

Capstone XI: Servicios avanzados en la nube

Cartelera

Juan Ignacio Alonso Barba, Jesus Martínez Gómez
Máster en Ciencia de Datos e Ingeniería de Datos en la Nube III
Universidad de Castilla-La Mancha

Marta Bellón Castro
Curso 2022-2023

Índice

- [1. Introducción](#)
- [2. Presentación de la arquitectura](#)
- [3. Acciones a realizar](#)

1. Introducción

El objetivo de este capstone es combinar de forma efectiva algunos de los conocimientos obtenidos en el módulo 11.

El Capstone posee un enfoque decididamente práctico, de forma que su resolución permitirá a los/as alumnos/as desplegar una pequeña aplicación en la nube que, de manera recurrente, consulte el estado de las carteleras de cine a través de llamadas a un API de terceros. El resultado de dichas consultas será almacenado a través de los servicios de almacenamiento de AWS. Además, los datos más relevantes serán almacenados en una base de datos no relacional, la cual permitirá realizar futuras consultas a través de un API construida sobre funciones Lambda.

La validación de la aplicación desarrollada se podrá realizar desde la libreta actual.

Cualquier alumno/a puede sentirse libre de ampliar los objetivos del Capstone.



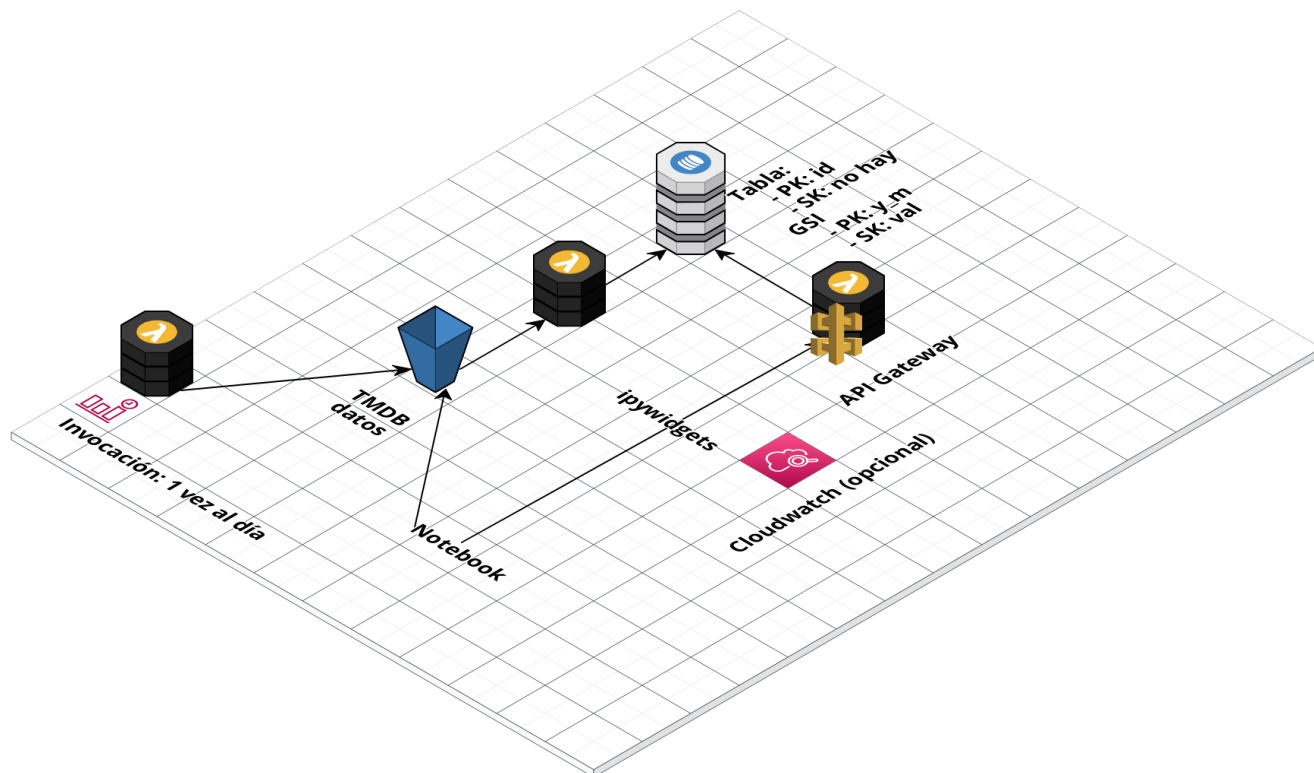
2. Presentación de la arquitectura

Durante el Capstone se desarrollará y desplegará una aplicación con la siguiente funcionalidad:

1. Consultará datos a través de un API externa, que permite obtener información sobre las películas que actualmente se encuentran proyectadas en los cines. Inicialmente, las consultas se harán a través del uso de la librería request dentro de la libreta actual
2. Almacenará los datos obtenidos con cada llamada, en forma de ficheros .json, usando el servicio de almacenamiento AWS S3. Estos ficheros se organizarán a través de un nombrado que refleje la fecha en la que han sido obtenidos
3. Mantendrá actualizada una base de datos, creada en AWS DynamoDB, con una serie de datos clave para toda película de la que hayamos obtenido información. El proceso de actualización se iniciará ante la presencia de nuevos ficheros en AWS S3.
4. Ofrecerá consultas a la base de datos que permitan obtener películas que cumplan determinadas condiciones
 - A. Películas lanzadas en mes en concreto
 - B. Películas con una valoración por encima de un umbral
5. Automatizará las consultas a la API externa a través del despliegue de una función AWS Lambda

6. Ofrecerá las consultas a la base de datos a través de un API desplegado con API Gateway + AWS Lambda

La arquitectura de la aplicación se muestra en la siguiente imagen:



3. Acciones a realizar

En esta sección se plantean una serie de ejercicios. Cada ejercicio amplía, de forma incremental, la funcionalidad de la aplicación.

EJERCICIO 0: Uso del API "The Movie Database API"

Tal y como se ha visto en otros módulos, para consumir el API debemos obtener un API Key que nos identifique durante nuestras consultas.

La API de TMDb requiere autenticación, por lo que para trabajar con ella es necesario, en primer lugar, disponer de un usuario. Una vez hecho el registro en el sitio, es necesario solicitar una clave para el uso de la API. Las instrucciones detalladas se muestran en esta página ([enlace \(https://developers.themoviedb.org/3/getting-started/introduction\)](https://developers.themoviedb.org/3/getting-started/introduction)). Este proceso es sencillo, y básicamente consiste en 3 pasos:

- Entrar en la configuración de la cuenta personal.
- Entrar en el menú de la API.
- Crear la API e identificar el token (lo usaremos como API KEY)

Una vez obtenido el API, podemos utilizarlo para realizar cualquier consulta de las incluidas en la documentación (<https://developers.themoviedb.org/3/movies> (<https://developers.themoviedb.org/3/movies>)).

Durante el capstone, haremos uso de la llamada `now_playing`, la cual nos da los siguientes datos para cada película proyectada actualmente en cines:

- `poster_path`: string con valores para generar una URL al poster de la película
- `adult`: valor boolean que nos indica si es una película para adultos o no
- `overview`: string con el resumen de la película
- `release_date`: string con la fecha de lanzamiento
- `id`: número entero que sirve de identificador único
- `popularity`: número entero con la popularidad actual
- `vote_average`: número entero con el valor medio de los votos realizados hasta la fecha

- vote_count: número entero que indica el número de votos realizados
- otros campos que podemos consultar desde <https://developers.themoviedb.org/3/movies/get-now-playing> (<https://developers.themoviedb.org/3/movies/get-now-playing>)

VALIDACIÓN

Utilizando alguna herramienta tipo POSTMAN, realiza una llamada a la URL https://api.themoviedb.org/3/movie/now_playing?api_key=TU_CLAVE (https://api.themoviedb.org/3/movie/now_playing?api_key=TU_CLAVE) reemplazando TU_CLAVE por el API Key personal.

Pegad los datos obtenidos para las 3 primeras películas en la siguiente celda:

```
"results": [{},{},{}] # DATOS DE PELÍCULAS
```

```
## Datos de las 3 primeras películas:
```

```
"results": [
  {
    "adult": false,
    "backdrop_path": "/e2Jd0sYMCe6qvMbswGQbM0Mzxt0.jpg",
    "genre_ids": [
      28,
      80,
      53
    ],
    "id": 385687,
    "original_language": "en",
    "original_title": "Fast X",
    "overview": "Over many missions and against impossible odds, Dom Toretto and his family have outsmarted, out-nerved and outdriven every foe in their path. Now, they confront the most lethal opponent they've ever faced: A terrifying threat emerging from the shadows of the past who's fueled by blood revenge, and who is determined to shatter this family and destroy everything—and everyone—that Dom loves, forever.",
    "popularity": 5796.751,
    "poster_path": "/fiVW06jE7z9Yn04trhaMEdc1SiC.jpg",
    "release_date": "2023-05-17",
    "title": "Fast X",
    "video": false,
    "vote_average": 7.3,
    "vote_count": 1926
  },
  {
    "adult": false,
    "backdrop_path": "/xXp7TbC0Ke4lB65ngkt3CuhsiAa.jpg",
    "genre_ids": [
      28,
      53
    ],
    "id": 697843,
    "original_language": "en",
    "original_title": "Extraction 2",
    "overview": "Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrates one of the world's deadliest prisons in order to save them. But when the extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rake and his team to Sydney, in order to get revenge.",
    "popularity": 3916.492,
    "poster_path": "/7gKI9hpEMcZUQpNgKrKdZJpbnNS.jpg",
    "release_date": "2023-06-09",
    "title": "Extraction 2",
    "video": false,
    "vote_average": 7.8,
    "vote_count": 680
  },
  {
    "adult": false,
    "backdrop_path": "/nGxUxi3PfXDRm7Vg95VBNgNM8yc.jpg",
    "genre_ids": [
      28,
      12,
      16,
      878
    ],
    "id": 569094,
    "original_language": "en",
    "original_title": "Spider-Man: Across the Spider-Verse",

```

```
    "overview": "After reuniting with Gwen Stacy, Brooklyn's full-time, friendly neighborhood Spider-Man is catapulted across the Multiverse, where he encounters the Spider Society, a team of Spider-People charged with protecting the Multiverse's very existence. But when the heroes clash on how to handle a new threat, Miles finds himself pitted against the other Spiders and must set out on his own to save those he loves most.",
    "popularity": 2073.855,
    "poster_path": "/8Vt6mWEReuy4Of61Lnj5Xj704m8.jpg",
    "release_date": "2023-05-31",
    "title": "Spider-Man: Across the Spider-Verse",
    "video": false,
    "vote_average": 8.7,
    "vote_count": 1614
  }
]
```

EJERCICIO 1: Consulta a través de requests

- Rellenad la celda de código a continuación para realizar una consulta a través del uso de la librería requests
 - Includ el valor del API KEY previamente obtenido
- Sabiendo que los datos se obtienen ordenados por popularidad, añadid un valor a cada resultado llamado rank con la posición de la película
 - La película en la posición 0 del array tendrá el valor de ranking 1
 - La película en la posición 1 del array tendrá el valor de ranking 2
 - La película en la posición n-1 del array tendrá el valor de ranking n

```
In [1]: import requests
import json

API_KEY = 'cc48d676183a961ea3d6c4299841b322' # "PEGAD EL VALOR ANTERIOR"
response = requests.get(f"https://api.themoviedb.org/3/movie/now_playing?api_key={API_KEY}")

# COMPLETAD CON CODIGO

# Verifica que la respuesta sea exitosa
if response.status_code == 200:

    # Carga los datos obtenidos en un diccionario
    results = response.json()['results']

    # Recorre cada resultado y agrega un valor de ranking
    for i, movie in enumerate(results):
        movie['rank'] = i + 1

    # Imprime los datos de las tres primeras películas
    for i in range(3):
        print(f"Puesto nº {results[i]['rank']}: {results[i]['title']}")

    # Imprime la info completa de las tres primeras películas en formato JSON
    print("\nLa información en formato JSON: \n")
    for i in range(3):
        result = results[i]
        print(json.dumps(result, indent=4))

else:
    # Imprime un mensaje de error en caso de que la respuesta no sea exitosa
    print("Error en la consulta, status code:", response.status_code)
```

Puesto nº 1: Fast X
Puesto nº 2: Extraction 2
Puesto nº 3: Spider-Man: Across the Spider-Verse

La información en formato JSON:

```
{
  "adult": false,
  "backdrop_path": "/e2Jd0sYMCe6qvMbswGQbM0Mzxt0.jpg",
  "genre_ids": [
    28,
    80,
    53
  ],
  "id": 385687,
  "original_language": "en",
  "original_title": "Fast X",
  "overview": "Over many missions and against impossible odds, Dom Toretto and his family have outsmarted, out-nerved and outdriven every foe in their path. Now, they confront the most lethal opponent they've ever faced: A terrifying threat emerging from the shadows of the past who's fueled by blood revenge, and who is determined to shatter this family and destroy everything\u2014and everyone\u2014that Dom loves, forever.",
  "popularity": 6048.312,
  "poster_path": "/fiVW06jE7z9YnO4trhaMEdc1SiC.jpg",
  "release_date": "2023-05-17",
  "title": "Fast X",
  "video": false,
  "vote_average": 7.3,
  "vote_count": 1951,
  "rank": 1
}
{
  "adult": false,
  "backdrop_path": "/xXp7TbC0Ke4lB65ngkt3CuhsiAa.jpg",
  "genre_ids": [
    28,
    53
  ],
  "id": 697843,
  "original_language": "en",
  "original_title": "Extraction 2",
  "overview": "Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrates one of the world's deadliest prisons in order to save them. But when the extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rake and his team to Sydney, in order to get revenge.",
  "popularity": 3872.49,
  "poster_path": "/7gKI9hpEMcZUQpNgKrkDzJpbnNS.jpg",
  "release_date": "2023-06-09",
  "title": "Extraction 2",
  "video": false,
  "vote_average": 7.8,
  "vote_count": 738,
  "rank": 2
}
{
  "adult": false,
  "backdrop_path": "/nGxUxi3PfXDRm7Vg95VBNgNM8yc.jpg",
  "genre_ids": [
    28,
    12,
    16,
    878
  ],
  "id": 569094,
  "original_language": "en",
  "original_title": "Spider-Man: Across the Spider-Verse",
  "overview": "After reuniting with Gwen Stacy, Brooklyn\u2019s full-time, friendly neighborhood Spider-Man is catapulted across the Multiverse, where he encounters the Spider Society, a team of Spider-People charged with protecting the Multiverse\u2019s very existence. But when the heroes clash on how to handle a new threat, Miles finds himself pitted against the other Spiders and must set out on his own to save those he loves most.",
  "popularity": 2354.257,
  "poster_path": "/8Vt6mWEReuy40f61Lnj5Xj704m8.jpg",
  "release_date": "2023-05-31",
  "title": "Spider-Man: Across the Spider-Verse",
  "video": false,
  "vote_average": 8.7,
```

```
    "vote_count": 1638,
    "rank": 3
}
```

VALIDACIÓN

Utilizando el código anterior, pegad los datos obtenidos para las 3 primeras películas (incluido el campo rank) en la siguiente celda:

```
"results": [{},{},{}] # DATOS DE PELÍCULAS
```

```
## Datos de Películas:
```

```
"results": {
  "adult": false,
  "backdrop_path": "/e2Jd0sYMCe6qvMbswGQbM0Mzxt0.jpg",
  "genre_ids": [
    28,
    80,
    53
  ],
  "id": 385687,
  "original_language": "en",
  "original_title": "Fast X",
  "overview": "Over many missions and against impossible odds, Dom Toretto and his family have outsmarted, out-nerved and outdriven every foe in their path. Now, they confront the most lethal opponent they've ever faced: A terrifying threat emerging from the shadows of the past who's fueled by blood revenge, and who is determined to shatter this family and destroy everything\u2014and everyone\u2014that Dom loves, forever.",
  "popularity": 5796.751,
  "poster_path": "/fiVW06jE7z9YnO4trhaMEdc1SiC.jpg",
  "release_date": "2023-05-17",
  "title": "Fast X",
  "video": false,
  "vote_average": 7.3,
  "vote_count": 1926,
  "rank": 1
}
{
  "adult": false,
  "backdrop_path": "/xXp7TbC0Ke4lB65ngkt3CuhsiAa.jpg",
  "genre_ids": [
    28,
    53
  ],
  "id": 697843,
  "original_language": "en",
  "original_title": "Extraction 2",
  "overview": "Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrates one of the world's deadliest prisons in order to save them. But when the extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rake and his team to Sydney, in order to get revenge.",
  "popularity": 3916.492,
  "poster_path": "/7gKI9hpEMcZUQpNgKrkdZJpbnNS.jpg",
  "release_date": "2023-06-09",
  "title": "Extraction 2",
  "video": false,
  "vote_average": 7.8,
  "vote_count": 680,
  "rank": 2
}
{
  "adult": false,
  "backdrop_path": "/nGxUxi3PfXDRm7Vg95VBNgNM8yc.jpg",
  "genre_ids": [
    28,
    12,
    16,
    878
  ],
  "id": 569094,
  "original_language": "en",
  "original_title": "Spider-Man: Across the Spider-Verse",
```

```
    "overview": "After reuniting with Gwen Stacy, Brooklyn\u2019s full-time, friendly neighborhood Spider-Man is catapulted across the Multiverse, where he encounters the Spider Society, a team of Spider-People charged with protecting the Multiverse\u2019s very existence. But when the heroes clash on how to handle a new threat, Miles finds himself pitted against the other Spiders and must set out on his own to save those he loves most.",
    "popularity": 2073.855,
    "poster_path": "/8Vt6mWREuy4Of61Lnj5Xj704m8.jpg",
    "release_date": "2023-05-31",
    "title": "Spider-Man: Across the Spider-Verse",
    "video": false,
    "vote_average": 8.7,
    "vote_count": 1614,
    "rank": 3
}
```



EJERCICIO 2: Subida de datos a S3

- Cread un bucket de S3 con el nombre `mcidaen3.capstone11.movies.raw.data.<tu-nombre>` .
 - Lo haremos desde la consola de AWS
- Rellenad la celda de código a continuación para que, tras consultar datos al API de películas e incluir el campo `rank`, se genere un objeto por cada resultado
- Path del objeto a crear `/movies/ID_MOVIE/AÑO_MES_DIA.json`
 - `ID_MOVIE` se corresponde con el id de la película
 - `AÑO/MES/DÍA` con la fecha de la consulta

Por ejemplo, si la consulta obtiene 4 películas, la ejecución de la celda debe generar 4 ficheros `.json` que se subirán, usando para ello la librería `boto3`, al bucket de S3 como objetos


```

In [2]: import boto3
        from datetime import datetime

NOMBRE_BUCKET = "mcidaen3.capstone11.movies.raw.data.mbc" # Creado desde la consola de AWS
now = datetime.now()
year_month_day = now.strftime("%Y_%m_%d") # Obtener dígitos del año, mes y día

# 1.- OBTENER DATOS DEL API MOVIES DB (COPIAD DEL EJERCICIO 1)

response = requests.get(f"https://api.themoviedb.org/3/movie/now_playing?api_key={API_KEY}")

if response.status_code == 200:
    data = response.json()
    results = data['results']

    # Agregar ranking a cada resultado
    for i, result in enumerate(results):
        result['rank'] = i + 1
else:
    results = []
    print(f'Error al obtener los datos: {response.text}')

# 2.- CONFIGURAD EL CLIENTE Y RECURSO DE ACCESO A S3, Y VALIDAD SU USO (CREACIÓN DE CARPETA)

s3_client = boto3.client('s3', region_name='us-east-1')
s3_resource = boto3.resource('s3', region_name='us-east-1')

try:
    s3_client.put_object(Bucket=NOMBRE_BUCKET, Key='movies/')
    print('Carpeta creada correctamente.')
except Exception as e:
    print(e)

# 3.- SUBID A S3 UN OBJETO POR RESULTADO OBTENIDO EN EL PASO 1

for result in results:
    movie_id = str(result['id'])
    file_name = f"movies/{movie_id}/{year_month_day}.json"
    s3_client.put_object(Bucket=NOMBRE_BUCKET, Key=file_name, Body=json.dumps(result))
    print(f'El objeto {file_name} se ha subido a S3.')

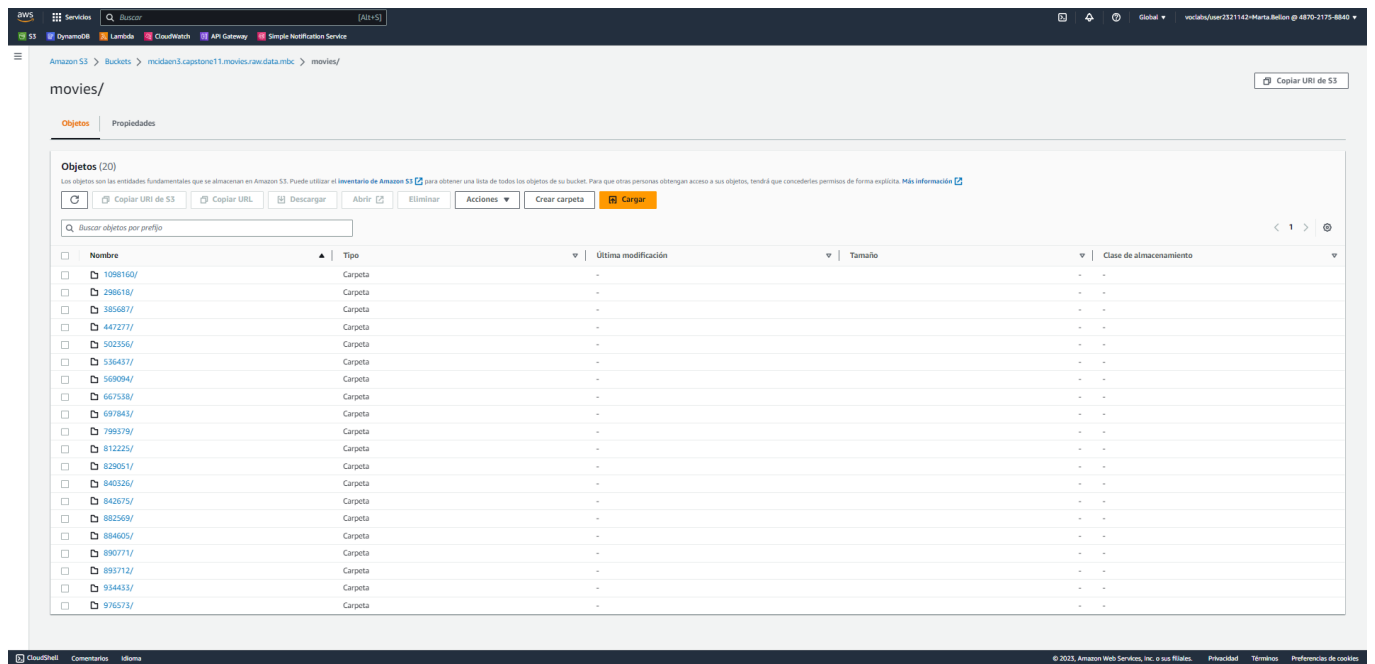
```

Carpeta creada correctamente.

El objeto movies/385687/2023_06_25.json se ha subido a S3.
 El objeto movies/697843/2023_06_25.json se ha subido a S3.
 El objeto movies/569094/2023_06_25.json se ha subido a S3.
 El objeto movies/502356/2023_06_25.json se ha subido a S3.
 El objeto movies/667538/2023_06_25.json se ha subido a S3.
 El objeto movies/536437/2023_06_25.json se ha subido a S3.
 El objeto movies/298618/2023_06_25.json se ha subido a S3.
 El objeto movies/976573/2023_06_25.json se ha subido a S3.
 El objeto movies/890771/2023_06_25.json se ha subido a S3.
 El objeto movies/447277/2023_06_25.json se ha subido a S3.
 El objeto movies/812225/2023_06_25.json se ha subido a S3.
 El objeto movies/882569/2023_06_25.json se ha subido a S3.
 El objeto movies/840326/2023_06_25.json se ha subido a S3.
 El objeto movies/799379/2023_06_25.json se ha subido a S3.
 El objeto movies/884605/2023_06_25.json se ha subido a S3.
 El objeto movies/1098160/2023_06_25.json se ha subido a S3.
 El objeto movies/934433/2023_06_25.json se ha subido a S3.
 El objeto movies/893712/2023_06_25.json se ha subido a S3.
 El objeto movies/842675/2023_06_25.json se ha subido a S3.
 El objeto movies/829051/2023_06_25.json se ha subido a S3.

VALIDACIÓN

- Ejecutad el código anterior y comprobad que se han generado entradas en S3
- Generad una imagen, con un pantallazo que muestre el contenido de la carpeta movies en S3, y guardadla en la ruta (junto al notebook) /img/eje_2.png



EJERCICIO 3: Almacenamiento de datos en DynamoDB

Para facilitar futuras consultas, vamos a almacenar datos de las películas en una base de datos de DynamoDB. Para ello, crearemos una tabla de DynamoDB a través de la consola de AWS con la siguiente configuración:

- Nombre: MoviesDB
- Partition Key: id (string)
- Global Secondary Index
 - Partition Key: y_m (string)
 - Representa la fecha de lanzamiento de la película (release_date)
 - Compuesto por el año (4 dígitos) y mes (2 dígitos). Por ejemplo 2023_01
 - Sort Key: val (número)

Para validar la tabla creada, vamos a generar una entrada a través de boto3. Para ello, rellenad la celda de código 3.1 de forma que el primer resultado consultado desde el API pueda guardarse en DynamoDB

```

In [7]: # CELDA 3.1

from decimal import Decimal

NOMBRE_TABLA = "MoviesDB"

# 1.- OBTENER DATOS DEL API MOVIES DB (COPIAD DEL EJERCICIO 1)

response = requests.get(f"https://api.themoviedb.org/3/movie/now_playing?api_key={API_KEY}")

# Verifica que la respuesta sea exitosa
if response.status_code == 200:

    # Carga los datos obtenidos en un diccionario
    results = response.json()['results']

    # Recorre cada resultado y agrega un valor de ranking
    for i, movie in enumerate(results):
        movie['rank'] = i + 1

    for i in range(3):
        result = results[i]

else:
    # Imprime un mensaje de error en caso de que la respuesta no sea exitosa
    print("Error en la consulta, status code:", response.status_code)

# 2.- CONFIGURAD EL RECURSO Y TABLA DE DYNAMO_DB

dynamodb_resource = boto3.resource('dynamodb', region_name='us-east-1')
dynamodb_table = dynamodb_resource.Table(NOMBRE_TABLA)

# 3.- MODIFICAD EL FORMATO DE LOS DATOS OBTENIDOS PARA QUE PUEDAN GUARDARSE EN DYNAMO_DB

for item in results:
    item["id"] = str(item["id"])
    item["y_m"] = item["release_date"][:7].replace("-", "_")
    item["val"] = float(item["popularity"])

# 4.- GUARDAD UNA ÚNICA ENTRADA EN DYNAMO DB, USANDO EL MÉTODO A CONTINUACIÓN

single_element = results[0]
dynamodb_table.put_item(Item=json.loads(json.dumps(single_element), parse_float=Decimal))

```

```

Out[7]: {'ResponseMetadata': {'RequestId': '3813VP56N4TP5BDM1GRRB02Q47VV4KQNSO5AEMVJF66Q9ASUAAJG',
'HTTPStatusCode': 200,
'HTTPHeaders': {'server': 'Server',
'date': 'Sun, 25 Jun 2023 13:14:37 GMT',
'content-type': 'application/x-amz-json-1.0',
'content-length': '2',
'connection': 'keep-alive',
'x-amzn-requestid': '3813VP56N4TP5BDM1GRRB02Q47VV4KQNSO5AEMVJF66Q9ASUAAJG',
'x-amz-crc32': '2745614147'},
'RetryAttempts': 0}}

```

VALIDACIÓN

- Ejecutad el código anterior y comprobad que se ha generado una entradas en DynamoDB
- Generad una imagen, con un pantallazo que muestre el contenido de la tabla movies en DynamoDB, y guardadla en la ruta (junto al notebook) /img/eje_3.png

Editar elemento
Puede agregar, eliminar o editar los atributos de un elemento. Es posible anidar atributos dentro de otros atributos hasta 32 niveles de profundidad. [Más información](#)

Formulario Vista JSON

Nombre de atributo	Valor	Tipo	
id - Clave de partición	697843	Cadena	
adult	<input type="radio"/> Verdadero <input checked="" type="radio"/> Falso	Booleano	Eliminar
backdrop_path	/xXp77bCOKe4lB65ngk13CuhsAa.jpg	Cadena	Eliminar
genre_ids	Insertar un campo	Lista	Eliminar
	0 28	Número	Eliminar
	1 53	Número	Eliminar
original_language	en	Cadena	Eliminar
original_title	Extraction 2	Cadena	Eliminar
overview	Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrates one of the world's deadliest prisons in order to save them. But when the extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rake and his team to	Cadena	Eliminar
poster_path	/7gKl9hpEMcZUQpNgKrkDzlpbnNS.jpg	Cadena	Eliminar
rank	2	Número	Eliminar

CloudShell Comentarios Idioma © 2023, Amazon Web Services, Inc. o sus filiales. Privacidad Términos Preferencias de cookies

NOTA: Puede que necesitemos cambiar el tipo de algunos campos para que puedan utilizarse como claves de partición u ordenación.



EJERCICIO 4: Automatización del proceso

Queremos automatizar el proceso de añadir/actualizar los datos en nuestra tabla de DynamoDB. Para ello, usaremos una función lambda con las siguientes características:

- Debe ejecutarse tras la creación de cualquier objeto en el bucket de S3 creado con anterioridad, bajo la ruta /movies
 - Lo conseguiremos a través de un disparador/trigger
- En caso de que no exista ninguna entrada previa para la película, se creará una entrada una vez se hayan generado campos adecuados para las claves de partición y ordenación
 - Clave de partición: id
 - GSI
 - Clave de partición: y_m
 - Clave de ordenación: val

Además, tendrá dos campos "especiales" con las siguientes características:

- rank_hist
 - Representará un diccionario con la evolución de la posición de la película en los rankings a lo largo de las semanas
 - Para ello, tendrá una entrada por cada semana en la que se haya consultado con el valor obtenido, donde fecha tendrá el formato año_nSemana (número de la semana/año en la que se ha consultado el ranking).
 - Año estará representado por 4 dígitos
 - nSemana estará representado por 2 dígitos
 - Si realizamos una petición a fecha 15 de Enero, estaremos en la semana 2, mientras que el 16 estaremos en la semana 3
 - Para la película más popular, añadiremos la entrada
 - rank_hist = {"2023_02", 1}
- until_date
 - Nos ayudará para consultar el número de días que una película se ha estado proyectando
 - Guardará la fecha de la consulta en el mismo formato que el campo release_date (proporcionado por el API MoviesDB)
 - yyyy-mm-dd
- En caso de que ya exista una entrada, actualizaremos sus valores de forma que "podamos" tener los siguientes cambios
 - Nuevo valor para la valoración actual
 - Campo val
 - Actualización del campo until_date
 - Nueva entrada en el campo rank_hist con el valor de la semana actual

Para ello, se deben realizar las siguientes acciones:

- Desarrollad en la celda de código 4.1 el código que se encargue de crear/actualizar entradas en DynamoDB, usando datos obtenidos previamente
- En la consola de AWS, cread una función lambda que se ejecute ante las acciones de S3 indicadas previamente
 - Incluir un disparador de S3 ante la creación de cualquier objeto
- Desplegad el código desarrollado en la celda 4.1 en la función lambda anterior
- Modificad el código para que los datos se obtengan desde el evento de la lambda
 - El evento incluye el Bucket y clave del objeto de S3 disparó la ejecución (fichero .json)
 - Utilizad boto3 para leer el contenido del fichero
 - La celda 4.2 incluye algún código que pueda servir de ayuda
- Desplegad la función lambda tras los cambios
- Ejecutad alguna celda anterior que genere entradas en S3 a partir de llamadas al API de películas y comprobad el resultado
 - También podemos bajar y subir alguno de los .json que ya estén en el bucket, usando la consola de AWS

⚠ NOTA Se valorará positivamente el uso de la función 'update_item' de boto3 con una 'update expression' que actualice los valores que pueden cambiar sin necesidad de hacer una 'query' previa de consulta.

In [8]: # CELDA 4.1

```
NOMBRE_TABLA = "MoviesDB"

year_number = now.isocalendar()[0]
week_number = now.isocalendar()[1]
year_week = f'{year_number}_{week_number}'

# 1.- OBTENER DATOS DEL API MOVIES DB (COPIAD DEL EJERCICIO 1)

response = requests.get(f"https://api.themoviedb.org/3/movie/now_playing?api_key={API_KEY}")

# Verifica que la respuesta sea exitosa
if response.status_code == 200:

    # Carga los datos obtenidos en un diccionario
    results = response.json()['results']

    # Recorre cada resultado y agrega un valor de ranking
    for i, movie in enumerate(results):
        movie['rank'] = i + 1

    for i in range(3):
        result = results[i]
else:
    # Imprime un mensaje de error en caso de que la respuesta no sea exitosa
    print("Error en la consulta, status code:", response.status_code)

# 2.- CONFIGURAD EL RECURSO Y TABLA DE DYNAMO_DB

dynamodb_resource = boto3.resource('dynamodb', region_name='us-east-1')
dynamodb_table = dynamodb_resource.Table(NOMBRE_TABLA)

# 3.- MODIFICAD EL FORMATO DE LOS DATOS OBTENIDOS PARA QUE PUEDAN GUARDARSE EN DYNAMO_DB

for item in results:
    item["id"] = str(item["id"])
    item["y_m"] = item["release_date"][:7].replace("-", "_")
    item["val"] = float(item["popularity"])

# 4.- GUARDA UNA ÚNICA ENTRADA EN DYNAMO DB, USANDO EL MÉTODO A CONTINUACIÓN
single_element = results[0]

# Define la clave de búsqueda
key = {'id': str(single_element['id'])}

# Define la expresión de actualización para los campos que cambian solamente
update_expression = 'SET #val = :val, #y_m = :y_m, #rank = :rank, #title = :title, #rank_hist = :rank_hist, #until_date = :until_date'

# Define los nombres de los campos
expression_attribute_names = {
    '#val': 'val',
    '#y_m': 'y_m',
    '#rank': 'rank',
    '#title': 'title',
    '#rank_hist': 'rank_hist',
    '#until_date': 'until_date',
    '#original_language': 'original_language',
    '#original_title': 'original_title',
    '#overview': 'overview',
    '#poster_path': 'poster_path',
    '#adult': 'adult',
    '#backdrop_path': 'backdrop_path',
    '#genre_ids': 'genre_ids',
    '#video': 'video',
    '#vote_average': 'vote_average',
    '#vote_count': 'vote_count'
}

# Creamos una variable "until_date" con el valor de la fecha actual en formato yyyy-mm-dd
until_date = datetime.now().strftime("%Y-%m-%d")

# Define los valores de los campos
expression_attribute_values = {
    ':val': Decimal(str(single_element['val'])),
    ':y_m': str(single_element['y_m']),
    ':rank': str(single_element['rank']),
    ':title': str(single_element['title']),
    ':rank_hist': str(single_element['rank_hist']),
    ':until_date': str(until_date),
    ':original_language': str(single_element['original_language']),
    ':original_title': str(single_element['original_title']),
    ':overview': str(single_element['overview']),
    ':poster_path': str(single_element['poster_path']),
    ':adult': str(single_element['adult']),
    ':backdrop_path': str(single_element['backdrop_path']),
    ':genre_ids': str(single_element['genre_ids']),
    ':video': str(single_element['video']),
    ':vote_average': str(single_element['vote_average']),
    ':vote_count': str(single_element['vote_count'])
}
```

```

        ':y_m': single_element['y_m'],
        ':rank': single_element['rank'],
        ':title': single_element['title'],
        ':until_date': str(until_date),
        ':original_language': single_element['original_language'],
        ':original_title': single_element['original_title'],
        ':overview': single_element['overview'],
        ':poster_path': single_element['poster_path'],
        ':adult': single_element['adult'],
        ':backdrop_path': single_element['backdrop_path'],
        ':genre_ids': single_element['genre_ids'],
        ':video': single_element['video'],
        ':vote_average': Decimal(str(single_element['vote_average'])),
        ':vote_count': Decimal(str(single_element['vote_count']))
    }

    # Crear rank_hist si no existe en el elemento
    if 'rank_hist' not in single_element:
        rank_hist = {year_week: single_element['rank']}
    else:
        # Recuperar el histórico de posiciones de la película
        rank_hist = single_element['rank_hist']
        # Añadir la nueva entrada al histórico de posiciones
        rank_hist[year_week] = single_element['rank']

    # Añadir la entrada rank_hist a expression_attribute_values
    expression_attribute_values[':rank_hist'] = rank_hist

    # Actualiza el elemento existente o crea uno nuevo si no existe
    response = dynamodb_table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues='ALL_NEW'
    )

    print('El elemento se ha creado/actualizado.')

```

El elemento se ha creado/actualizado.

CELDA 4.2 (función lambda pegada en AWS)

```

import boto3
import json
import botocore
from decimal import Decimal
from datetime import datetime

def lambda_handler(event, context):
    NOMBRE_TABLA = "MoviesDB"
    dynamodb_resource = boto3.resource('dynamodb', region_name='us-east-1')
    dynamodb_table = dynamodb_resource.Table(NOMBRE_TABLA)
    s3_resource = boto3.resource('s3', region_name='us-east-1')
    now = datetime.now()
    year_number = now.isocalendar()[0]
    week_number = now.isocalendar()[1]
    year_week = f'{year_number}_{week_number}'

    for record in event.get('Records', []):
        # 1 LEER DATOS DESDE S3
        bucket = record["s3"]["bucket"]["name"]
        key = record["s3"]["object"]["key"]
        obj = s3_resource.Object(bucket, key)
        element = json.load(obj.get()['Body'])
        # COMPLETAD CON CODIGO

        # 3.- MODIFICAD EL FORMATO DE LOS DATOS OBTENIDOS PARA QUE PUEDAN GUARDARSE EN DYNAMO_DB
        element["id"] = str(element["id"])
        element["y_m"] = element["release_date"][7:].replace("-", "_")
        element["val"] = float(element["popularity"])

        # Define la clave de búsqueda
        key = {'id': str(element['id'])}

```



```

# Define la expresión de actualización para los campos que cambian solamente
update_expression = 'SET #val = :val, #y_m = :y_m, #rank = :rank, #title = :title, #rank_hist =
:rank_hist, #until_date = :until_date, #original_language=:original_language, #original_title=:original_title,
#overview=:overview, #poster_path=:poster_path, #adult=:adult, #backdrop_path=:backdrop_path,
#genre_ids=:genre_ids, #video=:video, #vote_average=:vote_average, #vote_count=:vote_count'

# Define los nombres de los campos
expression_attribute_names = {
    '#val': 'val',
    '#y_m': 'y_m',
    '#rank': 'rank',
    '#title': 'title',
    '#rank_hist': 'rank_hist',
    '#until_date': 'until_date',
    '#original_language': 'original_language',
    '#original_title': 'original_title',
    '#overview': 'overview',
    '#poster_path': 'poster_path',
    '#adult': 'adult',
    '#backdrop_path': 'backdrop_path',
    '#genre_ids': 'genre_ids',
    '#video': 'video',
    '#vote_average': 'vote_average',
    '#vote_count': 'vote_count'
}

# Creamos una variable "until_date" con el valor de la fecha actual en formato yyyy-mm-dd
until_date = datetime.now().strftime("%Y-%m-%d")

# Define los valores de los campos
expression_attribute_values = {
    ':val': Decimal(str(element['val'])),
    ':y_m': element['y_m'],
    ':rank': element['rank'],
    ':title': element['title'],
    ':until_date': str(until_date),
    ':original_language': element['original_language'],
    ':original_title': element['original_title'],
    ':overview': element['overview'],
    ':poster_path': element['poster_path'],
    ':adult': element['adult'],
    ':backdrop_path': element['backdrop_path'],
    ':genre_ids': element['genre_ids'],
    ':video': element['video'],
    ':vote_average': Decimal(str(element['vote_average'])),
    ':vote_count': Decimal(str(element['vote_count']))
}

# Crear rank_hist si no existe en el elemento
if 'rank_hist' not in element:
    rank_hist = {year_week: element['rank']}
else:
    # Recuperar el histórico de posiciones de la película
    rank_hist = element['rank_hist']
    # Añadir la nueva entrada al histórico de posiciones
    rank_hist[year_week] = element['rank']

# Añadir la entrada rank_hist a expression_attribute_values
expression_attribute_values[':rank_hist'] = rank_hist

# Actualiza el elemento existente o crea uno nuevo si no existe
response = dynamodb_table.update_item(
    Key=key,
    UpdateExpression=update_expression,
    ExpressionAttributeNames=expression_attribute_names,
    ExpressionAttributeValues=expression_attribute_values,
    ReturnValues='ALL_NEW'
)

```

VALIDACIÓN

- Comprobad que al subir un fichero a S3 con datos de una película se ejecuta la lambda que acabamos de desplegar
- Comprobad que la función lambda se ejecuta sin errores
- Forzad la generación de ficheros en S3 a través de la ejecución de celdas anteriores

- Generad una imagen, con un pantallazo que muestre la monitorización de la función lambda durante los últimos 15 minutos, y guardadla en la ruta (junto al notebook) /img/eje_4_a.png
- Generad una imagen, con un pantallazo que muestre las entradas en la tabla de DynamoDB, y guardadla en la ruta (junto al notebook) /img/eje_4_b.png
- Pegad el contenido final de la lambda en la celda 4.3

CELDA 4.3

```
import boto3
import json
import botocore
from decimal import Decimal
from datetime import datetime

def lambda_handler(event, context):
    NOMBRE_TABLA = "MoviesDB"
    dynamodb_resource = boto3.resource('dynamodb', region_name='us-east-1')
    dynamodb_table = dynamodb_resource.Table(NOMBRE_TABLA)
    s3_resource = boto3.resource('s3', region_name='us-east-1')
    now = datetime.now()
    year_number = now.isocalendar()[0]
    week_number = now.isocalendar()[1]
    year_week = f'{year_number}_{week_number}'

    for record in event.get('Records', []):
        # 1 LEER DATOS DESDE S3
        bucket = record["s3"]["bucket"]["name"]
        key = record["s3"]["object"]["key"]
        obj = s3_resource.Object(bucket, key)
        element = json.load(obj.get()['Body'])
        # COMPLETAD CON CODIGO

        # 3.- MODIFICAD EL FORMATO DE LOS DATOS OBTENIDOS PARA QUE PUEDAN GUARDARSE EN DYNAMO_DB
        element["id"] = str(element["id"])
        element["y_m"] = element["release_date"][:7].replace("-", "_")
        element["val"] = float(element["popularity"])

        # Define la clave de búsqueda
        key = {'id': str(element['id'])}

        # Define la expresión de actualización para los campos que cambian solamente
        update_expression = 'SET #val = :val, #y_m = :y_m, #rank = :rank, #title = :title, #rank_hist = :rank_hist, #until_date = :until_date, #original_language=:original_language, #original_title=:original_title, #overview=:overview, #poster_path=:poster_path, #adult=:adult, #backdrop_path=:backdrop_path, #genre_ids=:genre_ids, #video=:video, #vote_average=:vote_average, #vote_count=:vote_count'

        # Define los nombres de los campos
        expression_attribute_names = {
            '#val': 'val',
            '#y_m': 'y_m',
            '#rank': 'rank',
            '#title': 'title',
            '#rank_hist': 'rank_hist',
            '#until_date': 'until_date',
            '#original_language': 'original_language',
            '#original_title': 'original_title',
            '#overview': 'overview',
            '#poster_path': 'poster_path',
            '#adult': 'adult',
            '#backdrop_path': 'backdrop_path',
            '#genre_ids': 'genre_ids',
            '#video': 'video',
            '#vote_average': 'vote_average',
            '#vote_count': 'vote_count'
        }

        # Creamos una variable "until_date" con el valor de la fecha actual en formato yyyy-mm-dd
        until_date = datetime.now().strftime("%Y-%m-%d")

        # Define los valores de los campos
        expression_attribute_values = {
            ':val': Decimal(str(element['val'])),
            ':y_m': element['y_m'],
            ':rank': element['rank'],
            ':title': element['title'],
            ':until_date': str(until_date),
        }
```

```

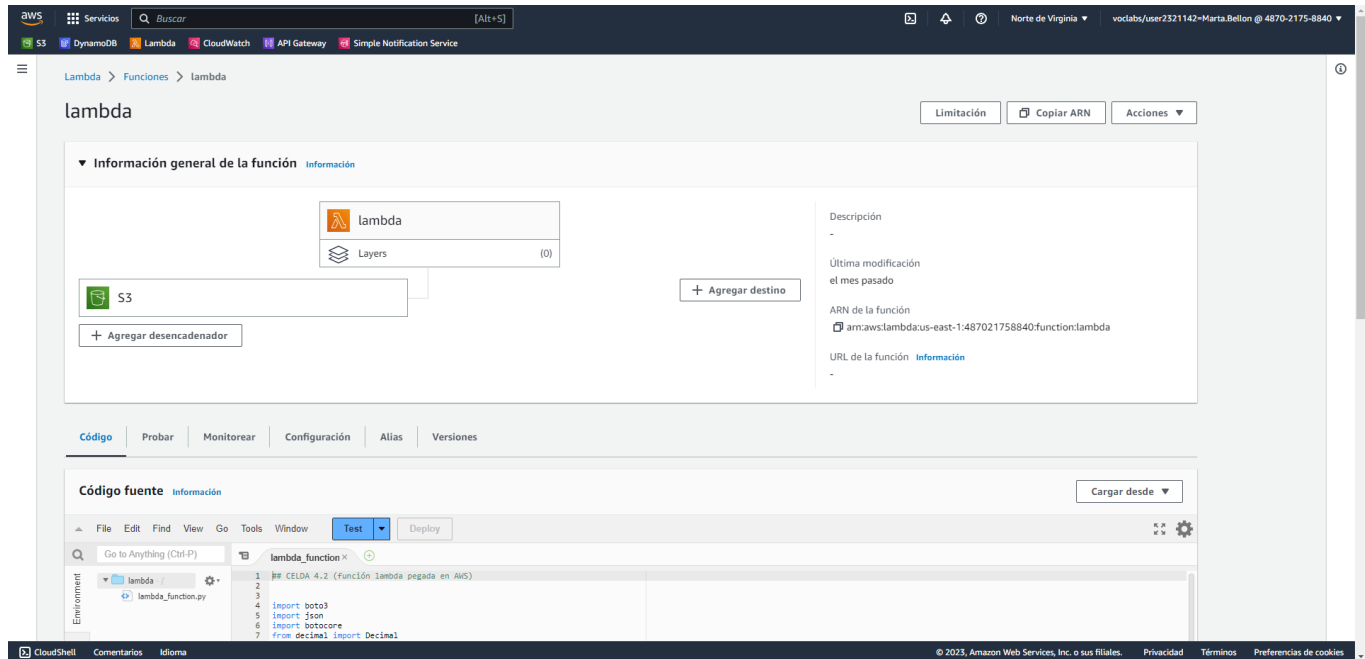
        ':original_language': element['original_language'],
        ':original_title': element['original_title'],
        ':overview': element['overview'],
        ':poster_path': element['poster_path'],
        ':adult': element['adult'],
        ':backdrop_path': element['backdrop_path'],
        ':genre_ids': element['genre_ids'],
        ':video': element['video'],
        ':vote_average': Decimal(str(element['vote_average'])),
        ':vote_count': Decimal(str(element['vote_count']))
    }

    # Crear rank_hist si no existe en el elemento
    if 'rank_hist' not in element:
        rank_hist = {year_week: element['rank']}
    else:
        # Recuperar el histórico de posiciones de la película
        rank_hist = element['rank_hist']
        # Añadir la nueva entrada al histórico de posiciones
        rank_hist[year_week] = element['rank']

    # Añadir la entrada rank_hist a expression_attribute_values
    expression_attribute_values[':rank_hist'] = rank_hist

    # Actualiza el elemento existente o crea uno nuevo si no existe
    response = dynamodb_table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues='ALL_NEW'
    )

```



aws

Servicios

Buscar

[Alt+S]

DynamoDB

Lambda

CloudWatch

API Gateway

Simple Notification Service

Norte de Virginia

vociabo/user2321142-Marta Bellon @ 4870-2175-8840

DynamoDB

Explorar elementos

MoviesDB

Vista previa automática

Ver los detalles de la tabla

Escanear o consultar elementos

Expanda para consultar o examinar elementos.

Elementos devueltos (20)

Acciones

Crear elemento

< 1 >

<input type="checkbox"/>	id	adult	backdrop_path	genre_ids	original_language	original_title	overview	popularity	poster_path	rank
<input type="checkbox"/>	697843	false	/xOp7TbCOke4L...	[[{"N": "28"}, {"N": "53"}]]	en	Extraction 2	Tasked with...		/7gKt9hpEMcZUQpK9kDzJpbNt5.jpg	2
<input type="checkbox"/>	1098160	false	/baE88dSR0byN...	[[{"N": "27"}, {"N": "53"}]]	en	The Tank	In 1978 Ore...		/2VxEtvgzOUJukatIZIKGn4borpgE.jpg	16
<input type="checkbox"/>	667538	false	/9NgtkUfLm9c...	[[{"N": "28"}, {"N": "12"}, {"N": "878"}]]	en	Transformers: Ris...	When a ne...		/gPbM0MK8CP8A174mJwGsADNYKD.jpg	5
<input type="checkbox"/>	840326	false	/94TIUEhuwv9P...	[[{"N": "28"}, {"N": "10752"}]]	fi	Sisu	Deep in the...		/ygO9l0wFMWymATCrhoQXk6gCEh.jpg	13
<input type="checkbox"/>	799379	false	/xKxv1WOIKfA...	[[{"N": "28"}, {"N": "53"}, {"N": "127"}]]	ko	복대사냥	While unde...		/dnWicB6fa7NvpGbguxWINPMc5f.jpg	14
<input type="checkbox"/>	884605	false	/cKE9qZqYf4ji...	[[{"N": "35"}, {"N": "10749"}]]	en	No Hard Feelings	On the brin...		/5xeNPGbM8lmVdJACUoGpXTBPx3.jpg	15
<input type="checkbox"/>	829051	false	/94eM5Yjge3IQ...	[[{"N": "35"}]]	en	About My Father	Encouraged...		/hQU7280QVv17pYMHyLzdNGVidBF.jpg	20
<input type="checkbox"/>	976573	false	/c5YLX73WkuC...	[[{"N": "16"}, {"N": "35"}, {"N": "10751"}, {"...	en	Elemental	In a city wh...		/6oH378KUfCEitLJkm07r97L0Rz.jpg	8
<input type="checkbox"/>	298618	false	/7e9MVGg8eFO...	[[{"N": "878"}, {"N": "28"}, {"N": "12"}]]	en	The Flash	When his at...		/ktdFPbFhUbaZ6OOOKxcv0Bm.jpg	7
<input type="checkbox"/>	536437	false	/QST84l8h4N2...	[[{"N": "9648"}, {"N": "53"}, {"N": "878"}]]	en	Hypnotic	A detective ...		/3hGkklwXguTlccgSISXUJZOW.jpg	6
<input type="checkbox"/>	882569	false	/lQwW349qR0g...	[[{"N": "10752"}, {"N": "28"}, {"N": "53"}]]	en	Guy Ritchie's The ...	During the ...		/KVG8zFFrpyYLoHCHuEeOGAd6Rui.jpg	10

CloudShell

Comentarios

Idioma

© 2023, Amazon Web Services, Inc. o sus filiales.

Privacidad

Términos

Preferencias de cookies

aws

Servicios

Buscar

[Alt+S]

DynamoDB

Lambda

CloudWatch

API Gateway

Simple Notification Service

Norte de Virginia

vociabo/user321142-Marta Bellon @ 4870-2175-8840

Editar elemento

Puede agregar, eliminar o editar los atributos de un elemento. Es posible anidar atributos dentro de otros atributos hasta 12 niveles de profundidad. Más información

Formulario

Vista JSON

Atributos

Agregar nuevo atributo

Nombre de atributo	Valor	Tipo	
id - Clave de partición	697843	Cadena	
adult	<input type="radio"/> Verdadero <input checked="" type="radio"/> Falso	Booleano	Eliminar
backdrop_path	/xOp7TbCOke4L...	Cadena	Eliminar
genre_ids	<div>Insertar un campo</div>	Lista	Eliminar
	0 28	Número	Eliminar
	1 53	Número	Eliminar
original_language	en	Cadena	Eliminar
original_title	Extraction 2	Cadena	Eliminar
overview	Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrates one of the world's deadliest prisons in order to save them. But when the extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rake and his team	Cadena	Eliminar
poster_path	/7gKt9hpEMcZUQpK9kDzJpbNt5.jpg	Cadena	Eliminar
rank	2	Número	Eliminar
rank_list	<div>Insertar un campo</div>	Lista	Eliminar
title	Extraction 2	Cadena	Eliminar
release_date	2023-06-25	Cadena	Eliminar
vote	3872.49	Número	Eliminar
video	<input type="radio"/> Verdadero <input checked="" type="radio"/> Falso	Booleano	Eliminar
vote_average	7.8	Número	Eliminar
vote_count	758	Número	Eliminar

CloudShell

Comentarios

Idioma

© 2023, Amazon Web Services, Inc. o sus filiales.

Privacidad

Términos

Preferencias de cookies

Screenshot of the AWS Lambda console showing the 'Información general de la función' (General function information) page for a function named 'lambda'. The page displays the function's configuration, including the runtime (Python 3.9), the handler, and the execution role. It also shows the function's status as 'Probar' (Test) and 'Monitorear' (Monitor). The 'Monitorear' tab is selected, showing the 'CloudWatch Logs' section. The logs display a table of recent invocations with columns for timestamp, request ID, log stream, duration, billed duration, memory size, and memory used.

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySizeInMB	MemoryUsedInMB
1	2023-08-25T13:34:40.220Z	115d0e27-716d-4ca3-bf97-c11c55a8efef	2023/08/25/[lambda-test]0b1a726957408121f209f22afce4c	678.26	679.0	128	49
2	2023-08-25T13:34:40.212Z	558dc088-27cf-4a8c-b452-1c6f1ec2ce00	2023/08/25/[lambda-test]443f3a4080c4f2d074247678a4a025	669.62	670.0	128	92
3	2023-08-25T13:34:40.208Z	476d2d0-1342-4a2d-935d-108b1114d2d0	2023/08/25/[lambda-test]7f98470993042c48a083081c2b04e4	666.38	667.0	128	92
4	2023-08-25T13:34:40.186Z	9f791a08-dc6c-4a5a-a0f5-eb42a3b39564	2023/08/25/[lambda-test]5d825f2c7f9a4080b5a178770f8eF	659.98	660.0	128	91
5	2023-08-25T13:34:40.184Z	d629c8e5-687f-408a-81d2-2986cd811ed	2023/08/25/[lambda-test]3d6c956c780b4155931f70b08216e522	644.66	645.0	128	91
6	2023-08-25T13:34:40.169Z	3407efc8-a640-4725-8793-2706122cc04	2023/08/25/[lambda-test]3f78062d4fa341a2ac1c0b0911cf07c	639.24	640.0	128	49
7	2023-08-25T13:34:40.165Z	80909e01-a037-4a0e-8055-7420f51260e8	2023/08/25/[lambda-test]30950f180f5a21915f90c584c1844	638.40	639.0	128	91
8	2023-08-25T13:34:40.162Z	637f20aa-8d72-4a2f-829e-467026f20b07	2023/08/25/[lambda-test]3842c580f70b40478a08200f70a0b87	628.42	621.0	128	45
9	2023-08-25T13:34:40.156Z	e62d2b38-68f7-66f7-bebc-1c742d677477	2023/08/25/[lambda-test]0217d1d44c048f8b77310310017c2c	627.28	618.0	128	49

EJERCICIO 5: Automatización de la captura de datos

Una vez tenemos todas las piezas del proceso, queremos proceder a automatizar la captura de datos, lo cual realizaremos a través de una función lambda desplegada en la nube.

Esta función lambda tendrá un código similar al encargado de consultar el API de películas y generar nuevos objetos en S3. Una vez desplegada, nos encargaremos de añadir un *trigger* que la ejecute cada 24 horas.

Para que la función tenga un comportamiento correcto, necesitaremos que maneje dependencias adicionales para usar la librería *requests*, lo cual puede realizarse a través de una *layer*. Además, puede que necesitemos modificar los parámetros por defecto (timeout, memoria, ...)

Se deben realizar las siguientes acciones:

- Desplegad una función lambda, similar a la utilizada en el ejercicio 2, que consulte el API de películas y genere como resultado objetos .json en S3
- Resolved los problemas que permitan utilizar la librería *requests* a través del uso de la *layer*
- Modificad los parámetros de la lambda timeout/memoria en caso que de problemas de ejecución
- Añadir un *trigger* para que la función se ejecute cada día

Ejemplo de generación de la layer en linux/mac (revisad en Windows)

```
$ virtualenv --python="/usr/bin/python3.9" .venv
$ source .venv/
$ source .venv/bin/activate
$ mkdir python
$ cd python/
$ pip install requests -t ./
$ cd ..
$ zip -r python_modules.zip .
```

VALIDACIÓN

- Tras todos los pasos, comprobad que la función se ejecuta sin problemas
- Modificad el trigger para que la función se ejecute cada minuto y esperad 5 minutos
- Generad una imagen con un pantallazo que muestre la monitorización de la función lambda durante los últimos 5 minutos, y guardadla en la ruta (junto al notebook) /img/eje_5.png
- Modificad el trigger para que la función se ejecute cada día

```
In [9]: import requests
import json
import boto3
from datetime import datetime

NOMBRE_BUCKET = "mcidaen3.capstone11.movies.raw.data.mbc"
API_KEY = 'cc48d676183a961ea3d6c4299841b322'
now = datetime.now()
year_month_day = now.strftime("%Y_%m_%d")
results = ""

def lambda_handler(event, context):
    # 1. Obtenngo los datos de la api
    response = requests.get(f"https://api.themoviedb.org/3/movie/now_playing?api_key={API_KEY}")

    if response.status_code == 200:
        # Resultados de la API
        results = response.json()['results']
        # Añado el ranking a cada resultado
        for i, result in enumerate(results):
            result['rank'] = i + 1
    else:
        print(f"Error al obtener los datos: {response.status_code}")
        return {"statusCode": response.status_code, "body": response.text}

    # 2. Configuro el cliente, el recurso de acceso a S3 y valido su uso
    s3_client = boto3.client('s3', region_name='us-east-1')
    try:
        s3_client.put_object(Bucket=NOMBRE_BUCKET, Key='movies/')
        print('La carpeta se ha creado correctamente.')
    except Exception as e:
        print(e)
        return {"statusCode": 500, "body": str(e)}

    # 3. Subo a S3 un objeto por resultado obtenido en el paso 1
    for result in results:
        movie_id = str(result['id'])
        file_name = f"movies/{movie_id}/{year_month_day}.json"
        try:
            s3_client.put_object(Bucket=NOMBRE_BUCKET, Key=file_name, Body=json.dumps(result))
            print(f'{file_name} se ha añadido a S3.')
        except Exception as e:
            print(e)
            return {"statusCode": 500, "body": str(e)}

    return {"statusCode": 200, "body": "Información guardada en S3."}
```

The screenshot displays the AWS Lambda console interface. At the top, the navigation bar shows various AWS services. The main content area is titled 'lambda2' and includes a 'Limitación' (Quota) button and a 'Copiar ARN' (Copy ARN) button. Below this, the 'Información general de la función' (Function general information) section is expanded, showing the function name 'lambda2', its layers (1), and its description. The 'EventBridge (CloudWatch Events)' section is also visible, with a button to 'Agregar desencadenador' (Add trigger). The 'Código' (Code) tab is selected, showing the function's code. Below the code, the 'Registros' (Logs) tab is active, displaying the 'CloudWatch Logs' section. This section includes a table of 'Recent invocations' with columns for #, Timestamp, RequestID, LogStream, DurationInMS, BilledDurationInMS, MemoryUsedInMB, and MemoryUsedInMB. The bottom of the console shows the footer with copyright information and links to Privacy, Terms, and Preferences.

aws Servicios Q Buscar [Alt+S]

S3 DynamoDB Lambda CloudWatch API Gateway Simple Notification Service

Lambda > Funciones > lambda2

Limitación Copiar ARN Acciones

Información general de la función Información

Código Probar **Monitorear** Configuración Alias Versiones

Métricas **Registros** Rastros

Ver registros de CloudWatch Ver rastros de X-Ray Ver Lambda Insights Ver perfiles de CodeGuru

CloudWatch Logs Información

Lambda registra todas las solicitudes gestionadas por la función y almacena automáticamente los registros generados por el código a través de Amazon CloudWatch Logs. Para validar el código, debe instrumentarlo con instrucciones de registro personalizadas. En las tablas siguientes se muestran las invocaciones de funciones más recientes y más caras de toda la actividad de las funciones. Para ver los registros correspondientes a un alias o una versión específicos de una función, visite la sección de Monitoreo en ese nivel.

1h 3h 12h 1d 3d 1sem. Personalizado (15m)

Recent invocations

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
1	2023-06-25T13:38:38.880Z	13943180-8776-413F-86C4-1F1C14266F	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1488.44	1489.0	128	49
2	2023-06-25T13:37:38.880Z	31281442-7316-4268-8c73-88186c3379c0	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1423.4	1429.0	128	49
3	2023-06-25T13:36:38.951Z	74042360-fd29-4052-a183-915c4e05f6e6	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1444.61	1445.0	128	49
4	2023-06-25T13:35:38.847Z	facdc08e-nf11-4741-8d5f-9d4461f0800	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1468.87	1461.0	128	49
5	2023-06-25T13:34:38.847Z	254e9d37-8b04-4f4a-d8e9-4534f6528970	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1412.39	1413.0	128	49
6	2023-06-25T13:33:38.886Z	5f31e615-3c88-4cc2-b0f0-5a148c7c8f2d	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1457.23	1458.0	128	49
7	2023-06-25T13:32:38.788Z	3af21a20-fb8d-4563-9924-610b9e080bcf	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1485.56	1486.0	128	49
8	2023-06-25T13:31:38.568Z	3764c932-198c-489f-8e48-4112c8c3d3d0	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1497.74	1498.0	128	49
9	2023-06-25T13:30:38.164Z	872b4293-3980-4968-5674c769388a	2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39	1539.37	1548.0	128	49

Most expensive invocations in GB-seconds (memory assigned * billed duration)

#	Timestamp	RequestID	LogStream	BilledDurationInMS	MemorySetInMB	BilledDurationInGBSeconds
---	-----------	-----------	-----------	--------------------	---------------	---------------------------

aws Servicios Q Buscar [Alt+S]

S3 DynamoDB Lambda CloudWatch API Gateway Simple Notification Service

CloudWatch > Grupos de registros > /aws/lambda/lambda2 > 2023/06/25/[LATEST]e4e2b8cc8994696837f8789f3077f39

Eventos de registro

Puede utilizar la barra de filtros a continuación para buscar y hacer coincidir términos, frases o valores en sus eventos de registro. Más información sobre los patrones de filtro

Buscar eventos

Borr ar 1 30 1 12 Personalizado (15m) Mostrar

Marca temporal Mensaje

No more records within selected time range [Volver a intentar](#)

2023-06-25T15:26:37.475+02:00	START RequestID: a7f8398b-bc77-4209-9d64-3269d729d7fc Version: SLATEST
2023-06-25T15:26:37.935+02:00	La carpeta se ha creado correctamente.
2023-06-25T15:26:37.981+02:00	movies/385687/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.019+02:00	movies/697843/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.065+02:00	movies/569894/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.101+02:00	movies/582356/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.144+02:00	movies/667938/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.188+02:00	movies/536437/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.236+02:00	movies/298618/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.281+02:00	movies/976573/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.326+02:00	movies/898771/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.374+02:00	movies/882569/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.416+02:00	movies/447277/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.457+02:00	movies/812225/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.508+02:00	movies/848326/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.539+02:00	movies/799379/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.588+02:00	movies/884685/2023_06_25.json se ha añadido a S3.
2023-06-25T15:26:38.626+02:00	movies/1895160/2023_06_25.json se ha añadido a S3.

EJERCICIO 6: API listado películas (dato mes y año)

Durante este ejercicio, desarrollaremos una función lambda ejecutada tras la invocación de un endpoint de API Gateway.

Para ello, generaremos una lambda que a partir de eventos con el siguiente formato: `{ 'pathParameters': { 'month': '1', 'year': '2023' } }`, genere un listado adecuado. Dicho listado se obtiene a partir de una consulta a la base de datos DynamoDB usando las siguientes características:

- Consulta de tipo *query*
- Consulta realizada sobre el índice secundario global
 - Se debe proporcionar un valor para el campo `y_m` para obtener los datos correctos
- Ordenación descendente, de forma que se obtengan en las primeras posiciones las películas con mayor valoración
- Limitar la consulta a 10 películas

Para que el resultado (listado de películas) pueda interpretarse correctamente por API Gateway, lo incluiremos como json en la respuesta de la lambda, en concreto en el campo `body` (comprobad el contenido del código de la celda 6.1).

Una vez desarrollada la lambda, la podemos validar en local realizando la siguiente llamada:

```
lambda_handler({'pathParameters': {'month': '12', 'year': '2022'}}, None)
```

Tras la validación de la lambda, desplegaremos un API a través de API Gateway que permita su invocación. Para ello, definiremos 1 recurso base llamado `/list` sobre el cual añadiremos 1 subrecurso `/list/{year}` y un (sub)subrecurso `/list/{year}/{month}`. Sobre el último subrecurso (`{month}`) añadiremos un método GET, el cual conectará las llamadas realizadas sobre

`API_URL/stage/list/{year}/{month}` con la lambda anterior, donde `{year}` y `{month}` serán reemplazados por valores reales.

El evento recibido por la lambda tendrá el formato previamente definido

```
{'pathParameters': {'month': 'MONTH', 'year': 'YEAR'}}
```

Se deben realizar las siguientes acciones:

- Desarrollar una lambda a partir de la siguiente celda de código, de forma que al ejecutarla con el evento previamente definido, realice una consulta a DynamoDB
- Validar la lambda desplegándola en AWS e invocándola con un evento compatible
- Desplegar un API en API Gateway (API REST no privado) con los recursos/métodos ya definidos
 - Árbol de recursos `/list/{year}/{month}`
 - Método: GET sobre `{month}`, invocando la lambda previamente desplegada, y activando la opción *lambda proxy integration*
- Validar el API de manera preliminar realizando llamadas a través de herramientas tipo POSTMAN
 - Posteriormente validaremos el API al completo

```
## He creado la lambda y la API siguiendo todos los pasos acordados en las clases.
## Al llamar a la función, me devuelve una lista en blanco pese a que en MoviesDB si existen archivos con
dicho mes y año.
## He revisado todo muchas veces e incluso he creado otras tablas y lambdas (con el mismo fallo), pero no
encuentro el
# porque del listado vacío.
```

In [10]: # CELDA 6.1

```
import boto3
import decimal
import json
from boto3.dynamodb.conditions import Key

NOMBRE_TABLA = "MoviesDB"
dynamo_resource = boto3.resource('dynamodb', region_name='us-east-1')
dynamo_table = dynamo_resource.Table(NOMBRE_TABLA)

class DecimalEncoder(json.JSONEncoder):      # Necesario para manejar los tipos Decimal
    def default(self, o):
        if isinstance(o, decimal.Decimal):
            return str(o)
        return super(DecimalEncoder, self).default(o)

def lambda_handler(event, context):
    year = int(event['pathParameters']['year'])
    month = int(event['pathParameters']['month'])
    y_m = f'{year}_{month:02d}'

    # REALIZAR CONSULTA A DYNAMO DB CON YEAR_MONTH, Y DEJAR EL RESULTADO (Lista) EN LA VARIABLE ITEMS

    response = dynamo_table.query(
        IndexName='y_m-index',
        KeyConditionExpression=Key('y_m').eq(y_m),
        ScanIndexForward=False,
        Limit=10
    )

    items = response['Items']

    return {
        'statusCode': 200,
        'body': json.dumps(items, cls=DecimalEncoder)
    }
```


In [11]: # VALIDACIÓN

```
event = {'pathParameters': {'month': '06', 'year': '2023'}}
lambda_handler(event, {})
```

```
Out[11]: {'statusCode': 200,
  'body': '[{"genre_ids": ["16", "14", "28", "12"], "original_title": "\\u6620\\u753b \\u30d6\\u30e9\\u30c3\\u30af\\u30af\\u30ed\\u30fc\\u30d0\\u30fc \\u9b54\\u6cd5\\u5e1d\\u306e\\u5263", "val": "889.941", "adult": false, "y_m": "2023_06", "overview": "As a lionhearted boy who can\\u2019t wield magic strives for the title of Wizard King, four banished Wizard Kings of yore return to crush the Clover Kingdom.", "vote_average": "8.6", "rank": "12", "rank_hist": {"2023_25": "12"}, "backdrop_path": "/rogeBJK44LtWynOqzMFmEF30T80.jpg", "original_language": "ja", "vote_count": "108", "until_date": "2023-06-25", "id": "812225", "poster_path": "/9YEGawvjaRgnyW6QVcUhFJPFdco.jpg", "video": false, "title": "Black Clover: Sword of the Wizard King"}, {"genre_ids": ["878", "28", "12"], "original_title": "The Flash", "val": "1230.087", "adult": false, "y_m": "2023_06", "overview": "When his attempt to save his family inadvertently alters the future, Barry Allen becomes trapped in a reality in which General Zod has returned and there are no Super Heroes to turn to. In order to save the world that he is in and return to the future that he knows, Barry\\'s only hope is to race for his life. But will making the ultimate sacrifice be enough to reset the universe?", "vote_average": "6.8", "rank": "7", "rank_hist": {"2023_25": "7"}, "backdrop_path": "/7e9MVGg8efOhoA2R9XhZcGwTC5Z.jpg", "original_language": "en", "vote_count": "554", "until_date": "2023-06-25", "id": "298618", "poster_path": "/rktDFPbfHfUbaRZ6000KsXcv0Bm.jpg", "video": false, "title": "The Flash"}, {"genre_ids": ["16", "35", "10751", "14", "10749"], "original_title": "Elemental", "val": "1146.21", "adult": false, "y_m": "2023_06", "overview": "In a city where fire, water, land and air reside side by side, a fiery young woman and a go-with-the-flow guy will discover something elemental: how much they have in common.", "vote_average": "7.5", "rank": "8", "rank_hist": {"2023_25": "8"}, "backdrop_path": "/cSYLX73WskxCgvpN3MtrKYUSj1T.jpg", "original_language": "en", "vote_count": "122", "until_date": "2023-06-25", "id": "976573", "poster_path": "/6oH378KUfCEitzJkm07r97L0RsZ.jpg", "video": false, "title": "Elemental"}, {"genre_ids": ["35", "10749"], "original_title": "No Hard Feelings", "val": "516.859", "adult": false, "y_m": "2023_06", "overview": "On the brink of losing her childhood home, Maddie discovers an intriguing job listing: wealthy helicopter parents looking for someone to \\u201cdate\\u201d their introverted 19-year-old son, Percy, before he leaves for college. To her surprise, Maddie soon discovers the awkward Percy is no sure thing.", "vote_average": "6.2", "rank": "15", "rank_hist": {"2023_25": "15"}, "backdrop_path": "/cKE9qZqYtF4jimf0GF0qKfy7NEU.jpg", "original_language": "en", "vote_count": "42", "until_date": "2023-06-25", "id": "884605", "poster_path": "/5xeNPGbM8ImVdJACUoGpXT8Pxx3.jpg", "video": false, "title": "No Hard Feelings"}, {"genre_ids": ["28", "12", "878"], "original_title": "Transformers: Rise of the Beasts", "val": "1744.341", "adult": false, "y_m": "2023_06", "overview": "When a new threat capable of destroying the entire planet emerges, Optimus Prime and the Autobots must team up with a powerful faction known as the Maximals. With the fate of humanity hanging in the balance, humans Noah and Elena will do whatever it takes to help the Transformers as they engage in the ultimate battle to save Earth.", "vote_average": "7.2", "rank": "5", "rank_hist": {"2023_25": "5"}, "backdrop_path": "/9NgktUFLm9cnFDfaekx2R0h84f.jpg", "original_language": "en", "vote_count": "410", "until_date": "2023-06-25", "id": "667538", "poster_path": "/gPbM0MK8CP8A174rmUwGsADNYKD.jpg", "video": false, "title": "Transformers: Rise of the Beasts"}, {"genre_ids": ["28", "53"], "original_title": "Extraction 2", "val": "3872.49", "adult": false, "y_m": "2023_06", "overview": "Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrates one of the world\\'s deadliest prisons in order to save them. But when the extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rake and his team to Sydney, in order to get revenge.", "vote_average": "7.8", "rank": "2", "rank_hist": {"2023_25": "2"}, "backdrop_path": "/xXp7TbCOke4lB65ngkt3CuhsiAa.jpg", "original_language": "en", "vote_count": "738", "until_date": "2023-06-25", "id": "697843", "poster_path": "/7gKI9hpEMcZUQpNgKrkDzJpbnNS.jpg", "video": false, "title": "Extraction 2"}]]}]}
```

```
In [12]: # VALIDACIÓN DE LA API

import requests
URL = 'https://nawg4eewpk.execute-api.us-east-1.amazonaws.com/prod/list/2023/06'

response = requests.get(f'{URL}').json()
response
```

```

Out[12]: [{ 'genre_ids': ['16', '14', '28', '12'],
  'original_title': '映画 ブラッククロバー 魔法帝の剣',
  'val': '889.941',
  'adult': False,
  'y_m': '2023_06',
  'overview': 'As a lionhearted boy who can't wield magic strives for the title of Wizard King, four banished Wizard Kings of yore return to crush the Clover Kingdom.',
  'vote_average': '8.6',
  'rank': '12',
  'rank_hist': {'2023_25': '12'},
  'backdrop_path': '/rogeBJK44LtWynOqzMfMEF30T80.jpg',
  'original_language': 'ja',
  'vote_count': '108',
  'until_date': '2023-06-25',
  'id': '812225',
  'poster_path': '/9YEGawvjaRgnyW6QVcUhFJFPDco.jpg',
  'video': False,
  'title': 'Black Clover: Sword of the Wizard King'},
  { 'genre_ids': ['878', '28', '12'],
  'original_title': 'The Flash',
  'val': '1230.087',
  'adult': False,
  'y_m': '2023_06',
  'overview': "When his attempt to save his family inadvertently alters the future, Barry Allen becomes trapped in a reality in which General Zod has returned and there are no Super Heroes to turn to. In order to save the world that he is in and return to the future that he knows, Barry's only hope is to race for his life. But will making the ultimate sacrifice be enough to reset the universe?",
  'vote_average': '6.8',
  'rank': '7',
  'rank_hist': {'2023_25': '7'},
  'backdrop_path': '/7e9MVGg8ef0hoA2R9XhZcGWT5Z.jpg',
  'original_language': 'en',
  'vote_count': '554',
  'until_date': '2023-06-25',
  'id': '298618',
  'poster_path': '/rktDFPbHfUbaArZ6000KsXcv0Bm.jpg',
  'video': False,
  'title': 'The Flash'},
  { 'genre_ids': ['16', '35', '10751', '14', '10749'],
  'original_title': 'Elemental',
  'val': '1146.21',
  'adult': False,
  'y_m': '2023_06',
  'overview': 'In a city where fire, water, land and air residents live together, a fiery young woman and a go-with-the-flow guy will discover something elemental: how much they have in common.',
  'vote_average': '7.5',
  'rank': '8',
  'rank_hist': {'2023_25': '8'},
  'backdrop_path': '/cSYLX73WskxCgvpN3MtRkYUSj1T.jpg',
  'original_language': 'en',
  'vote_count': '122',
  'until_date': '2023-06-25',
  'id': '976573',
  'poster_path': '/6oH378KUfCEitzJkm07r97L0RsZ.jpg',
  'video': False,
  'title': 'Elemental'},
  { 'genre_ids': ['35', '10749'],
  'original_title': 'No Hard Feelings',
  'val': '516.859',
  'adult': False,
  'y_m': '2023_06',
  'overview': 'On the brink of losing her childhood home, Maddie discovers an intriguing job listing: wealthy helicopter parents looking for someone to "date" their introverted 19-year-old son, Percy, before he leaves for college. To her surprise, Maddie soon discovers the awkward Percy is no sure thing.',
  'vote_average': '6.2',
  'rank': '15',
  'rank_hist': {'2023_25': '15'},
  'backdrop_path': '/cKE9qZqYtF4jimf0GFOqKfy7NEU.jpg',
  'original_language': 'en',
  'vote_count': '42',
  'until_date': '2023-06-25',
  'id': '884605',
  'poster_path': '/5xeNPGbM8ImVdJACUoGpXT8Pxx3.jpg',
  'video': False,
  'title': 'No Hard Feelings'},
  { 'genre_ids': ['28', '12', '878'],

```

```

'original_title': 'Transformers: Rise of the Beasts',
'val': '1744.341',
'adult': False,
'y_m': '2023_06',
'overview': 'When a new threat capable of destroying the entire planet emerges, Optimus Prime and the Autobots must team up with a powerful faction known as the Maximals. With the fate of humanity hanging in the balance, humans Noah and Elena will do whatever it takes to help the Transformers as they engage in the ultimate battle to save Earth.',
'vote_average': '7.2',
'rank': '5',
'rank_hist': {'2023_25': '5'},
'backdrop_path': '/9NgktUFLm9cnFDfaekx2R0h84f.jpg',
'original_language': 'en',
'vote_count': '410',
'until_date': '2023-06-25',
'id': '667538',
'poster_path': '/gPbM0MK8CP8A174rmUwGsADNYKD.jpg',
'video': False,
'title': 'Transformers: Rise of the Beasts'},
{'genre_ids': ['28', '53'],
'original_title': 'Extraction 2',
'val': '3872.49',
'adult': False,
'y_m': '2023_06',
'overview': "Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrates one of the world's deadliest prisons in order to save them. But when the extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rake and his team to Sydney, in order to get revenge.",
'vote_average': '7.8',
'rank': '2',
'rank_hist': {'2023_25': '2'},
'backdrop_path': '/xXp7TbC0Ke4lB65ngkt3CuhsiAa.jpg',
'original_language': 'en',
'vote_count': '738',
'until_date': '2023-06-25',
'id': '697843',
'poster_path': '/7gKI9hpEMcZUQpNgKrkdZJpbnNS.jpg',
'video': False,
'title': 'Extraction 2'}}]

```

EJERCICIO 7: API detalles película (dado su id)

Durante este ejercicio, desarrollaremos otra función lambda ejecutada tras la invocación de otro endpoint de API Gateway.

Para ello, generaremos una lambda (desarrollada inicialmente en la celda 7.1) que a partir de eventos con el siguiente formato:

`{'pathParameters': {'id': '76600'}}`, obtenga detalles para la película con el id proporcionado. Los detalles se obtienen a partir de una consulta a la base de datos DynamoDB usando las siguientes características:

- Consulta de tipo *get_item()*
- Consulta realizada sobre la tabla (usando la clave de partición)
 - Se debe proporcionar un valor para el campo id

Para que el resultado (detalles de la película) pueda interpretarse correctamente por API Gateway, lo incluiremos como json en la respuesta de la lambda, en concreto en el campo `body` (comprobad el contenido del código incluido).

Una vez desarrollada la lambda, la podemos validar en local realizando la siguiente llamada:

```
lambda_handler({'pathParameters': {'id': '76600'}}, None)
```

Tras la validación de la lambda, desplegaremos un API a través de API Gateway que permita su invocación. Para ello, definiremos 1 nuevo recurso base llamado `/movies` sobre el cual añadiremos 1 subrecurso `/movies/{id}`. Sobre el subrecurso (*{id}*) añadiremos un método GET, el cual conectará las llamadas realizadas sobre `API_URL/stage/movies/{id}` con la lambda anterior, donde *{id}* será reemplazado por un valor real.

El evento recibido por la lambda tendrá el formato previamente definido

```
{'pathParameters': {'id': 'MOVIE_ID'}}
```

Se deben realizar las siguientes acciones:

- Desarrollad una lambda a partir de la siguiente celda de código, de forma que al ejecutarla con el evento previamente definido, realice una consulta a DynamoDB
- Validad la lambda desplegándola en AWS e invocándola con un evento compatible
- Cread los nuevos recursos/métodos sobre el API de API Gateway previamente desplegado

- Arbol de recursos /movies/{id}
- Método: GET sobre {id}, invocando la lambda previamente desplegada, y activando la opción *lambda proxy integration*
- Desplegad de nuevo el API usando la opción implementar API
- Validad el API de manera preliminar realizando llamadas a través de herramientas tipo POSTMAN
 - Posteriormente validaremos el API al completo

In [13]: # CELDA 7.1

```
import boto3
import decimal
import json
from boto3.dynamodb.conditions import Key

NOMBRE_TABLA = "MoviesDB"
dynamo_resource = boto3.resource('dynamodb', region_name='us-east-1')
dynamo_table = dynamo_resource.Table(NOMBRE_TABLA)

class DecimalEncoder(json.JSONEncoder):      # Necesario para manejar Los tipos Decimal
    def default(self, o):
        if isinstance(o, decimal.Decimal):
            return str(o)
        return super(DecimalEncoder, self).default(o)

def lambda_handler(event, context):
    movie_id = event['pathParameters']['id']
    # REALIZAR CONSULTA A DYNAMO DB CON movie_id, Y DEJAR EL RESULTADO (DICT) EN LA VARIABLE ITEM
    response = dynamo_table.get_item(
        Key={'id': movie_id}
    )
    item = response.get('Item', {})

    return {
        'statusCode': 200,
        'body': json.dumps(item, cls=DecimalEncoder)
    }
```

In [14]: # VALIDACIÓN

```
event = {'pathParameters': {'id': '697843'}}
print(lambda_handler(event, {}))
```

```
{'statusCode': 200, 'body': '{"genre_ids": ["28", "53"], "original_title": "Extraction 2", "val": "3872.49",
"adult": false, "y_m": "2023_06", "overview": "Tasked with extracting a family who is at the mercy of a Georgi
an gangster, Tyler Rake infiltrates one of the world\'s deadliest prisons in order to save them. But when the
extraction gets hot, and the gangster dies in the heat of battle, his equally ruthless brother tracks down Rak
e and his team to Sydney, in order to get revenge.", "vote_average": "7.8", "rank": "2", "rank_hist": {"2023_2
5": "2"}, "backdrop_path": "/xXp7TbC0Ke4lB65ngkt3CuhsiAa.jpg", "original_language": "en", "vote_count": "738",
"until_date": "2023-06-25", "id": "697843", "poster_path": "/7gKI9hpEMcZUQpNgKrKdZJpbnNS.jpg", "video": false,
"title": "Extraction 2"}'}
```

```
In [15]: # VALIDACIÓN DE LA API

import requests
URL = 'https://nawg4eewpk.execute-api.us-east-1.amazonaws.com/prod/movies/697843' ##### 'https://n87f3s9pte.e

response = requests.get(f'{URL}').json()
response
```

```
Out[15]: {'genre_ids': ['28', '53'],
'original_title': 'Extraction 2',
'val': '3872.49',
'adult': False,
'y_m': '2023_06',
'overview': "Tasked with extracting a family who is at the mercy of a Georgian gangster, Tyler Rake infiltrat
es one of the world's deadliest prisons in order to save them. But when the extraction gets hot, and the gangs
ter dies in the heat of battle, his equally ruthless brother tracks down Rake and his team to Sydney, in order
to get revenge.",
'vote_average': '7.8',
'rank': '2',
'rank_hist': {'2023_25': '2'},
'backdrop_path': '/xXp7TbC0Ke4lB65ngkt3CuhsiAa.jpg',
'original_language': 'en',
'vote_count': '738',
'until_date': '2023-06-25',
'id': '697843',
'poster_path': '/7gKI9hpEMcZUQpNgKrkDzJpbnNS.jpg',
'video': False,
'title': 'Extraction 2'}
```

EJERCICIO 8: Validación del API a través del uso de IPWidgets.

Para validar que todo funciona correctamente, hemos incluido una serie de elementos visuales (widgets) que permiten realizar consultas al API previamente desplegado. Para ello, únicamente debemos incluir en la siguiente celda la URL del API desplegado, incluyendo la etapa/stage, la cual habremos usado en validaciones preliminares.

Se deben realizar las siguientes acciones:

- Verificad el funcionamiento de los widgets usando la URL ya proporcionada (solución)
- Modificad la celda posterior para incluir la URL del API desplegado previamente
- Verificad el funcionamiento de los widgets

```
In [16]: #BASE_URL = 'https://x525ljspii.execute-api.us-east-1.amazonaws.com/dev' # URL SOLUCIÓN --> PARA VALIDAR
BASE_URL = 'https://nawg4eewpk.execute-api.us-east-1.amazonaws.com/prod'
```


In [25]: %matplotlib inline

```
import requests
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display, HTML
from matplotlib.ticker import MaxNLocator

def display_movie_info(year, month):
    url = BASE_URL + f'/list/{year}/{month}'
    res = requests.get(url)
    data = res.json()
    #print(data)

    dropdown_movies = widgets.Dropdown(
        options=[(x['title'], x['id']) for x in data],
        description='Película'
    )
    display(dropdown_movies)
    movie_data = widgets.interactive_output(display_movie, {'_id': dropdown_movies})
    display(movie_data)

def display_movie(_id):
    url = BASE_URL + f'/movies/{_id}'
    res = requests.get(url)
    data = res.json()
    if data:
        output1 = widgets.Output()
        with output1:
            poster_path = 'http://image.tmdb.org/t/p/w185'+data['poster_path']
            html = HTML(f'''
                <p>
                    <b>Estreno:</b> {data["y_m"]}</br>
                    <b>Hasta:</b> {data["until_date"]}</br>
                    <b>Votos:</b> {data["vote_count"]}</br>
                    <b>Media:</b> {data["vote_average"]}
                </p>
                
            ''')
            display(html)

        output2 = widgets.Output()
        with output2:
            #ranks = [int(value) for value in data['rank_history'].values()]
            #weeks = list(data['rank_history'].keys())
            ## Yo Le Llame rank_hist
            ranks = [int(value) for value in data['rank_hist'].values()]
            weeks = list(data['rank_hist'].keys())

            # Configurar grafica
            fig = plt.figure(figsize=(8, 5), dpi=100)
            ax = fig.gca()
            ax.yaxis.set_major_locator(MaxNLocator(integer=True))
            plt.xticks(range(len(weeks)), weeks)
            plt.ylim(0, max(ranks) + 1)

            plt.title("Ranking semanal")
            plt.xlabel("Semana")
            plt.ylabel("Posición")

            # Dibujar grafica
            plt.plot(ranks, marker='o', linewidth=2)
            plt.show()

        two_columns = widgets.HBox([output1, output2])
        display(two_columns)

dropdown_year = widgets.Dropdown(
    options=[2022, 2023],
    description='Año'
)

dropdown_month = widgets.Dropdown(
    options=range(1, 13),
    description='Mes'
```



```
)
movie_info = widgets.interactive_output(display_movie_info, {'year': dropdown_year, 'month': dropdown_month})

display(dropdown_year)
display(dropdown_month)
display(movie_info)
```

```
Dropdown(description='Año', options=(2022, 2023), value=2022)
```

```
Dropdown(description='Mes', options=(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), value=1)
```

```
Output()
```

EJERCICIO 9 (OPCIONAL): Monitorización a través de Cloudwatch.


Si se desea profundizar/investigar sobre el uso de Cloudwatch, se propone un ejercicio opcional a tal efecto.

AWS Cloudwatch es un servicio de monitorización que permite crear paneles/dashboards mostrando datos sobre el funcionamiento de otros servicios. Por ejemplo, podemos mostrar las invocaciones totales a las funciones lambda o las capacidades de lectura de una tabla de DynamoDB.

El ejercicio propone la generación de un panel que muestre datos, que desde vuestro punto de vista, sean relevantes para la monitorización de la aplicación en su conjunto. Este ejercicio se realizará totalmente a través de la consola de AWS.

VALIDACIÓN

- Generad una imagen con un pantallazo que muestre el panel de Cloudwatch desarrollado, y guardadla en la ruta (junto al notebook) `/img/eje_9.png`

 Validacion 9

