import requests import psycopg2 from psycopg2 import Error

from datetime import datetime

Configuración de conexión a PostgreSQL

db_host = 'postgres' db_port = '5432' db_name = 'jsonplaceholder_data' db_user = 'postgres' db_password = '1234567890'

Función para obtener datos de la API y cargar en PostgreSQL

def extract_and_load_data(event, context): connection = None

```
try:
    connection = psycopg2.connect(
        user=db_user,
        password=db_password,
        host=db_host,
        port=db_port,
        database=db_name,
       # client_encoding='utf-8'
    )

    cursor = connection.cursor()

    # Endpoint URLs
    users_url = 'https://jsonplaceholder.typicode.com/users'
    posts_url = 'https://jsonplaceholder.typicode.com/posts'

    # Obtener datos de usuarios
    response = requests.get(users_url)
    users_data = response.json()

    # Obtener datos de publicaciones
    response = requests.get(posts_url)
    posts_data = response.json()

    # Crear tablas en PostgreSQL (si es necesario, generalmente no debería ser necesario cada vez)
    create_tables(cursor)

    # Insertar o actualizar datos de usuarios
    upsert_users(cursor, users_data)

    # Insertar o actualizar datos de publicaciones
    upsert_posts(cursor, posts_data)

    # Confirmar cambios en la base de datos
    connection.commit()

    print(f"{datetime.now()} - Actualización diaria completada")

except (Exception, Error) as error:
    print(f"{datetime.now()} - Error durante la actualización diaria: {error}")
    raise

finally:
    if connection is not None:
        cursor.close()
        connection.close()  # Cerrar la conexión aquí
```

Función para crear tablas en PostgreSQL

def create_tables(cursor): create_users_table = """ CREATE TABLE IF NOT EXISTS Usuarios (id SERIAL PRIMARY KEY, name VARCHAR(255), username VARCHAR(255), email VARCHAR(255)); """

```
create_posts_table = """
CREATE TABLE IF NOT EXISTS Publicaciones (
    id SERIAL PRIMARY KEY,
    user_id INTEGER,
    title VARCHAR(255),
    body TEXT,
    FOREIGN KEY (user_id) REFERENCES Usuarios (id)
);
"""

cursor.execute(create_users_table)
cursor.execute(create_posts_table)
```

Función para insertar o actualizar datos de usuarios en PostgreSQL

def upsert_users(cursor, users_data): upsert_query = """ INSERT INTO Usuarios (id, name, username, email) VALUES (%s, %s, %s, %s) ON CONFLICT (id) DO UPDATE SET name = EXCLUDED.name, username = EXCLUDED.username, email = EXCLUDED.email; """

```
for user in users_data:
    user_tuple = (user['id'], user['name'], user['username'], user['email'])
    cursor.execute(upsert_query, user_tuple)
```

Función para insertar o actualizar datos de publicaciones en

PostgreSQL

def upsert_posts(cursor, posts_data): upsert_query = """ INSERT INTO Publicaciones (id, user_id, title, body) VALUES (%s, %s, %s, %s) ON CONFLICT (id) DO UPDATE SET user_id = EXCLUDED.user_id, title = EXCLUDED.title, body = EXCLUDED.body; """

```
for post in posts_data:
    post_tuple = (post['id'], post['userId'], post['title'], post['body'])
    cursor.execute(upsert_query, post_tuple)
```

Este fragmento permite la ejecución local

if **name** == "**main**": extract_and_load_data(None, None)

Script para Extraer y Cargar Datos en PostgreSQL

Este script en Python extrae datos de la API de JSONPlaceholder y los carga en una base de datos PostgreSQL. Incluye funciones para la creación de tablas y la inserción/actualización de datos.

Código

import requests

import psycopg2 from psycopg2 import Error from datetime import datetime

Configuración de conexión a PostgreSQL

db_host = 'postgres' db_port = '5432' db_name = 'jsonplaceholder_data' db_user = 'postgres' db_password = '1234567890'

Función para obtener datos de la API y cargar en PostgreSQL

def extract_and_load_data(event, context): connection = None

```
try:
    connection = psycopg2.connect(
        user=db_user,
        password=db_password,
        host=db_host,
        port=db_port,
        database=db_name,
       # client_encoding='utf-8'
    )

    cursor = connection.cursor()

    # Endpoint URLs
    users_url = 'https://jsonplaceholder.typicode.com/users'
    posts_url = 'https://jsonplaceholder.typicode.com/posts'

    # Obtener datos de usuarios
    response = requests.get(users_url)
    users_data = response.json()

    # Obtener datos de publicaciones
    response = requests.get(posts_url)
    posts_data = response.json()

    # Crear tablas en PostgreSQL (si es necesario, generalmente no debería ser necesario cada vez)
    create_tables(cursor)

    # Insertar o actualizar datos de usuarios
    upsert_users(cursor, users_data)

    # Insertar o actualizar datos de publicaciones
    upsert_posts(cursor, posts_data)

    # Confirmar cambios en la base de datos
    connection.commit()

    print(f"{datetime.now()} - Actualización diaria completada")

except (Exception, Error) as error:
    print(f"{datetime.now()} - Error durante la actualización diaria: {error}")
    raise

finally:
    if connection is not None:
        cursor.close()
        connection.close()  # Cerrar la conexión aquí
```

Función para crear tablas en PostgreSQL

def create_tables(cursor): create_users_table = """ CREATE TABLE IF NOT EXISTS Usuarios (id SERIAL PRIMARY KEY, name VARCHAR(255), username VARCHAR(255), email VARCHAR(255)); """

```
create_posts_table = """
CREATE TABLE IF NOT EXISTS Publicaciones (
    id SERIAL PRIMARY KEY,
    user_id INTEGER,
    title VARCHAR(255),
    body TEXT,
    FOREIGN KEY (user_id) REFERENCES Usuarios (id)
);
"""

cursor.execute(create_users_table)
cursor.execute(create_posts_table)
```

Función para insertar o actualizar datos de usuarios en PostgreSQL

def upsert_users(cursor, users_data): upsert_query = """ INSERT INTO Usuarios (id, name, username, email) VALUES (%s, %s, %s, %s) ON CONFLICT (id) DO UPDATE SET name = EXCLUDED.name, username = EXCLUDED.username, email = EXCLUDED.email; """

```
for user in users_data:
    user_tuple = (user['id'], user['name'], user['username'], user['email'])
    cursor.execute(upsert_query, user_tuple)
```

Función para insertar o actualizar datos de publicaciones en

PostgreSQL

def upsert_posts(cursor, posts_data): upsert_query = """ INSERT INTO Publicaciones (id, user_id, title, body) VALUES (%s, %s, %s, %s) ON CONFLICT (id) DO UPDATE SET user_id = EXCLUDED.user_id, title = EXCLUDED.title, body = EXCLUDED.body; """

```
for post in posts_data:
    post_tuple = (post['id'], post['userId'], post['title'], post['body'])
    cursor.execute(upsert_query, post_tuple)
```

Este fragmento permite la ejecución local

if **name** == "**main**": extract_and_load_data(None, None)