Report: Reddit Data Pipelines Project

Abbassi Moetez

February 2025

1 Introduction:

This report presents the creation of a seamless ETL process by the integration of several tools and software including Docker, Apache Airflow, Celery Executor, PostgreSQL, S3 Buckets, AWS Glue, Athena, and Redshift. The main goal of this project is to automate the data collection and processing workflow in order to store and analyze real-time Reddit data efficiently.

2 Project Setup:

To orchestrate the data pipelines, I have set up a workflow using Apache Airflow with Celery Executor. The overall workflow was initiated inside a Docker environment.

The process starts with configuring a "docker-compose.yml" file, which is mainly downloaded from the official Apache Airflow website and edited to match the goal of my project. The content of "docker-compose.yml" includes the characteristics of my Airflow environment, which contains the following items:

- **x-airflow-common:** Ensures consistency of multiple Airflow Services by defining common configurations.
- **Postgres Service**: The back-end database for Airflow, which will store the data and record the executions.
- **Redis Service**: Redis acts as a message broker for Celery, facilitating communication between the Airflow scheduler and Celery workers. When the Airflow scheduler assigns a task, it sends a message to Redis. Celery workers then pick up the task from Redis and execute it.
- **Airflow Webserver**: The UI to manage Airflow DAGs and tasks.
- **Airflow Scheduler:** Continuously scan the DAG file to ensure that Airflow runs in accordance with the required configurations.
- **Airflow Worker:**: Execute tasks given by Celery in a scalable manner.
- **Airflow Trigger:** Orders the scheduling and execution of tasks by conditions or events that determine when a task should be executed within a DAG.
- **Volumes:** Stores the Postgres database and allows communication between containers.

- **Airflow Init:** Initializes the Database, creates the root user followed by the needed permissions.

Then, the images of Redis, Postgres, and Apache Airflow are built using Docker, along with the needed requirements from "requirements.txt" by using the command "docker compose up -d –build". The Airflow server will be hosted and we will get the following interface:

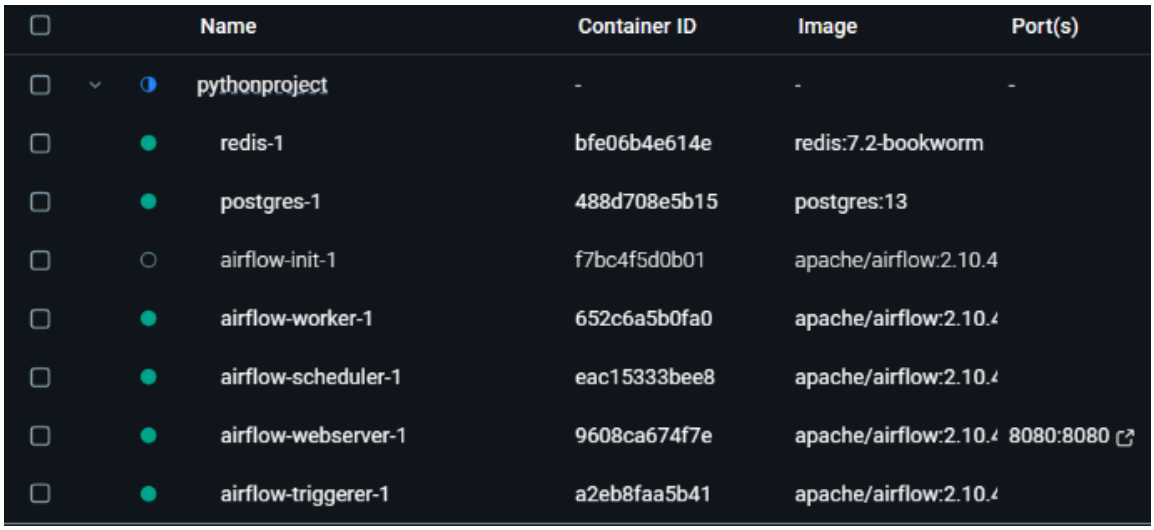

Figure 1: Docker UI after building the images and running the Webserver

3 Reddit ETL Process

The first ETL process is included in the "reddit-etl.py" file, and it has these following items: - **Extraction:**

First, we login into Reddit and create an app with the type 'script' to retrieve the data. We will receive secret keys to access Reddit's API from our app, and then we will add these keys to our configuration file to ensure the automation of the extraction process. We use the "praw" library in Python to fetch the reddit data from a subreddit of our choice (dataengineering subreddit in this project) The obtained dataframe will include several features such as comments, posts, date, etc.

- **Transformation:**

The reddit data will be simply redesigned to ensure consistency within the data and make it easier to manipulate in the AWS.

-Loading:

We obtain the final dataframe which will be automatically processed by Airflow, and it will be available almost instantly in the Webserver:

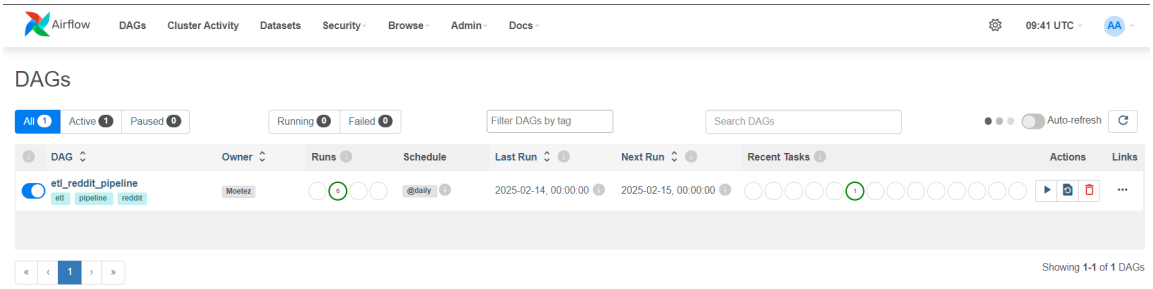

Figure 2: Airflow UI after obtaining Reddit Data

4 AWS Data Processing:

4.1 Uploading data to AWS S3:

During this phase, we will be making a pipeline to connect our Airflow DAG to an S3 Bucket in Amazon Web Services.

We start by getting our AWS confidential keys and adding them to the configuration file. Then, the "aws-etl.py" automatically connects an S3 bucket to our Airflow DAGs, fetches our final dataframe and stores it as a CSV file.

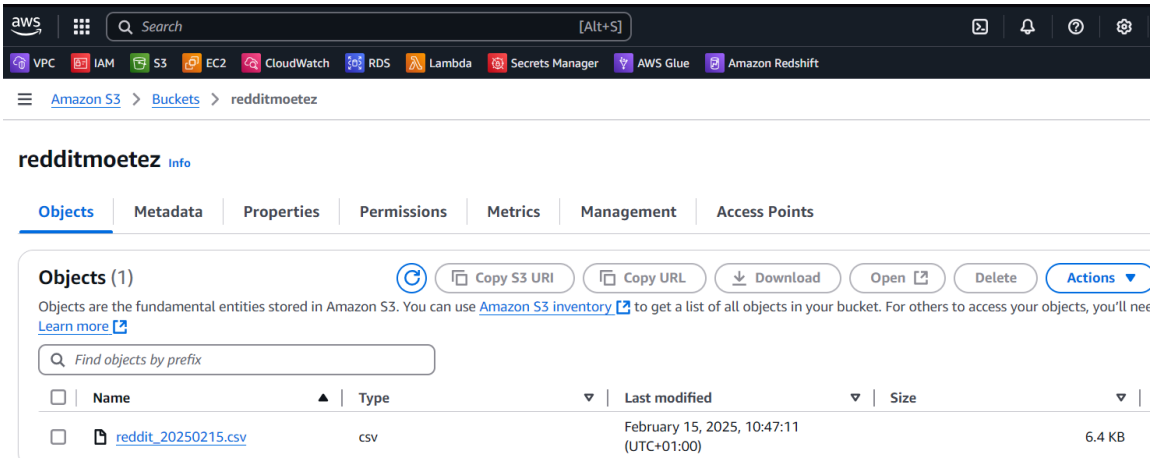

Figure 3: Uploading data to S3 Bucket

4.2 Data Processing with several AWS Services:

4.2.1 AWS Glue:

AWS Glue can create a Crawler that catalogs the data stored in S3 bucket, it also has another internal ETL Process to prepare the data for further analysis.

The Crawler's ETL Process in this project is simple as it only combines the columns 'Edited, Spoiler, Stickied' into one column, and then drops the previous

columns as it will have no use. After completing the internal ETL Process, the transformed dataframe will be loaded to our S3 bucket.

Running the Crawler will result in the following UI:

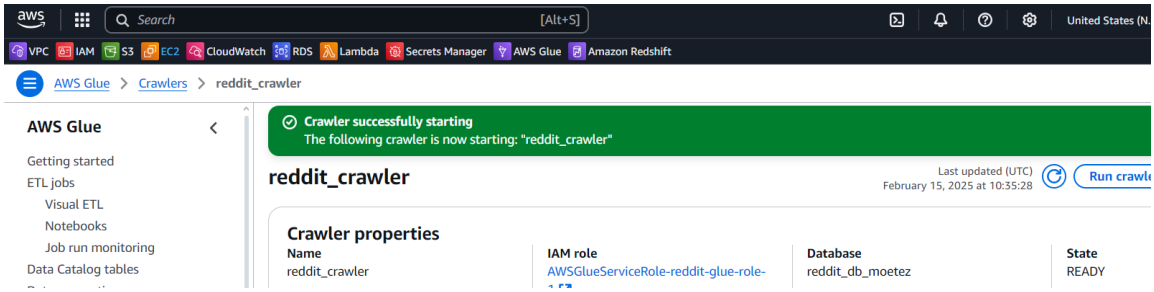

Figure 4: Successfully ran and started Crawler in AWS Glue

4.2.2 AWS Athena:

After obtaining our transformed data from AWS Glue and storing it in our S3 bucket, AWS Athena will allow us to run SQL queries directly on S3-stored data.

The benefit of Athena in this case is that it allows for EDA and Data Visualization without the need to load the transformed data into a database.

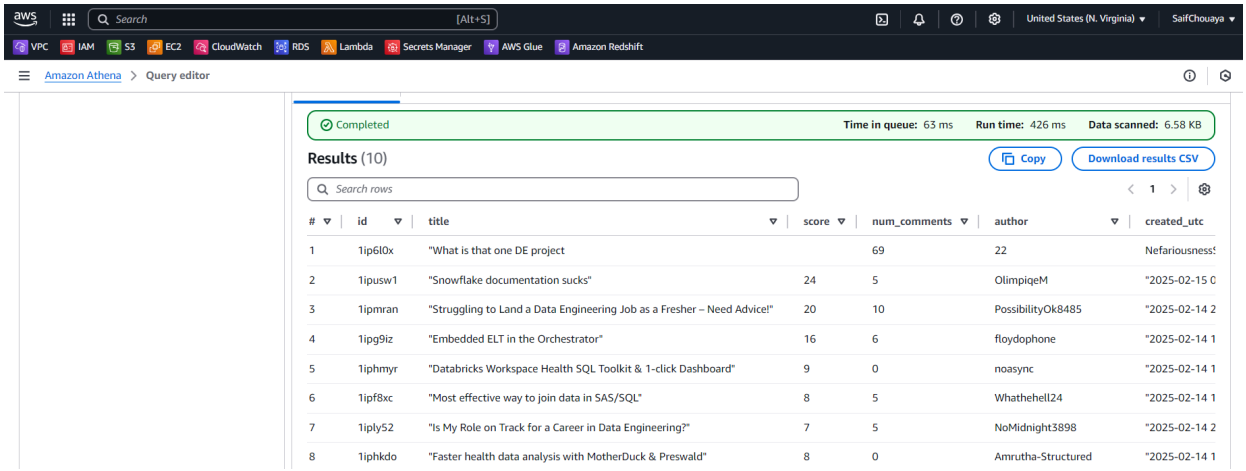

Figure 5: AWS Athena UI

4.2.3 AWS Redshift:

Unlike AWS Athena, we start by loading the structured dataframe from S3 bucket into AWS Redshift, which will ensure efficient querying and will allow us to make dashboards and analytics on our dataframe.

The purpose of Redshift in this project's case is very similar to Athena, but it still allows us to perform more complex queries and aggregations that will derive great insights from our data.

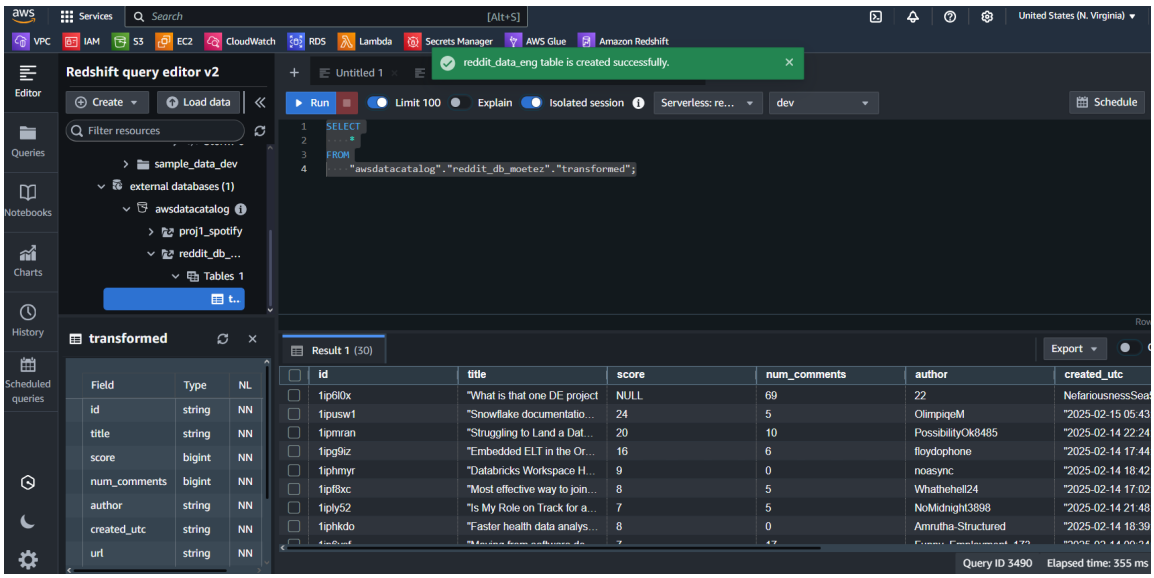

Figure 6: AWS Redshift: Displaying the Table

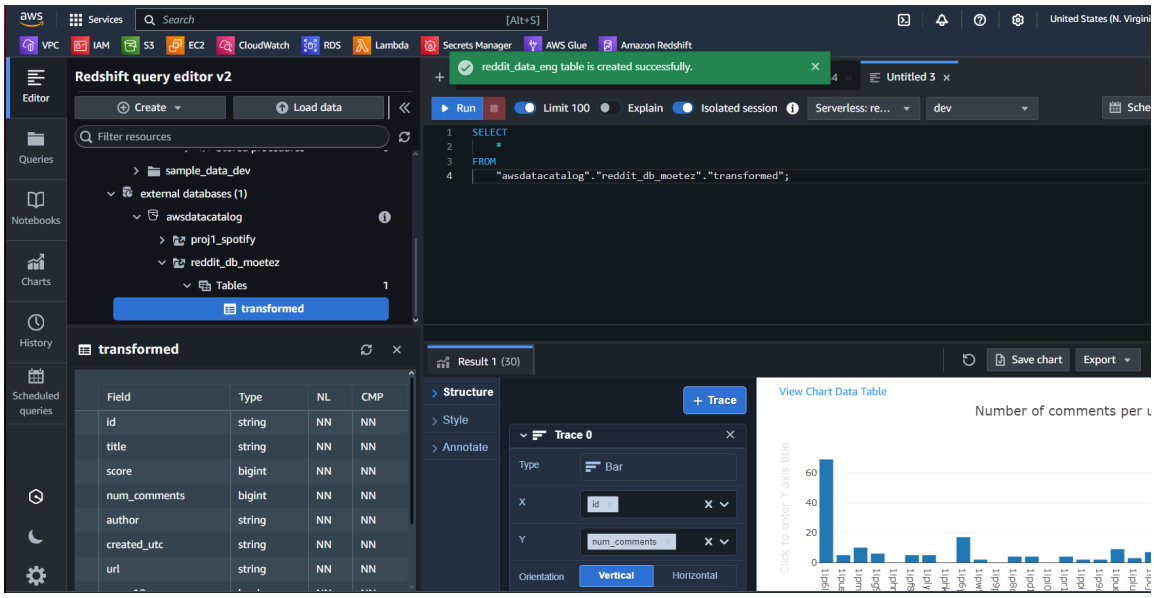

Figure 7: AWS Redshift: Data Visualization

5 Conclusion

This project successfully demonstrated the end-to-end process of building a data pipeline using Reddit API, Airflow, Celery, Postgres, Docker, Redis, S3, AWS Glue, Athena, and Redshift. It automates the process of Extraction, Transforming and Loading Reddit's subreddit data while ensuring efficient querying and visualization through Amazon Web Services. The project still needs several future enhancements, including real-time streaming, Machine Learning integration and extended data analysis.