Implementing a Serverless Architecture with AWS Managed Services

Traditionally, applications run on servers. These can be physical ('bare metal') servers or virtual environments running on top of physical servers, but they still need servers to be purchased and provisioned, and for capacity to be managed. On the other hand, AWS Lambda can run serverless code without having to pre-allocate servers. Simply provide the code and define a trigger and the function can run whenever required — once per week or hundreds of times per second, and you only pay for what you use.

This lab demonstrates how to trigger an AWS Lambda function when a file is uploaded to Amazon S3. The file will be loaded into an Amazon DynamoDB table, and the data will be available for viewing on a Dashboard page that pulls the data directly from DynamoDB. The solution is **completely serverless, automatically scalable and incurs very little cost.**

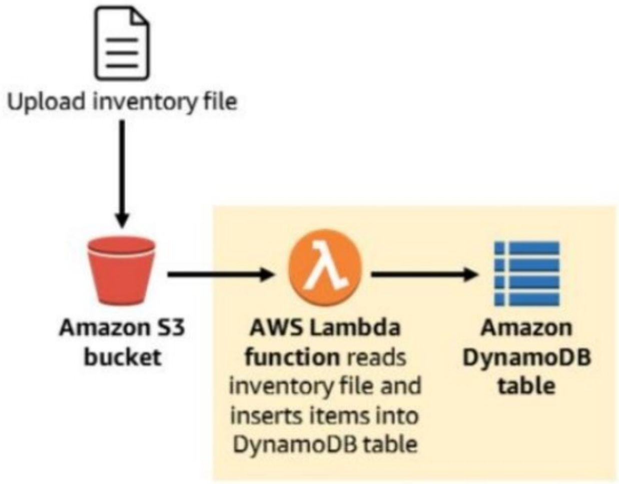
The system **does not use Amazon EC2.** The system will automatically scale when it is used and incurs practically no cost when it is not being used (just a few cents for data storage).

In this lab you will:

- Use AWS managed services to implement a serverless architecture
- Trigger AWS Lambda functions from Amazon S3 and Amazon DynamoDB
- Configure Amazon SNS to send notifications

Scenario

You are creating an inventory tracking system. Stores from around the world will upload an inventory file to Amazon S3 and your team wants to be able to view the inventory levels and send a notification when inventory levels are low.


Upload inventory file
Amazon S3 bucket
AWS Lambda function reads inventory file and inserts items into DynamoDB table
Amazon DynamoDB table

The scenario workflow is:

- You will **upload** an inventory file to an Amazon S3 bucket
- This will **trigger an AWS Lambda function** that will read the file and insert items into an **Amazon DynamoDB table**

Duration

This lab takes approximately **20 minutes** to complete.

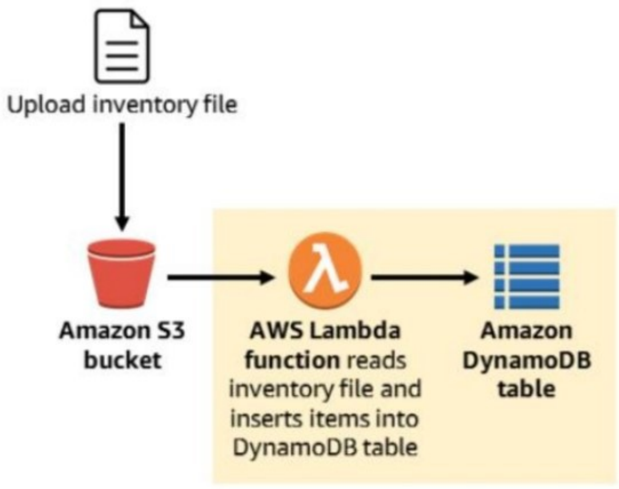
Accessing the AWS Management Console

- Click Start Lab to launch the lab.
- Click Open Console
- Sign in to the AWS Management Console using the credentials shown to the left of these instructions.

⚠ Please do not change the Region during this lab.

Task 1: Create a Lambda Function to Load Data

In this task, you will create an **AWS Lambda function** that will process an inventory file. The Lambda function will read the file and insert information into an Amazon DynamoDB table.


Upload inventory file
Amazon S3 bucket
AWS Lambda function reads inventory file and inserts items into DynamoDB table
Amazon DynamoDB table

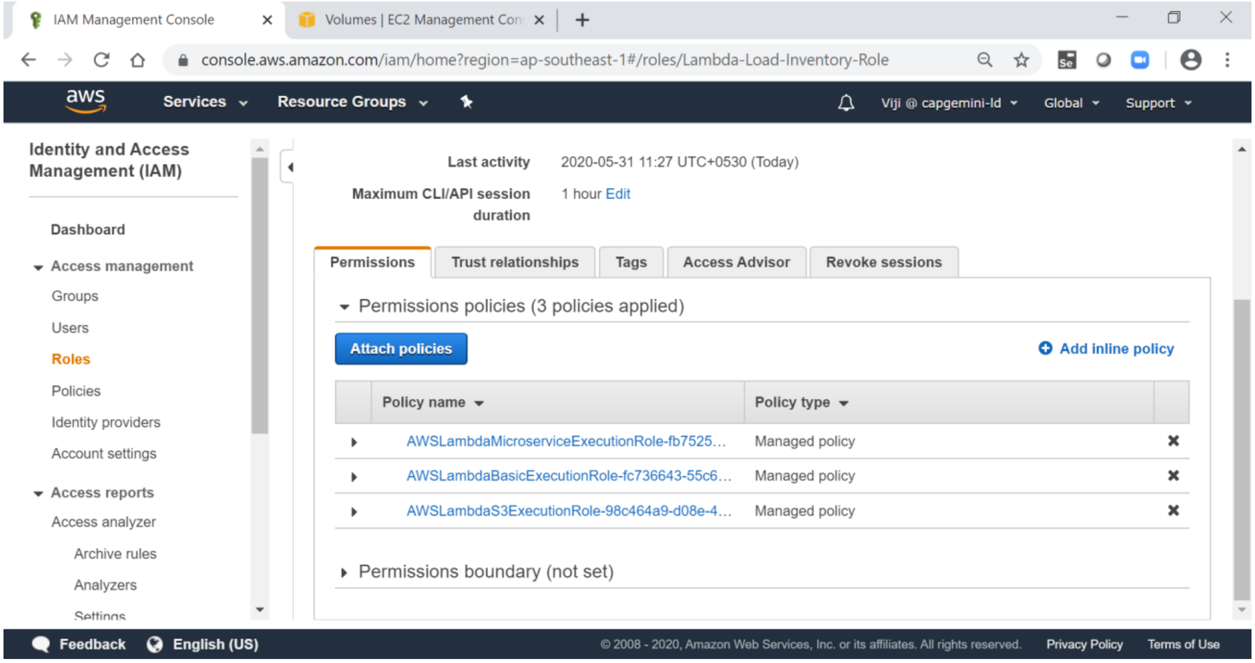
1. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.

2. Click **Create a function**

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

3. Configure the following:

- **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



Last activity 2020-05-31 11:27 UTC+0530 (Today)

Maximum CLI/API session duration 1 hour Edit

Permissions policies (3 policies applied)

Policy name	Policy type
AWSLambdaMicroserviceExecutionRole-fb7525…	Managed policy
AWSLambdaBasicExecutionRole-fc736643-55c6…	Managed policy
AWSLambdaS3ExecutionRole-98c464a9-d08e-4…	Managed policy

Permissions boundary (not set)

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

4. Click **Create function**

5. Scroll down to the **Function code** section, then delete all of the code that appears in the code editor.

6. Copy and paste the following code into the **Function code** editor:

```
# 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 executed 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(
     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 is performing the following steps:

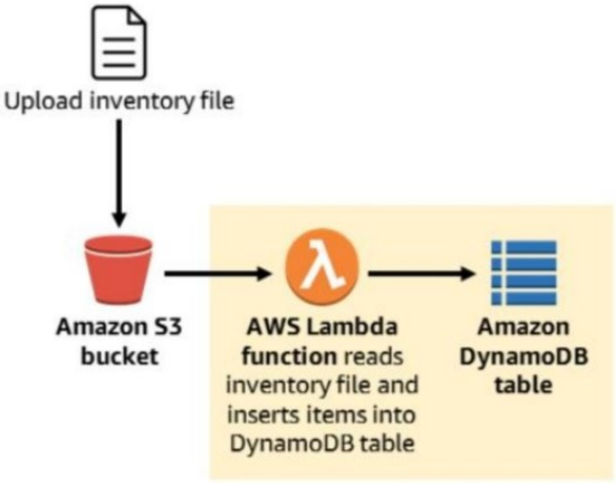
- Download the file from Amazon S3 that triggered the event
- Loop through each line in the file
- Insert the data into the **DynamoDB Inventory table**

7. Click **Save** at the top of the page.

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

Task 2: Configure an Amazon S3 Event

Stores from around the world will provide inventory files to load into the inventory tracking system. Rather than uploading files via FTP, the stores can upload directly to Amazon S3. This can be done via a web page, a script or as part of a program. Once a file is received, the AWS Lambda function will be triggered and it will load the inventory into a DynamoDB table.


Upload inventory file
Amazon S3 bucket
AWS Lambda function reads inventory file and inserts items into DynamoDB table
Amazon DynamoDB table

In this task you will create an Amazon S3 bucket and configure it to trigger the Lambda function.

8. On the **Services** menu, click **S3**.

9. Click **+ Create bucket**

Each bucket must have a unique name, so you will add a random number to the bucket name. For example: *inventory-123*

10. For **Bucket name** enter: *inventory-123* (Replacing 123 with a random number)

11. Click **Create**

💬 If you receive an error stating The **requested bucket name is not available**, then click the first **Edit** link, change the bucket name and try again until it is accepted.

You will now configure the bucket to automatically trigger the Lambda function whenever a file is uploaded.

12. Click the name of your *inventory-bucket.*

13. Click the **Properties** tab.

14. Scroll down to **Advanced settings**, then click **Events**. You will configure an event to trigger when an object is created in the S3 bucket.

15. Click + Add notification then configure:

- **Name**: *Load-Inventory*
- **Events**: ☑ All object create events
- **Send to: Lambda Function**
- **Lambda**: *Load-Inventory*
- Click **Save**

This will tell Amazon S3 to trigger the Load-Inventory Lambda function you created earlier whenever an object is created in the bucket.

Your bucket is now ready to receive inventory files!

Task 3: Test the Loading Process

You are now ready to test the loading process. You will upload an inventory file, then check that it loaded successfully.

16. Right-click this link to download a Zip file: **inventory-files.zip** (please take it from share folder)

17. Unzip the file.

The Zip file contains multiple inventory CSV files that you can use to test the system. Here is the contents of the Berlin file:

```
store,item,count

Berlin,Echo Dot,12

Berlin,Echo (2nd Gen),19

Berlin,Echo Show,18

Berlin,Echo Plus,0

Berlin,Echo Look,10

Berlin,Amazon Tap,15
```

18. Return to your S3 bucket in the console by clicking the **Overview** tab.

19. Click **Upload** and upload one of the CSV files to the bucket. (You can choose any of the inventory files.)

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

Lab Complete

Congratulations! You have completed the lab.