

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_pass}@{rds_host}/{rds_db}')
        df.to_sql(rds_table, con=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.LazySimpleTextSerDe',
                        '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.LazySimpleTextSerDe',
                        'Parameters': {'field.delim': ','}
                    }
                },
                'TableType': 'EXTERNAL_TABLE'
            }
        )
        print('Fallback to AWS Glue successful')
    except ClientError as e:
        print(f'Glue error: {e}')

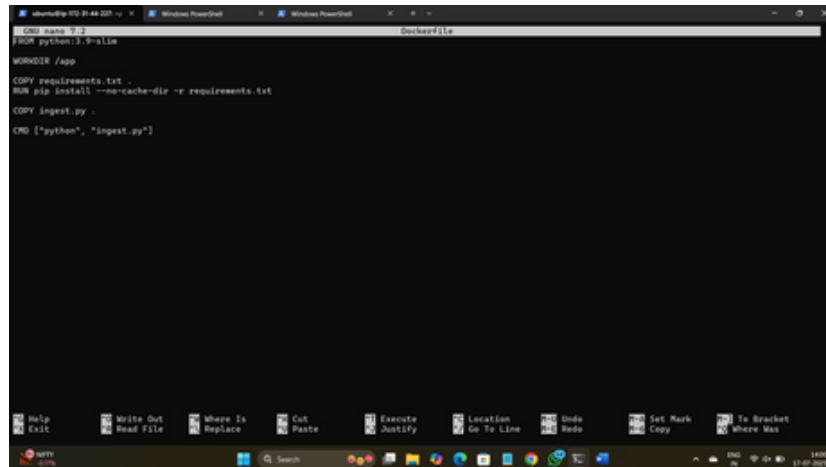
def main():
    df = read_from_s3()
    if not push_to_rds(df):
        fallback_to_glue(df)

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



```
shank@ip-172-31-44-225: ~  
LUD name: 2.2  
FROM python:3.9-slim  
WORKDIR /app  
COPY requirements.txt .  
RUN pip install --no-cache-dir -r requirements.txt  
COPY ingest.py .  
CMD ["python", "ingest.py"]
```

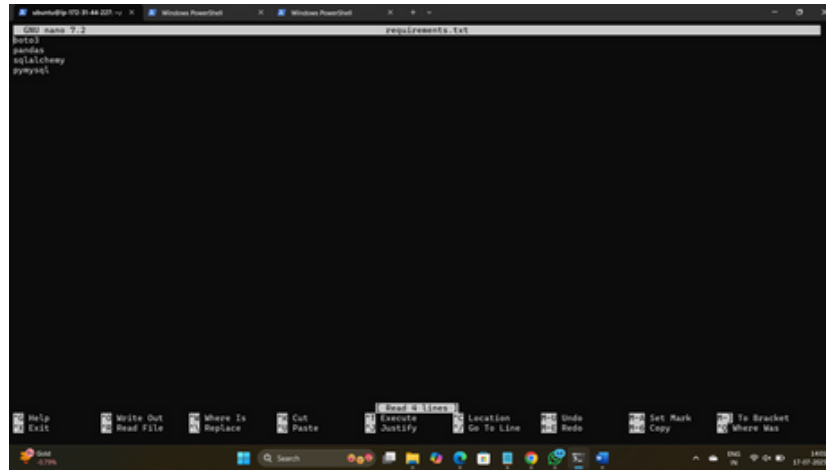
The screenshot shows a Dockerfile editor window titled "Dockerfile". The content of the Dockerfile is as follows:

```
LUD name: 2.2  
FROM python:3.9-slim  
WORKDIR /app  
COPY requirements.txt .  
RUN pip install --no-cache-dir -r requirements.txt  
COPY ingest.py .  
CMD ["python", "ingest.py"]
```

Below the editor, there is a toolbar with various icons and a status bar at the bottom showing system information like CPU usage, memory, and the date/time.

Step 3: Requirements File

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



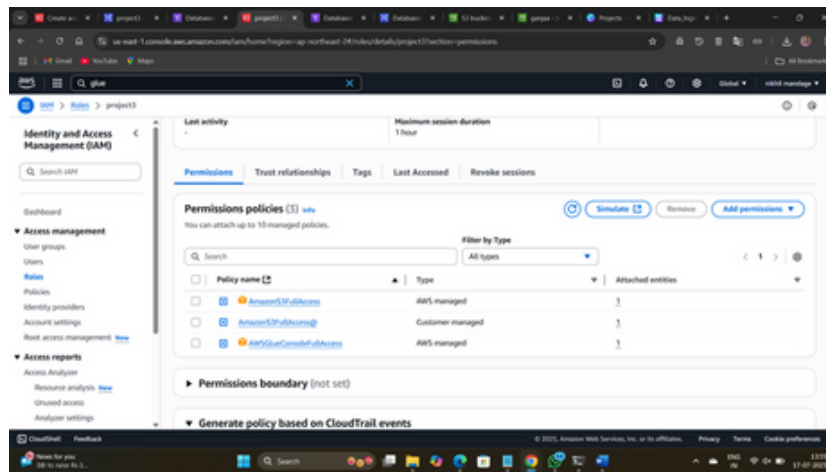
```
Windows PowerShell
PS C:\Users\user> cat .\Requirements.txt
boto3
pandas
sqlalchemy
pymysql
```

Step 4: Create database

Step 5 : Create S3 bucket

Step 6: Create AWS Glue Database

AWSGlueConsoleFullAccess



Step 8: Image Build and Container Run

- Build the Docker image using `docker build -t data-ingestor .`
- Run the container with AWS credentials passed as environment variables

```
Windows PowerShell
User sessions running outdated binaries:
ubuntu@0 session 81: sshd[1868,1158]
ubuntu@0 user manager service: systemd[1851]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-44-227:~$ sudo usermod -s docker $USER
ubuntu@ip-172-31-44-227:~$ cat
ubuntu@ip-172-31-44-227:~$ newgrp docker
ubuntu@ip-172-31-44-227:~$ mkdir ~/data-ingestion-project && cd ~/data-ingest

ubuntu@ip-172-31-44-227:~/data-ingestion-project$ docker build -t s3-to-rds-ingestor .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  6.104kB
Step 1/6 : FROM python:3.9-alim
3.9-alim: Pulling from library/python
3d49a4985ed8: Pull complete
a81768876831: Pull complete
80abc91c4528: Pull complete
9f1673b82588: Pull complete
Digest: sha256:c2a5f6d78e0b9f0108882c2f61c3281e95c914013e21c3ba79b0baad8e67
Status: Downloaded newer image for python:3.9-alim
--> a1ecda2ef264
Step 2/6 : WORKDIR /app
--> Running in 00cd2f79db2e
--> Removed intermediate container 00cd2f79db2e
--> Sch8dd0v7ad
Step 3/6 : COPY requirements.txt .
--> 149b2c281818
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
--> Running in f33ad7f37d7
Collecting boto3
  Downloading boto3-1.39.7-py3-none-any.whl (139 kB)
    139.9/139.9 kB 6.7 MB/s eta 0:00:00
Collecting pandas
  Downloading pandas-2.3.1-cp39-cp39-manylinux_2_37_x86_64_manylinux2014_x86_64.whl (12.4 MB)
    12.4/12.4 MB 114.4 MB/s eta 0:00:00
```

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

mysql> EXIT;
bye

ubuntu@ip-172-31-44-227:~$ docker run \
  --rm --ACCESS_KEY_ID=AKIAIYBMA5Y235IE888 \
  --rm --SECRET_ACCESS_KEY_ID=70M6vA8U2EuvQfHm9Twn1z97Wn6lGmCY \
  --rm --DEFAULT_REGION=ap-northeast-2 \
  --rm --BUCKET=ganja \
  --rm --S3_KEY=data/student_records.csv \
  --rm --RDS_HOST=project3.c32maagw6dc.ap-northeast-2.rds.amazonaws.com \
  --rm --RDS_USER=admin \
  --rm --RDS_PASS=project3 \
  --rm --RDS_DB=project3 \
  --rm --TABLE=students \
  --rm --GLUE_DB=student_data_db \
  --rm --GLUE_TABLE=student_fallback_table \
  --rm --GLUE_S3_PATH=3://ganja/data/ \
  s3-to-rds-ingestor

Data uploaded to RDS
ubuntu@ip-172-31-44-227:~$ history
1 clear
2 docker run --rm --ACCESS_KEY_ID=AKIAIYBMA5Y235IE888 --rm --SECRET_ACCESS_KEY_ID=70M6vA8U2EuvQfHm9Twn1z97Wn6lGmCY --rm --DEFAULT_REGION=ap-northeast-2 --rm --BUCKET=ganja --rm --S3_KEY=data/student_records.csv --rm --RDS_HOST=project3.c32maagw6dc.ap-northeast-2.rds.amazonaws.com --rm --RDS_USER=admin --rm --RDS_PASS=project3 --rm --RDS_DB=project3 --rm --TABLE=students --rm --GLUE_DB=student_data_db --rm --GLUE_TABLE=student_fallback_table --rm --GLUE_S3_PATH=3://ganja/data/ s3-to-rds-ingestor
3 sudo apt install mysql-client -y
4 mysql -h project3.c32maagw6dc.ap-northeast-2.rds.amazonaws.com -u admin -p
5 docker run --rm --ACCESS_KEY_ID=AKIAIYBMA5Y235IE888 --rm --SECRET_ACCESS_KEY_ID=70M6vA8U2EuvQfHm9Twn1z97Wn6lGmCY --rm --DEFAULT_REGION=ap-northeast-2 --rm --BUCKET=ganja --rm --S3_KEY=data/student_records.csv --rm --RDS_HOST=project3.c32maagw6dc.ap-northeast-2.rds.amazonaws.com --rm --RDS_USER=admin --rm --RDS_PASS=project3 --rm --RDS_DB=project3 --rm --TABLE=students --rm --GLUE_DB=student_data_db --rm --GLUE_TABLE=student_fallback_table --rm --GLUE_S3_PATH=3://ganja/data/ s3-to-rds-ingestor
6 history
ubuntu@ip-172-31-44-227:~$ client_loop: send disconnect: Connection reset
PS C:\Users\mamba\Downloads> ssh -i .\pr3.pem ubuntu@3.36.74.218
```

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

```

ubuntu@ip-172-31-44-127:~$ docker run \
  --rm AWS_ACCESS_KEY_ID=AKIA4YBNASV1J35IE888 \
  --rm AWS_SECRET_ACCESS_KEY=sg70M4+4a0uifc0qF8m9Fam1z9TN0a1gMCY \
  --rm AWS_DEFAULT_REGION=ap-northeast-2 \
  --rm S3_BUCKET=project3 \
  --rm S3_BUCKET_PATH=/records.csv \
  --rm RDS_HOST=project3.c32mnaqumdc.ap-northeast-2.rds.amazonaws.com \
  --rm RDS_USER=admin \
  --rm RDS_PASS=project3 \
  --rm RDS_DB=project3 \
  --rm RDS_TABLE=students \
  --rm GLUE_TABLE=project3-fallback-table \
  --rm GLUE_S3_PATH=s3://project3/data/ \
  s3-to-rds-ingester
X Failed to upload to RDS: (mysql.err.OperationalError) [1007, "Unknown database 'project3'"]
(Background on this error at: https://sqlalche.me/e/20/c4q8)
● Fallback to AWS Glue successful
ubuntu@ip-172-31-44-127:~$ sudo apt install mysql-client -y

mysql -h project3.c32mnaqumdc.ap-northeast-2.rds.amazonaws.com -u admin -p
Reading package lists... Done
Building dependency tree... Done
The following packages were automatically installed and are no longer required:
  galera-4 liblogi-fast-perl liblogi-go-perl libncurses-perl
  libnssdp-initfiles-perl libnssdp-mysql-perl libnssdp-perl
  libnssdp-locale-perl libnssdp-his libnssdp-perl libnssdp-his
  libnssdp-server-perl libnssdp-target-perl libnssdp-template-perl
  libnssdp-data-perl libnssdp-message-perl libnssdp-his-perl
  libnssdp-mediatypes-perl libnssdp-his libnssdp-client21 libnssdp-his
  libnssdp-data-perl libnssdp-perl libnssdp-mysql libnssdp-common
  mariadb-server-core po sedat
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  mysql-client-8.0 mysql-client-core-8.0
The following packages will be REMOVED:
  mariadb-client mariadb-client-core mariadb-plugin-provider-brp2
  mariadb-plugin-provider-lsa mariadb-plugin-provider-lsa
  mariadb-plugin-provider-lsa mariadb-plugin-provider-magpy
  mariadb-server

```

```
Windows PowerShell
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or 'h' for help. Type 'c' to clear the current input statement.

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

mysql> EXIT;
Bye

ubuntu@ip-172-31-44-227:~$ docker run \
  --rm --ACCESS_KEY_ID=AKIAVBNASYSZ3SIE888 \
  --rm --SECRET_ACCESS_KEY=STY08Fv4d0EEUvQfHm9Twn1z97Wnlg0NCY \
  --rm --DEFAULT_REGION=ap-northeast-2 \
  --rm --S3_BUCKET=ganja \
  --rm --S3_KEY=data/student_records.csv \
  --rm --RDS_HOST=project3.c12maaqwmbic.ap-northeast-2.rds.amazonaws.com \
  --rm --RDS_USER=admin \
  --rm --RDS_DB=project3 \
  --rm --RDS_TABLE=students \
  --rm --GLUE_DB=student_data_db \
  --rm --GLUE_TABLE=student_fallback_table \
  --rm --GLUE_S3_PA=ganja/Data/ \
  s3-to-rds-ingester
Data uploaded to RDS
ubuntu@ip-172-31-44-227:~$ history
1 clear
2 docker run --rm --ACCESS_KEY_ID=AKIAVBNASYSZ3SIE888 --rm --SECRET_ACCESS_KEY=STY08Fv4d0EEUvQfHm9Twn1z97Wnlg0NCY --rm --DEFAULT_REGION=ap-northeast-2 --rm --S3_BUCKET=ganja --rm --S3_KEY=data/student_records.csv --rm --RDS_HOST=project3.c12maaqwmbic.ap-northeast-2.rds.amazonaws.com --rm --RDS_USER=admin --rm --RDS_DB=project3 --rm --RDS_TABLE=students --rm --GLUE_DB=student_data_db --rm --GLUE_TABLE=student_fallback_table --rm --GLUE_S3_PA=ganja/Data/ s3-to-rds-ingester
3 sudo apt install mysql-client -y
4 mysql -h project3.c12maaqwmbic.ap-northeast-2.rds.amazonaws.com -u admin -p
5 docker run --rm --ACCESS_KEY_ID=AKIAVBNASYSZ3SIE888 --rm --SECRET_ACCESS_KEY=STY08Fv4d0EEUvQfHm9Twn1z97Wnlg0NCY --rm --DEFAULT_REGION=ap-northeast-2 --rm --S3_BUCKET=ganja --rm --S3_KEY=data/student_records.csv --rm --RDS_HOST=project3.c12maaqwmbic.ap-northeast-2.rds.amazonaws.com --rm --RDS_USER=admin --rm --RDS_DB=project3 --rm --RDS_TABLE=students --rm --GLUE_DB=student_data_db --rm --GLUE_TABLE=student_fallback_table --rm --GLUE_S3_PA=ganja/Data/ s3-to-rds-ingester
6 history
ubuntu@ip-172-31-44-227:~$ client_loop: send disconnect: Connection reset
```

7. Results

- Data was successfully read from S3.- Data was inserted into RDS database.- On RDS failure, fallback to AWS Glue succeeded.- Docker logs showed successful execution.[Screenshot Placeholder - Final logs or Glue catalog screen]


8. Conclusion


This project demonstrates a robust, scalable, and automated data ingestion system using cloud-native tools and containerization. The Dockerized Python application ensures consistent deployment, while AWS S3, RDS, and Glue provide scalable storage, compute, and fallback capabilities. The implementation strengthens cloud architecture reliability and fault tolerance.


repository link:


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


nikhilmandage/**S3-RDS-Glue-Fallback**




 1
Contributor

 0
Issues

 0
Stars

 0
Forks



Name- Nikhil Mandage

Cloud & DevOps Intern Cravita Technologies

