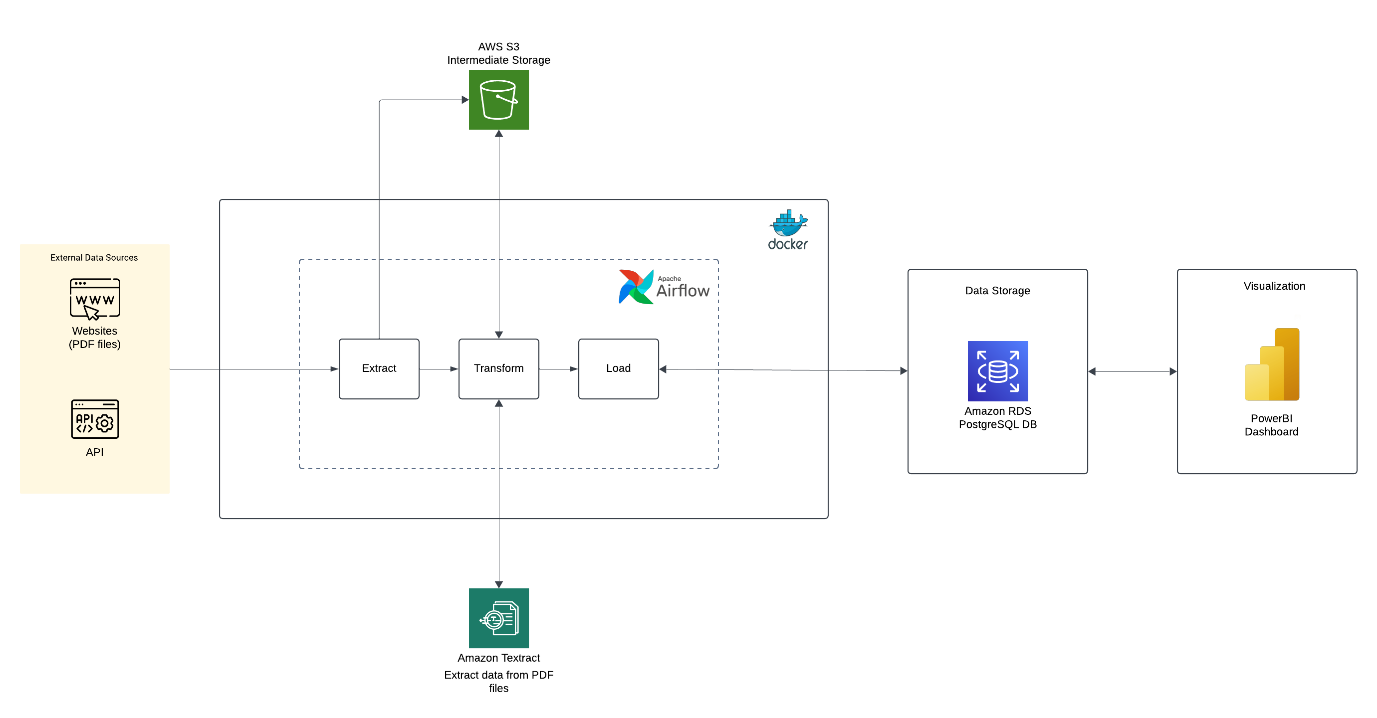
**Federal Spending Tracker Tool:**

**Overview**

The Federal Spending Tracker Tool is designed to automate the extraction, transformation, and loading (ETL) of data related to federal spending, with a focus on housing programs in Puerto Rico. This document provides an overview of the tool's architecture, detailed procedural guidelines, and instructions for running the ETL pipelines.

The system employs AWS services for data storage and processing, Apache Airflow for workflow orchestration, and AWS RDS PostgreSQL for data storage. The pipelines are intended to handle data from multiple sources, including PDFs, APIs, and manually uploaded documents.

**Architecture Overview**



The architecture diagram provides a high-level view of the tool, highlighting data sources, AWS services, Apache Airflow orchestration, and the final database. The architecture includes the following key components:

* **Data Sources**: Web pages, APIs, and manual uploads of pdf files.
* **AWS Services**: S3 for storage, Textract for document processing, and RDS PostgreSQL for data storage.
* **Workflow Orchestration**: Apache Airflow to automate and manage the ETL processes.

**Database Schema Overview**A screenshot of a computer

Description automatically generated

The database schema diagram presents the structure of the PostgreSQL database used for storing processed data.

**ETL Pipelines:**

**Pipeline 1 & 2: Webpage PDF Data Extraction (ETL\_financial\_reports, ETL\_history\_reports)**

1. **Data Extraction**
   * Extract PDF files from designated web pages.
   * Store the PDF files in the AWS S3 raw bucket.
2. **Data Processing**
   * Use AWS Textract to extract relevant information from the PDF files.
   * Convert the extracted data into a structured format (CSV).
   * Save the processed data in the S3 processed bucket.
3. **Data Loading**
   * Fetch the processed CSV files from the S3 processed bucket.
   * Ingest the data into AWS RDS PostgreSQL tables.

**Pipeline 3: USA Spending Data Extraction (ETL\_usa\_spending)**

1. **Data Extraction**
   * Use Python's requests library to make GET requests to the API.
   * Store the retrieved data in JSON format in the AWS S3 raw bucket.
2. **Data Processing**
   * Transform the JSON data as needed (e.g., normalizing nested fields).
   * Save the processed data in the S3 processed bucket.
3. **Data Loading**
   * Load the processed data into PostgreSQL tables.

**Pipeline 4: Quarterly Performance Reports (ETL\_quarterly\_performance\_reports)**

* **Manual Task**: Manually download the PDF files from the provided sources and upload specific pages to the S3 raw bucket.

1. **Data Extraction and Processing**
   * Extract and process the data from the uploaded PDF pages using AWS Textract.
   * Save the processed output in the S3 processed bucket.
2. **Data Loading**
   * Load the processed data into PostgreSQL tables.

**Scheduling**

* All ETL pipelines are scheduled to run either daily or monthly to ensure up-to-date data availability.

**Error Handling**

* **Data Extraction Failures**: Retry logic is implemented for web scraping and API requests to handle intermittent network issues.
* **Data Transformation Errors**: Validation checks are applied to processed data, and any discrepancies are logged for further analysis.

**Security Considerations**

* **AWS Credentials**: Stored securely using Apache Airflow's connections feature.
* **Data Encryption**: Data in S3 is encrypted, and access is restricted using IAM policies.

**Manual Intervention Points**

* For the quarterly performance reports pipeline, manual intervention is required to download and upload specific pages from PDF files to the S3 bucket (raw).

**Lessons Learned**

* Automation of manual tasks significantly enhances pipeline efficiency.
* Incorporating data quality checks early in the pipeline prevents downstream issues.

**Steps to Run the Program**

1. Clone the Repository

* Clone the repository from GitHub:

git clone https://github.com/leelasagar1/federal\_spending\_tracking\_tool.git

cd federal\_spending\_tracking\_tool

1. Configure AWS Credentials
   * Update the AWS configuration file (config/aws\_config.json) with your AWS details:

{

"aws\_access\_key\_id": "ADD AWS ACCESS KEY",

"aws\_secret\_access\_key": "ADD AWS SECRET ACCESS KEY",

"region\_name": "ADD REGION"

}

1. Set Up S3 Buckets in AWS
   * Create two S3 buckets: one for raw data and one for processed data.
   * Update the S3 configuration file (config/S3\_config.json) with bucket details:

{

"raw\_bucket": "project-raw-data-files",

"processed\_bucket": "project-processed-data"

}

1. Set Up Apache Airflow Using Docker Compose

* First-time setup commands for Apache Airflow:

docker-compose up airflow-init

* After initializing Airflow, start all services:

docker-compose up -d

* To shut down the Airflow server and stop all containers:

docker-compose down

1. Running the Pipelines
   * To trigger the pipelines manually, go to the Airflow web UI (usually accessible at http://localhost:8080) and enable the DAGs as needed.
2. Manual Task Execution (Quarterly Performance Reports)
   * Download the PDF reports manually and upload them to the S3 raw bucket.
3. Validation and Verification
   * Validate the data in PostgreSQL tables using SQL queries to ensure data integrity.