

Project Report: Data Ingestion from S3 to RDS with Fallback to AWS Glue using Dockerized Python Application

1. Introduction

This project focuses on building a resilient and automated data ingestion pipeline using AWS services and Docker. The pipeline reads data from an Amazon S3 bucket, attempts to upload it to an Amazon RDS (MySQL-compatible) database, and if that fails, falls back to AWS Glue for data cataloging. The entire process is containerized using Docker to ensure portability and ease of deployment.

2. Objectives

The objective of this project is to:- Automate data ingestion from S3 to RDS- Implement a fallback mechanism using AWS Glue- Package the solution using Docker for scalable deployment

3. Requirements

Software Requirements:- Python 3.9+- Docker- AWS CLI (configured with credentials)- boto3, pandas, sqlalchemy, pymysql
AWS Resources:- S3 Bucket (for CSV file)- RDS MySQL-compatible instance- AWS Glue (Database &

Table)Configuration Parameters:- S3 Bucket Name and CSV File Key- RDS DB Endpoint, Username, Password, DB Name, Table Name- Glue Database Name, Table Name, and S3 Location

4. Technology Stack

- Programming Language: Python 3.9+- Cloud Services: Amazon S3, Amazon RDS, AWS Glue- Containerization: Docker- Libraries: boto3, pandas, sqlalchemy, pymysql

5. System Architecture Diagram

The architecture includes the following components:- Docker container running a Python script- S3 bucket as the data source- RDS MySQL-compatible database for primary ingestion- AWS Glue as a fallback cataloging system

+-----+

| CSV File |

| (Stored in S3) |

+-----+-----+

|

v

+-----+-----+

| Dockerized |

| Python Script |

| (Runs in EC2 or |

| local container)|

+-----+-----+

|

v

+-----+-----+

| Attempt to push |

| data to RDS |

| (MySQL-compatible)|

+-----+-----+

|

| Success

v

+-----+

| Data Stored in |

| RDS |

+-----+

|

| Failure

v

+-----+-----+

| Fallback to |

| AWS Glue |

+-----+-----+

|

v

+-----+

| Glue Table in |

| Data Catalog |

| with S3 Location |

+-----+

6. Implementation Steps

Step 1: Python Script Development

- Read CSV file from S3 using boto3
- Parse the data using pandas
- Attempt to upload to RDS using SQLAlchemy
- If RDS fails, create a table in AWS Glue and register the S3 location


```

#!/usr/bin/env python3
import boto3
import pandas as pd
from sqlalchemy import create_engine
from botocore.exceptions import ClientError

# Load ENV
s3_bucket = os.getenv("S3_BUCKET")
s3_key = os.getenv("S3_KEY")
rds_host = os.getenv("RDS_HOST")
rds_user = os.getenv("RDS_USER")
rds_pass = os.getenv("RDS_PASS")
rds_db = os.getenv("RDS_DB")
rds_table = os.getenv("RDS_TABLE")
glue_db = os.getenv("GLUE_DB")
glue_table = os.getenv("GLUE_TABLE")
glue_s3_path = os.getenv("GLUE_S3_PATH")

def read_from_s3():
    s3 = boto3.client('s3')
    obj = s3.get_object(Bucket=s3_bucket, Key=s3_key)
    df = pd.read_csv(obj['Body'])
    return df

def push_to_rds(df):
    try:
        engine = create_engine(f'mysql+pymysql://{rds_user}@{rds_host}/{rds_db}')
        df.to_sql(rds_table, engine, if_exists='replace', index=False)
        print('Data uploaded to RDS')
        return
    except Exception as e:
        print(f'Failed to upload to RDS: {e}')
        return False

def fallback_to_glue(df):
    try:
        glue = boto3.client('glue')
        glue.create_database(DatabaseInput={'Name': glue_db})
        glue.create_table(
            DatabaseName=glue_db,
            TableInput={
                'Name': glue_table,
                'StorageDescriptor': {
                    'Columns': [{'Name': col, 'Type': 'string'} for col in df.columns],
                    'Location': glue_s3_path,
                    'InputFormat': 'org.apache.hadoop.mapred.TextInputFormat',
                    'OutputFormat': 'org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat',
                    'SerdeInfo': {
                        'SerializationLibrary': 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe',
                        'Parameters': {'field.delim': ','}
                    }
                },
                'TableType': 'EXTERNAL_TABLE'
            }
        )
        print('Fallback to AWS Glue successful')
    except ClientError as e:
        print(f'Glue error: {e}')

def main():
    df = read_from_s3()
    if not push_to_rds(df):
        fallback_to_glue(df)

if __name__ == "__main__":
    main()

```

```

#!/usr/bin/env python3
except Exception as e:
    print(f'Failed to upload to RDS: {e}')
    return False

def fallback_to_glue(df):
    try:
        glue = boto3.client('glue')
        glue.create_database(DatabaseInput={'Name': glue_db})
        glue.create_table(
            DatabaseName=glue_db,
            TableInput={
                'Name': glue_table,
                'StorageDescriptor': {
                    'Columns': [{'Name': col, 'Type': 'string'} for col in df.columns],
                    'Location': glue_s3_path,
                    'InputFormat': 'org.apache.hadoop.mapred.TextInputFormat',
                    'OutputFormat': 'org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat',
                    'SerdeInfo': {
                        'SerializationLibrary': 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe',
                        'Parameters': {'field.delim': ','}
                    }
                },
                'TableType': 'EXTERNAL_TABLE'
            }
        )
        print('Fallback to AWS Glue successful')
    except ClientError as e:
        print(f'Glue error: {e}')

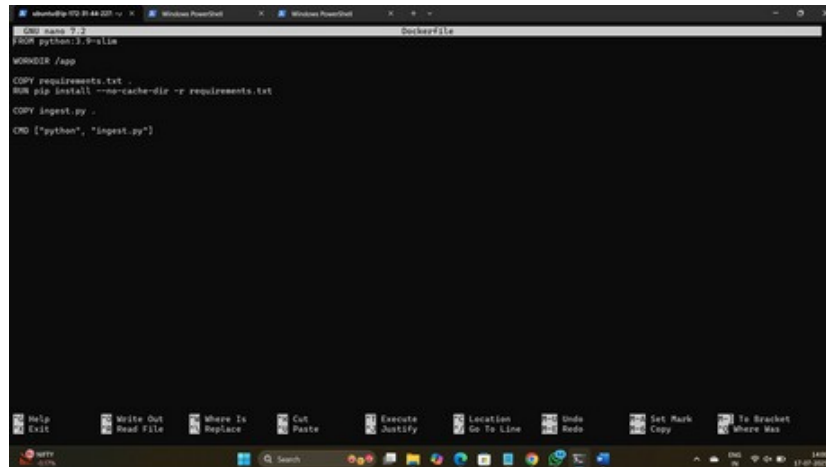
def main():
    df = read_from_s3()
    if not push_to_rds(df):
        fallback_to_glue(df)

if __name__ == "__main__":
    main()

```

Step 2: Dockerfile Creation

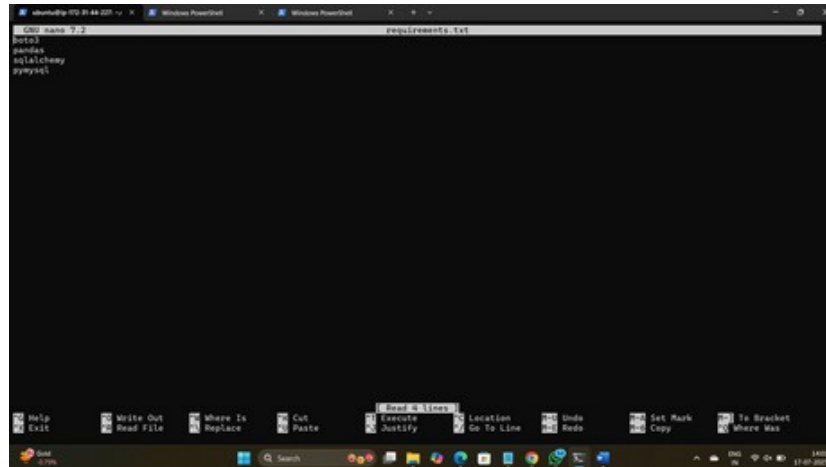
- Use Python 3.9 base image
- Install necessary dependencies
- Copy Python script into container
- Run script on container startup

A screenshot of a code editor window showing a Dockerfile. The editor has a dark theme and a menu bar at the bottom with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, and Where Was. The Dockerfile content is as follows:

```
FROM name:3.2
RUN python3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY ingest.py .
CMD ["python", "ingest.py"]
```

Step 3: Requirements File

- Include all necessary Python dependencies (boto3, pandas, sqlalchemy, pymysql)



Step 4: Create database

Step 5 : Create S3 bucket

Step 6: Create AWS Glue Database


```

GNU nano 7.2                                data_ingest.py
import pandas as pd
import boto3
import pymysql
from sqlalchemy import create_engine
import os

# Load env vars
s3_bucket = os.environ.get("S3_BUCKET")
s3_key = os.environ.get("S3_KEY")
rds_host = os.environ.get("RDS_HOST")
rds_user = os.environ.get("RDS_USER")
rds_password = os.environ.get("RDS_PASSWORD")
rds_db = os.environ.get("RDS_DB")
glue_db = os.environ.get("GLUE_DB")
glue_table = os.environ.get("GLUE_TABLE")
glue_s3_path = os.environ.get("GLUE_S3_PATH")

def read_csv_from_s3():
    s3 = boto3.client('s3')
    obj = s3.get_object(Bucket=s3_bucket, Key=s3_key)
    df = pd.read_csv(obj['Body'])
    return df

def push_to_rds(df):
    try:
        conn_str = f"mysql+pymysql://{rds_user}:{rds_password}@{rds_host}/{rds_db}"
        engine = create_engine(conn_str)
        df.to_sql('students', conn=engine, index=False, if_exists='replace')
        print("Data uploaded to RDS successfully.")
        return True
    except Exception as e:
        print("RDS upload failed:", str(e))
        return False

def fallback_to_glue():
    glue = boto3.client('glue')
    try:
        response = glue.create_table(
            DatabaseName=glue_db,
            TableInput={
                'Name': glue_table,
                'StorageDescriptor': {
                    'Columns': [

```

```

mysql> CREATE DATABASE project3;
Query OK, 1 row affected (0.01 sec)

mysql> EXIT;
Bye

ubuntu@ip-172-31-46-217:~$ docker run \
-e AWS_ACCESS_KEY_ID=AKIAHYBNASVSI35I8BB \
-e AWS_SECRET_ACCESS_KEY=Se7X0bfvAdU5IEuvQ9fHm9fms1z97W4lg0NCY \
-e AWS_DEFAULT_REGION=ap-northeast-2 \
-e S3_BUCKET=ganjaa \
-e S3_KEY=data/student_records.csv \
-e RDS_HOST=project3.c32maq4w4dc.ap-northeast-2.rds.amazonaws.com \
-e RDS_USER=admin \
-e RDS_PASS=project3 \
-e RDS_DB=project3 \
-e RDS_TABLE=students \
-e GLUE_DB=student_data_db \
-e GLUE_TABLE=student_fallback_table \
-e GLUE_S3_PATH=s3://ganjaa/data/ \
s3-to-rds-ingester

Data uploaded to RDS
ubuntu@ip-172-31-46-217:~$ history
1 clear
2 docker run -e AWS_ACCESS_KEY_ID=AKIAHYBNASVSI35I8BB -e AWS_SECRET_ACCESS_KEY=Se7X0bfvAdU5IEuvQ9fHm9fms1z97W4lg0NCY -e AWS_DEFAULT_REGION=ap-northeast-2 -e S3_BUCKET=ganjaa -e S3_KEY=data/student_records.csv -e RDS_HOST=project3.c32maq4w4dc.ap-northeast-2.rds.amazonaws.com -e RDS_USER=admin -e RDS_PASS=project3 -e RDS_DB=project3 -e RDS_TABLE=students -e GLUE_DB=student_data_db -e GLUE_TABLE=student_fallback_table -e GLUE_S3_PATH=s3://ganjaa/data/ s3-to-rds-ingester
3 sudo apt install mysql-client -y
4 mysql -h project3.c32maq4w4dc.ap-northeast-2.rds.amazonaws.com -u admin -p
5 docker run -e AWS_ACCESS_KEY_ID=AKIAHYBNASVSI35I8BB -e AWS_SECRET_ACCESS_KEY=Se7X0bfvAdU5IEuvQ9fHm9fms1z97W4lg0NCY -e AWS_DEFAULT_REGION=ap-northeast-2 -e S3_BUCKET=ganjaa -e S3_KEY=data/student_records.csv -e RDS_HOST=project3.c32maq4w4dc.ap-northeast-2.rds.amazonaws.com -e RDS_USER=admin -e RDS_PASS=project3 -e RDS_DB=project3 -e RDS_TABLE=students -e GLUE_DB=student_data_db -e GLUE_TABLE=student_fallback_table -e GLUE_S3_PATH=s3://ganjaa/data/ s3-to-rds-ingester
6 history
ubuntu@ip-172-31-46-217:~$ client_loop: send disconnect: Connection reset
PS C:\Users\munda\Downloads> ssh -i .\proj3.pem ubuntu@3.36.74.219

```

Step 9: Data Verification

- Confirm that data was uploaded to RDS (MySQL Workbench or CLI)
- If RDS fails, confirm Glue table creation in AWS Glue Data Catalog

repository link:

<https://github.com/rohinipandhare12/S3-RDS-Glue-Fallback>

Name- Rohii Pandhare

Cloud & DevOps Intern Cravita Technologies

