Oakvale Data Lakehouse

This project implements a complete data lakehouse architecture on AWS for Oakvale. The system extracts movie data, processes it through a three-layer architecture (bronze, silver, gold), and makes it available for analytics.

Architecture Overview

The data lakehouse architecture consists of:

1. **Data Generation**: Lambda function generates fake movie data using Python's Faker library
2. **Storage**: Three S3 buckets for raw data, lakehouse data (in Delta format), and Glue scripts
3. **Processing**: AWS Glue jobs for ETL processing through three layers:
    - Bronze: Raw data from the source
    - Silver: Cleaned and transformed data
    - Gold: Aggregated data for analytics

Components

S3 Buckets

- **oakvale-raw-data**: Stores raw JSON data from the Lambda function
- **oakvale-lakehouse**: Stores processed data in Delta Lake format
- **oakvale-glue-scripts**: Stores AWS Glue ETL scripts

Glue Databases

- **oakvale_bronze**: Raw data in Delta format
- **oakvale_silver**: Cleaned and normalized data
- **oakvale_gold**: Aggregated data for analytics

ETL Process

1. **Lambda Function**: Generates movie data and stores it in the raw bucket
2. **Bronze Layer**: Reads raw JSON data and converts it to Delta format
3. **Silver Layer**: Cleans and normalizes the data (fixes data types, normalizes categories)
4. **Gold Layer**: Creates aggregated metrics (by genre, studio, release year)

Project Structure

```
├── extract_api_data/
│   └── api_data.py          # Lambda function for data generation
├── glue_scripts/
│   ├── bronze_glue_script.py # ETL script for bronze layer
│   ├── silver_glue_script.py # ETL script for silver layer
│   └── gold_glue_script.py   # ETL script for gold layer
├── delta_jar/
```

```
│   ├── delta-core_2.12-2.1.0.jar  # Delta Lake core JAR
│   └── delta-storage-2.1.0.jar    # Delta Lake storage JAR
├── main.tf                     # Main Terraform configuration
├── variables.tf                # Terraform variables
├── glue_jobs.tf                # Glue jobs configuration
├── download_delta_jars.ps1  # PowerShell script to download Delta JARs
└── download_delta_jars.sh   # Shell script to download Delta JARs
```

Deployment

1. Ensure you have AWS credentials configured
2. Install Terraform
3. Download the required Delta jar files using the provided scripts:
   - For Windows:

     ```
     .\download_delta_jars.ps1
     ```

   - For Linux/Mac:

     ```
     chmod +x download_delta_jars.sh
     ./download_delta_jars.sh
     ```

4. Package Lambda function:

   ```
   pip install faker boto3 -t ./lambda_package
   cp extract_api_data/api_data.py ./lambda_package/
   cd lambda_package && zip -r ../lambda_package.zip . && cd ..
   ```

5. Initialize and apply Terraform:

   ```
   terraform init
   terraform apply
   ```

Glue Jobs Configuration

The project uses a dynamic approach to define Glue jobs using Terraform's `for_each` functionality. This makes it easy to maintain and update the jobs as needed. The configuration includes:

- Job parameters like worker type, timeout, and retry settings
- Default arguments including paths for source and destination data
- Integration with Delta Lake via JAR files

- Auto-scaling capabilities for efficient resource usage
- Workflow orchestration with conditional triggers

Data Flow

1. Lambda function generates movie data daily and stores it in the raw bucket
2. Glue workflow triggers the bronze job to process raw data
3. Upon completion, the silver job is triggered to clean and normalize data
4. Finally, the gold job creates aggregated metrics for analytics

Accessing the Data

The processed data can be accessed using:

- AWS Athena for SQL queries
- AWS QuickSight for visualization
- Any tool that supports Glue Data Catalog as a metadata store