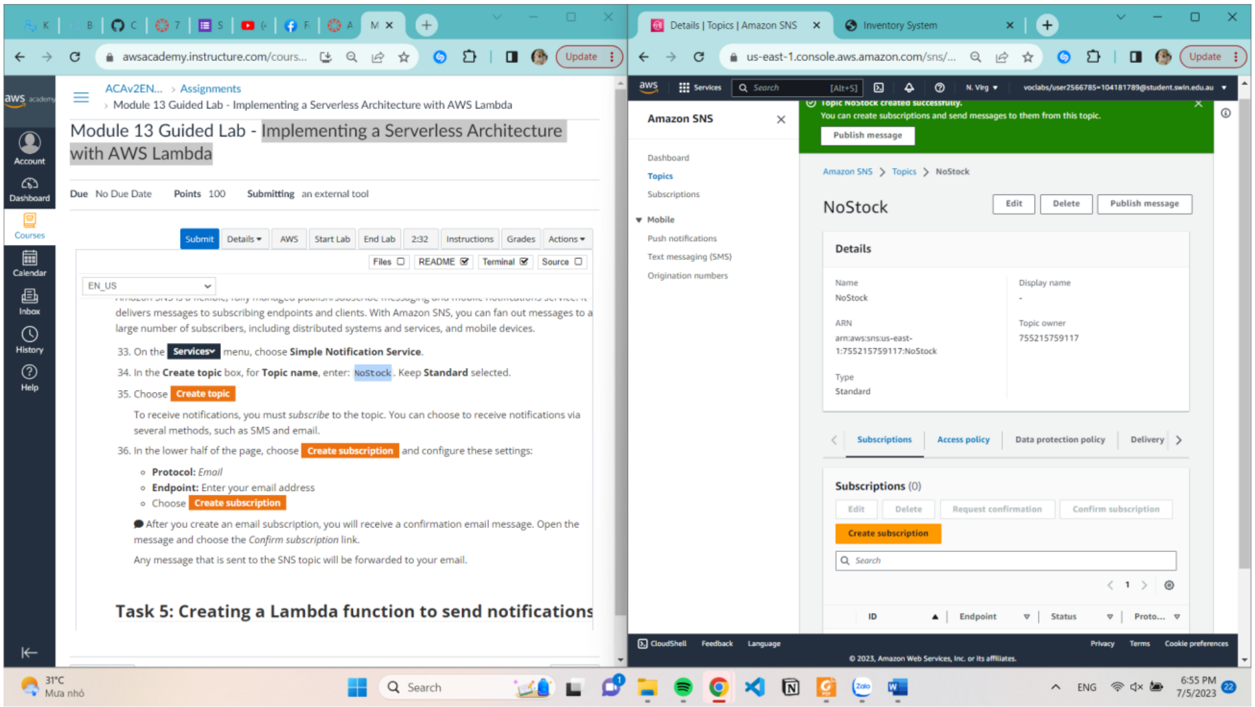
Store: All Stores ▾

Store	Item	Count
All Stores	Echo Show	16
All Stores	Echo Plus	0
All Stores	Echo Look	25
All Stores	Echo Dot	19
All Stores	Echo (2nd Gen)	20
All Stores	Amazon Tap	6

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

Task 4: Configuring notifications

Create topic



ACAv2EN... > Assignments > Module 13 Guided Lab - Implementing a Serverless Architecture with AWS Lambda

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

Due No Due Date Points 100 Submitting an external tool

delivers messages to subscribing endpoints and clients. With Amazon SNS, you can fan out messages to a large number of subscribers, including distributed systems and services, and mobile devices.

33. On the Services menu, choose **Simple Notification Service**.
34. In the **Create topic** box, for **Topic name**, enter: `NoStock`. Keep **Standard** selected.
35. Choose Create topic

To receive notifications, you must *subscribe* to the topic. You can choose to receive notifications via several methods, such as SMS and email.

36. In the lower half of the page, choose Create subscription and configure these settings:
    - **Protocol:** *Email*
    - **Endpoint:** Enter your email address
    - Choose Create subscription

After you create an email subscription, you will receive a confirmation email message. Open the message and choose the *Confirm subscription* link.

Any message that is sent to the SNS topic will be forwarded to your email.

Task 5: Creating a Lambda function to send notifications

Topic NoStock created successfully.
You can create subscriptions and send messages to them from this topic.

Amazon SNS > Topics > NoStock

NoStock

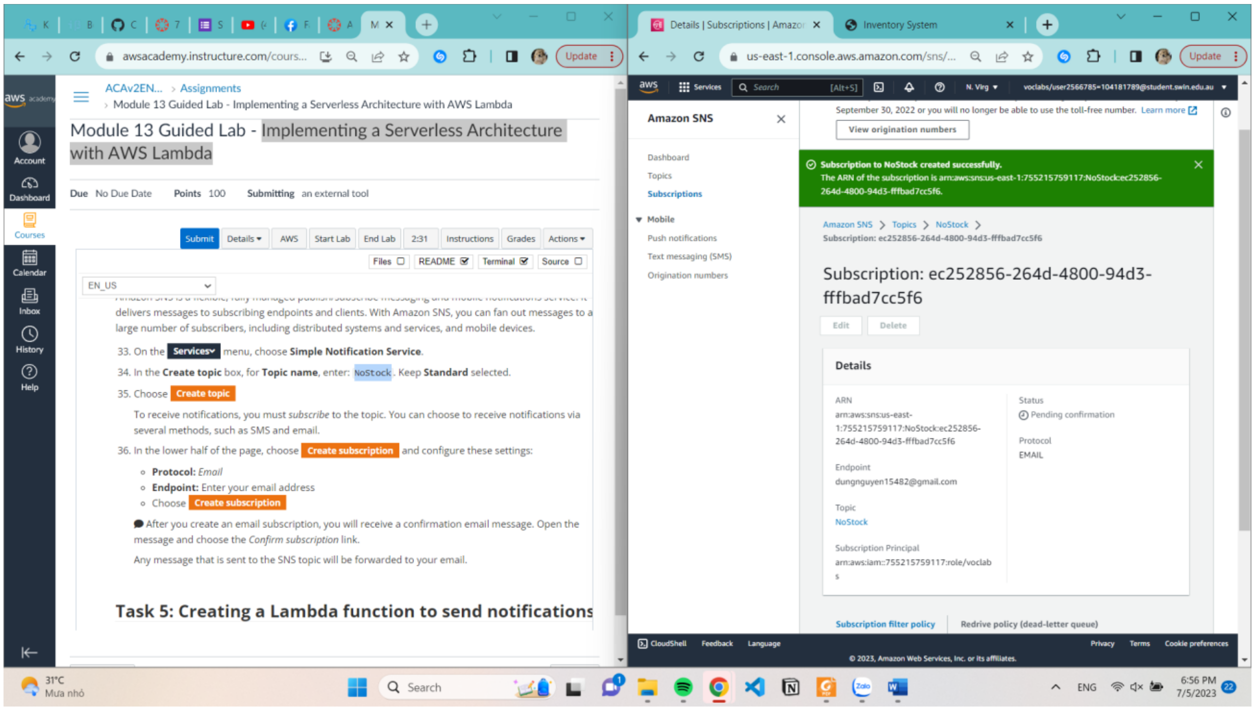
Details

Name: NoStock
Display name: -
ARN: arn:aws:sns:us-east-1:755215759117:NoStock
Topic owner: 755215759117
Type: Standard

Subscriptions (0)

ID	Endpoint	Status	Proto...

Create subscription



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

33. On the Services menu, choose **Simple Notification Service**.
34. In the **Create topic** box, for **Topic name**, enter: `NoStock`. Keep **Standard** selected.
35. Choose Create topic
36. In the lower half of the page, choose Create subscription and configure these settings:

September 30, 2022 or you will no longer be able to use the toll-free number. Learn more

Subscription to NoStock created successfully.
The ARN of the subscription is arn:aws:sns:us-east-1:755215759117:NoStock:ec252856-264d-4800-94d3-fffbad7cc5f6.

Amazon SNS > Topics > NoStock > Subscription: ec252856-264d-4800-94d3-fffbad7cc5f6

Subscription: ec252856-264d-4800-94d3-fffbad7cc5f6

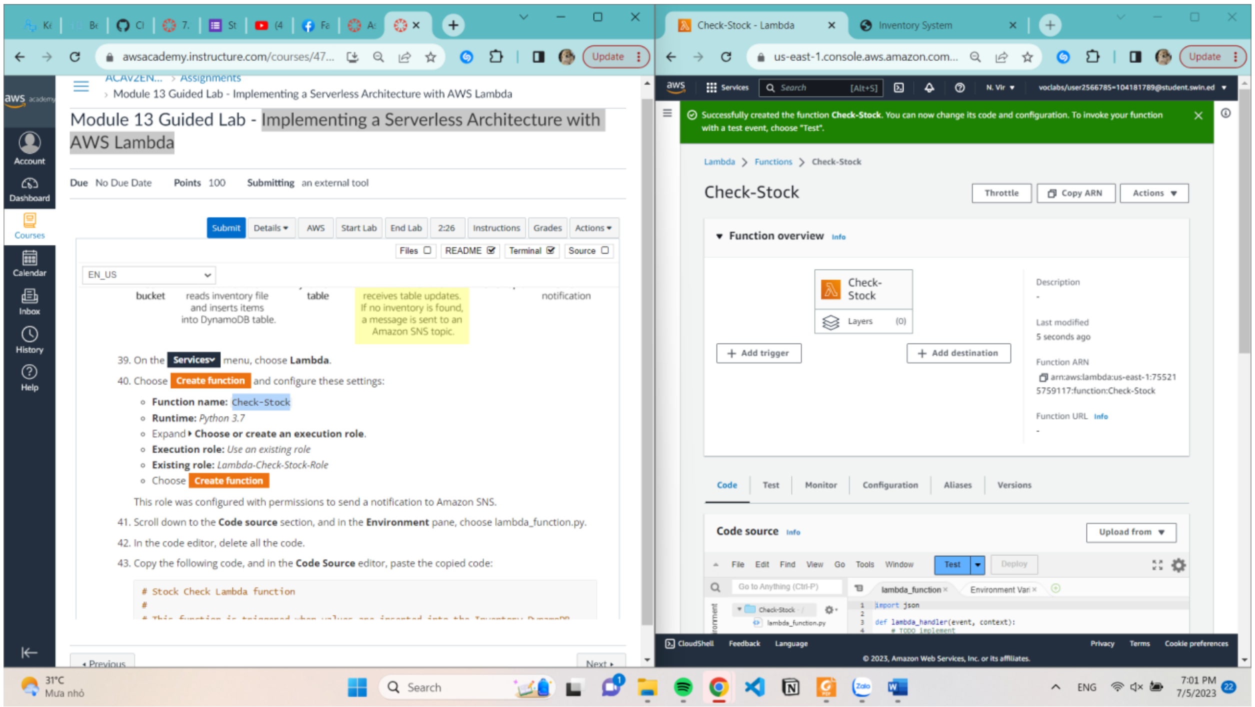
Details

ARN: arn:aws:sns:us-east-1:755215759117:NoStock:ec252856-264d-4800-94d3-fffbad7cc5f6
Status: Pending confirmation
Protocol: EMAIL
Endpoint: dungnguyen15482@gmail.com
Topic: NoStock
Subscription Principal: arn:aws:iam::755215759117:role/voclabs

View origination numbers
Edit
Delete
Subscription filter policy
Redrive policy (dead-letter queue)
6:56 PM
7/5/2023

Task 5: Creating a Lambda function to send notifications

Create Lambda function



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

Due No Due Date Points 100 Submitting an external tool

bucket reads inventory file and inserts items into DynamoDB table. table receives table updates. If no inventory is found, a message is sent to an Amazon SNS topic. notification

39. On the Services menu, choose **Lambda**.
40. 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

    This role was configured with permissions to send a notification to Amazon SNS.
41. Scroll down to the **Code source** section, and in the **Environment** pane, choose lambda_function.py.
42. In the code editor, delete all the code.
43. Copy the following code, and in the **Code Source** editor, paste the copied code:

```
# Stock Check Lambda function
#
```

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

Lambda > Functions > Check-Stock

Check-Stock

Function overview

Description
-

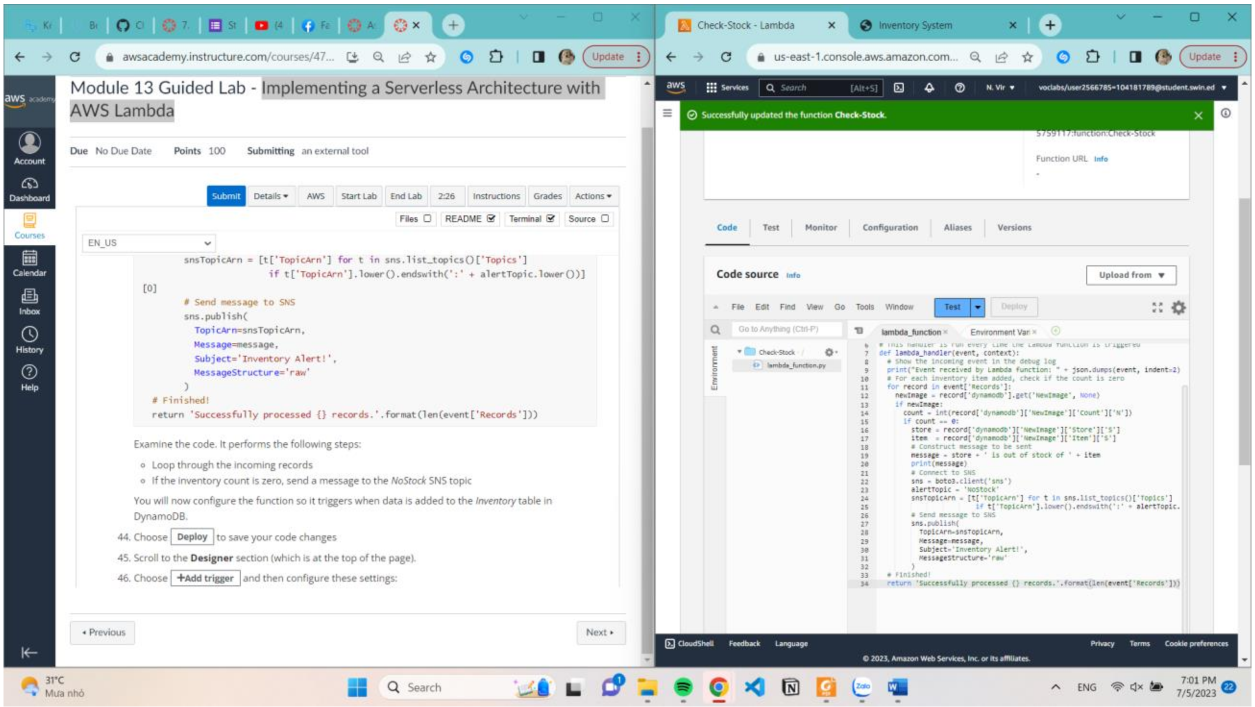
Last modified
5 seconds ago

Function ARN
arn:aws:lambda:us-east-1:755215759117:function:Check-Stock

Function URL
-

```
import json

def lambda_handler(event, context):
    # TODO implement
```



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

Due No Due Date Points 100 Submitting an external tool

```
        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']))
```

Examine the code. It performs the following steps:

- Loop through the incoming records
- If the inventory count is zero, send a message to the *NoStock* SNS topic

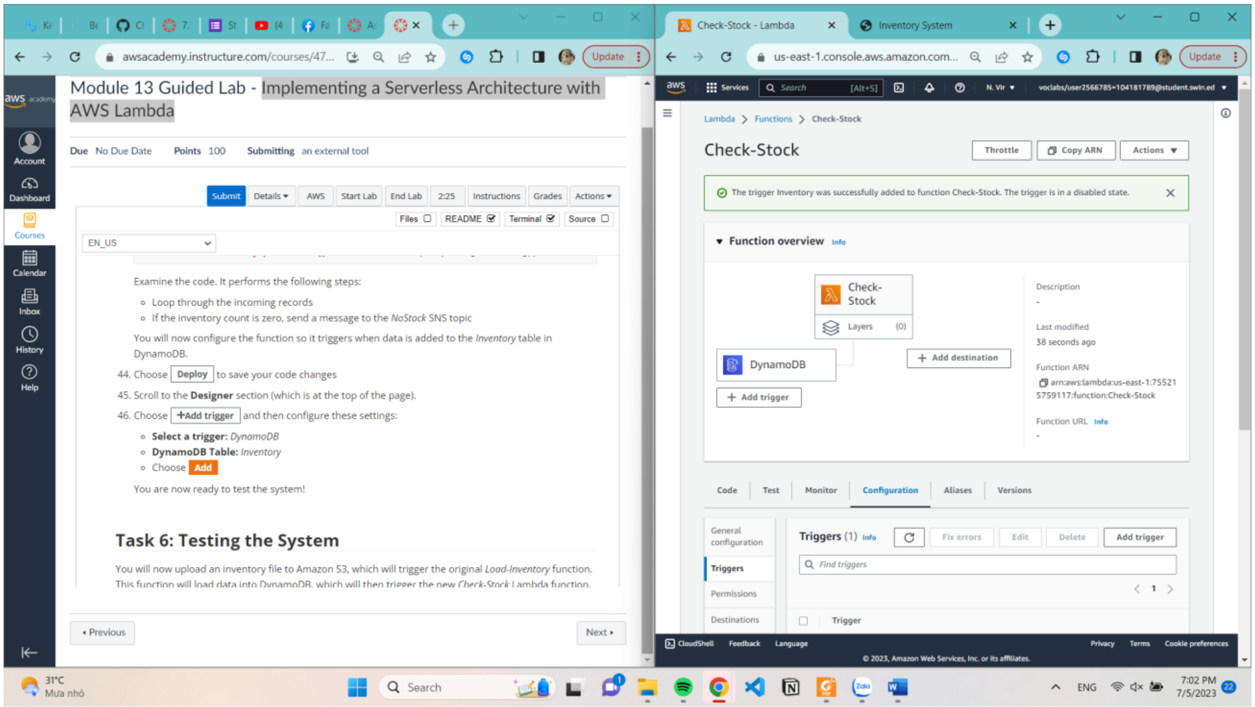
You will now configure the function so it triggers when data is added to the *Inventory* table in DynamoDB.

44. Choose Deploy to save your code changes
45. Scroll to the **Designer** section (which is at the top of the page).
46. Choose +Add trigger and then configure these settings:

Successfully updated the function **Check-Stock**.

```
# 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.
        # Send message to SNS
        sns.publish(
          TopicArn=snsTopicArn,
          Message=message,
          Subject='Inventory Alert!',
          MessageStructure='raw'
        )
  # Finished!
  return 'Successfully processed {} records.'.format(len(event['Records']))
```

Adding trigger



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

Due No Due Date **Points** 100 **Submitting** an external tool

Examine the code. It performs the following steps:

- Loop through the incoming records
- If the inventory count is zero, send a message to the *NoStock* SNS topic

You will now configure the function so it triggers when data is added to the *Inventory* table in DynamoDB.

44. Choose Deploy to save your code changes
45. Scroll to the **Designer** section (which is at the top of the page).
46. Choose +Add trigger and then configure these settings:
    - **Select a trigger:** *DynamoDB*
    - **DynamoDB Table:** *Inventory*
    - Choose Add

You are now ready to test the system!

Task 6: Testing the System

You will now upload an inventory file to Amazon S3, which will trigger the original *Load-Inventory* function. This function will load data into DynamoDB, which will then trigger the new *Check-Stock* Lambda function.

Lambda > Functions > Check-Stock

Check-Stock

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

Function overview

Description: -

Last modified: 38 seconds ago

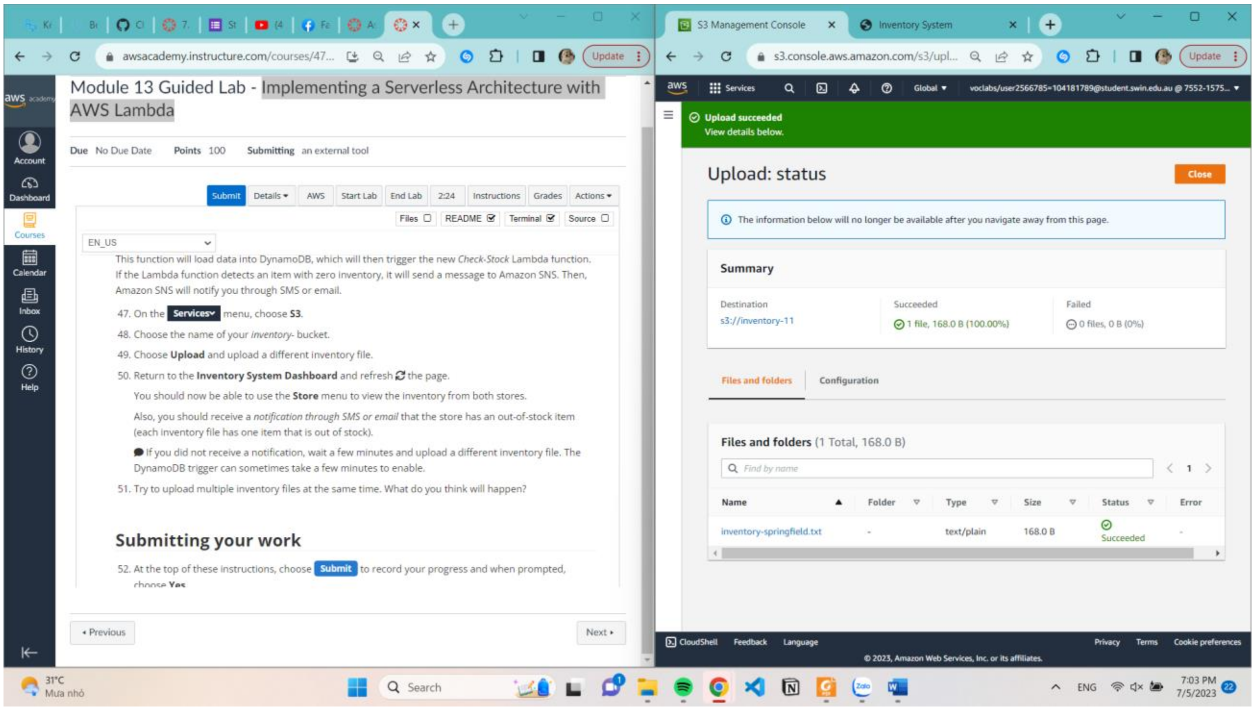
Function ARN: arn:aws:lambda:us-east-1:755215759117:function:Check-Stock

Function URL: -

Triggers (1)

Task 6: Testing the System

Upload different inventory file



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

Due No Due Date **Points** 100 **Submitting** an external tool

This function will load data into DynamoDB, which will then trigger the new *Check-Stock* Lambda function. If the Lambda function detects an item with zero inventory, it will send a message to Amazon SNS. Then, Amazon SNS will notify you through SMS or email.

47. On the Services menu, choose **S3**.
48. Choose the name of your *inventory-* bucket.
49. Choose **Upload** and upload a different inventory file.
50. Return to the **Inventory System Dashboard** and refresh the page.

    You should now be able to use the **Store** menu to view the inventory from both stores.

    Also, you should receive a *notification through SMS or email* that the store has an out-of-stock item (each inventory file has one item that is out of stock).

    If you did not receive a notification, wait a few minutes and upload a different inventory file. The DynamoDB trigger can sometimes take a few minutes to enable.
51. Try to upload multiple inventory files at the same time. What do you think will happen?

Submitting your work

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

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-11	1 file, 168.0 B (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 168.0 B)

Name	Folder	Type	Size	Status	Error
inventory-springfield.txt	-	text/plain	168.0 B	Succeeded	-