Report

Overview

This report documents the challenges encountered and solutions implemented while connecting to a PostgreSQL database using AWS Lambda and Amazon RDS Proxy. The objective was to enhance the efficiency of cloud architecture deployment and data engineering tasks. By addressing various technical issues, this process also served to strengthen practical application skills in a real-world setting.

Experience

While using AWS Lambda with an RDS proxy to connect to PostgreSQL, I encountered issues like timeouts and permission errors. To resolve these, I reviewed IAM roles, VPC configurations, and security group settings, and opted to store credentials directly in environment variables instead of using Secrets Manager. Leveraging CloudWatch Logs for real-time debugging helped me troubleshoot effectively and deepen my understanding of the AWS environment.

Code

```
import os
import psycopg2
import uuid
import json


def handler(event, context):
  # 환경 변수에서 자격 증명을 직접 가져옴
  db_user = os.environ['DB_USER']
  db_password = os.environ['DB_PASSWORD']
  db_host = os.environ['DB_HOST']
  db_name = os.environ['DB_NAME']


  # 데이터베이스 연결 시도
  try:
    print(f"Attempting to connect to DB at host: {db_host}")
    connection = psycopg2.connect(
        host=db_host,
        database=db_name,
        user=db_user,
        password=db_password
    )
    print("Database connection established successfully.")
except Exception as e:
    print("Database connection failed:", str(e))
    return {"statusCode": 500, "body": f"Database connection failed: {str(e)}"}

cursor = connection.cursor()

try:
    # 삽입할 데이터 준비
    records_to_insert = [
        (
            uuid.UUID(record['id']),
            uuid.UUID(record['device_id']),
            record['event_type'],
            json.dumps(record['event_values']),  # JSON 형식일 경우 변환
            record['timestamp']
        )
        for record in event['Records']
    ]

    # Batch Insert로 데이터베이스에 한 번에 삽입
    sql = """
    INSERT INTO your_table_name (id, device_id, event_type, event_values, timestamp)
    VALUES (%s, %s, %s, %s, %s)
    """
    cursor.executemany(sql, records_to_insert)
    connection.commit()
    print("Data inserted successfully.")

except Exception as e:
    print("Failed to insert data:", str(e))
```

```
        return {"statusCode": 500, "body": f"Data insertion failed: {str(e)}"}
    finally:
        # 커서와 연결을 닫기
        cursor.close()
        connection.close()
        print("Database connection closed.")

    return {"statusCode": 200, "body": "Data inserted successfully"}
```