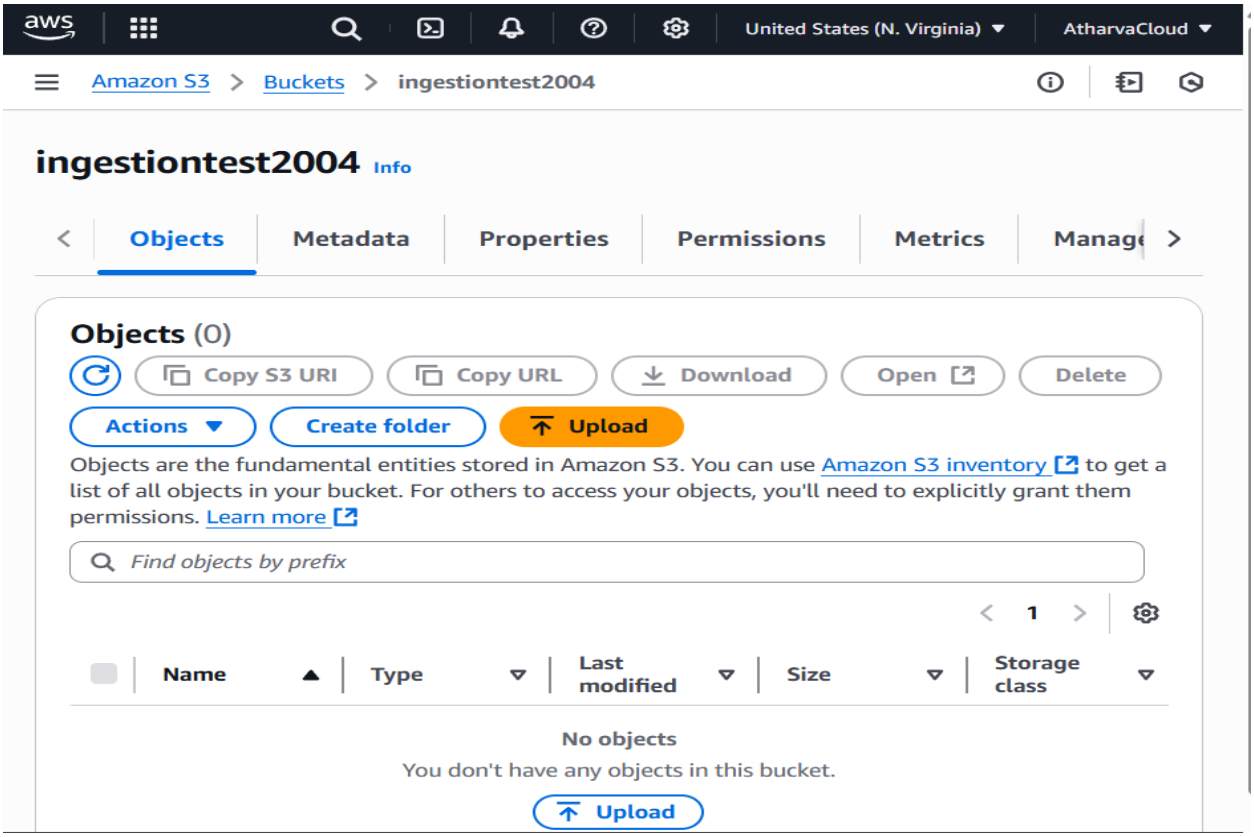
Project Title: Data Ingestion from S3 to RDS with Fallback to AWS Glue using Dockerized Python Application

Objective: To build a Dockerized Python application that automates reading a CSV file from Amazon S3, inserts it into an Amazon RDS (MySQL-compatible) database, and falls back to AWS Glue Data Catalog if RDS fails. This ensures a reliable and fault-tolerant data ingestion pipeline.

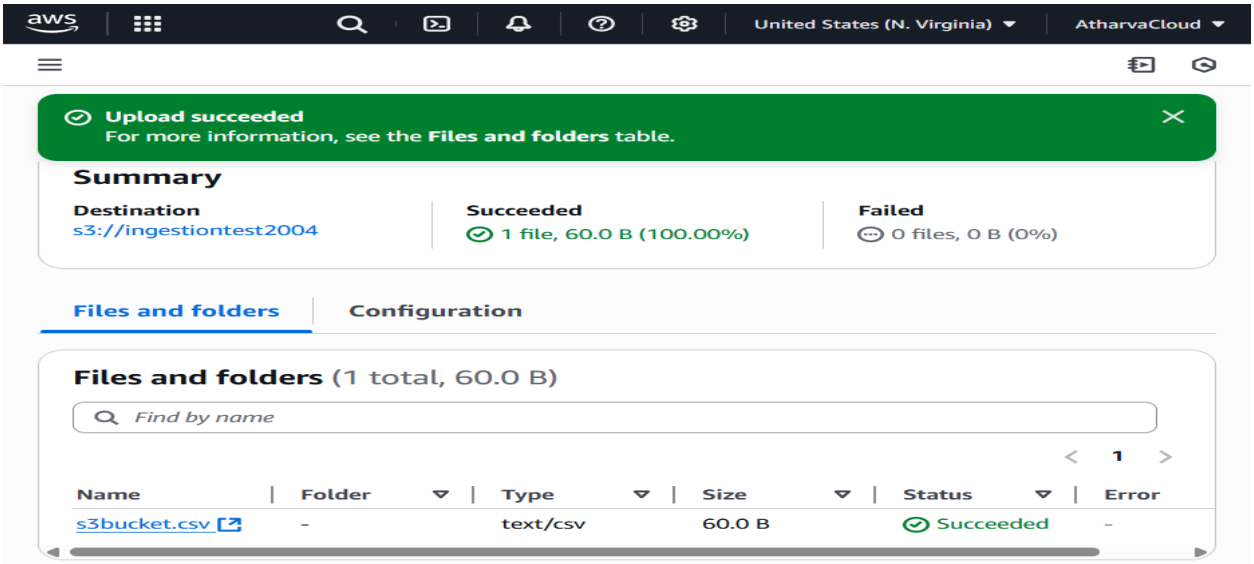
1. Created an S3 Bucket:

- **Service:** Amazon S3
- **Bucket Name:** `ingestiontest2004`
- **Why:** To store the CSV file as the input source for the ingestion pipeline.



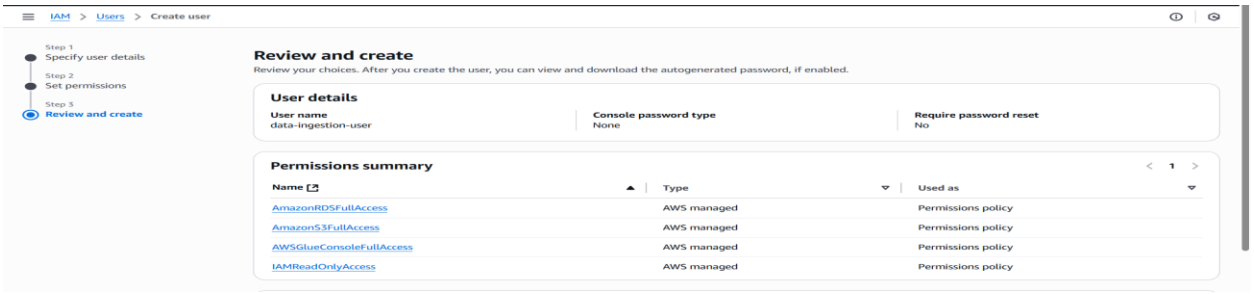
2. Uploaded CSV File to S3:

- **File:** `s3bucket.csv`
- **Why:** This file is used as the input for ingestion into RDS.



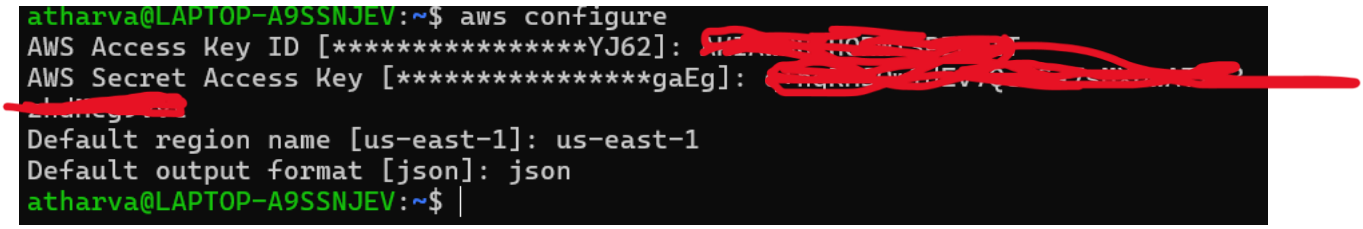
3. Created IAM User and Attached Permissions:

- **IAM User:** `s3-rds-glue-user`
- **Policies Attached:**
    1)`AmazonS3FullAccess`
    2)`AmazonRDSFullAccess`
    3)`AWSGlueConsoleFullAccess`
    4)`IAMFullAccess`
- **Why:** To allow the script to access and operate on S3, RDS, and Glue services securely.



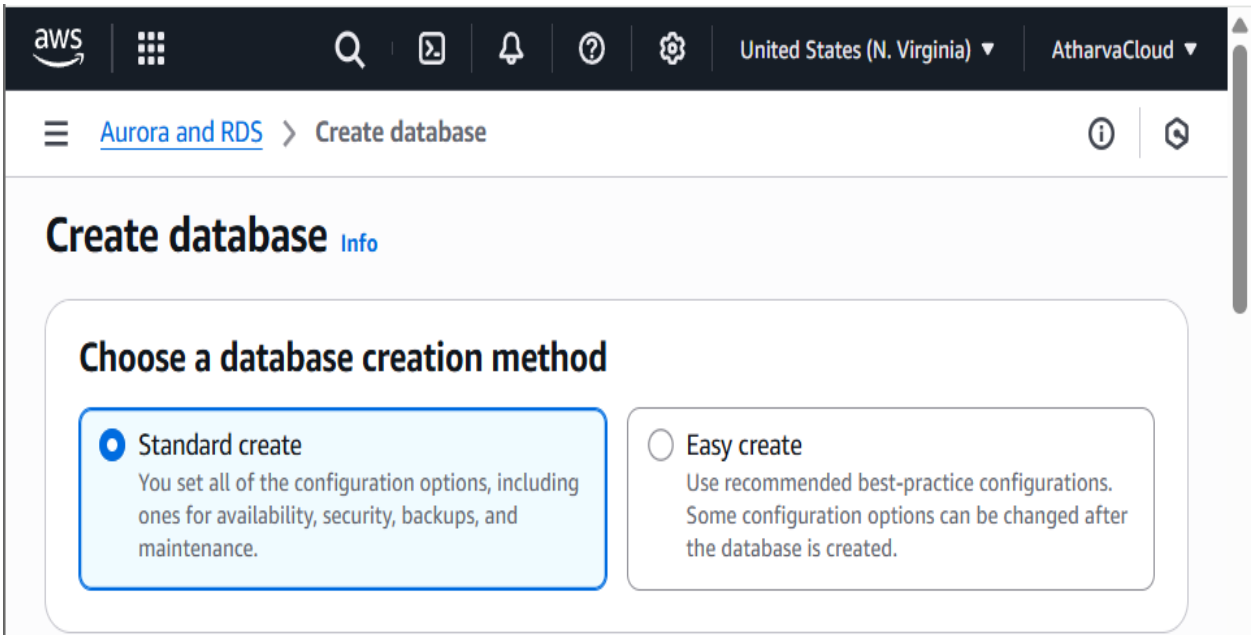
4. Configured AWS CLI:

- **Command Used:** `aws configure`
- **Inputs:** Access Key, Secret Key, Region (`us-east-1`), Output format (`json`)
- **Why:** To test access and verify permissions from the command line.

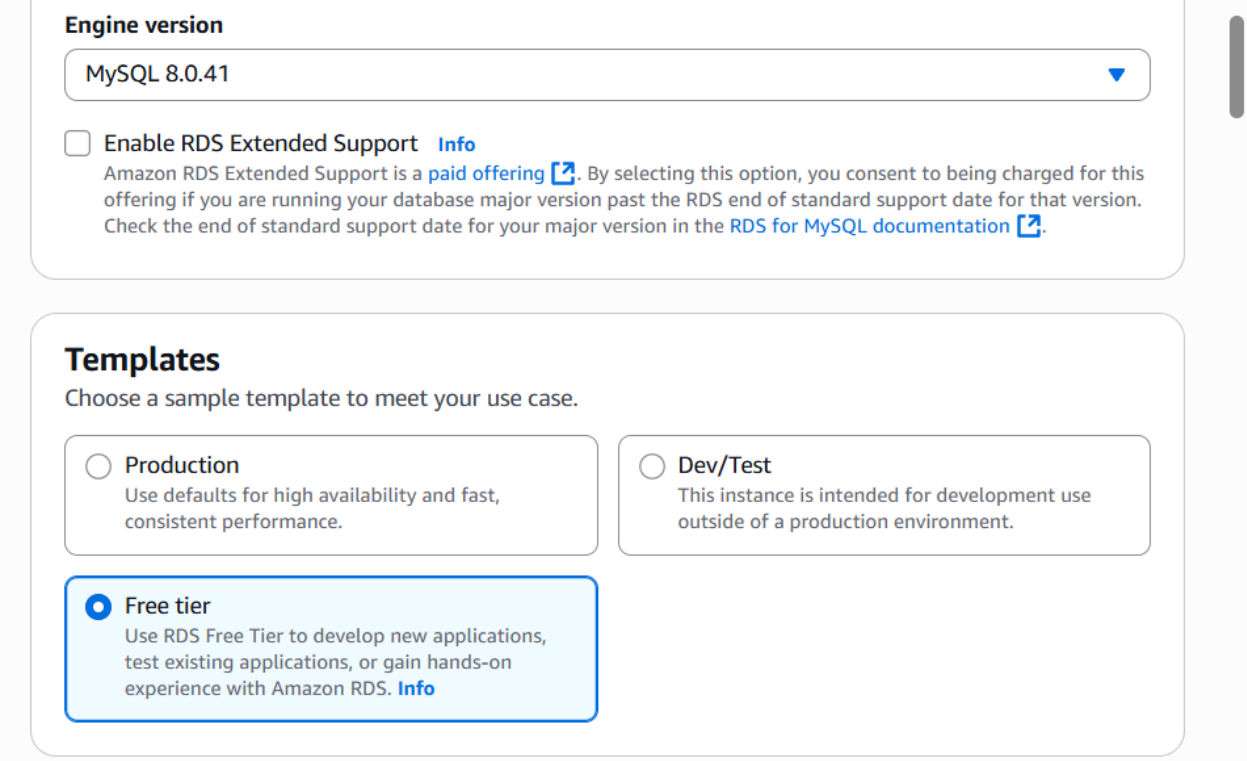


5. Created RDS MySQL Instance:

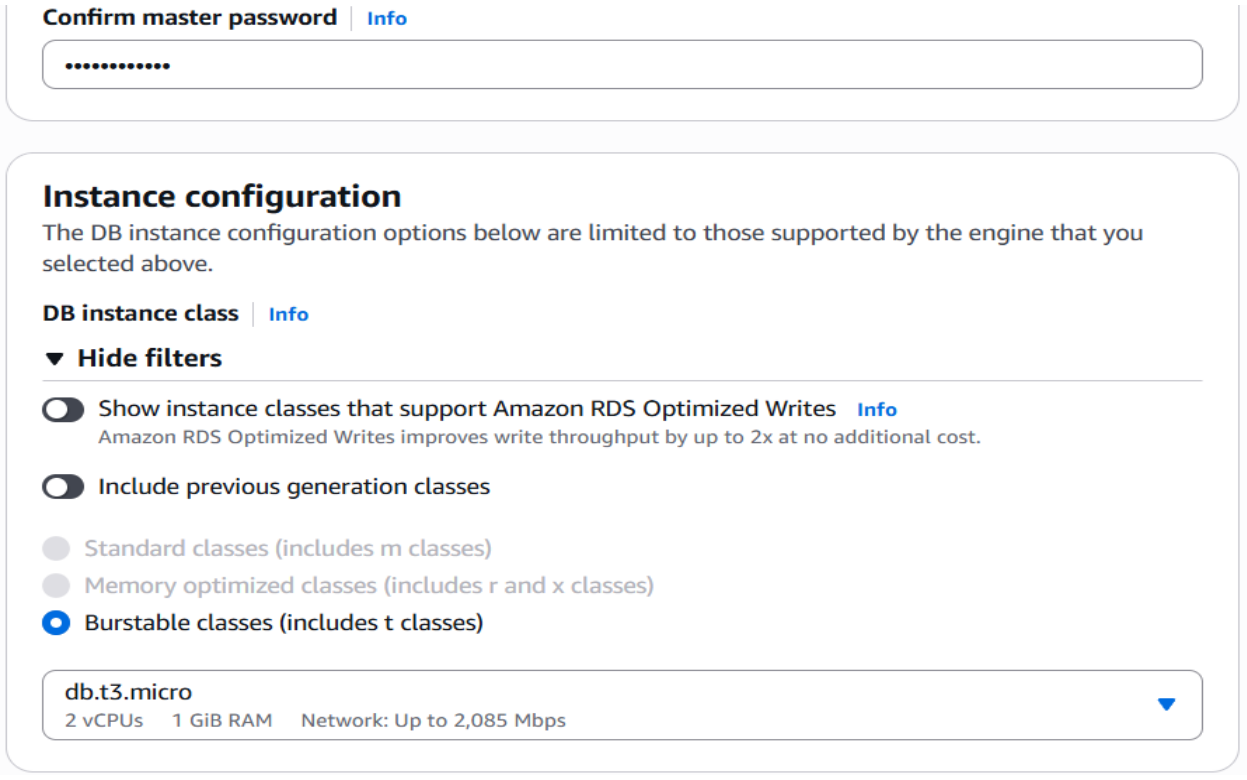
- **Settings:**
  - Engine: MySQL
  - Instance Type: `db.t3.micro`
  - Public Access: Enabled
- **Why:** This is the target database for inserting ingested data.



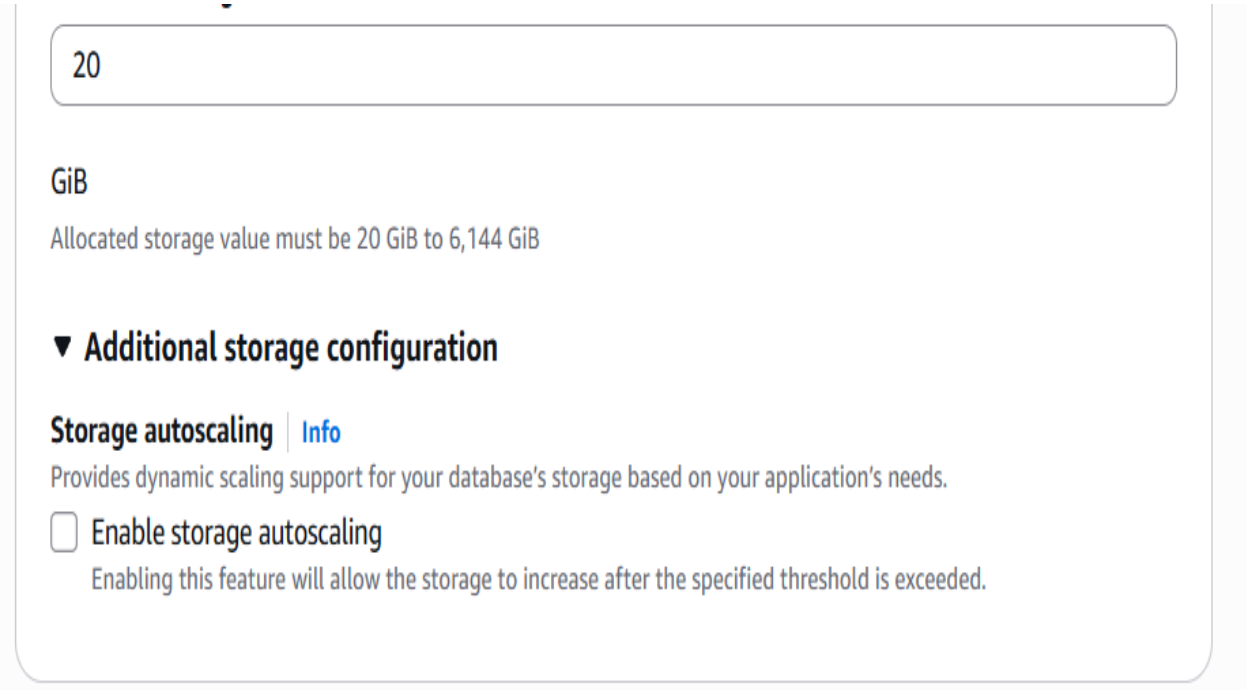
selected version and templates:



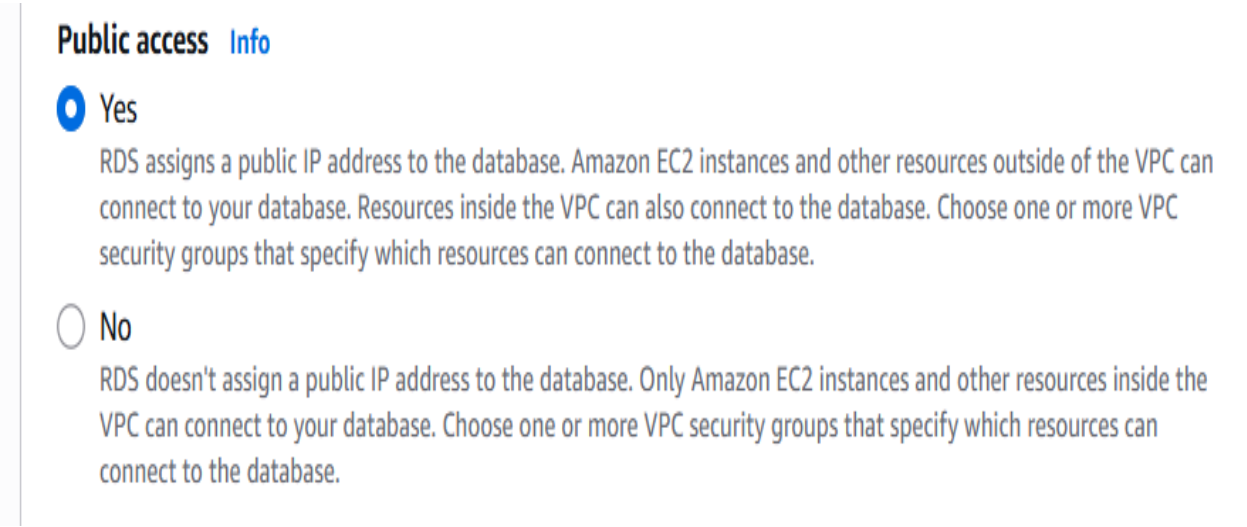
selected instance configuration to db.t3.micro



unchecked storage autoscaling:



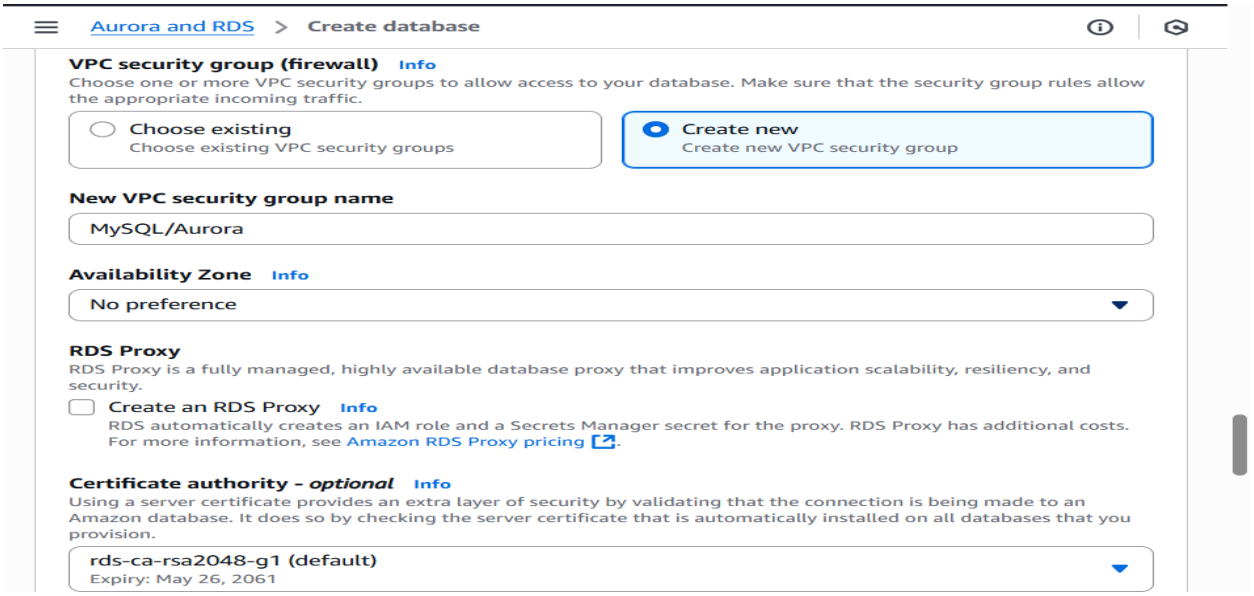
public access- yes



6. Configured Security Group for RDS:

- **Inbound Rule:** Port 3306 open to `0.0.0.0/0`

- **Why:** To allow Docker container (or local machine) to connect to RDS for data insertion.



7. Installed MySQL Client Locally:

- **Command:** `sudo apt install mysql-client`
- **Why:** To manually connect and verify if data has been inserted into RDS.

```
atharva@LAPTOP-A9SSNJEV:~$ python3 --version
Python 3.12.3
atharva@LAPTOP-A9SSNJEV:~$ pip3 --version
pip 24.0 from /usr/lib/python3/dist-packages/pip (python 3.12)
atharva@LAPTOP-A9SSNJEV:~$ docker --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
atharva@LAPTOP-A9SSNJEV:~$ mysql --version
mysql  Ver 8.0.42-0ubuntu0.24.04.1 for Linux on x86_64 ((Ubuntu))
atharva@LAPTOP-A9SSNJEV:~$ aws --version
aws-cli/2.27.35 Python/3.13.3 Linux/6.6.87.2-microsoft-standard-WSL2 exe/x86
_64.ubuntu.24
atharva@LAPTOP-A9SSNJEV:~$
```

8. Created Database and Table in RDS:

- **Database:** `ingestion_db`
- **Table:** `users`
- **Why:** The script requires a valid database and table to insert the data.

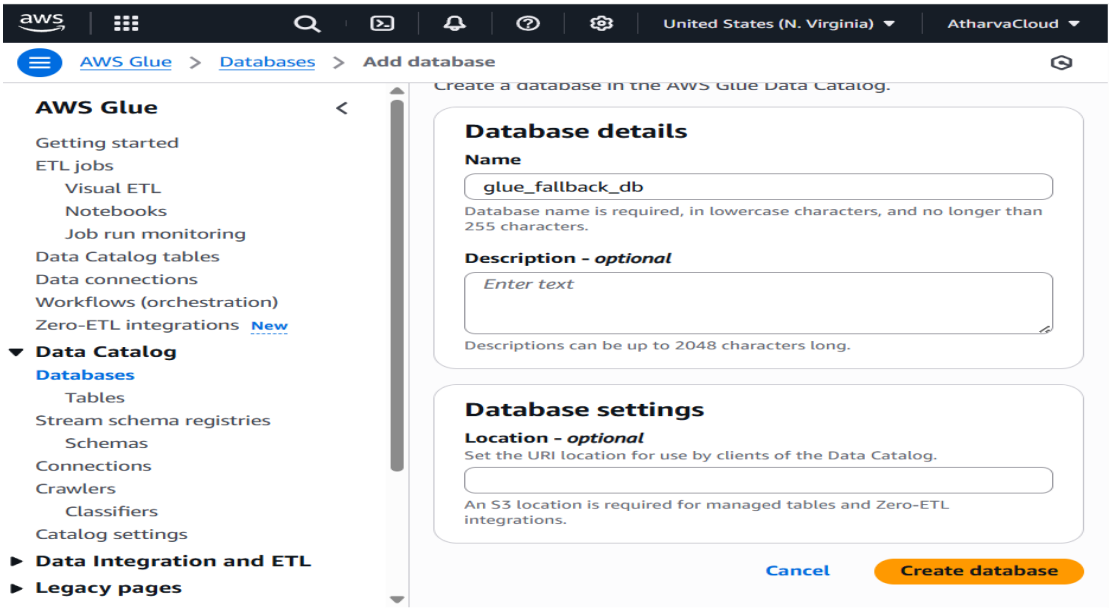
```
mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ingestion_db       |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.51 sec)

mysql> use ingestion_db
;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

9.Created AWS Glue Database:

- **Name:** `glue_fallback_db`
- **Why:** Used as a fallback when RDS ingestion fails.

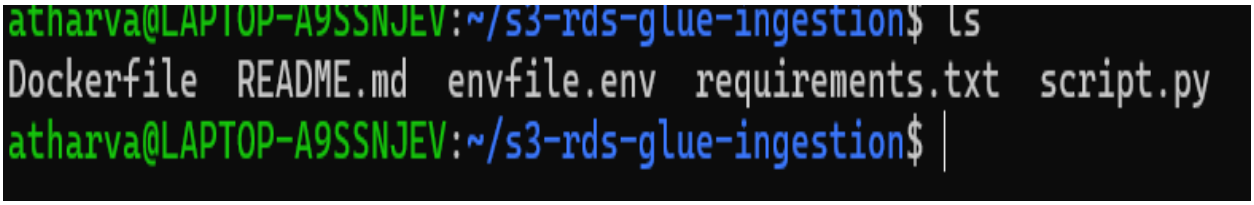


10. Installed Python Dependencies:

- `boto3`, `pandas`, `sqlalchemy`, `pymysql`
- **Why:** Required for connecting to AWS services, data parsing, and database insertion.

- **`pip install boto3`** – Installs AWS SDK for Python to interact with AWS services like S3, RDS, and Glue using `boto3.client()` and `boto3.resource()`.
- **`pip install pandas`** – Installs pandas for reading and processing CSV data from S3 using `pd.read_csv()` and exporting DataFrames to SQL or files.
- **`pip install sqlalchemy`** – Installs SQLAlchemy to create a connection engine to RDS and insert DataFrames into tables using `df.to_sql()`.
- **`pip install pymysql`** – Installs the MySQL connector required by SQLAlchemy to communicate with RDS using the dialect `mysql+pymysql://` in connection strings.

11. Created Project Folder Structure:



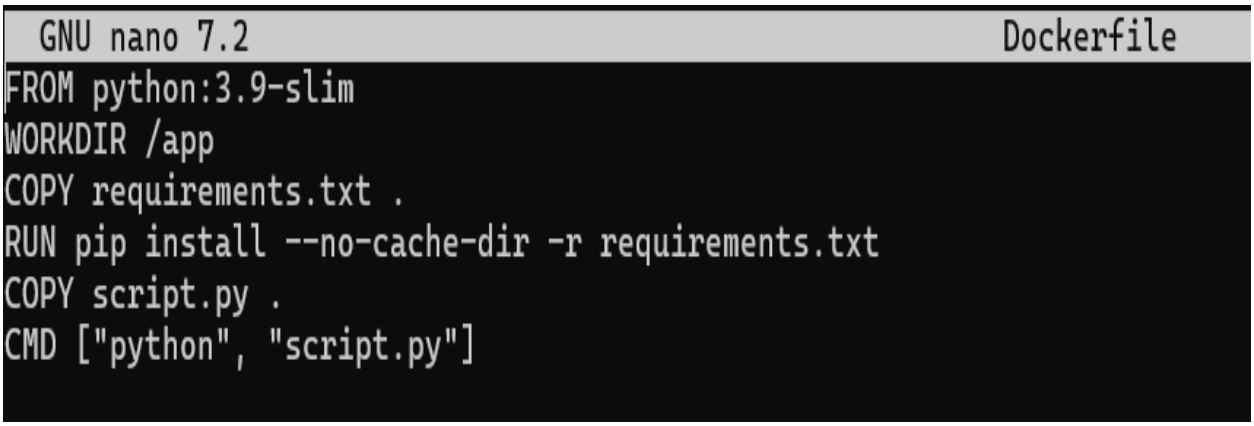
12. Created script.py:

- Contains logic to:
  1)Read from S3
  2)Insert into RDS
  3)Fallback to Glue if RDS fails
- Uses `os.environ.get()` for security.

13. Created requirements.txt:

Why: Lists dependencies for Docker

14.Created Dockerfile:



Why: Packages the app for consistent deployment.

15.Created envfile.env:

- Stores environment variables:
  - AWS keys
  - S3 bucket info
  - RDS credentials
  - Glue fallback config
- **Why:** Keeps sensitive credentials secure and out of source code.

16. Built Docker Image:

docker build --no-cache -t s3-rds-glue-app .

O/P:

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 9.216kB Step 1/6 : FROM python:3.9-slim 3.9-slim: Pulling from library/python 59e22667830b: Already exists 91067d9e3807: Already exists 02ba7daa58b6: Already exists 58b24570cddd: Already exists Digest: sha256:969463b94ba821ddf40b0ebed54ef4252161a8af1ea0de0ba9eab44b41b48308 Status: Downloaded newer image for python:3.9-slim ---> 563a905f7a66 Step 2/6 : WORKDIR /app ---> Running in f3439421821f ---> Removed intermediate container f3439421821f ---> 272fe75420d0 Step 3/6 : COPY requirements.txt . ---> 7c86b96a31bc Step 4/6 : RUN pip install --no-cache-dir --default-timeout=120 -r requirements.txt ---> Running in 08a3cfdfbfcd Collecting boto3 Downloading boto3-1.39.10-py3-none-any.whl (139 kB)

139.9/139.9 kB 214.0 kB/s eta 0:00:00 Collecting pandas Downloading pandas-2.3.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.4 MB)

12.4/12.4 MB 507.9 kB/s eta 0:00:00 Collecting sqlalchemy Downloading sqlalchemy-2.0.41-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.2 MB)

— 3.2/3.2 MB 497.7 kB/s eta 0:00:00 Collecting pymysql Downloading PyMySQL-1.1.1-py3-none-any.whl (44 kB)

— 45.0/45.0 kB 53.4 MB/s eta 0:00:00 Collecting s3transfer<0.14.0,>=0.13.0 Downloading s3transfer-0.13.1-py3-none-any.whl (85 kB)

85.3/85.3 kB 521.9 kB/s eta 0:00:00 Collecting jmespath<2.0.0,>=0.7.1 Downloading jmespath-1.0.1-py3-none-any.whl (20 kB) Collecting botocore<1.40.0,>=1.39.10 Downloading botocore-1.39.10-py3-none-any.whl (13.9 MB)

13.9/13.9 MB 954.7 kB/s eta 0:00:00 Collecting numpy>=1.22.4 Downloading numpy-2.0.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (19.5 MB)

— 19.5/19.5 MB 1.2 MB/s eta 0:00:00 Collecting tzdata>=2022.7 Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)

347.8/347.8 kB 1.6 MB/s eta 0:00:00 Collecting pytz>=2020.1 Downloading pytz-2025.2-py2.py3-none-any.whl (509 kB)

509.2/509.2 kB 848.0 kB/s eta 0:00:00 Collecting python-dateutil>=2.8.2 Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)

229.9/229.9 kB 895.6 kB/s eta 0:00:00 Collecting typing-extensions>=4.6.0 Downloading typing_extensions-4.14.1-py3-none-any.whl (43 kB)

— 43.9/43.9 kB 2.0 MB/s eta 0:00:00 Collecting greenlet>=1 Downloading greenlet-3.2.3-cp39-cp39-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (580 kB)

580.8/580.8 kB 1.2 MB/s eta 0:00:00 Collecting urllib3<1.27,>=1.25.4 Downloading urllib3-1.26.20-py2.py3-none-any.whl (144 kB)

144.2/144.2 kB 2.0 MB/s eta 0:00:00 Collecting six>=1.5 Downloading six-1.17.0-py2.py3-none-any.whl (11 kB) Installing collected packages: pytz, urllib3, tzdata, typing-extensions, six, pymysql, numpy, jmespath, greenlet, sqlalchemy, python-dateutil, pandas, botocore, s3transfer, boto3 Successfully installed boto3-1.39.10 botocore-1.39.10 greenlet-3.2.3 jmespath-1.0.1 numpy-2.0.2 pandas-2.3.1 pymysql-1.1.1 python-dateutil-2.9.0.post0 pytz-2025.2 s3transfer-0.13.1 six-1.17.0 sqlalchemy-2.0.41 typing-extensions-4.14.1 tzdata-2025.2 urllib3-1.26.20 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

[notice] A new release of pip is available: 23.0.1 -> 25.1.1 [notice] To update, run: pip install --upgrade pip ---> Removed intermediate container 08a3cfdfbfcd ---> 02c765da8aec Step 5/6 : COPY script.py . ---> aec8a3abe23e Step 6/6 : CMD ["python", "script.py"] ---> Running in 4273046a5329 ---> Removed intermediate container 4273046a5329 ---> c35def22c497 Successfully built c35def22c497 Successfully tagged s3-rds-glue-app:latest atharva@LAPTOP-A9SSNJEV:~/s3-rds-glue-ingestion$ docker run --rm

17.Ran Docker Container:

```
atharva@LAPTOP-A9SSNJEV:~/s3-rds-glue-ingestion$ docker run --rm \
  -e AWS_ACCESS_KEY_ID=AKIAWNHTHORWZTM66C63 \
  -e AWS_SECRET_ACCESS_KEY=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 \
  -e AWS_DEFAULT_REGION=us-east-1 \
  -e S3_BUCKET_NAME=ingestiontest2004 \
  -e CSV_FILE_KEY=s3bucket.csv \
  -e RDS_HOST=rds-ingestion.cq5yqyu20eg1.us-east-1.rds.amazonaws.com \
  -e RDS_USER=admin \
  -e RDS_PASSWORD=RdsIngest#2025 \
  -e RDS_DB_NAME=ingestion_db \
  -e RDS_TABLE_NAME=users \
  -e GLUE_DB_NAME=glue_fallback_db \
  -e GLUE_TABLE_NAME=users_fallback \
  -e GLUE_S3_PATH=s3://ingestiontest2004/fallback-data/ \
  s3-rds-glue-app
📥 Reading CSV from S3...
🔗 Connecting to RDS and uploading data...
✅ Data uploaded to RDS successfully.
```

18. Verified RDS Insertion:

- Used MySQL client to check if records appeared in `users` table.

```
atharva@LAPTOP-A9SSNJEV:~/s3-rds-glue-ingestion$ mysql -h rds-ingestion.cq5y
qyu20eg1.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 90
Server version: 8.0.41 Source distribution

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

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 statemen
t.
```

```
mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ingestion_db       |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.51 sec)

mysql> use ingestion_db
;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

Data Stored:

```
mysql> select * from users
    -> ;
+------+----------+------+
| id   | name     | age  |
+------+----------+------+
|    1 | Atharva  |   30 |
|    2 | Om       |   25 |
|    3 | Tejas    |   40 |
|    4 | Shreyas  |   22 |
+------+----------+------+
4 rows in set (0.56 sec)
```

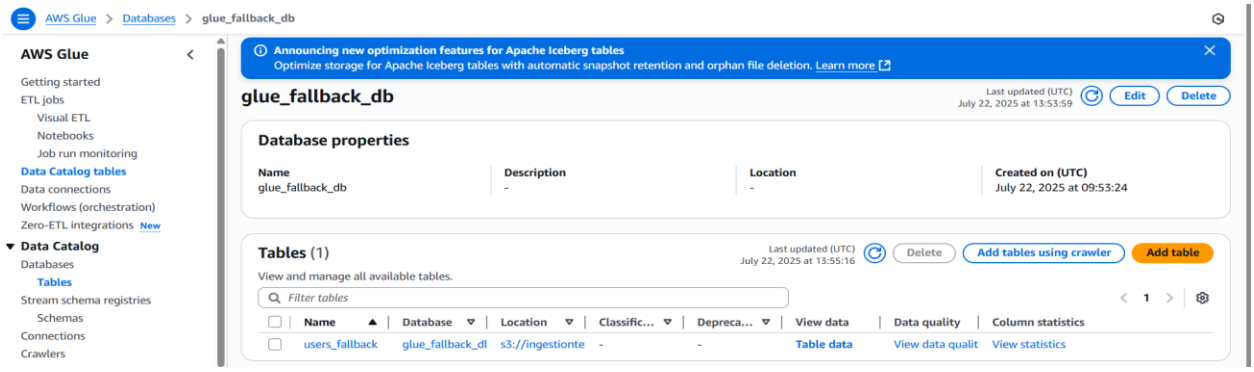
19. Simulated RDS Failure:

- Changed RDS password in env file to an incorrect one.
- **Result:**

```
atharva@LAPTOP-A9SSNJEV:~/s3-rds-glue-ingestion$ docker run --rm   -e AWS_AC
CESS_KEY_ID=AKIAWNHTHORWZTM66C63   -e AWS_SECRET_ACCESS_KEY=JZIJuyjyY7UQLQ0H
gNtqotHrFOL5chtMH1tWfjKM   -e AWS_DEFAULT_REGION=us-east-1   -e S3_BUCKET_NA
ME=ingestiontest2004   -e CSV_FILE_KEY=s3bucket.csv   -e RDS_HOST=rds-ingest
ion.cq5yqyu20eg1.us-east-1.rds.amazonaws.com   -e RDS_USER=admin   -e RDS_PA
SSWORD=RdsIngest#20   -e RDS_DB_NAME=ingestion_db   -e RDS_TABLE_NAME=users
  -e GLUE_DB_NAME=glue_fallback_db   -e GLUE_TABLE_NAME=users_fallback   -e
GLUE_S3_PATH=s3://ingestiontest2004/fallback-data/   s3-rds-glue-app
📥 Reading CSV from S3...
🔗 Connecting to RDS and uploading data...
❌ Error uploading to RDS: (pymysql.err.OperationalError) (1045, "Access den
ied for user 'admin'@'152.58.16.117' (using password: YES)")
(Background on this error at: https://sqlalche.me/e/20/e3q8)
📥 Reading CSV from S3...
⚠️ Upload to RDS failed. Falling back to Glue...
✅ Fallback: Glue Table created successfully.
```

20. Verified Glue Table:

- Checked AWS Glue Console > `glue_fallback_db`
- Table `users_fallback` successfully created with correct schema.



SUMMARY:

This project is a Dockerized Python application that automates data ingestion from an Amazon S3 CSV file into an Amazon RDS (MySQL) database. If the RDS upload fails, the system automatically falls back to AWS Glue Data Catalog. It ensures a reliable and fault-tolerant data pipeline using AWS services like S3, RDS, Glue, and IAM.