Entregables y paso a paso AWS – Avance 4

NOTA IMPORTANTE: El siguiente paso a paso solamente lo debe ejecutar si quiere aprender a implementar los recursos de AWS: API Gateway, S3, Lambda, RDS, DynamoDB. Si lo decide hacer, tenga en cuenta que deberá crear una cuenta de AWS, si es su primera será gratuita, y podrá ejecutar el proceso sin costo alguno, pero aún así deberá asociar una tarjeta de crédito cuando esté creando la cuenta. Si no es su primera cuenta de AWS le cobrarán por el uso de los servicios. NO SE LE EVALUARÁ POR REALIZAR ESTO

En caso contrario, los entregables de este avance serán 2:

1. El diagrama de Arquitectura AWS (aws_architecture_diagram.png)
2. El archivo AWS_Análisis_Arquitectura.pdf, donde debe documentar TODO el proceso, sin ejecutarlo. Debe responder claramente:
    a. ¿Qué recursos se crean en los archivos lambda_functions.py y aws_setup.py?, y qué funcionalidad tiene cada uno?
    b. Profundice en la explicación de la función lambda verificar-entrega
    c. Profundice en la explicación de la función migrar_datos_postgresql()

Uso de recursos de AWS

Este paso a paso asume que usted ya tiene descargado python>=3.9 en su máquina local.

Si tiene alguna duda o no puede avanzar contacte al profesor.

1. Ingrese a https://signin.aws.amazon.com/signup?request_type=register y cree una cuenta AWS Free Tier. Le pedirá asociar una tarjeta de crédito. Recuerde que los créditos bajo la cuenta gratuita (200 USD) solo están disponibles si esta es su primera cuenta de AWS. Si no es su primera cuenta de AWS le cobrarán por uso de los servicios
2. Configure AWSCLI
    a. ¿Qué es AWS CLI? Es una herramienta de línea de comandos para controlar servicios AWS desde tu terminal (sin usar la web).
    b. ¿Dónde ejecutar los comandos?
    En Windows: Abre Command Prompt (CMD) o PowerShell
    En Mac/Linux: Abre la Terminal
    c. Instalar AWS CLI: pip install awscli boto3
    d. Verificar instalación: Ejecute aws –version, y luego pip show boto3
    Deberá ver algo como

```
PS C:\Users\alexv> aws --version
aws-cli/2.21.1 Python/3.12.6 Windows/11 exe/AMD64
PS C:\Users\alexv> pip show boto3
Name: boto3
Version: 1.35.62
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: C:\Users\alexv\anaconda3\Lib\site-packages
Requires: botocore, jmespath, s3transfer
Required-by: langchain-aws
PS C:\Users\alexv>
```

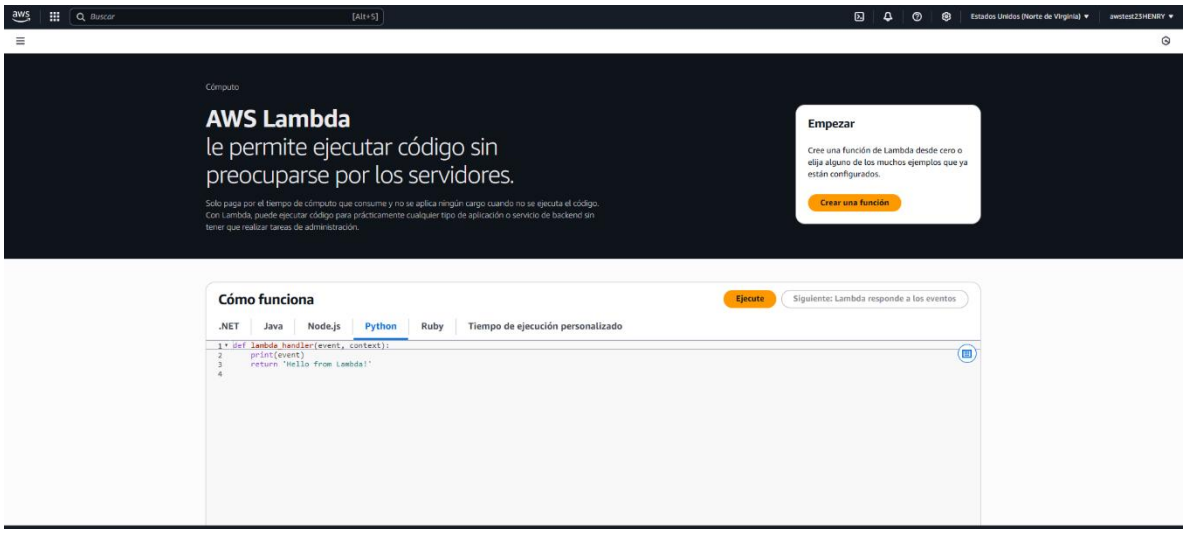
3. Obtener credenciales de AWS:
    a. Ve a la consola AWS (web)
    b. Click en tu nombre (arriba derecha) → Security credentials (Credenciales de seguridad)
    c. En la sección Access keys (Claves de acceso), click Create access key (Crear clave de acceso)
    d. Selecciona Command Line Interface (CLI)
    e. Guarda el Access Key ID y Secret Access Key

4. Configurar AWS CLI:
5. En PowerShell escribe: aws configure
6. Te pedirá 4 cosas:
        AWS Access Key ID [None]: (pega tu Access Key)
        AWS Secret Access Key [None]: (pega tu Secret Key)
        Default region name [None]: us-east-1
        Default output format [None]: json
7. Verificar que funciona: aws sts get-caller-identity. GUARDA LAS CREDENCIALES
8. Si todo está bien, verás tu información de cuenta AWS.

Ejecución funciones lambda

Para ejecutar las funciones lambda, siga los siguientes pasos:

1. Estando en la consola de AWS, en la barra superior izquierda en donde dice Buscar escriba Lambda y seleccione el servicio, luego deberá ver un panel como el siguiente



2. Primera función - Verificar Entrega
    a. Click en "Crear una función"
    b. Selecciona "Crear desde cero"
    c. Configuración:
        i. Nombre: fleetlogix-verificar-entrega

ii. Tiempo de ejecución: Python 3.9 o 3.10
iii. Arquitectura: x86_64

d. Click "Crear función"

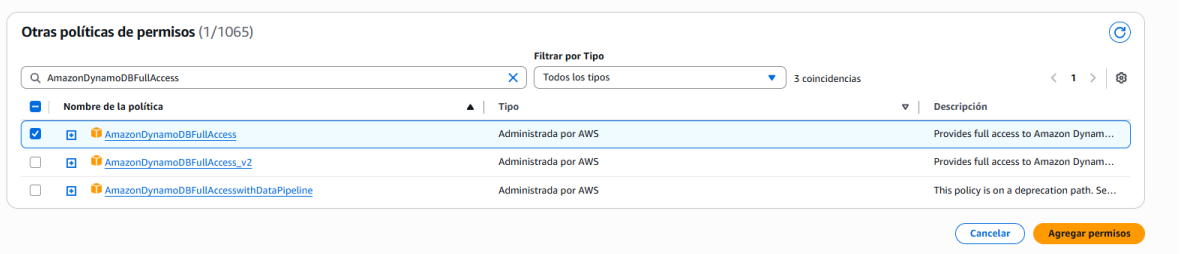
3. En el editor de código:
   a. Borra el código de ejemplo
   b. Copia y pega solo la primera función de lambda_functions.py:

```python
import json
import boto3
from datetime import datetime, timedelta
from decimal import Decimal

dynamodb = boto3.resource('dynamodb')

def lambda_handler(event, context):
    # Copiar aquí el contenido de lambda_verificar_entrega
    # Pero renombrar la función a lambda_handler
```

4. Configurar permisos:
   a. Ve a Configuración → Permisos
   b. Click en el rol de ejecución: fleetlogix-verificar-entrega-role-xxxxxxx
   c. Te llevará a la consola IAM
   d. Click en **"Agregar permisos"** → **"Asociar políticas"**
   e. Busca y agrega: AmazonDynamoDBFullAccess.



Necesitas tener permisos para acceder a DynamoDB, en el código la función intenta:

1. Conectarse a una tabla DynamoDB llamada deliveries_status
2. Leer datos de esa tabla

```python
# Conectar a tabla DynamoDB
table = dynamodb.Table('deliveries_status')

# Buscar entrega
response = table.get_item(
    Key={'delivery_id': delivery_id}
)
```

5. Vuelve a la pestaña de Lambda
    a. Click en "Deploy"
    b. Para probar, click en "Test"
    c. Crea un evento de prueba con:
        i. En la ventana que aparece, escribe:
        Event name: TestEntrega
        Event JSON: Reemplaza todo con:

```json
{
  "delivery_id": "12345",
  "tracking_number": "FL202412345"
}
```

        ii. Click en Save
    d. Click en Test

    e. VAS A VER UN ERROR COMO:

```
Status: Succeeded
Test Event Name: TestEntrega

Response:
{
  "statusCode": 500,
  "body": "{\"error\": \"An error occurred (ResourceNotFoundException) when calling the GetItem operation: Requested resource not found\"}"
}

Function Logs:
START RequestId: ccb51dca-915a-44b3-afd8-d4a15dc3a365 Version: $LATEST
END RequestId: ccb51dca-915a-44b3-afd8-d4a15dc3a365
REPORT RequestId: ccb51dca-915a-44b3-afd8-d4a15dc3a365  Duration: 323.51 ms Billed Duration: 324 ms Memory Size: 128 MB Max Memory Used: 81 MB  Init
Duration: 499.28 ms

Request ID: ccb51dca-915a-44b3-afd8-d4a15dc3a365
```

Esto es NORMAL porque:

- La función está buscando la tabla deliveries_status en DynamoDB
- Esa tabla no existe (se crearía con aws_setup.py)
- Este error demuestra que:
    La función Lambda está funcionando correctamente
    Está intentando conectarse a DynamoDB
    Solo falta crear la infraestructura (tablas)

6. Repetir los pasos para las otras 2 funciones:

- fleetlogix-calcular-eta. Al igual que la primera, solo necesita permisos a AmazonDynamoDBFullAccess

- fleetlogix-alerta-desvio. Esta función necesita 2 permisos AmazonDynamoDBFullAccess y AmazonSNSFullAccess

Importante: En Lambda, la función principal SIEMPRE debe llamarse lambda_handler, no el nombre original

Ejecución aws_setup

Anteriormente en la sección Uso de recursos de AWS se configuraron las credenciales de AWS. Ahora desde su máquina local ejecute **aws_setup.py,** este proceso creará automáticamente:

- **S3**
- **RDS PostgreSQL**
- **DynamoDB (4 tablas)**
- **Backups configurados**

Nota: RDS es lo más caro (~$15/mes). Los demás son casi gratis con uso bajo.