Bluebikes Data Pipeline Documentation

1. Introduction

This document describes the Bluebikes data pipeline, built using Apache Airflow. The purpose of this pipeline is to automate the process of extracting Bluebikes trip data from an AWS S3 bucket, preprocessing it, and loading the cleaned data into a Google Cloud Platform (GCP) bucket. The data pipeline is designed to handle monthly updates of Bluebikes data, allowing for efficient, scalable, and consistent data ingestion and processing.

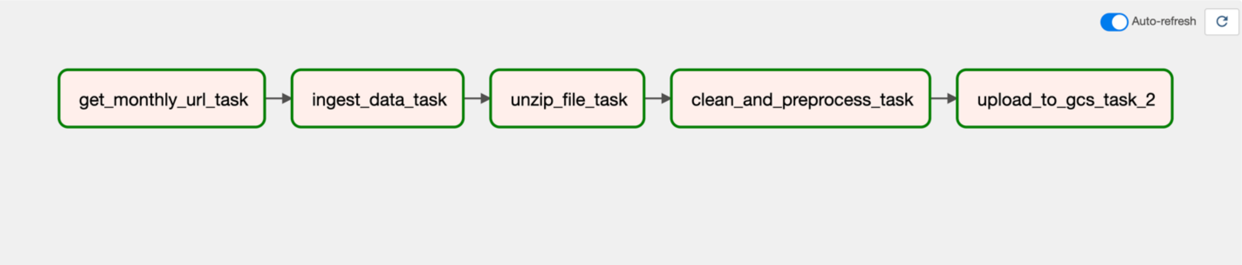
Our pipeline is implemented using Airflow's Directed Acyclic Graph (DAG) capabilities, ensuring a clear and structured flow of tasks. This setup allows for easy monitoring and maintenance of each step in the data pipeline.

2. Architecture Overview

The Bluebikes data pipeline consists of the following stages:

- **Data Extraction**: Retrieves monthly Bluebikes trip data from an AWS S3 bucket.
- **Data Ingestion**: Ingests the extracted data file into the pipeline.
- **Data Unzipping**: Unzips the downloaded file to make it available for processing.
- **Data Cleaning and Preprocessing**: Performs data cleaning, such as handling missing values, removing invalid data points, and standardizing formats.
- **Data Upload to GCP**: Saves the preprocessed data in a designated folder in a GCP bucket for future analysis.

Below is a visual representation of the DAG that controls this workflow:



3. Detailed DAG Description

Each task in the pipeline is represented by a node in the DAG. The following tasks are included:

- **get_monthly_url_task**: Fetches the URL for the latest monthly Bluebikes data from AWS S3. The URL points to the specific data file required for the current month.
- **ingest_data_task**: Ingests the data file from the AWS S3 bucket into the pipeline for processing.
- **unzip_file_task**: Unzips the data file to access the raw data. This step is essential for handling compressed datasets.
- **clean_and_preprocess_task**: Cleans and preprocesses the data by removing null values, correcting data formats, and ensuring data consistency.
- **upload_to_gcs_task**: Uploads the preprocessed data to a GCP bucket in a specific folder named Preprocessed_data, making it readily accessible for future analysis and storage.

The tasks are organized in the following order:

get_monthly_url_task >> ingest_data_task >> unzip_file_task >>
clean_and_preprocess_task >> upload_to_gcs_task

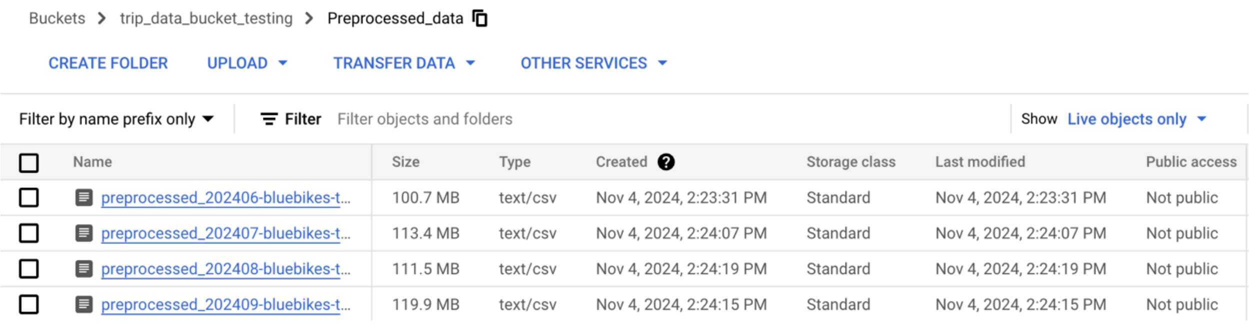
Commands to Run the Pipeline

To set up and run the Airflow pipeline, use the following Docker commands:

1. **Build the container (without cache)**: docker-compose build --no-cache
2. **Initialize Airflow**: docker-compose run airflow-init
3. **Start Airflow in detached mode**: docker-compose up -d
4. **Shut down Airflow and remove unused containers**: docker-compose down --remove-orphans

These commands ensure that Airflow is properly set up and ready to orchestrate the tasks within the pipeline.

4. GCP Bucket with Preprocessed Data

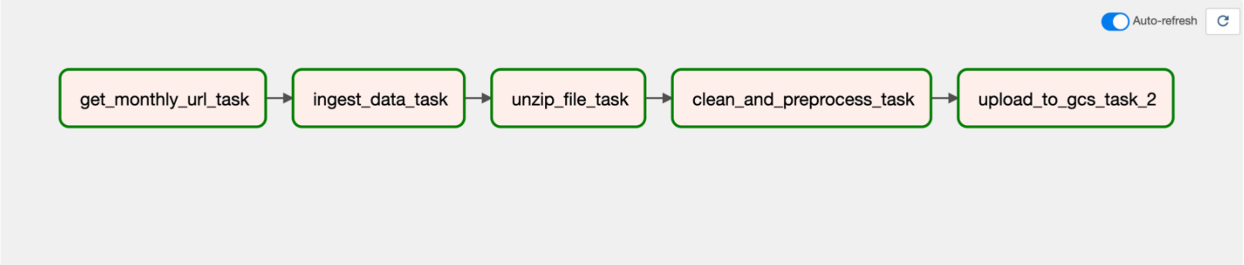

Buckets > trip_data_bucket_testing > Preprocessed_data
CREATE FOLDER
UPLOAD
TRANSFER DATA
OTHER SERVICES
Filter by name prefix only
Filter
Filter objects and folders

Show Live objects only

☐	Name	Size	Type	Created	Storage class	Last modified	Public access
☐	preprocessed_202406-bluebikes-t...	100.7 MB	text/csv	Nov 4, 2024, 2:23:31 PM	Standard	Nov 4, 2024, 2:23:31 PM	Not public
☐	preprocessed_202407-bluebikes-t...	113.4 MB	text/csv	Nov 4, 2024, 2:24:07 PM	Standard	Nov 4, 2024, 2:24:07 PM	Not public
☐	preprocessed_202408-bluebikes-t...	111.5 MB	text/csv	Nov 4, 2024, 2:24:19 PM	Standard	Nov 4, 2024, 2:24:19 PM	Not public
☐	preprocessed_202409-bluebikes-t...	119.9 MB	text/csv	Nov 4, 2024, 2:24:15 PM	Standard	Nov 4, 2024, 2:24:15 PM	Not public

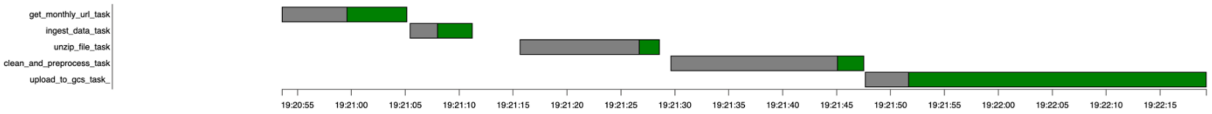
This screenshot shows the structure of the GCP bucket where preprocessed Bluebikes data files are stored. Each file is labeled with a timestamp and contains clean, ready-to-use data. These files are stored in a folder named Preprocessed_data in the GCP bucket trip_data_bucket_testing.

5. Airflow DAG



The above screenshot illustrates the DAG in the Airflow UI, showcasing the structured flow from data extraction to uploading the preprocessed data to GCP.

6. Task Timeline



This screenshot provides insight into the timeline and duration of each task within the DAG. Each task's runtime is represented, helping us identify any bottlenecks or tasks that may require optimization.

7. Performance Analysis and Monitoring

- **Long-Running Tasks**:
    - clean_and_preprocess_task and upload_to_gcs_task are the longest tasks, indicating potential areas for optimization.

- **Sequential Execution**:
    - Tasks are run sequentially due to dependencies, limiting opportunities for parallelism and contributing to the overall duration.
- **Potential Bottlenecks**:
    - clean_and_preprocess_task is the main bottleneck due to the time it takes for data processing.
    - upload_to_gcs_task also takes significant time, likely due to file size or network upload speed.
- **Successful Execution**:
    - All tasks completed successfully without retries, suggesting the pipeline is reliable.
- **Optimization Recommendations**:
    - Optimize clean_and_preprocess_task by streamlining data operations.
    - Explore options to speed up upload_to_gcs_task, such as file compression or improved network bandwidth.

8. Challenges and Solutions

During the setup and execution of this pipeline, we encountered the following challenges:

- **Data Ingestion and Unzipping**: Initially, we had issues with managing the compressed data files from S3. We addressed this by incorporating a specific unzipping task within the DAG.
- **Docker Configuration for Airflow**: Ensuring compatibility with Docker and Airflow required adjustments to the Dockerfile and docker-compose configurations. We resolved these issues by following Airflow's recommended Docker setup.

By documenting these challenges and their solutions, we ensure that future improvements and troubleshooting can build upon these experiences.

9. File Details

The data files are in CSV format and contain monthly Bluebikes trip data. Each file includes multiple fields such as trip duration, start and end locations, and timestamps. The clean_and_preprocess_task standardizes these fields, removes missing values, and organizes the data for storage in GCP.

10. Conclusion

This Airflow-based data pipeline efficiently automates the extraction, transformation, and loading of Bluebikes data from AWS S3 to GCP. With a structured DAG and task-oriented workflow, the pipeline can handle monthly data updates reliably and consistently. This document outlines the key steps, configurations, and improvements, providing a complete view of the pipeline’s capabilities and future potential.