





CloudOps Guild  
*Together, towards mastery in Cloud and DevOps*

---

## Lab Cloud – Parte 2: Guarda tus reseñas en DynamoDB con API Gateway + Lambda

 **Objetivo:** Extender nuestra API RESTful sin servidores para guardar datos en una base de datos NoSQL con DynamoDB

 **Nivel:** Intermedio

---

### Tabla de contenidos

- Escenario real: Guardar reseñas en base de datos
  - ¿Qué es DynamoDB?
  - Paso a paso:
    1. Crear la tabla DynamoDB
    2. Dar permisos a Lambda
    3. Actualizar la función Lambda
    4. Probar el flujo completo
  - Validación en consola
  - Retos comunes
  - Próximos pasos
-



CloudOps Guild  
*Together, towards mastery in Cloud and DevOps*



## Escenario real

Ahora que CloudFizz ya puede recibir reseñas desde una API RESTful, necesitan guardarlas en una base de datos para poder listarlas y analizarlas más adelante. Vamos a usar **Amazon DynamoDB**, la base de datos serverless de AWS, ideal para este tipo de apps ligeras y altamente disponibles.

---



## ¿Qué es DynamoDB?

Amazon DynamoDB es una base de datos NoSQL completamente administrada, altamente escalable y muy rápida. No tienes que preocuparte por servidores, backups, ni mantenimiento.

---



## Paso a paso: Conectando la API a DynamoDB

---

### ♦ Paso 1: Crear la tabla en DynamoDB

1. Ve a **DynamoDB** desde la consola de AWS
2. Haz clic en **Crear tabla**
3. Configura:
  - **Nombre de tabla:** `Cafereview`
  - **Clave principal (Partition Key):** `id` (tipo `String`)
4. Desmarca la opción de “sort key” por ahora
5. Deja las demás configuraciones por defecto y crea la tabla



CloudOps Guild  
*Together, towards mastery in Cloud and DevOps*

☰ [DynamoDB](#) > [Tables](#) > Create table

## Create table

### Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

#### Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.).

#### Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

1 to 255 characters and case sensitive.

#### Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

1 to 255 characters and case sensitive.

### Table settings

#### ☒ Default settings

The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

#### ☐ Customize settings

Use these advanced features to make DynamoDB work better.

## ♦ Paso 2: Dar permisos a Lambda

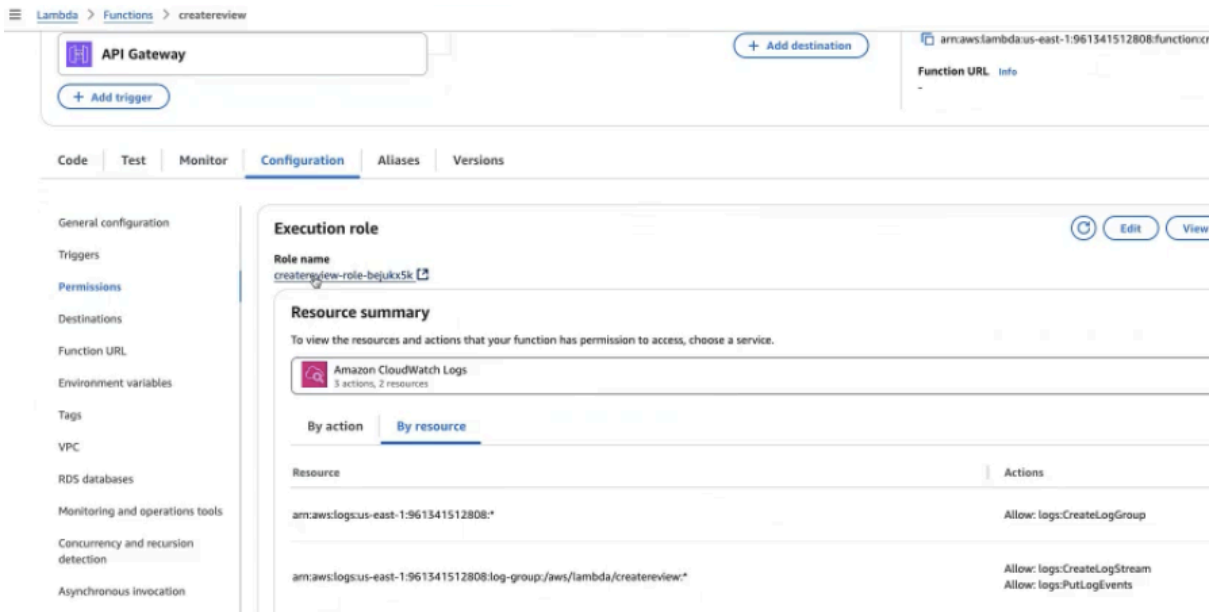
Tu función Lambda necesita permiso para escribir en la tabla.

1. Ve al servicio **Lambda**
2. Abre la función **crearreview**
3. En la pestaña **Permisos**, haz clic en el nombre del rol de ejecución (IAM Role)
4. Dentro del rol, haz clic en **Agregar permisos** → **Adjuntar políticas**
5. Busca y selecciona la política **AmazonDynamoDBFullAccess**  
👉 (En producción se recomienda una política más restringida)



# CloudOps Guild

*Together, towards mastery in Cloud and DevOps*



### ◆ Paso 3: Actualizar la función Lambda

Reemplaza el código actual de la función con lo siguiente:

```
import json
import uuid
import boto3

dynamodb = boto3.resource('dynamodb')
tabla = dynamodb.Table('Cafereview')

def lambda_handler(event, context):
    try:
        data = event # Ya viene en JSON gracias al mapping template

        nueva_resena = {
            "id": str(uuid.uuid4()),
            "usuario": data.get("usuario", "anónimo"),
            "comentario": data.get("comentario", "")
        }
```



CloudOps Guild  
*Together, towards mastery in Cloud and DevOps*

```
}  
  
tabla.put_item(Item=nueva_resena)  
  
return {  
    'statusCode': 201,  
    'body': json.dumps({  
        "mensaje": "¡Reseña guardada exitosamente!",  
        "reseña": nueva_resena  
    }, ensure_ascii=False)  
}  
  
except Exception as e:  
    return {  
        'statusCode': 400,  
        'body': json.dumps({'error': str(e)}, ensure_ascii=False)  
    }
```

🔒 Este código:

- Genera un ID único con `uuid4()`
- Extrae los campos del cuerpo del JSON
- Guarda la reseña en la tabla DynamoDB

---

#### ♦ Paso 4: Probar el flujo completo

Ejecuta el mismo comando `curl` desde CMD:

```
curl -X POST https://<tu-api-id>.execute-api.<region>.amazonaws.com/dev/resenas -H  
"Content-Type: application/json" -d "{\"usuario\": \"Carlos\", \"comentario\": \"Me encantó el  
café orgánico!\"}"
```



CloudOps Guild  
*Together, towards mastery in Cloud and DevOps*

✓ Si todo está bien, obtendrás una respuesta con `statusCode 201` y la reseña que se guardó.

---

### Verifica en la consola

1. Ve a DynamoDB → Tablas → `Cafereview`
  2. Haz clic en **Explorar tabla**
  3. Verás las reseñas creadas con su `id`, `usuario` y `comentario`
- 

### Retos comunes

Error o síntoma	Solución
AccessDeniedException en Lambda	Verifica que el rol tenga permisos de escritura en DynamoDB
La tabla aparece vacía	Asegúrate de llamar a <code>put_item()</code> y que el nombre esté correcto
Timeout o errores en Lambda	Verifica el nombre exacto de la tabla y región configurada

---

### Próximos pasos

¡Ahora tu API no solo recibe datos, sino que también los guarda!



**CloudOps Guild**  
*Together, towards mastery in Cloud and DevOps*

Puedes seguir con:

- Endpoint **GET** para leer las reseñas
  - Validación de campos en Lambda
  - Manejo de errores más detallado
  - Filtrar por usuario o fecha
  - Conectarlo a una app móvil o frontend en React/Vue
-