Individual Project Report - Rishabh Singh - G26592249

Project Overview

The project focused on leveraging big data and analytics tools to gather, store, and visualize extensive baseball game data. Specifically, the data encompassed Major League Baseball (MLB) games from 1901 to 2023. My role in this project was multifaceted, involving setting up a data pipeline, data storage, and data visualization.

Setting up Airflow and DAGs

- **Task:** Establish an Apache Airflow environment and create Directed Acyclic Graphs (DAGs).
- **Objective:** Automate the data extraction process from the MLB schedule API and handle subsequent data processing tasks.
- **Execution:** I successfully set up an Airflow environment, leveraging its scheduling and workflow management capabilities. Two DAGs were written:
    - **DAG 1:** Extracted data from the MLB schedule API. This process involved fetching game data spanning over a century (1901-2023).
    - **DAG 2:** Called the game_pk API using the game IDs (game_pk) extracted by DAG 1.

Data Storage in S3

- **Task:** Store the retrieved data in an AWS S3 bucket.
- **Data Volume:** The initial data from the schedule API totaled approximately 8 GB. The data from the game_pk API was substantially larger, around 175 GB.
- **Structure and Format:**
    - The schedule API data was stored along with a CSV file containing all game IDs.
    - The game_pk data was organized in a structured folder hierarchy: year/month/day/game_pk.json.
- **Data Lake:** Both sets of data were integrated into our S3-based data lake, providing a centralized and scalable storage solution.

Data Visualization in Google Colab

- **Task:** Develop data visualizations to represent the cleaned and processed MLB data.
- **Tools Used:** Google Colab, leveraging Python libraries for data analysis and visualization.
- **Process:** Utilized the cleaned data prepared by my teammate. This data was extracted from our S3 data lake and then analyzed and visualized in Google Colab. The visualizations focused on

providing insights into various aspects of the MLB data, such as game trends, player performance, and historical comparisons.

Challenges Faced and Overcome

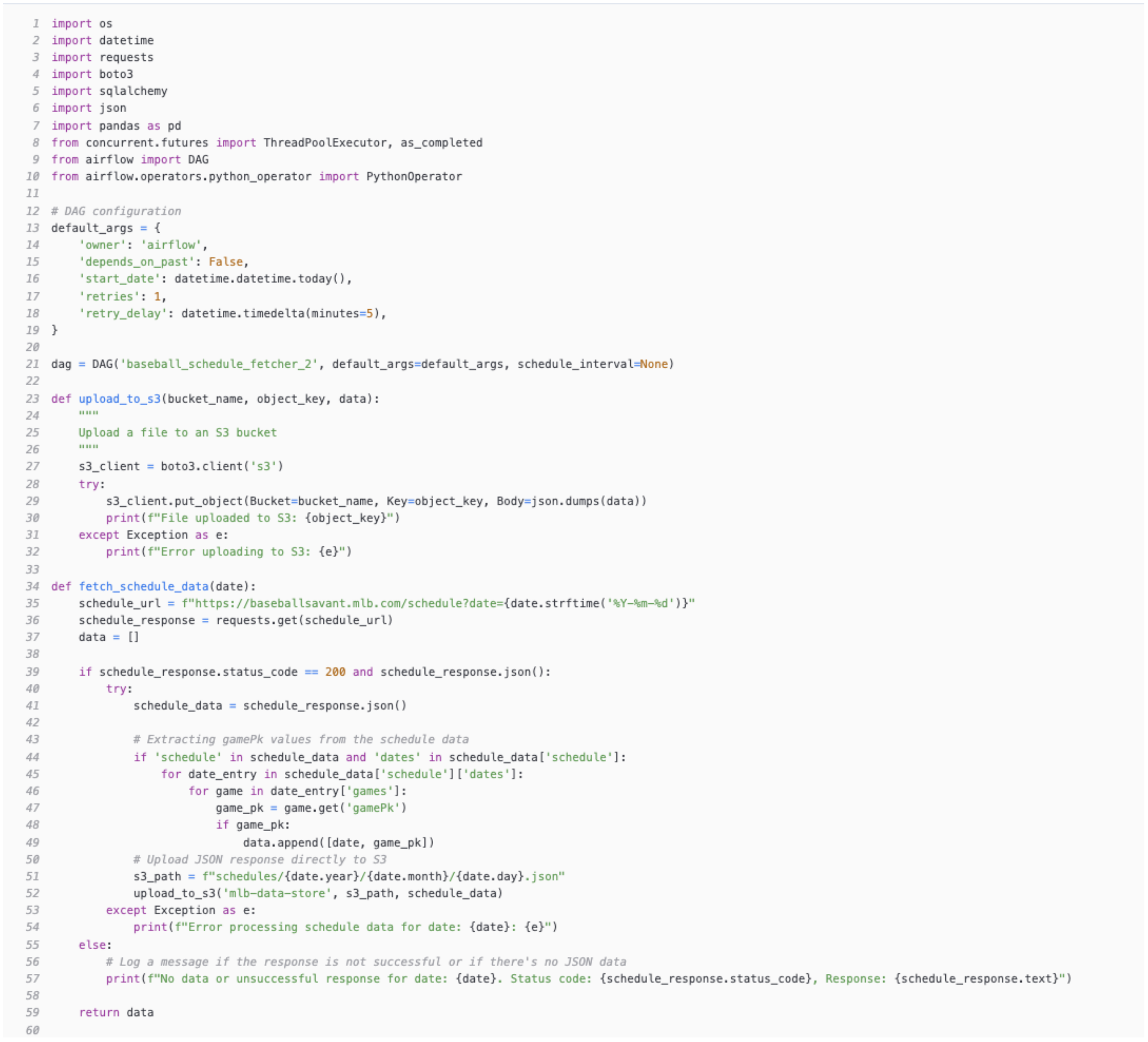
- **Data Volume Management:** Handling large volumes of data (especially the 175 GB from the game_pk API) was challenging. Efficient data processing and storage strategies were crucial.
- **Complex Data Extraction:** Extracting and structuring data from APIs spanning over a century required meticulous planning and execution.
- **Visualization of Complex Data Sets:** Converting large and complex data sets into comprehensible visualizations demanded a deep understanding of both the data and the visualization tools.

Conclusion

This project not only demonstrated my ability to handle large-scale data using modern big data tools but also emphasized the importance of clear and insightful data visualization. Working with historical MLB data presented unique challenges and learning opportunities, especially in terms of data management and visualization techniques.

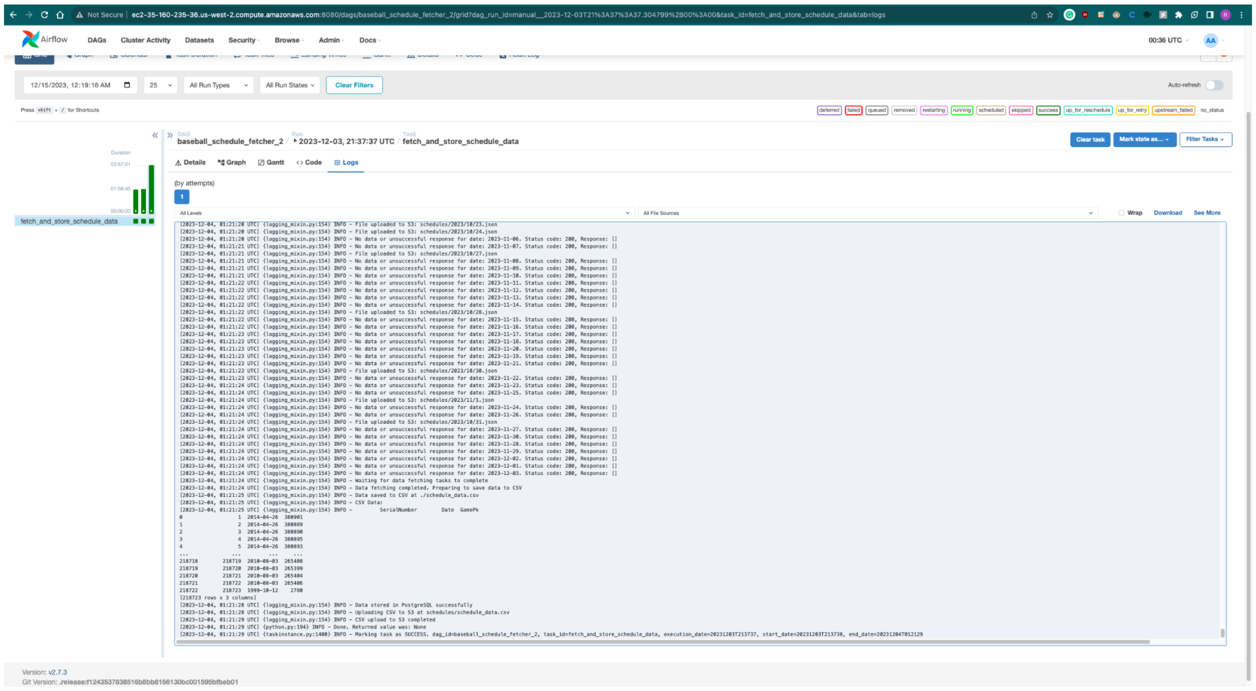
Screenshots Attached

DAG 1



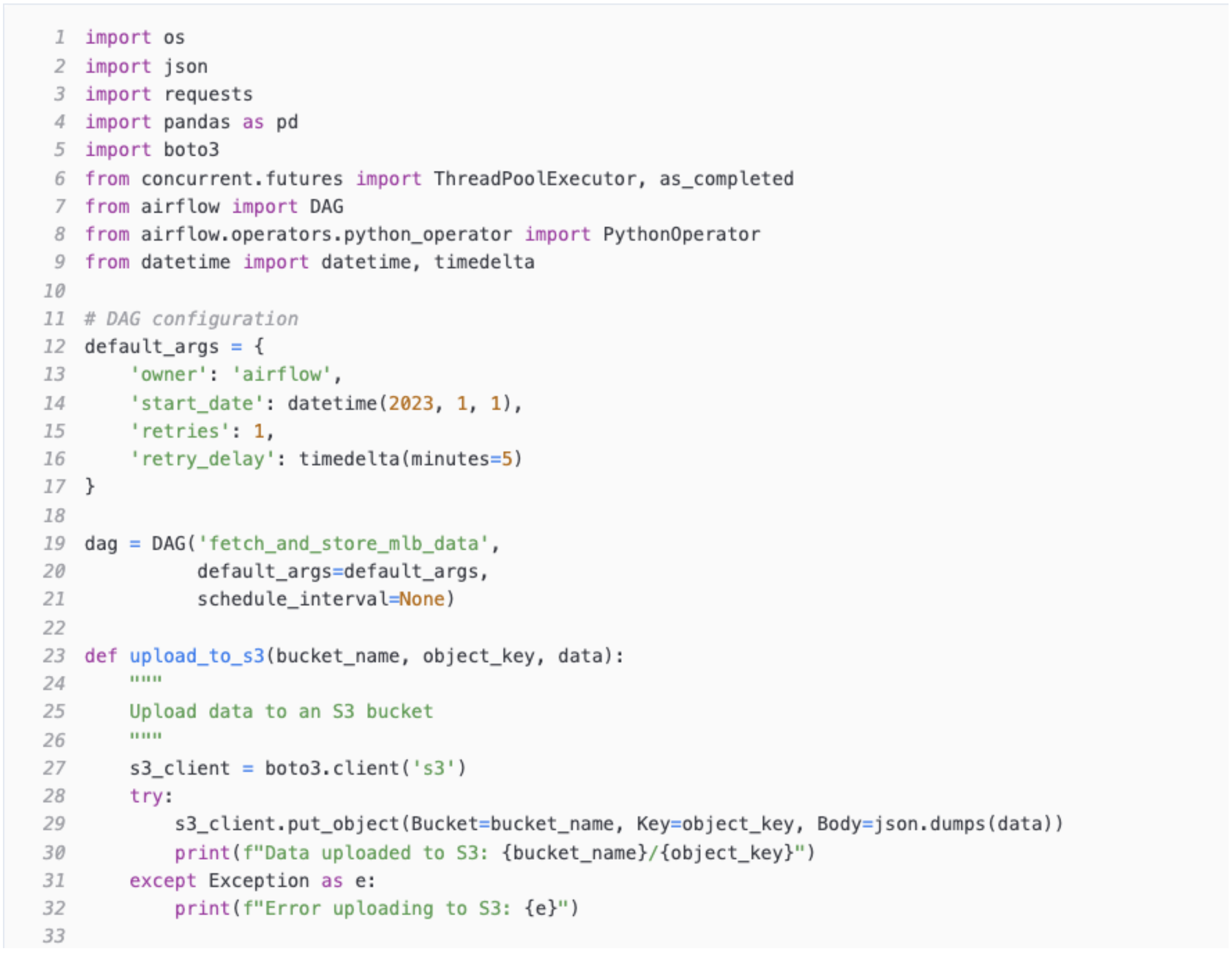
```python
import os
import datetime
import requests
import boto3
import sqlalchemy
import json
import pandas as pd
from concurrent.futures import ThreadPoolExecutor, as_completed
from airflow import DAG
from airflow.operators.python_operator import PythonOperator

# DAG configuration
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime.datetime.today(),
    'retries': 1,
    'retry_delay': datetime.timedelta(minutes=5),
}

dag = DAG('baseball_schedule_fetcher_2', default_args=default_args, schedule_interval=None)

def upload_to_s3(bucket_name, object_key, data):
    """
    Upload a file to an S3 bucket
    """
    s3_client = boto3.client('s3')
    try:
        s3_client.put_object(Bucket=bucket_name, Key=object_key, Body=json.dumps(data))
        print(f"File uploaded to S3: {object_key}")
    except Exception as e:
        print(f"Error uploading to S3: {e}")

def fetch_schedule_data(date):
    schedule_url = f"https://baseballsavant.mlb.com/schedule?date={date.strftime('%Y-%m-%d')}"
    schedule_response = requests.get(schedule_url)
    data = []

    if schedule_response.status_code == 200 and schedule_response.json():
        try:
            schedule_data = schedule_response.json()

            # Extracting gamePk values from the schedule data
            if 'schedule' in schedule_data and 'dates' in schedule_data['schedule']:
                for date_entry in schedule_data['schedule']['dates']:
                    for game in date_entry['games']:
                        game_pk = game.get('gamePk')
                        if game_pk:
                            data.append([date, game_pk])
            # Upload JSON response directly to S3
            s3_path = f"schedules/{date.year}/{date.month}/{date.day}.json"
            upload_to_s3('mlb-data-store', s3_path, schedule_data)
        except Exception as e:
            print(f"Error processing schedule data for date: {date}: {e}")
    else:
        # Log a message if the response is not successful or if there's no JSON data
        print(f"No data or unsuccessful response for date: {date}. Status code: {schedule_response.status_code}, Response: {schedule_response.text}")

    return data

```

```python
def fetch_and_store_data(ds, **kwargs):
    print("Starting data fetch and store process")

    start_date = datetime.date(1901, 1, 1)
    end_date = datetime.date.today()
    print(f"Fetching data from {start_date} to {end_date}")

    dates = [start_date + datetime.timedelta(days=i) for i in range((end_date - start_date).days + 1)]

    futures = []
    with ThreadPoolExecutor(max_workers=20) as executor:
        for date in dates:
            futures.append(executor.submit(fetch_schedule_data, date))
            print(f"Scheduled data fetch for date: {date}")

    print("Waiting for data fetching tasks to complete")
    results = []
    for future in as_completed(futures):
        results.extend(future.result())

    print("Data fetching completed. Preparing to save data to CSV")
    # Convert results to DataFrame and save as CSV
    df = pd.DataFrame(results, columns=['Date', 'GamePk'])
    df.insert(0, 'SerialNumber', range(1, 1 + len(df)))
    csv_file_path = './schedule_data.csv'
    df.to_csv(csv_file_path, index=False)
    print(f"Data saved to CSV at {csv_file_path}")

    # Read and print the CSV file
    df = pd.read_csv(csv_file_path)
    print("CSV Data:")
    print(df)

    # Store data in PostgreSQL
    # Replace with your PostgreSQL connection details
    postgres_connection = 'postgresql+psycopg2://airflow:airflow@localhost/airflow'
    engine = sqlalchemy.create_engine(postgres_connection)
    try:
        df.to_sql('schedule_data', engine, index=False, if_exists='replace')
        print("Data stored in PostgreSQL successfully")
    except Exception as e:
        print(f"Error storing data in PostgreSQL: {e}")

    # Upload the CSV file to S3
    s3_csv_path = f"schedules/schedule_data.csv"
    print(f"Uploading CSV to S3 at {s3_csv_path}")
    s3_client = boto3.client('s3')
    s3_client.upload_file(csv_file_path, 'mlb-data-store', s3_csv_path)
    print("CSV upload to S3 completed")

# Task definition
fetch_store_task = PythonOperator(
    task_id='fetch_and_store_schedule_data',
    provide_context=True,
    python_callable=fetch_and_store_data,
    dag=dag,
)

fetch_store_task

```



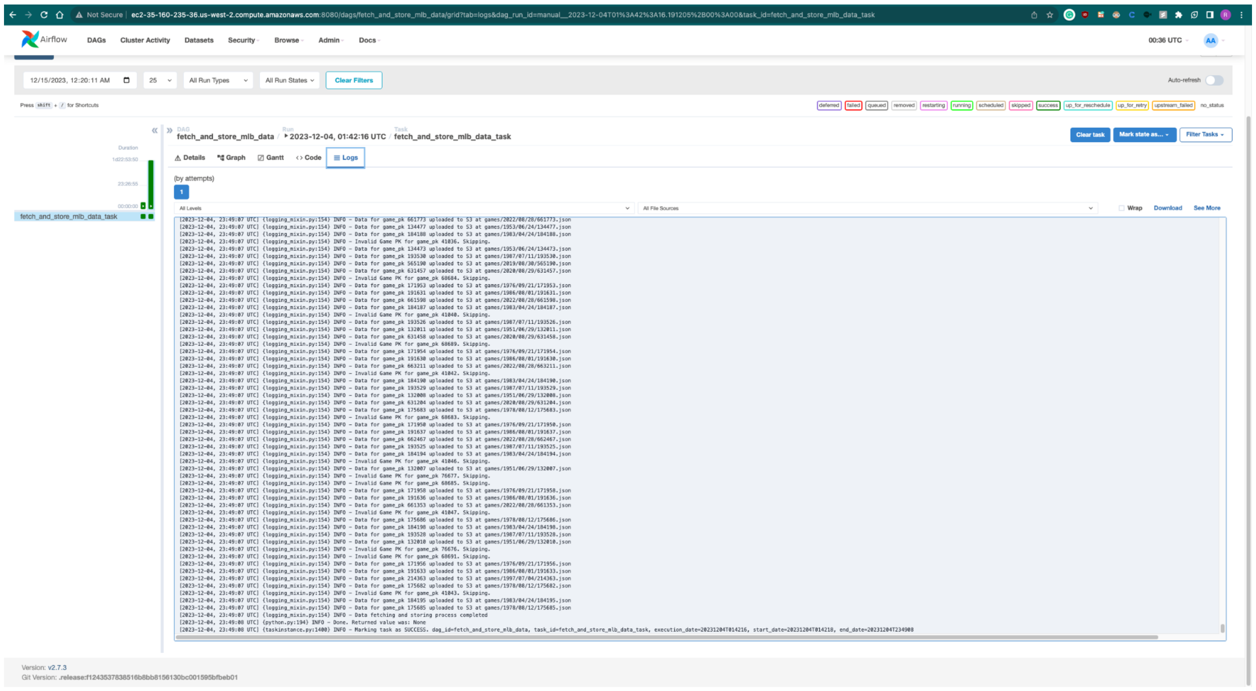
```
[2023-12-04, 01:21:20 UTC] {logging_mixin.py:154} INFO - File uploaded to S3: schedules/2023/10/23.json
[2023-12-04, 01:21:20 UTC] {logging_mixin.py:154} INFO - File uploaded to S3: schedules/2023/10/24.json
[2023-12-04, 01:21:20 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-06. Status code: 200, Response: []
[2023-12-04, 01:21:21 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-07. Status code: 200, Response: []
[2023-12-04, 01:21:21 UTC] {logging_mixin.py:154} INFO - File uploaded to S3: schedules/2023/10/27.json
[2023-12-04, 01:21:21 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-08. Status code: 200, Response: []
[2023-12-04, 01:21:21 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-09. Status code: 200, Response: []
[2023-12-04, 01:21:21 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-10. Status code: 200, Response: []
[2023-12-04, 01:21:22 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-11. Status code: 200, Response: []
[2023-12-04, 01:21:22 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-12. Status code: 200, Response: []
[2023-12-04, 01:21:22 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-13. Status code: 200, Response: []
[2023-12-04, 01:21:22 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-14. Status code: 200, Response: []
[2023-12-04, 01:21:22 UTC] {logging_mixin.py:154} INFO - File uploaded to S3: schedules/2023/10/28.json
[2023-12-04, 01:21:22 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-15. Status code: 200, Response: []
[2023-12-04, 01:21:22 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-16. Status code: 200, Response: []
[2023-12-04, 01:21:23 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-17. Status code: 200, Response: []
[2023-12-04, 01:21:23 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-18. Status code: 200, Response: []
[2023-12-04, 01:21:23 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-20. Status code: 200, Response: []
[2023-12-04, 01:21:23 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-19. Status code: 200, Response: []
[2023-12-04, 01:21:23 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-21. Status code: 200, Response: []
[2023-12-04, 01:21:23 UTC] {logging_mixin.py:154} INFO - File uploaded to S3: schedules/2023/10/30.json
[2023-12-04, 01:21:23 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-22. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-23. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-25. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - File uploaded to S3: schedules/2023/11/1.json
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-24. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-26. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - File uploaded to S3: schedules/2023/10/31.json
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-27. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-30. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-28. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-11-29. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-12-02. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-12-01. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - No data or unsuccessful response for date: 2023-12-03. Status code: 200, Response: []
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - Waiting for data fetching tasks to complete
[2023-12-04, 01:21:24 UTC] {logging_mixin.py:154} INFO - Data fetching completed. Preparing to save data to CSV
[2023-12-04, 01:21:25 UTC] {logging_mixin.py:154} INFO - Data saved to CSV at ./schedule_data.csv
[2023-12-04, 01:21:25 UTC] {logging_mixin.py:154} INFO - CSV Data:
[2023-12-04, 01:21:25 UTC] {logging_mixin.py:154} INFO -         SerialNumber        Date  GamePk
0                  1  2014-04-26  380901
1                  2  2014-04-26  380889
2                  3  2014-04-26  380890
3                  4  2014-04-26  380895
4                  5  2014-04-26  380893
...              ...         ...     ...
218718        218719  2010-08-03  265408
218719        218720  2010-08-03  265399
218720        218721  2010-08-03  265404
218721        218722  2010-08-03  265406
218722        218723  1999-10-12    2780
[218723 rows x 3 columns]
[2023-12-04, 01:21:28 UTC] {logging_mixin.py:154} INFO - Data stored in PostgreSQL successfully
[2023-12-04, 01:21:28 UTC] {logging_mixin.py:154} INFO - Uploading CSV to S3 at schedules/schedule_data.csv
[2023-12-04, 01:21:29 UTC] {logging_mixin.py:154} INFO - CSV upload to S3 completed
[2023-12-04, 01:21:29 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-12-04, 01:21:29 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=baseball_schedule_fetcher_2, task_id=fetch_and_store_schedule_data, execution_date=20231203T213737, start_date=20231203T213738, end_date=20231204T012129
```

DAG 2



```python
import os
import json
import requests
import pandas as pd
import boto3
from concurrent.futures import ThreadPoolExecutor, as_completed
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta

# DAG configuration
default_args = {
    'owner': 'airflow',
    'start_date': datetime(2023, 1, 1),
    'retries': 1,
    'retry_delay': timedelta(minutes=5)
}

dag = DAG('fetch_and_store_mlb_data',
          default_args=default_args,
          schedule_interval=None)

def upload_to_s3(bucket_name, object_key, data):
    """
    Upload data to an S3 bucket
    """
    s3_client = boto3.client('s3')
    try:
        s3_client.put_object(Bucket=bucket_name, Key=object_key, Body=json.dumps(data))
        print(f"Data uploaded to S3: {bucket_name}/{object_key}")
    except Exception as e:
        print(f"Error uploading to S3: {e}")

```

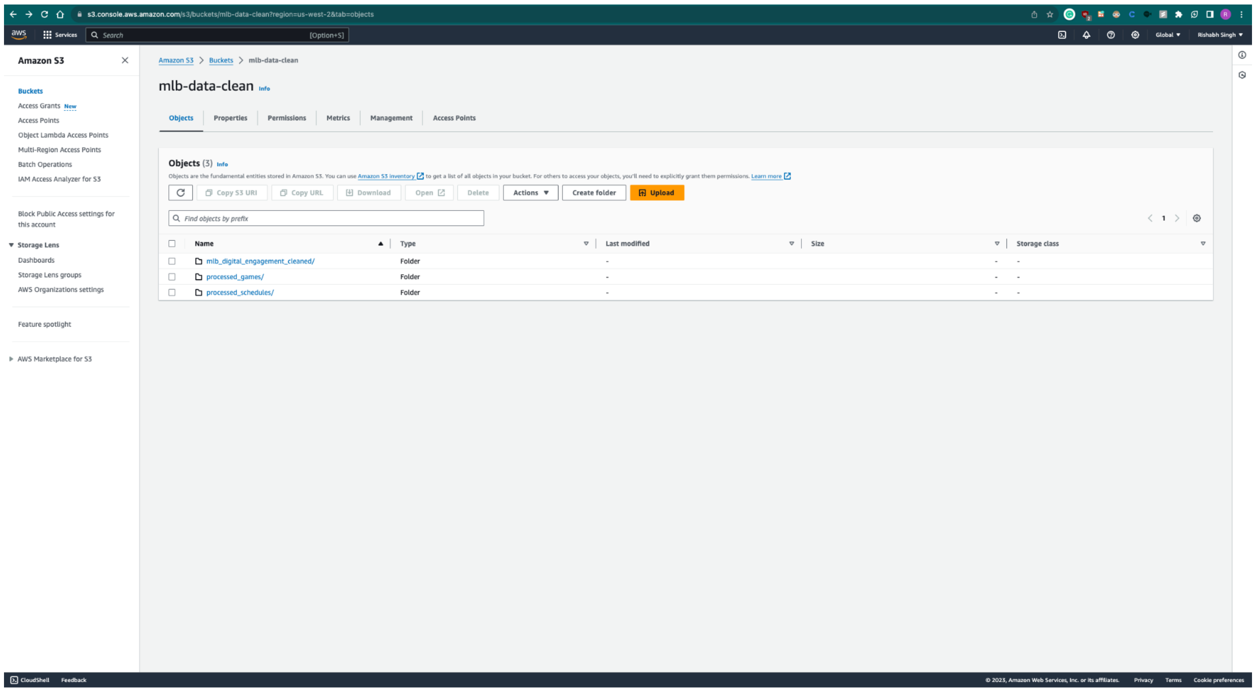
```python
def fetch_and_save_game_data(date, game_pk, bucket):
    print(f"Fetching data for game_pk {game_pk}...")
    api_url = f"https://baseballsavant.mlb.com/gf?game_pk={game_pk}"
    try:
        response = requests.get(api_url)
        if response.status_code == 200:
            data = response.json()
            if "error" in data and data["error"] == "Invalid Game PK.":
                return f"Invalid Game PK for game_pk {game_pk}. Skipping."

            object_key = f"games/{date.year}/{date.month:02d}/{date.day:02d}/{game_pk}.json"
            upload_to_s3(bucket, object_key, data)
            return f"Data for game_pk {game_pk} uploaded to S3 at {object_key}"
        else:
            return f"Failed to fetch data for game_pk {game_pk}: Status code {response.status_code}"
    except Exception as e:
        return f"Exception for game_pk {game_pk}: {e}"

def fetch_game_data_and_store():
    print("Loading DataFrame from CSV...")
    df = pd.read_csv('./schedule_data.csv')
    print(df.head())
    print("DataFrame loaded. Processing data.")
    bucket = 'mlb-data-store'  # Replace with your S3 bucket name

    with ThreadPoolExecutor(max_workers=20) as executor:
        futures = {executor.submit(fetch_and_save_game_data, pd.to_datetime(row['Date']), row['GamePk'], bucket): row for index, row in df.iterrows()}

    for future in as_completed(futures):
        result = future.result()
        print(result)

    print("Data fetching and storing process completed")

fetch_store_task = PythonOperator(
    task_id='fetch_and_store_mlb_data_task',
    python_callable=fetch_game_data_and_store,
    dag=dag,
)

fetch_store_task
```



```
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 661773 uploaded to S3 at games/2022/08/28/661773.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 134477 uploaded to S3 at games/1953/06/24/134477.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 184188 uploaded to S3 at games/1983/04/24/184188.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 41036. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 134473 uploaded to S3 at games/1953/06/24/134473.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 193530 uploaded to S3 at games/1987/07/11/193530.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 565190 uploaded to S3 at games/2019/08/30/565190.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 631457 uploaded to S3 at games/2020/08/29/631457.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 68684. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 171953 uploaded to S3 at games/1976/09/21/171953.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 191631 uploaded to S3 at games/1986/08/01/191631.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 661598 uploaded to S3 at games/2022/08/28/661598.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 184187 uploaded to S3 at games/1983/04/24/184187.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 41040. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 193526 uploaded to S3 at games/1987/07/11/193526.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 132011 uploaded to S3 at games/1951/06/29/132011.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 631458 uploaded to S3 at games/2020/08/29/631458.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 68689. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 171954 uploaded to S3 at games/1976/09/21/171954.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 191630 uploaded to S3 at games/1986/08/01/191630.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 663211 uploaded to S3 at games/2022/08/28/663211.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 41042. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 184190 uploaded to S3 at games/1983/04/24/184190.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 193529 uploaded to S3 at games/1987/07/11/193529.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 132008 uploaded to S3 at games/1951/06/29/132008.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 631204 uploaded to S3 at games/2020/08/29/631204.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 175683 uploaded to S3 at games/1978/08/12/175683.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 68683. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 171950 uploaded to S3 at games/1976/09/21/171950.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 191637 uploaded to S3 at games/1986/08/01/191637.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 662467 uploaded to S3 at games/2022/08/28/662467.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 193525 uploaded to S3 at games/1987/07/11/193525.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 184194 uploaded to S3 at games/1983/04/24/184194.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 41046. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 132007 uploaded to S3 at games/1951/06/29/132007.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 76677. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 68685. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 171958 uploaded to S3 at games/1976/09/21/171958.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 191636 uploaded to S3 at games/1986/08/01/191636.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 661353 uploaded to S3 at games/2022/08/28/661353.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 41047. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 175686 uploaded to S3 at games/1978/08/12/175686.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 184198 uploaded to S3 at games/1983/04/24/184198.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 193528 uploaded to S3 at games/1987/07/11/193528.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 132010 uploaded to S3 at games/1951/06/29/132010.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 76676. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 68691. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 171956 uploaded to S3 at games/1976/09/21/171956.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 191633 uploaded to S3 at games/1986/08/01/191633.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 214363 uploaded to S3 at games/1997/07/04/214363.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 175682 uploaded to S3 at games/1978/08/12/175682.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Invalid Game PK for game_pk 41043. Skipping.
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 184195 uploaded to S3 at games/1983/04/24/184195.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data for game_pk 175685 uploaded to S3 at games/1978/08/12/175685.json
[2023-12-04, 23:49:07 UTC] {logging_mixin.py:154} INFO - Data fetching and storing process completed
[2023-12-04, 23:49:08 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-12-04, 23:49:08 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=fetch_and_store_mlb_data, task_id=fetch_and_store_mlb_data_task, execution_date=20231204T014216, start_date=20231204T014218, end_date=20231204T234908
```

Version: v2.7.3
Git Version: .release:f243537838516b8bb8156130bc001595bfbeb01

S3



s3.console.aws.amazon.com/s3/buckets/mlb-data-clean?region=us-west-2&tab=objects

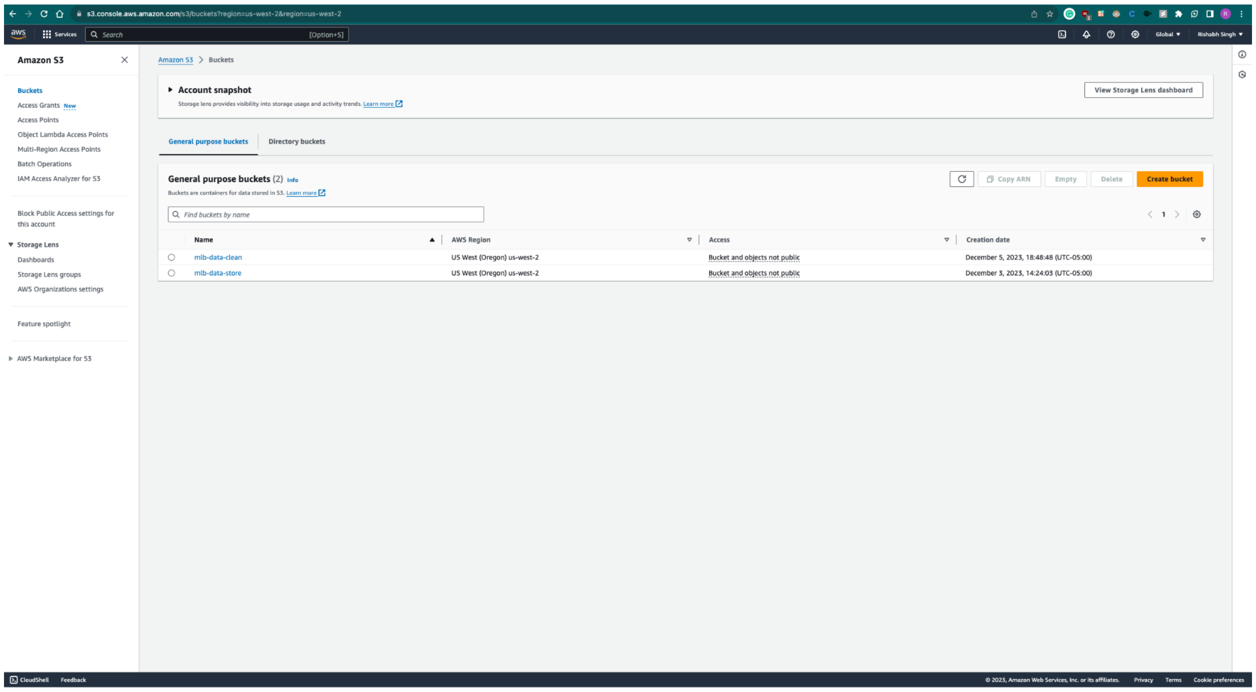
Amazon S3 > Buckets > mlb-data-clean

mlb-data-clean Info

Objects (3) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
mlb_digital_engagement_cleaned/	Folder	-	-	-
processed_games/	Folder	-	-	-
processed_schedules/	Folder	-	-	-



s3.console.aws.amazon.com/s3/buckets?region=us-west-2&region=us-west-2

Amazon S3 > Buckets

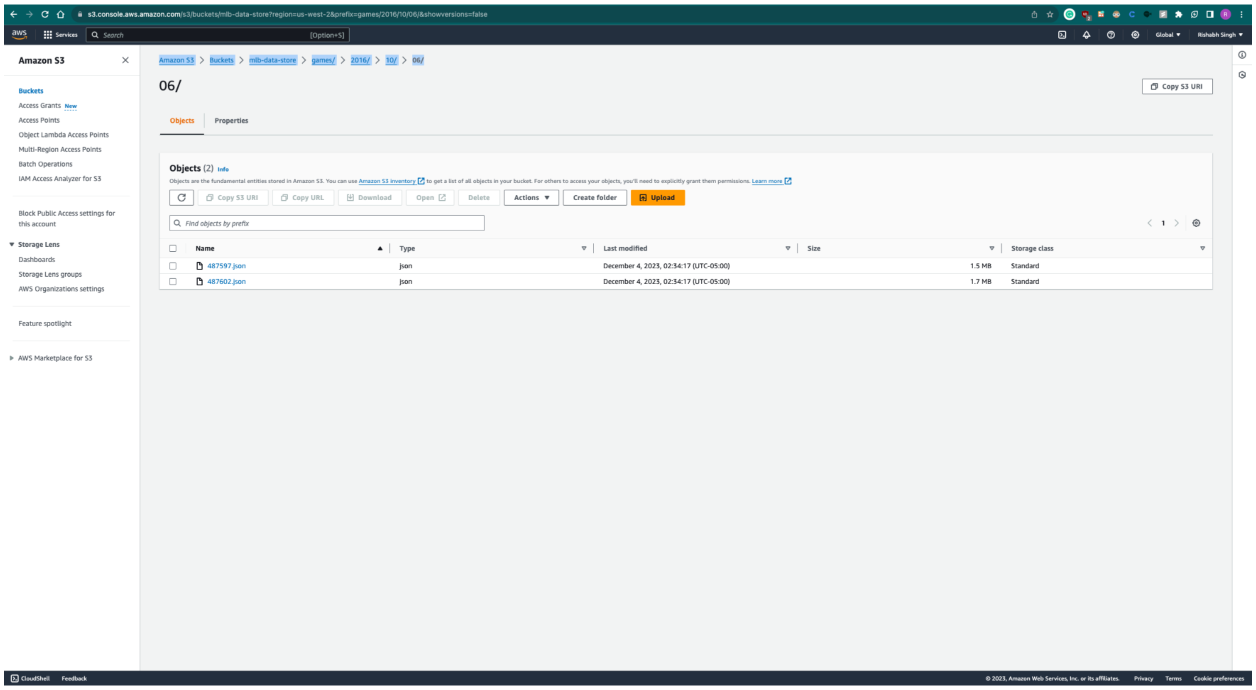
Account snapshot

Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (2) Info

Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
mlb-data-clean	US West (Oregon) us-west-2	Bucket and objects not public	December 5, 2023, 18:48:48 (UTC-05:00)
mlb-data-store	US West (Oregon) us-west-2	Bucket and objects not public	December 3, 2023, 14:24:03 (UTC-05:00)



s3.console.aws.amazon.com/s3/buckets/mlb-data-store?region=us-west-2&prefix=games/2016/10/06/&showversions=false

Amazon S3 > Buckets > mlb-data-store > games/ > 2016/ > 10/ > 06/

06/

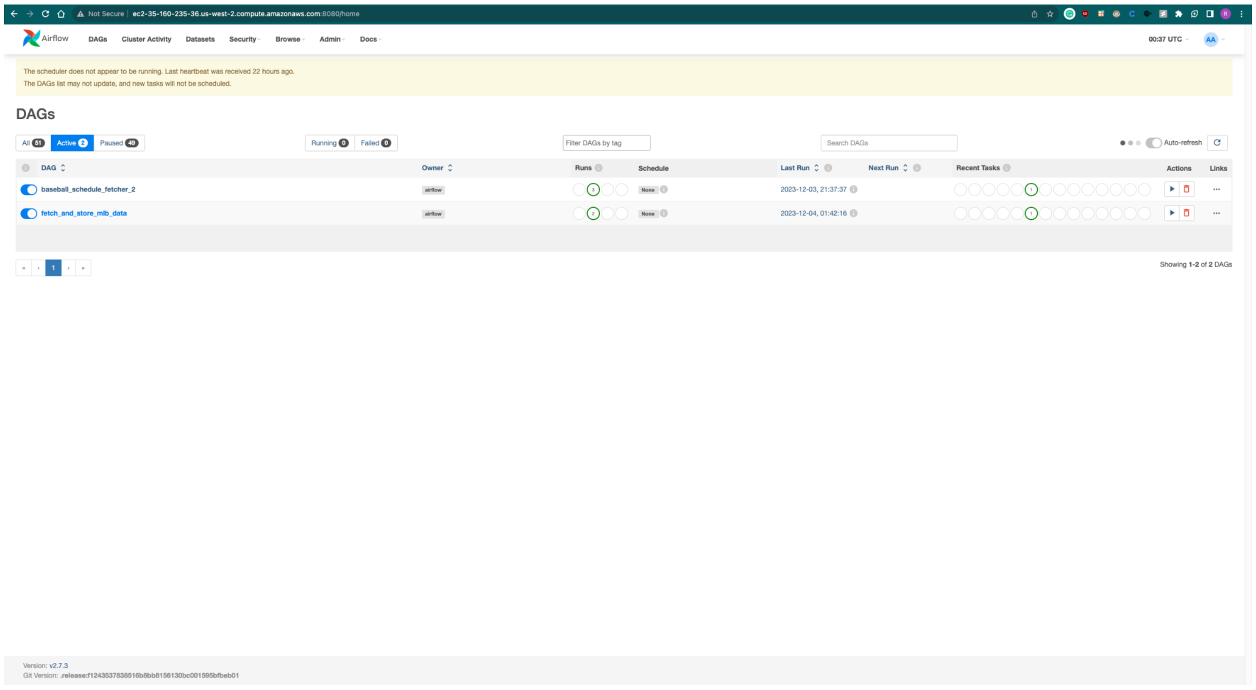
Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
487597.json	json	December 4, 2023, 02:34:17 (UTC-05:00)	1.5 MB	Standard
487602.json	json	December 4, 2023, 02:34:17 (UTC-05:00)	1.7 MB	Standard

© 2023, Amazon Web Services, Inc. or its affiliates.

Airflow



ec2-35-160-235-36.us-west-2.compute.amazonaws.com:8080/home

The scheduler does not appear to be running. Last heartbeat was received 22 hours ago.
The DAGs list may not update, and new tasks will not be scheduled.

DAGs

All 51 | Active 2 | Paused 49 | Running 0 | Failed 0

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
baseball_schedule_fetcher_2	airflow	3	None	2023-12-03, 21:37:37		1
fetch_and_store_mlb_data	airflow	2	None	2023-12-04, 01:42:16		1

Showing 1-2 of 2 DAGs

Version: v2.7.3
Git Version: .release:f1243537838516b8bb8156130bc001595bfbeb01