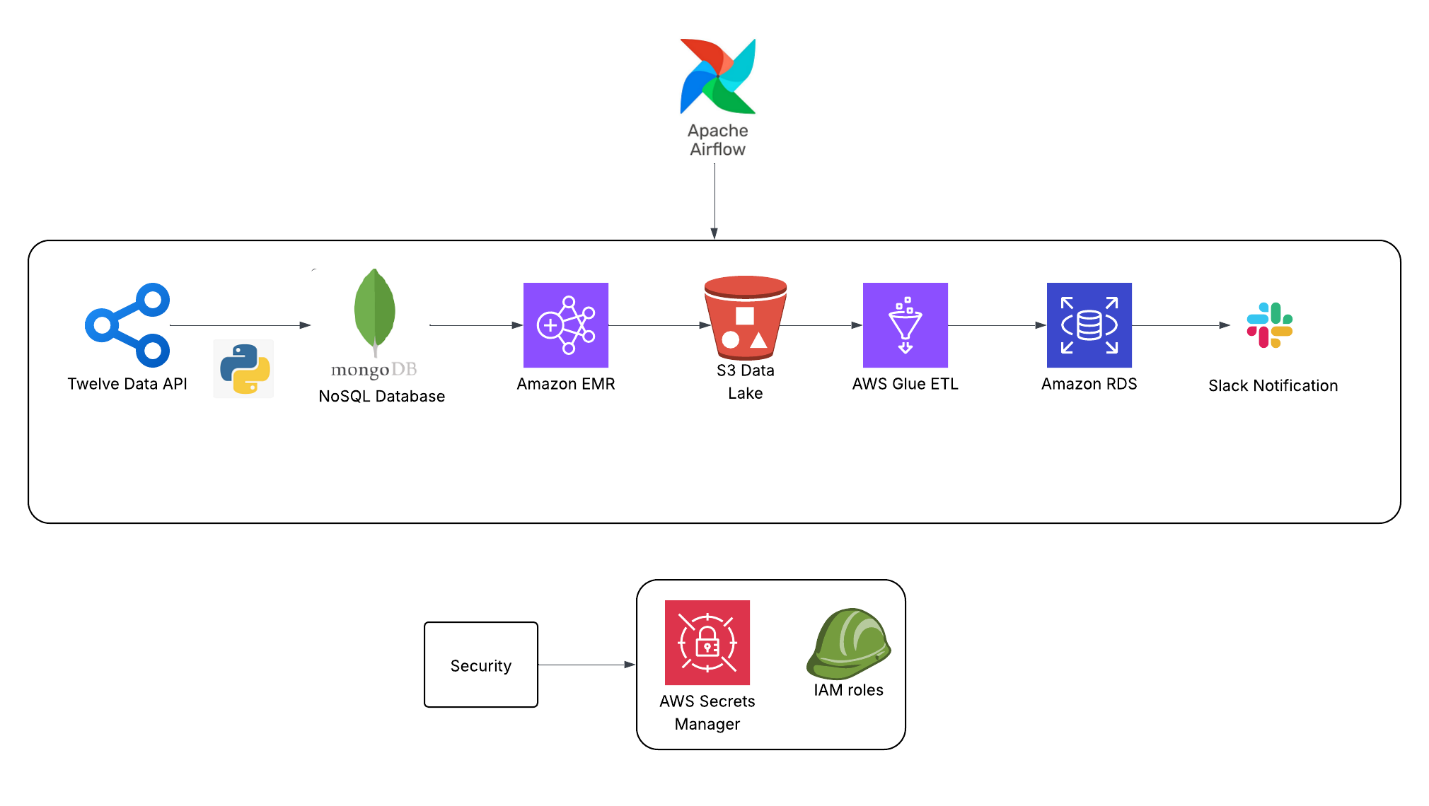
# Stock Market Data Engineering Pipeline with Twelve Data, MongoDB, AWS EMR and RDS



**Problem Statement:**

Build a scalable and automated data pipeline that fetches stock price data every 5 minutes for companies like Apple, Google, Tesla, Microsoft, and Nvidia from TwelveData API. The pipeline should store raw data in MongoDB, transform it into Parquet using PySpark on AWS EMR, aggregate it using AWS Glue, and store final data in RDS for analytics.

**Business Use Cases:**

Real-time stock trend monitoring for financial analysts

Historical stock data archiving for research and reporting

Automated price alerts for trading bots

Scalable infrastructure for financial data aggregation

**Tools Used:**

Initial data Ingestion: TwelveData API, Python

Staging Layer: MongoDB Atlas

Data Processing: AWS EMR with PySpark, AWS Glue

Data Pipeline Orchestration: Apache Airflow

Monitoring: Apache Airflow, Slack Notifications

Security: AWS Secrets Manager, IAM Roles

Version Control: GitHub

Host & Run Airflow and initial Ingestion Code: AWS EC2 instance

Programming Languages Used: Python, SQL, Pyspark API

**Data Set Info:**

Source: TwelveData API

Format: JSON

Variables:

symbol: Stock symbol (e.g., AAPL, TSLA)

datetime: Timestamp

open, high, low, close: Stock prices

volume: Volume of trade

**Data Set Explanation:**

* Each API call gives OHLCV (Open, High, Low, Close, Volume) data at 5-minute

intervals.

* The data must be validated (e.g., non-null prices, chronological order).
* MongoDB will be the staging layer before processing.
* PySpark will convert JSON → DataFrame → Parquet (with schema).
* S3 stores Parquet files partitioned by Company.
* Glue jobs summarize the data.
* RDS holds the cleaned, aggregated output.

**Project Set Up requirements and steps:**

 **AWS EC2 Instance for Apache Airflow**

* Use an **AWS EC2 instance** to host Apache Airflow and the Python code responsible for initial data ingestion from the **TwelveData API**. (Alternatively, **AWS Lambda** can be used for Python code execution.)

 **IAM Roles and Permissions**

* Ensure the EC2 instance has the necessary **IAM roles** to interact with AWS services like **EMR**, **S3**, **Glue**, **RDS**, and **Secrets Manager**.

 **VPC and Subnet Configuration**

* Deploy the EC2 instance in the appropriate **VPC** and **Subnet**. Make sure the **Security Group** allows inbound traffic on **port 8080** to access the **Airflow Web UI**.

 **MongoDB Atlas Setup**

* Set up a **MongoDB Atlas** database, and create the required collections as part of the code execution (as opposed to doing it manually).

 **S3 Bucket for Code and Bootstrapping**

* Store the **Pyspark** code for **AWS EMR** data processing and the **requirements.sh** file (for EMR bootstrapping) in an **S3 bucket**.

 **AWS Glue Catalog and Crawler**

* Update the **AWS Glue Catalog** with the location of **EMR output** and **RDS database tables**. Use an **AWS Glue Crawler** to automate metadata discovery where applicable.

 **JDBC Connector Setup**

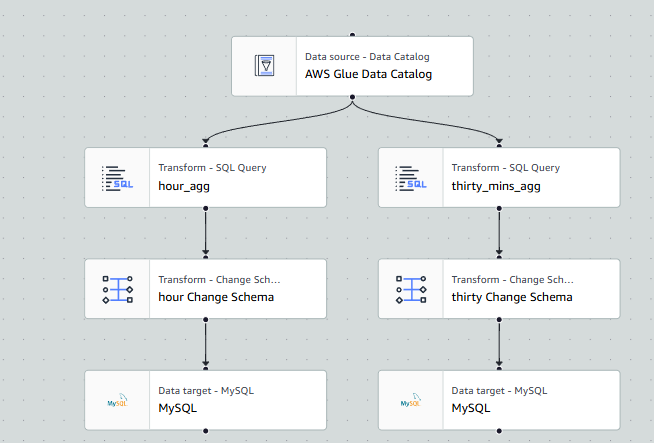
* Create a **JDBC Connector** in **AWS Glue** to connect to the **AWS RDS** database.

 **VPC Configuration for Glue**

* As **AWS RDS** is hosted within a **VPC**, ensure that **AWS Glue** and the JDBC connector are also hosted within the same **VPC** to ensure secure communication.

 **AWS Glue ETL Job Setup**

* Set up the **AWS Glue ETL job** using **Glue Visual ETL**.



* **AWS RDS Database Setup**
  + Create the required **AWS RDS databases** to store the final results of your ETL process.
* **AWS EMR Cluster Configuration**
  + **AWS EMR** will be automatically created and terminated as part of the **Airflow DAG**. Ensure that **EMR** has the appropriate permissions to fetch data from **MongoDB Atlas**.
* **Apache Airflow Installation and Setup**
  + Install **Apache Airflow** on the EC2 instance.
  + Set up the **Airflow webserver** and **scheduler** to manage your DAGs.
* **Airflow Connection Configurations**
  + Set up an **SSH connection** in Airflow to your EC2 instance for accessing **MongoDB**.
  + Set up an **AWS connection** in Airflow for access to **EMR**, **RDS**, and **Glue**.
  + Set up a **Slack connection** in Airflow for sending notifications on task statuses.
* **Define Airflow Tasks**
  + Define all the tasks based on the provided **DAG file** and ensure proper configuration of task dependencies.
* **Schedule the DAG**
  + Set up the DAG's **schedule** (or trigger it manually, depending on your use case).
* **Additional Resources**
  + Please refer to the **requirements text file** attached in each subdirectory for more detailed information on specific configurations and setup details.