Laboratorio CRUD

Creación de una aplicación CRUD sin servidor

Información general del laboratorio

Objetivos

Al final de este laboratorio, podrá hacer lo siguiente:

- Crear una tabla de DynamoDB
- Crear funciones Lambda para leer y escribir datos de una table DynamoDB
- Crear y probar la API de API Gateway
- Implementar y probar la API

Iniciar laboratorio (loguearse en consola de aws)

Tarea 1: crear una tabla de DynamoDB

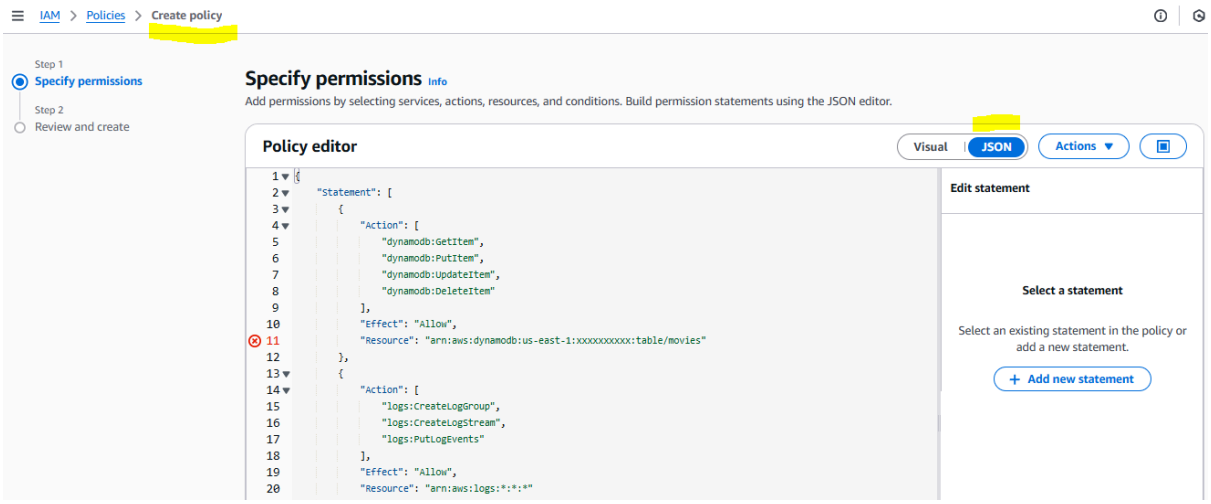
Cree una tabla de DynamoDB llamada movies a través de las siguientes opciones:

- Table name (Nombre de la tabla): ingrese `movies`
  Nota: anote el nombre de esta tabla, ya que lo necesitará más adelante.
- Partition key (Clave de partición): ingrese `title` y luego seleccione String (Cadena)
- Sort key (Clave de ordenación) (opcional): *dejar en blanco*
- Table settings (Configuración de la tabla): seleccione Default settings (Configuración predeterminada)

Nota: toma algunos segundos aprovisionar la tabla. Apenas termine, el Status (Estado) de la tabla será Active (Activo) y estará lista para usar. Puede seleccionar la tabla movies y navegar por las pestañas de navegación a fin de ver las opciones de configuración disponibles para la tabla.

Tarea 2: crear funciones de Lambda

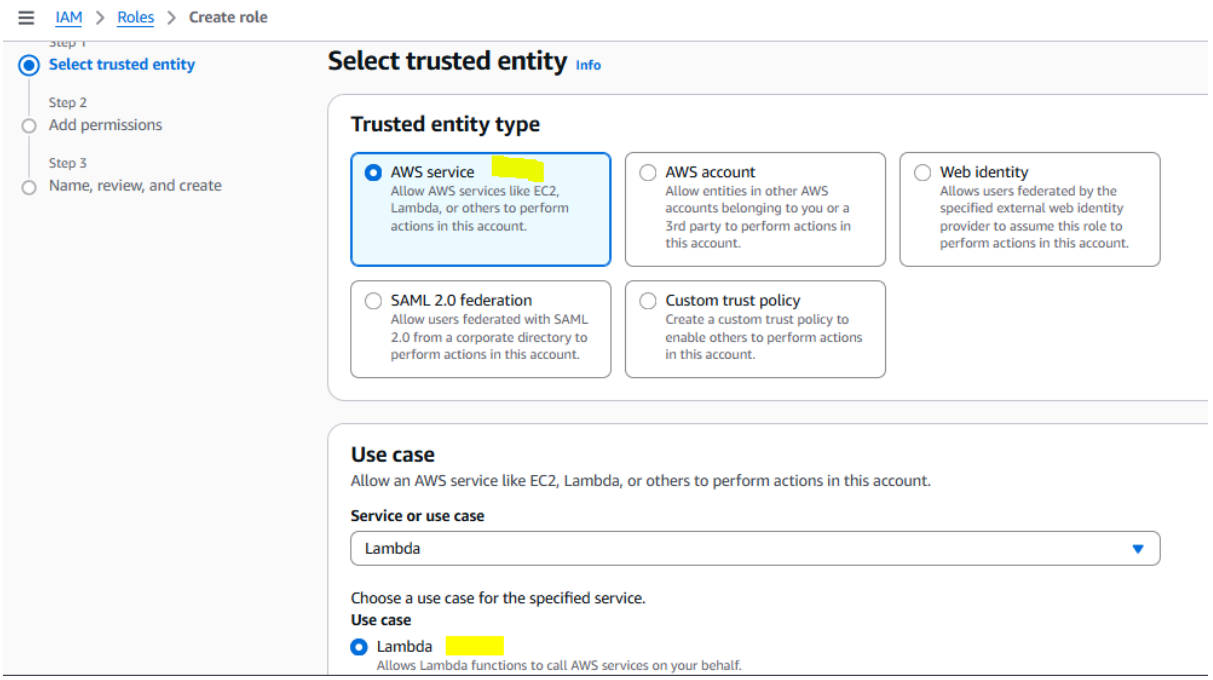
a) Cree la policy y rol para las lambdas. Buscar en la consola ***IAM/Policies*** llamada: **lambda-dynamodb-policy**



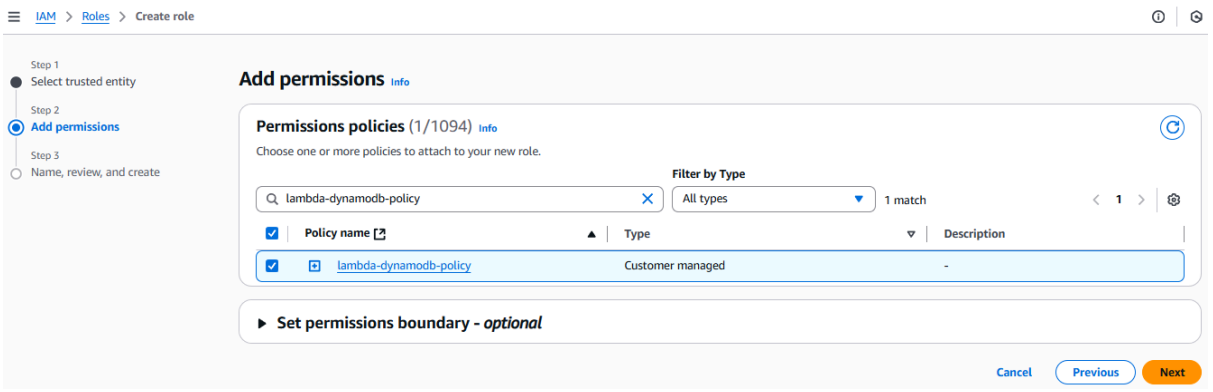
En el Policy editor pegue los siguiente, donde xxxxxxxxxx es su numero de cuenta de aws:

```
{
   "Statement": [
      {
         "Action": [
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb:DeleteItem"
         ],
         "Effect": "Allow",
         "Resource": "arn:aws:dynamodb:us-east-1:xxxxxxxxxx:table/movies"
      },
      {
         "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
         ],
         "Effect": "Allow",
         "Resource": "arn:aws:logs:*:*:*"
      }
   ],
   "Version": "2012-10-17"
}
```

b) Crear el siguiente rol de IAM. En la consola de AWs buscar **IAM/Roles**.



Elegir lambda-dynamo-policy



Finalmente en nombre : **lambda-movies-role**→ Create**.**

Nota: Este rol se adjuntará a las 4 lambda por crear.

Tarea 2.1: crear una función de Lambda movie

- Cree una función de Lambda movie a través de las siguientes opciones:
  - Create function (Crear función): seleccione **Author from scratch** (Crear desde cero)
  - Function name (Nombre de función): ingrese `create_movie`
  - Runtime (Tiempo de ejecución): seleccione **Python 3.9**
  - Architecture (Arquitectura): seleccione x86_64
- Permisos:
  - Expanda Change default execution role (Cambiar el rol de ejecución predeterminado)

- Execution role (Rol de ejecución): seleccione Use an existing role (Utilizar un rol existente)
- Existing role (Rol existente): seleccione **lambda-movies-role**

Nota: este rol otorga a la función de Lambda y a otras funciones de Lambda los permisos que necesitan para interactuar con DynamoDB.

- Copie y pegue el siguiente código para reemplazar el código existente en el archivo lambda_function.py.

```
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('movies')

def lambda_handler(event, context):
    year = event['year'] if event['year'] else '0'
    title = event['title'] if event['title'] else ''
    actors = event['actors'] if event['actors'] else ''

    response = table.put_item(
         Item={
             'year': year,
             'title': title,
             'info': {
                 'actors': actors
             }
         }
     )

    return {
         "statusCode": 200,
         "headers": {
             "Content-Type": "application/json"
         },
         "body": response
     }
```

- Implemente y pruebe la función de Lambda a través de las siguientes opciones:
  - Test event action (Acción de evento de prueba): seleccione Create new event (Crear evento nuevo)
  - Event name (Nombre de evento): ingrese `CreateMovieTest`
  - Event sharing settings (Configuración para compartir eventos): seleccione Private (Privado)
  - Template (Plantilla) (*Opcional*): ingrese `hello-world`
  - Event JSON: copie y pegue el siguiente código:

```
{
  "year": 1972,
  "title": "The Godfather",
  "actors": ["Marlon Brando", "Al Pacino", "James Caan"]
```

```
}
```

3. Para crear dos elementos más en la tabla movies modifique el código anterior JSON Code con la siguiente información y ejecute la prueba otra vez:

- Primer elemento:
  - year (año): `1972`
  - title (título): `Deliverance`
  - actors (actores): `Jon Voight`, `Burt Reynolds`, `Ned Beatty`
- Segundo elemento:
  - year (año): `1994`
  - title (título): `The Shawshank Redemption`
  - actors (actores): `Tim Robbins`, `Morgan Freeman`, `Bob Gunton`

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un HTTPStatusCode igual a 200 y los Elementos creados en la tabla movies de DynamoDB.

Tarea 2.2: crear una función de Lambda get movie (obtener movie)

- Cree una función de Lambda get movie (obtener movie) a través de las siguientes opciones:
  - Create function (Crear función): seleccione **Author from scratch** (Crear desde cero)
  - Function name (Nombre de función): ingrese `get_movie`
  - Runtime (Tiempo de ejecución): seleccione **Python 3.9**
  - Architecture (Arquitectura): seleccione x86_64
- Permisos:
  - Expanda Change default execution role (Cambiar rol de ejecución predeterminado)
  - Execution role (Rol de ejecución): seleccione Use an existing role (Utilizar un rol existente)
  - Existing role (Rol existente): seleccione **lambda-movies-role**
- Copie y pegue el siguiente código para reemplazar el código existente en el archivo lambda_function.py.

```
import boto3
import json

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('movies')

def lambda_handler(event, context):
    title = event['title']
    response = table.get_item(
       Key={'title': title}
    )
    return {
        'statusCode': 200,
        'headers': {
            'Content-Type': 'application/json'
```

```
        },
        "body": response
    }
```

- Implemente y pruebe la función de Lambda a través de las siguientes opciones:
    - Test event action (Acción de evento de prueba): seleccione Create new event (Crear evento nuevo)
    - Event name (Nombre de evento): ingrese `GetMovieTest`
    - Event sharing settings (Configuración para compartir eventos): seleccione Private (Privado)
    - Template (Plantilla) (*opcional*): ingrese `hello-world`
    - Event JSON: copie y pegue el siguiente código

```
{
  "title": "The Shawshank Redemption"
}
```

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un HTTPStatusCode igual a 200 y un Elemento con los datos relacionados al año y al título de la tabla movie que utilizó.

Tarea 2.3: crear una función de Lambda update movie (actualizar movie)

- Cree una función de Lambda update movie (actualizar movie) a través de las siguientes opciones:
    - Create function (Crear función):x seleccione **Author from scratch** (Crear desde cero)
    - Function name (Nombre de función): ingrese `update_movie`
    - Runtime (Tiempo de ejecución): seleccione **Python 3.9**
    - Architecture (Arquitectura): seleccione x86_64
- Permisos:
    - Expanda Change default execution role (Cambiar rol de ejecución predeterminado)
    - Execution role (Rol de ejecución): seleccione Use an existing role (Utilizar un rol existente)
    - Existing role (Rol existente): seleccione **lambda-movies-role**
- Copie y pegue el siguiente código para reemplazar el código existente en el archivo lambda_function.py.

```
import boto3
import decimal
from decimal import Decimal

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('movies')

def lambda_handler(event, context):
    year = event['year']
    title = event['title'] if event['title'] else ''
```

```
    rating = event['rating'] if event['rating'] else '0.0'
    plot = event['plot'] if event['plot'] else ''
    response = table.update_item(
        Key={
            'title': title
        },
        UpdateExpression="set info.rating=:r, info.plot=:p",
        ExpressionAttributeValues={
                ':r': Decimal(str(rating)),
                ':p': plot
        },
        ReturnValues="UPDATED_NEW"
    )
    return {
        "statusCode": 200,
        "headers": {
            "Content-Type": "application/json"
        },
        "body": response
    }
```

- Implemente y pruebe la función de Lambda a través de las siguientes opciones:
    - Test event action (Acción de evento de prueba): seleccione Create new event (Crear evento nuevo)
    - Event name (Nombre de evento): ingrese `UpdateMovieTest`
    - Event sharing settings (Configuración para compartir eventos): seleccione Private (Privado)
    - Template (Plantilla) (*opcional*): ingrese `hello-world`
    - Event JSON: copie y pegue el siguiente código

```
{
  "year": 1972,
  "title": "The Godfather",
  "rating": "9.2",
  "plot": "The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son."
}
```

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un HTTPStatusCode igual a 200 y una clave de Atributo con los datos actualizados relacionados a la tabla movie que utilizó.

Tarea 2.4: crear una función de Lambda delete movie (eliminar movie)

- Cree una función de Lambda delete movie (eliminar movie) a través de las siguientes opciones:
    - Create function (Crear función): seleccione **Author from scratch** (Crear desde cero)

    - Function name (Nombre de función): ingrese `delete_movie`
    - Runtime (Tiempo de ejecución): seleccione **Python 3.9**
    - Architecture (Arquitectura): seleccione x86_64
- Permisos:
    - Expanda Change default execution role (Cambiar rol de ejecución predeterminado)
    - Execution role (Rol de ejecución): seleccione Use an existing role (Utilizar un rol existente)
    - Existing role (Rol existente): seleccione **lambda-movies-role**
- Copie y pegue el siguiente código para reemplazar el código existente en el archivo lambda_function.py.

```
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('movies')

def lambda_handler(event, context):
    year = event['year']
    title = event['title']
    response = table.delete_item(
        Key={
            'title': title
        },
        ConditionExpression = "attribute_exists(info.actors)",
        ReturnValues="ALL_OLD"
    )
    return {
        "statusCode": 200,
        "headers": {
            "Content-Type": "application/json"
        },
        "body": response
    }
```

- Implemente y pruebe la función de Lambda a través de las siguientes opciones:
    - Test event action (Acción de evento de prueba): seleccione Create new event (Crear evento nuevo)
    - Event name (Nombre de evento): ingrese `DeleteMovieTest`
    - Event sharing settings (Configuración para compartir eventos): seleccione Private (Privado)
    - Template (Plantilla) (*opcional*): ingrese `hello-world`
    - Event JSON: copie y pegue el siguiente código

```
{
  "year": 1972,
  "title": "The Godfather"
}
```

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un HTTPStatusCode igual a 200 y el Elemento eliminado de la tabla movies de DynamoDB.

Tarea 3: crear y probar una API de API Gateway

Cree una API de API Gateway e intégrela con las funciones de Lambda creadas recientemente.

Tarea 3.1: crear una API de API Gateway

- Cree una API REST a través de las siguientes opciones:
    - Seleccione un API Type (Tipo de API): seleccione Rest API
      Nota: no haga clic en la API REST que indica Private (Privada) junto a ella.
      Para la configuración de la API, puede utilizar las siguientes opciones:
        - Create new API (Crear nueva API): seleccione New API (Nueva API)
        - API name (Nombre de API): ingrese
          `Movies API`
        - Description (Descripción): ingrese
          `Movies API that connects a web endpoint to several Lambda functions`

Tarea 3.2: crear un recurso de API de API Gateway

- Cree un recurso a través de las siguientes opciones:
    - Seleccione Create Resource (Crear un recurso).
    - Desactive Proxy resource (Recurso proxy).
    - Resource name (Nombre del recurso): ingrese
      `Movies`
    - Resource Path (Ruta del recurso): ingrese /
    - Enable API Gateway CORS (Habilitar CORS de la API Gateway):

Tarea 3.3: crear y probar un método POST de API Gateway

- Cree el método POST para la API de API Gateway para crear un nuevo elemento en la tabla movies de DynamoDB a través de las siguientes opciones:
    - Integration type (Tipo de integración): seleccione Lambda Function (Función Lambda)
    - Desactive Lambda proxy integration (Integración del proxy de Lambda).
    - Lambda Function (Función Lambda): ingrese
      `create_movie`
    - Habilite Default Timeout (Tiempo de espera predeterminado).
- Seleccione la pestaña Test (Probar) para probar el método POST con la siguiente carga en el Request Body (Cuerpo de la solicitud):

```
{
  "year": 1972,
  "title": "The Godfather",
  "actors": ["Marlon Brando", "Al Pacino", "James Caan"]
}
```

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un StatusCode igual a 200 y un Elemento creado en la tabla movies de DynamoDB.

Tarea 3.4: crear y probar el método PUT de API Gateway

- Cree el método PUT para la API de API Gateway para crear un nuevo Elemento en la tabla movies de DynamoDB a través de las siguientes opciones:
    - Integration type (Tipo de integración): seleccione Lambda Function (Función Lambda)
    - Desactive Lambda proxy integration (Integración del proxy de Lambda).
    - Lambda Function (Función Lambda): ingrese `update_movie`
    - Seleccione el Default timeout (Tiempo de espera predeterminado).
- Seleccione la pestaña Test (Probar) para probar el método PUT con la siguiente carga en el Request Body (Cuerpo de la solicitud):

```
{
  "year": 1972,
  "title": "The Godfather",
  "rating": "9.2",
  "plot": "The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son."
}
```

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un StatusCode igual a 200 y los Atributos actualizados en la tabla movies de DynamoDB.

Tarea 3.5: crear y probar el método DELETE de API Gateway

- Cree el método DELETE para la API de API Gateway para eliminar un elemento existente en la tabla movies de DynamoDB a través de las siguientes opciones:
    - Integration type (Tipo de integración): seleccione Lambda Function (Función Lambda)
    - Desactive Lambda proxy integration (Integración del proxy de Lambda).
    - Lambda Function (Función Lambda): ingrese `delete_movie`
    - Elija el Default Timeout (Tiempo de espera predeterminado).
- Seleccione la pestaña Test (Probar) para probar el método DELETE con la siguiente carga en el Request Body (Cuerpo de la solicitud):

```
{
  "year": 1972,
  "title": "The Godfather"
}
```

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un StatusCode igual a 200 y el Elemento eliminado de la tabla movies de DynamoDB.

Tarea 3.6: crear y probar el método GET de API Gateway

- Cree el método GET para la API de API Gateway para recuperar un elemento de la tabla movies de DynamoDB por su título.
- Establecerá la API para que responda a una ruta como la siguiente: `/Movies/<title>`
- Utilice las siguientes opciones para crear una ruta:

    Create Resource (Crear recurso) con el recurso /Movies seleccionado.

    - Desactive Proxy resource (Recurso proxy).
    - Resource Name (Nombre del recurso): ingrese `title`
    - Resource Path (Ruta del recurso): ingrese Movies/
    - Desactive la casilla de verificación Enable API Gateway CORS (Habilitar API Gateway CORS):

    Mantenga su recurso /{title} seleccionado para configurar el método GET a través de las siguientes opciones:

    - Integration type (Tipo de integración): seleccione Lambda Function (Función Lambda)
    - Desactive Lambda proxy integration (Integración del proxy de Lambda).
    - Lambda Function (Función Lambda): ingrese `get_movie`
    - Seleccione el Default timeout (Tiempo de espera predeterminado).
- Cambie la Integration Request (Solicitud de integración) con la siguiente información:

    Request body passthrough (Acceso directo al cuerpo de la solicitud): seleccione When there are no templates defined (recommended) (Cuando no hay plantillas definidas [recomendado])

    Mapping Templates (Plantillas de mapeo)

    - Add mapping template (Agregar plantilla de mapeo): en el campo de texto Content-Type (Tipo de contenido) ingrese `application/json`
    - Template body (Cuerpo de la plantilla): ingrese lo siguiente:

```
{
    "title": "$input.params('title')"
}
```

- Seleccione la pestaña Test (Probar) para probar el método GET con el valor `The Shawshank Redemption` en el campo de la Path (Ruta) {title} usando el comando de la Query String (cadena de la consulta) `title=The Shawshank Redemption`.

Nota: si la prueba se ejecuta correctamente, verá una Respuesta con un StatusCode igual a 200 y la información relacionada con el Elemento de la tabla movies de DynamoDB.

Tarea 4: implementar la API

Después de crear la API, debe implementarla para que se pueda acceder a ella desde una aplicación externa.

- Implemente la API a través de las siguientes opciones:
    - Resources (Recursos): seleccione la / superior en el árbol de la API.
    - Stage name (Nombre de la etapa): ingrese `dev`

Nota: después de implementar la API, tendrá la Invoke URL (URL de invocación) que se utilizará para acceder a la API. Cópiela a su editor de texto porque la utilizará para la tarea de testing.

Tarea 5: probar la API

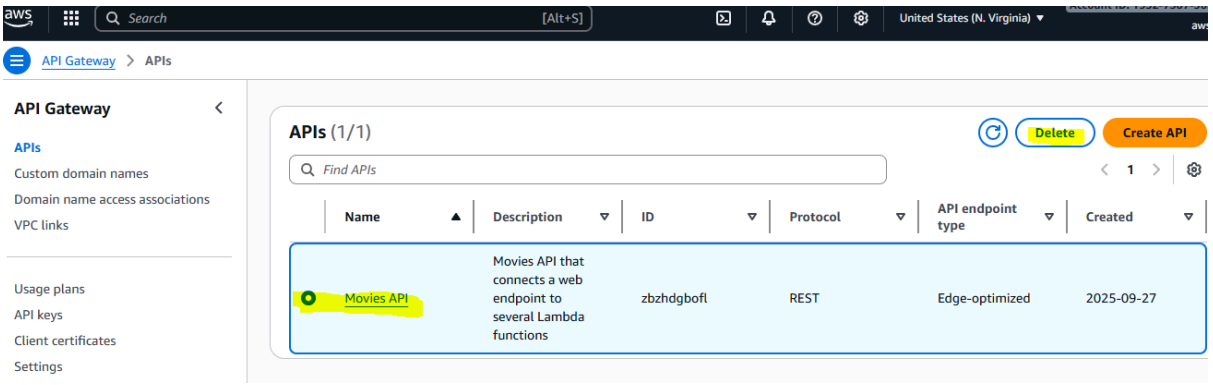
A continuación, debe probar el punto de enlace de la API de API Gateway como si una aplicación externa accediera a ella. Utilizará Postman.

Testear todos los métodos.

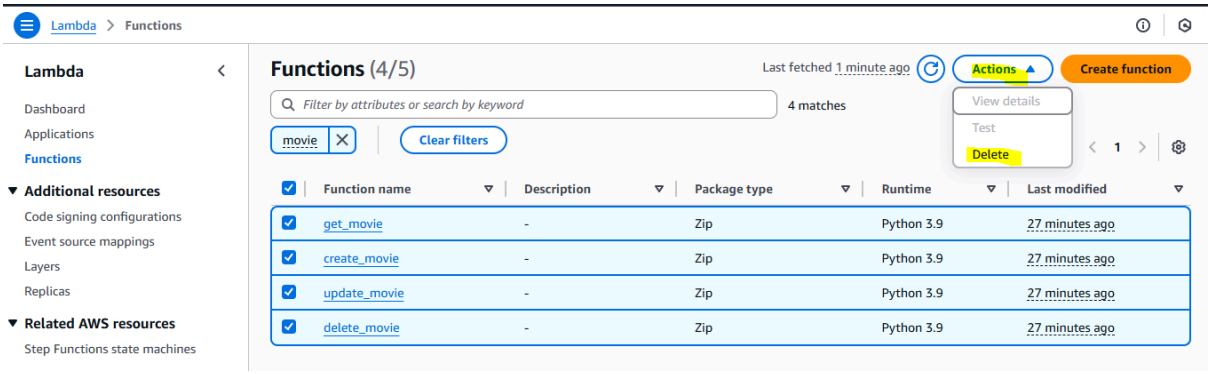
Tarea 6: Limpiar el ambiente

Es IMPORTANTE que luego de hacer todo el laboratorio se limpie el ambiente (eliminación de recursos).

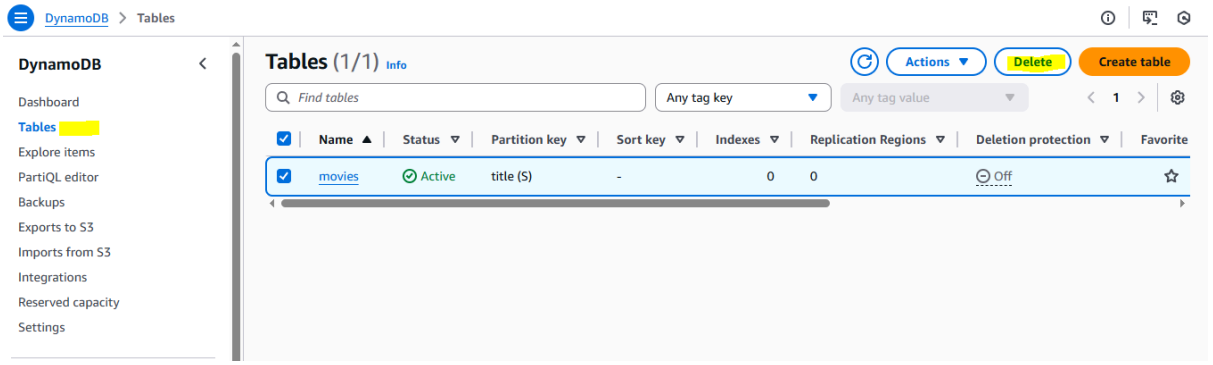
1- Ingrese en la consola al Servicio de AWS: **Api Gateway y elimine los recursos: seleccionar y click en Delete**



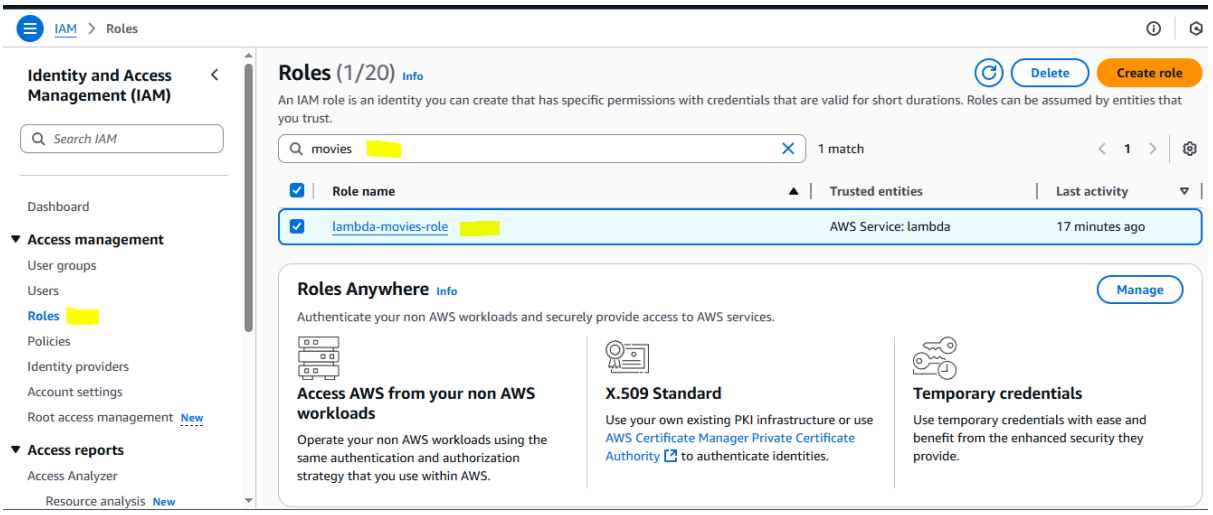
2-Ingrese por la consola al servicio **Lambda** y elimine las 4 lambdas (Seleccionar + Actions/Delete)



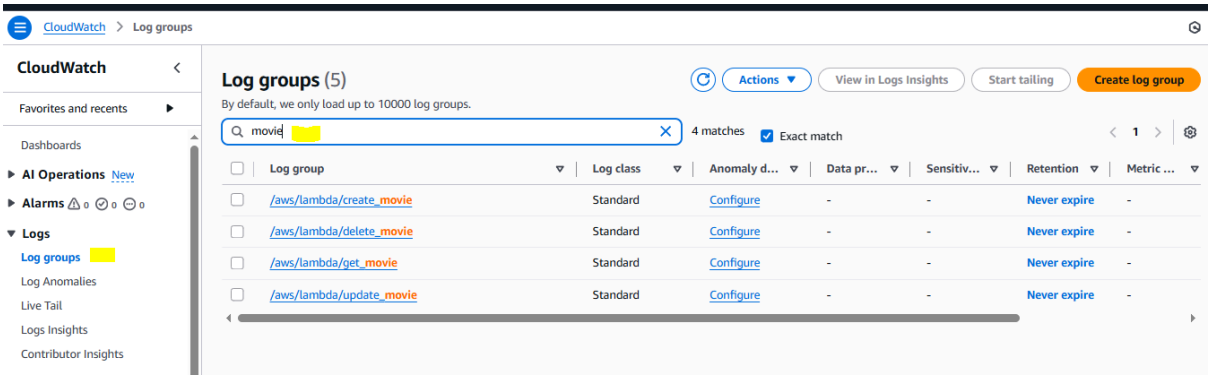
3-Ingrese por la consola al servicio DynamoDB y elimine la tabla movies (Seleccionar en Tables + Delete)



4-Ingrese por la consola al servicio de **IAM** y elimine el rol creado: **lambda-movies-role. Luego Puede eliminar la Policy asociada.**



5-Eliminar registros/grupos de CloudWatch



Conclusión

Aprendió a realizar correctamente lo siguiente:

- Crear una tabla de DynamoDB.
- Crear funciones Lambda para leer y escribir datos de una tabla DynamoDB.
- Crear y probar la API de API Gateway desde la consola de API Gateway.
- Implementar y probar el punto de enlace de la API de API Gateway.