

## **Implementación de una arquitectura sin servidor con AWS Managed Services**

Tradicionalmente, las aplicaciones se ejecutan en servidores. Estos pueden ser servidores físicos o entornos virtuales que se ejecutan sobre servidores físicos, pero aún necesitan que se compren y aprovisionar servidores, y que se pueda administrar la capacidad. Por otro lado, AWS Lambda puede ejecutar código sin servidor sin tener que preasignar servidores. Simplemente proporcione el código y defina un disparador y la función pueda ejecutarse cuando sea necesario: una vez por semana o ciento de veces por segundo, y solo paga por lo que usa.

Este laboratorio muestra cómo activar una función AWS Lambda cuando se carga un archivo en Amazon S3. El archivo se cargará en una tabla de Amazon DynamoDB, y los datos estarán disponibles para su visualización en una página del Panel que extrae los datos directamente de DynamoDB. La solución es completamente sin servidor, escalable automáticamente e incurre en muy poco costo.

El sistema no usa Amazon EC2. El sistema escalará automáticamente cuando se use e incurre en prácticamente ningún costo cuando no se usa (solo unos pocos centavos para el almacenamiento de datos).

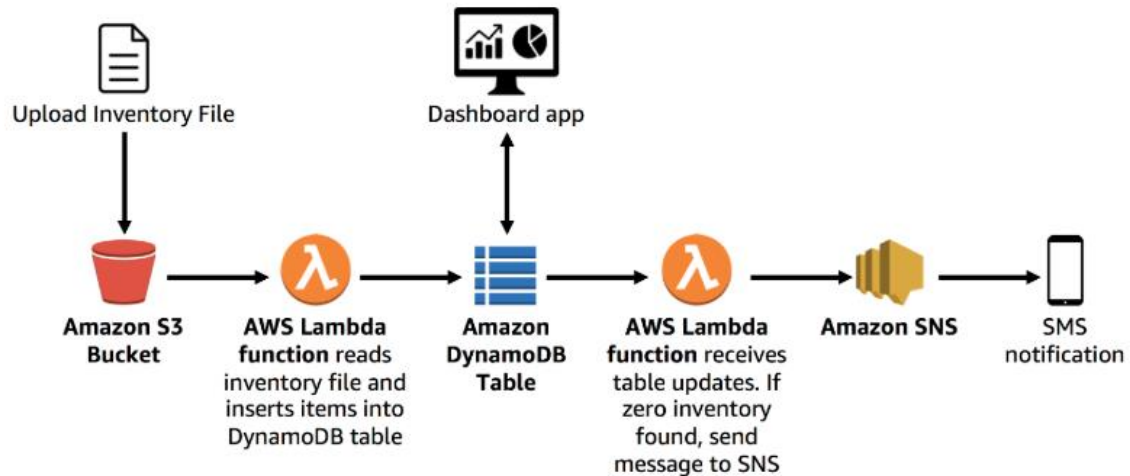
### **En este laboratorio podrás:**

- Use los servicios administrados de AWS para implementar una arquitectura sin servidor.
- Activa las funciones de AWS Lambda de Amazon S3 y DynamoDB.
- Configure Amazon SNS para enviar notificaciones.

### **Escenario:**

Estás creando un sistema de seguimiento de inventario. Las tiendas de todo el mundo subirán un archivo de inventario a Amazon S3 y su equipo desea poder ver los niveles de inventario y enviar una notificación cuando los niveles de inventario sean bajos.

Para poder entenderlo mejor se adjunta arquitectura de la solución:



El flujo de trabajo del escenario es:

- Va a cargar un archivo de inventario a un depósito de Amazon S3.
- Esto activará una **función AWS Lambda** que leerá el archivo e insertará elementos en una **tabla de Amazon DynamoDB**.
- Una aplicación de tablero basada en la web sin servidor utilizará Amazon Cognito para autenticarse en AWS y luego acceder a la tabla DynamoDB para mostrar los niveles de inventario.
- Otra función de AWS Lambda recibirá actualizaciones de la tabla DynamoDB y enviará un mensaje a un tema de **Amazon Simple Notification Service (SNS)** cuando un artículo de inventario este agotado.
- Amazon SNS le **enviará un SMS o una notificación por correo electrónico** para solicitar un inventario adicional.

Manos a la obra:

- 1) Ingresar con el usuario y contraseña que le asignó el docente a la consola de AWS
- 2) Antes de empezar el laboratorio, crear la política que contengan los siguientes permisos:
  - a) CloudWatch: Para poder monitorear los logs arrojados
  - b) S3: Para nuestro repositorio de datos
  - c) DynamoDB: Nuestra base de datos NoSql que almacenará los csv o json cargados.
  - d) Lambda: Infraestructura como código que permitirá poner la función a usar.
  - e) SNS: Permitirá emitir una notificación de mensaje.

Para ello:

Ingresar a IAM, click en políticas y click en crear política

Identity and Access Management (IAM)

Dashboard

Access management

- Groups
- Users
- Roles
- Policies**
- Identity providers
- Account settings

Access reports

- Access analyzer

Create policy Policy actions

Filter policies

	Policy name	Type	Used as
<input type="radio"/>	<a href="#">AmazonDMSRedshiftS3Role</a>	AWS managed	None
<input type="radio"/>	<a href="#">AmazonS3FullAccess</a>	AWS managed	Permissions policy (7
<input type="radio"/>	<a href="#">AmazonS3ReadOnlyAccess</a>	AWS managed	None
<input type="radio"/>	<a href="#">QuickSightAccessForS3Stor...</a>	AWS managed	None
<input type="radio"/>	<a href="#">s3-json-dynamodb</a>	Customer managed	Permissions policy (1

Y añadir los permisos

Visual editor JSON Import managed policy

Expand all Collapse all

▼ S3 (All actions) Clone Remove

► Service S3

► Actions Manual actions \*

▼ Resources ☐ Specific ☒ All resources close

► Request conditions Specify request conditions (optional)

+ Add additional permissions

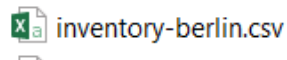
acter count: 110 of 6144.

Cancel

Review policy

Luego crear un rol que llame a ese conjunto de políticas que as creado.

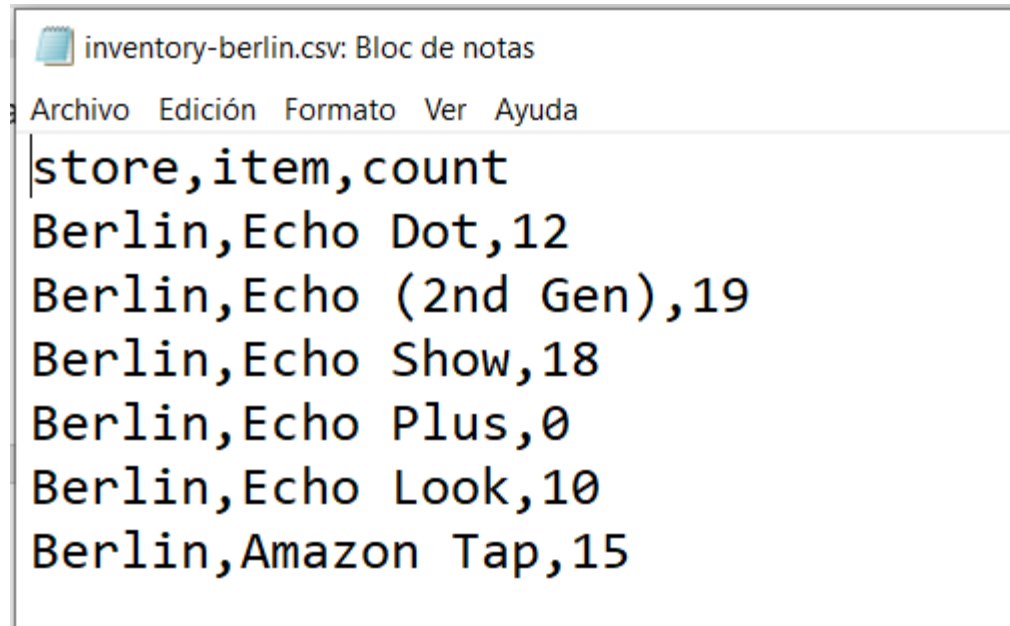
- 3) Crear un repositorio de S3 con un nombre de inventario-s3 y descomprimir el .rar inventory-files.zip y subir el archivo:



08/03/2019 12:04

Archivo de valores...

#### Estructura del archivo



- 4) Crear una tabla de DynamoDB:

Create DynamoDB table Tutorial ?

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name\*  ⓘ

Primary key\* Partition key

String ⓘ

☒ Add sort key

String ⓘ

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☒ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.

ⓘ You do not have the required role to enable Auto Scaling by default.  
Please refer to [documentation](#).

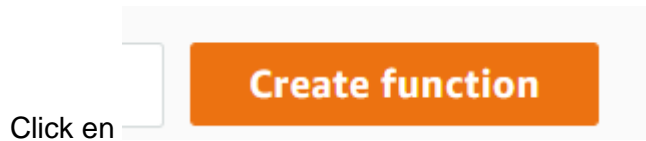
+ Add tags NEW!

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel Create

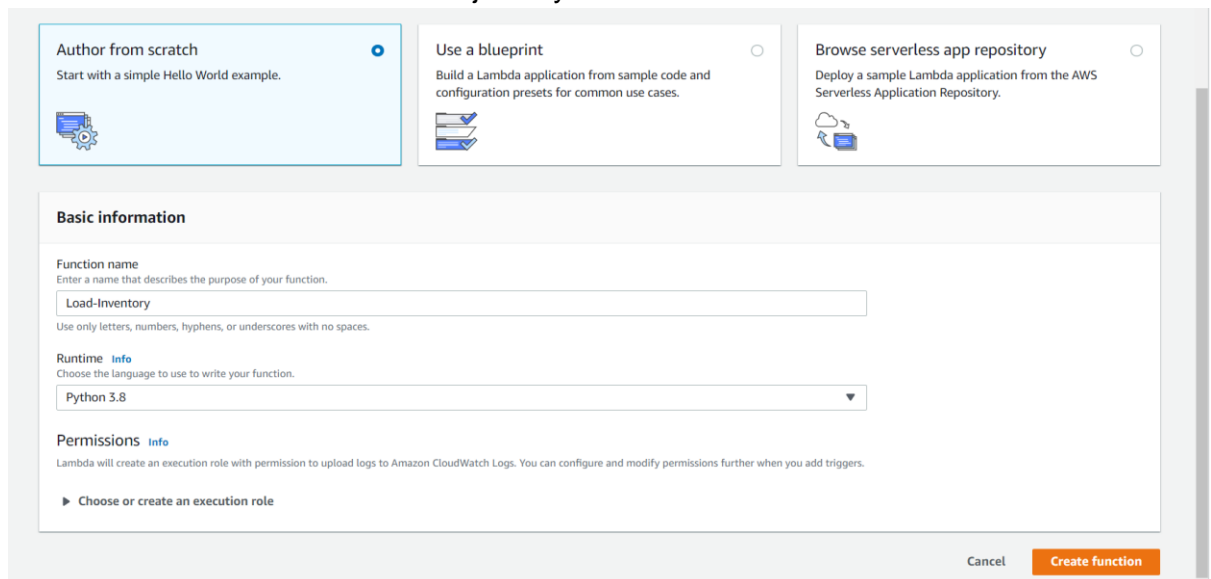
5) Ahora vamos a crear la función Lambda

En aws Management Console, click en el menú de servicios y buscar lambda



Ingresar:

- **Function Name: Load-Inventory**
- **Runtime:** python 3.8
- **Expandir choose or create an execution role**  
Usar el rol existente creado en paso anterior  
Nombre del rol : s3 –json-dynamodb

A screenshot of the AWS Lambda 'Create function' wizard. At the top, there are three tabs: 'Author from scratch' (selected), 'Use a blueprint', and 'Browse serverless app repository'. Below the tabs is a 'Basic information' section. It contains three fields: 'Function name' with the value 'Load-Inventory', 'Runtime' with the value 'Python 3.8', and 'Permissions' with a dropdown menu showing 'Choose or create an execution role'. At the bottom right of the wizard, there are two buttons: 'Cancel' and 'Create function'.

Este rol permite que la función Lambda tenga permisos de Amazon s3 y Amazon DynamoDB, luego click en Create Function

```
import json, urllib, boto3, csv

s3_client = boto3.client('s3')
s3 = boto3.resource('s3')

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Inventory');

def lambda_handler(event, context):
    bucket = event['Records'][0]['s3']['bucket']['name']

    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])
    localFilename = '/tmp/inventory.txt'
    s3.meta.client.download_file(bucket, key, localFilename)

    with open(localFilename) as csvfile:
        reader = csv.DictReader(csvfile, delimiter=',')
        rowCount = 0

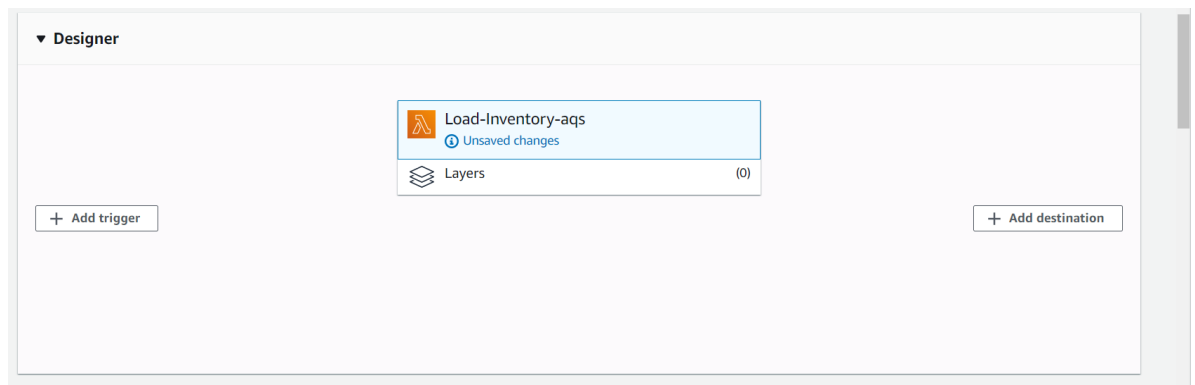
        for row in reader:
            rowCount += 1

            print(row['store'], row['item'], row['count'])

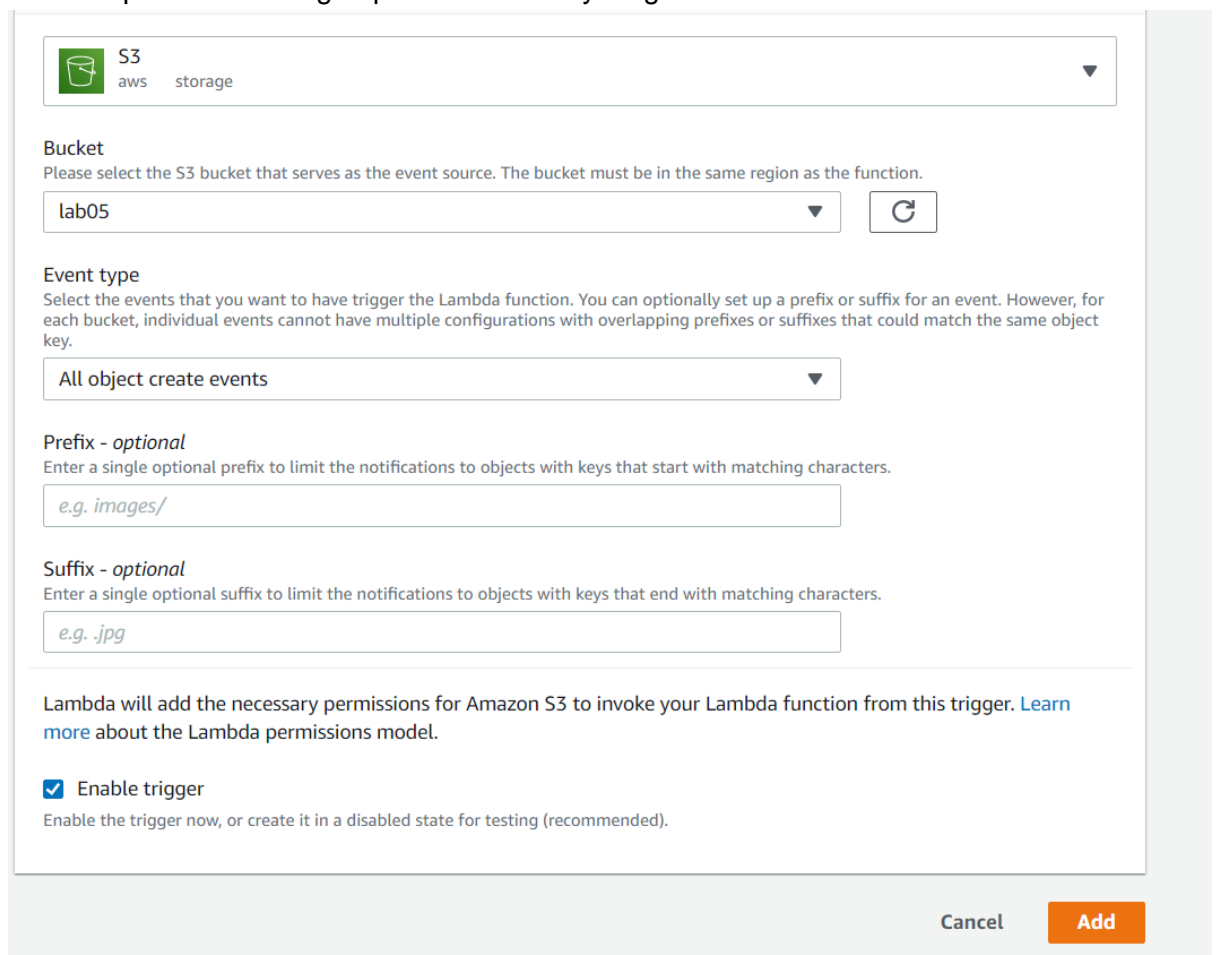
            table.put_item(
                Item={
                    'store': row['store'],
                    'item': row['item'],
                    'count': int(row['count'])})
        return "%d counts inserted" % rowCount
```

6) Ahora vamos a crear el trigger

Click en add trigger y escoger s3



Deberá quedar una imagen parecido a esto y luego click en crear.



7) Ahora para poder probar que todo esté bien crearemos una plantilla, para ello hacer click en configure test events.

Throttle

Qualifiers ▼

Actions ▼

Select a test event ▲

Test

Save

Configure test events

s. The function is now receiving events from the trigger.

Y cambiar los parámetros de acuerdo del cuadrado azul, por el nombre creado en s3 y nombre del archivo.

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☒ Create new test event

☐ Edit saved test events

Event template

Amazon S3 Put ▼

Event name

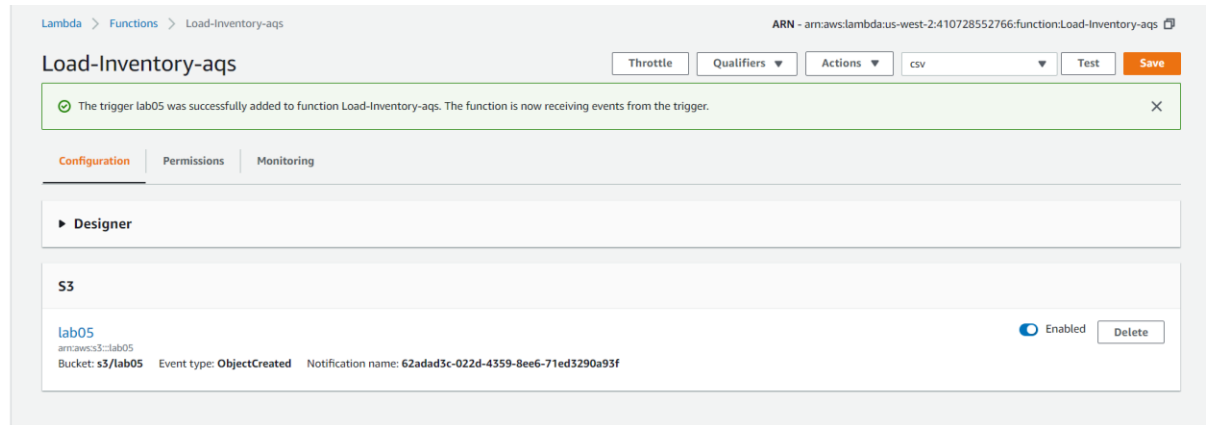
CSV

```
1 {
2   "Records": [
3     {
4       "eventVersion": "2.0",
5       "eventSource": "aws:s3",
6       "awsRegion": "us-west-2",
7       "eventTime": "1970-01-01T00:00:00.000Z",
8       "eventName": "ObjectCreated:Put",
9       "userIdentity": {
10        "principalId": "EXAMPLE"
11      },
12      "requestParameters": {
13        "sourceIPAddress": "127.0.0.1"
14      },
15      "responseElements": {
16        "x-amz-request-id": "EXAMPLE123456789",
17        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
18      },
19      "s3": {
20        "s3SchemaVersion": "1.0",
21        "configurationId": "testConfigRule",
22        "bucket": {
23          "name": "lab05",
24          "ownerIdentity": {
25            "principalId": "EXAMPLE"
26          },
27          "arn": "arn:aws:s3:::example-bucket"
28        },
29        "object": {
30          "key": "inventory-berlin.csv",
31          "size": 1024,
```

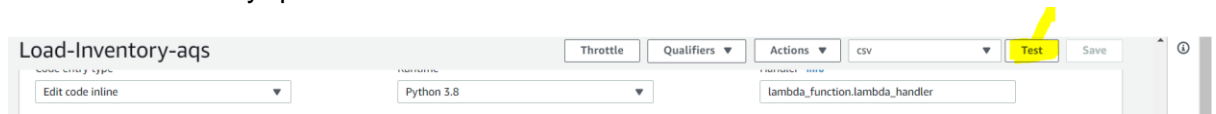
Cancel Create



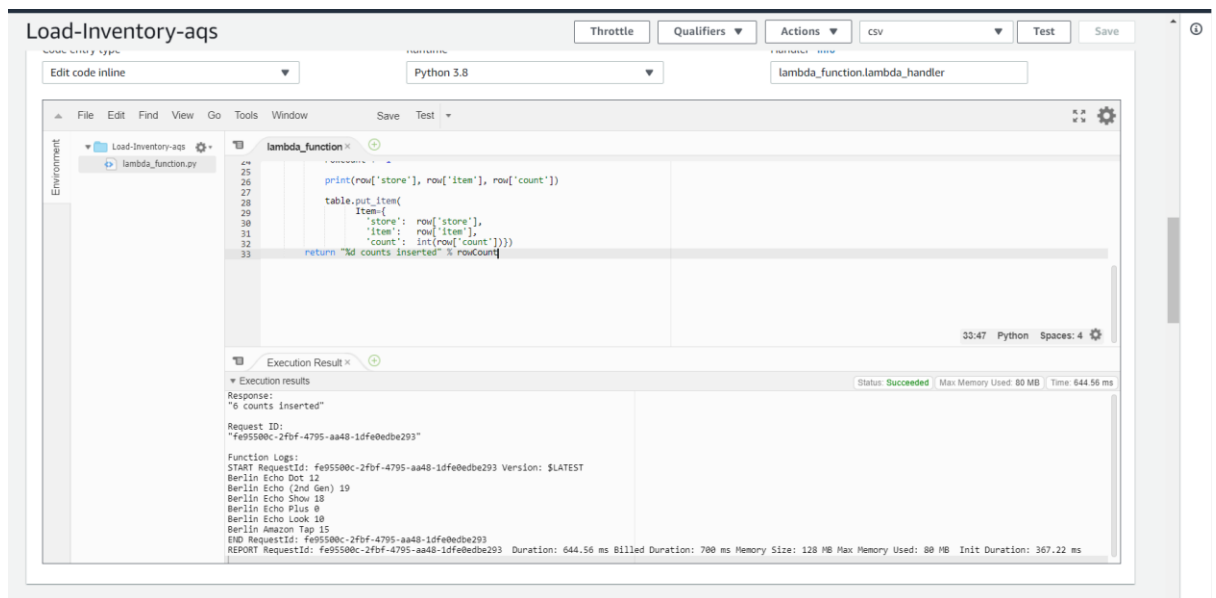
Click en créate y luego click en sabe



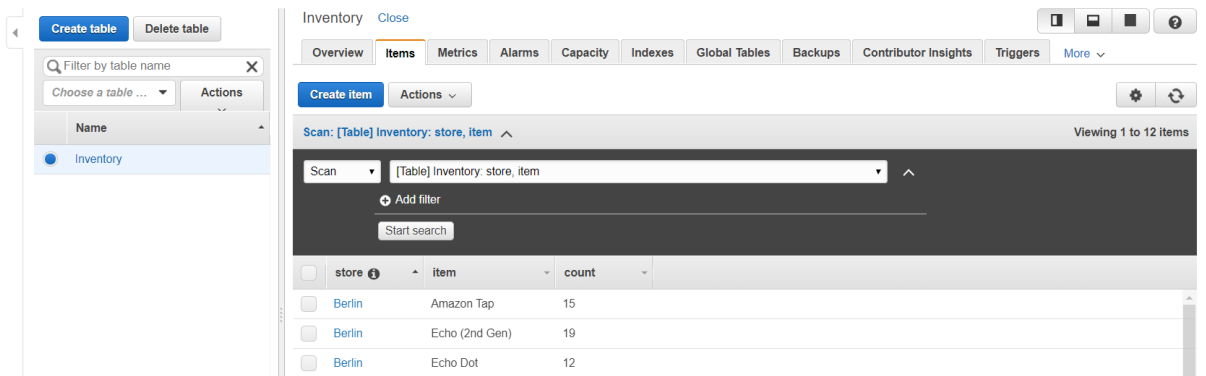
Para hacer el test y que tu función está bien debes hacer click en test



Saliendo algo parecido a esto:



Y así mismo se insertará los registros en DynamoDB

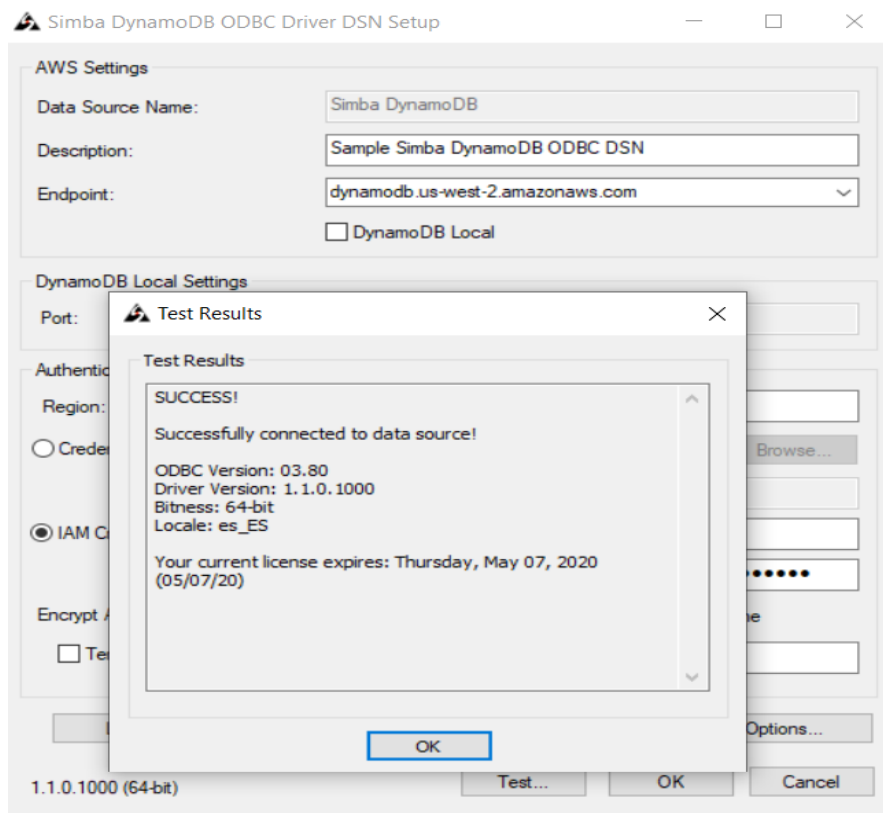


Luego instalaremos power bi

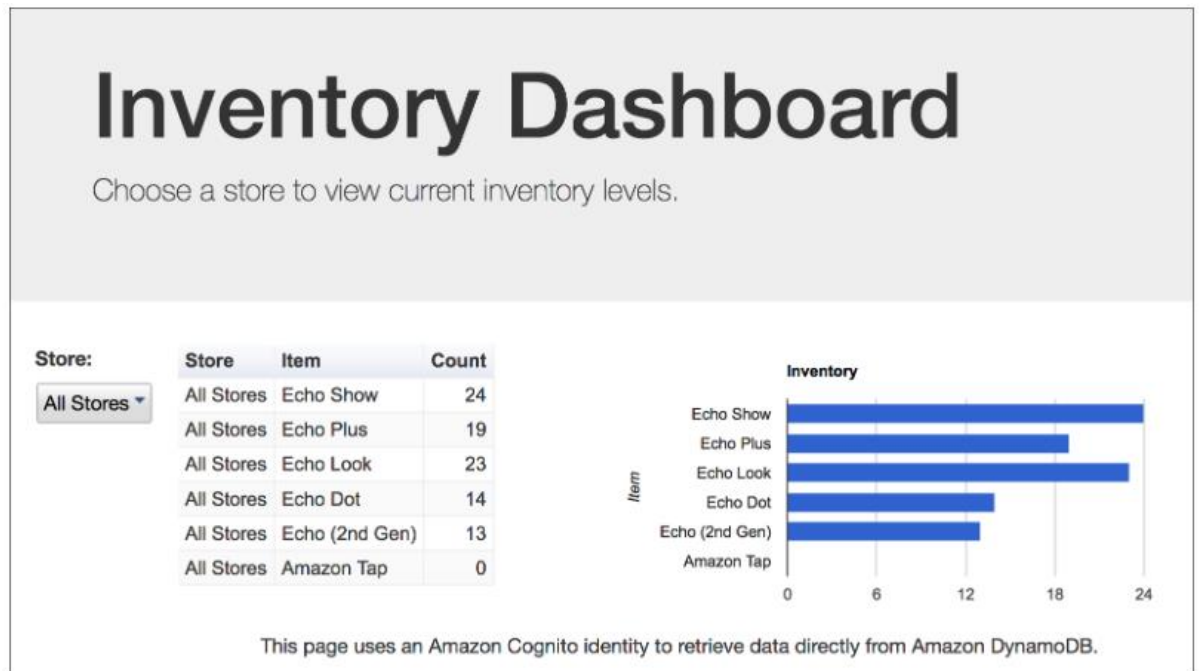
<https://www.microsoft.com/en-us/download/confirmation.aspx?id=58494>

<https://docs.aws.amazon.com/general/latest/gr/ddb.html>

<https://www.simba.com/checkout/order-received/?prod=DynamoDB&type=evaluation>



8) En el power bi se crear un dashboard parecido a esto



9) Vamos a ver las configuraciones de notificaciones

Desea notificar al personal de gestión de inventario cuando una tienda se queda sin existencias de un artículo. Para esta funcionalidad de notificación sin servidor, utilizará **Amazon simple notification service (SNS)**

Create topic

Click en

## Create topic

### Details

#### Name

NoStock

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

#### Display name - optional

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. [Info](#)

NoStock

Maximum 100 characters, including hyphens (-) and underscores (\_).

## Click en crear suscripción

nostock

EditDeletePublish message

Details

Name  
nostock

ARN  
arn:aws:sns:us-west-2:410728552766:nostock

Display name  
nostock

Topic owner  
410728552766

Subscriptions

Access policy

Delivery retry policy (HTTP/S)

Delivery status logging

Encryption

Tags

Subscriptions (0)

EditDeleteRequest confirmationConfirm subscriptionCreate subscription

Search

< 1 > ⚙

ID

Endpoint

Status

Protocol

No subscriptions found

You don't have any subscriptions to this topic.

Create subscription

Details

Topic ARN  

arn:aws:sns:us-west-2:410728552766:nostock X

Protocol  
The type of endpoint to subscribe  

Email

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

aquevedos91@gmail.com

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

► Subscription filter policy - optional

This policy filters the messages that a subscriber receives. Info

► Redrive policy (dead-letter queue) - optional

Send undeliverable messages to a dead-letter queue. Info

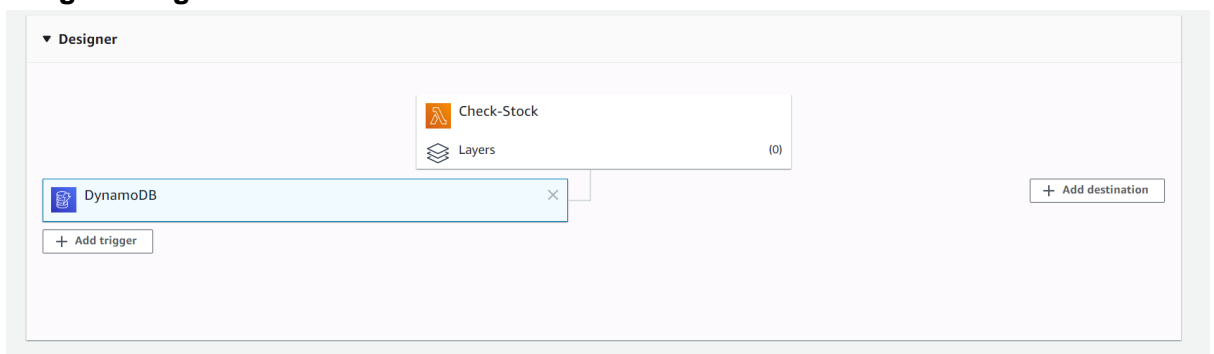
Cancel

Create subscription

10) Crear una función lambda que permitirá enviar el msn.

- **Name:** Check-Stock
- **Runtime:** *Python 3.8*
- Expand ☐ **Choose or create an execution role.**
- **Execution Role:** *Use an existing role*
- **Existing role:** *Lambda-Check-Stock-Role*
- Click **Create function**

Luego configurar de esta manera



```
console.log('Loading function');

var AWS = require('aws-sdk');
AWS.config.region = 'us-west-2'

exports.handler = function(event, context) {

    console.log("\n\nLoading handler\n\n");
    var sns = new AWS.SNS();
    sns.publish({
        Message: 'a new book has been published to dynamodb',
        TopicArn: 'arn:aws:sns:us-west-2:410728552766:nostock'
    }, function(err, data) {
        if(err){
            console.log(err.stack);
            return;
        }
        console.log('push sent');
        console.log(data);
        context.done(null, 'function finished');
    });
};
```

snsfunction

Throttle Qualifiers Actions sns Test Save

Code entry type Edit code inline Runtime Node.js 12.x Handler Info index.handler

Environment

File Edit Find View Go Tools Window Save Test

index.js

```
5
6 exports.handler = function(event, context) {
7   console.log("\n\nLoading handler\n\n");
8   var sns = new AWS.SNS();
9   sns.publish({
10    Message: 'a new book has been published to dynamodb',
11    TopicArn: 'arn:aws:sns:us-west-2:41872852766:nostock'
12  }, function(err, data) {
13    if(err){
14      console.log(err.stack);
15    }
16    return;
17  });
18  console.log("push sent");
19  console.log(data);
20  context.done(null, 'function finished');
21  };
22
```

Execution Result

Execution results Status: Succeeded Max Memory Used: 83 MB Time: 814.86 ms

Function Logs:

START RequestId: 4016cd3b-6a29-472e-81eb-f390ff4b213b Version: \$LATEST

2020-04-18T06:16:08.853Z 4016cd3b-6a29-472e-81eb-f390ff4b213b INFO

Loading handler

2020-04-18T06:16:09.625Z 4016cd3b-6a29-472e-81eb-f390ff4b213b INFO push sent

2020-04-18T06:16:09.627Z 4016cd3b-6a29-472e-81eb-f390ff4b213b INFO {

ResponseMetadata: { RequestId: '9e04883e-29e8-568e-bdde-df2ab4818041' },

MessageId: '8544ba1b-8264-576d-87f4-ed0f1f0da1a2'

END RequestId: 4016cd3b-6a29-472e-81eb-f390ff4b213b

REPORT RequestId: 4016cd3b-6a29-472e-81eb-f390ff4b213b Duration: 814.86 ms Billed Duration: 900 ms Memory Size: 128 MB Max Memory Used: 83 MB Init Duration: 364.95 ms

<https://github.com/julielkinsfembotit/CFtemplates/blob/master/AWSVPC2LAtemplate>

# Deploying a basic infrastructure in AWS using Cloud Formation

