### **1. Overview**

This project implements a **near real-time attribution pipeline** on **Google Cloud Platform (GCP)** using **Cloud Composer (Airflow)**, **BigQuery**. The pipeline simulates streaming GA4-style event data, processes it through structured transformations, and computes **First-Click** and **Last-Click** attribution metrics. A **layered data architecture**—comprising raw, staging, and final tables—ensures data quality and scalability. The **final attribution table** is connected to **Power BI** for real-time visualization and channel performance analysis.

### **2. Table Architecture**

Three BigQuery tables were created to support a standard ELT (Extract, Load, Transform) flow:

|  | | **Layer** | **Table Name** | | --- | --- | | Raw | endless-upgrade-475014-b1.g4a\_events.event\_raw | | Staging | endless-upgrade-475014-b1.g4a\_events.event\_stg | | Final | endless-upgrade-475014-b1.g4a\_events.user\_attribution | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

### **Purpose of Each Layer**

* **Raw Layer:** Captures incoming data directly from the ingestion process without modification. This acts as a source of truth for replay or auditing.
* **Staging Layer:** Removes the duplicate data from the raw layer and insert only the new records only.This layer prepares the data for aggregation and modeling.
* **Final Layer (Attribution):** Stores aggregated user-level metrics including **first-click** and **last-click** . This table is the primary source for reporting and visualization.

### 

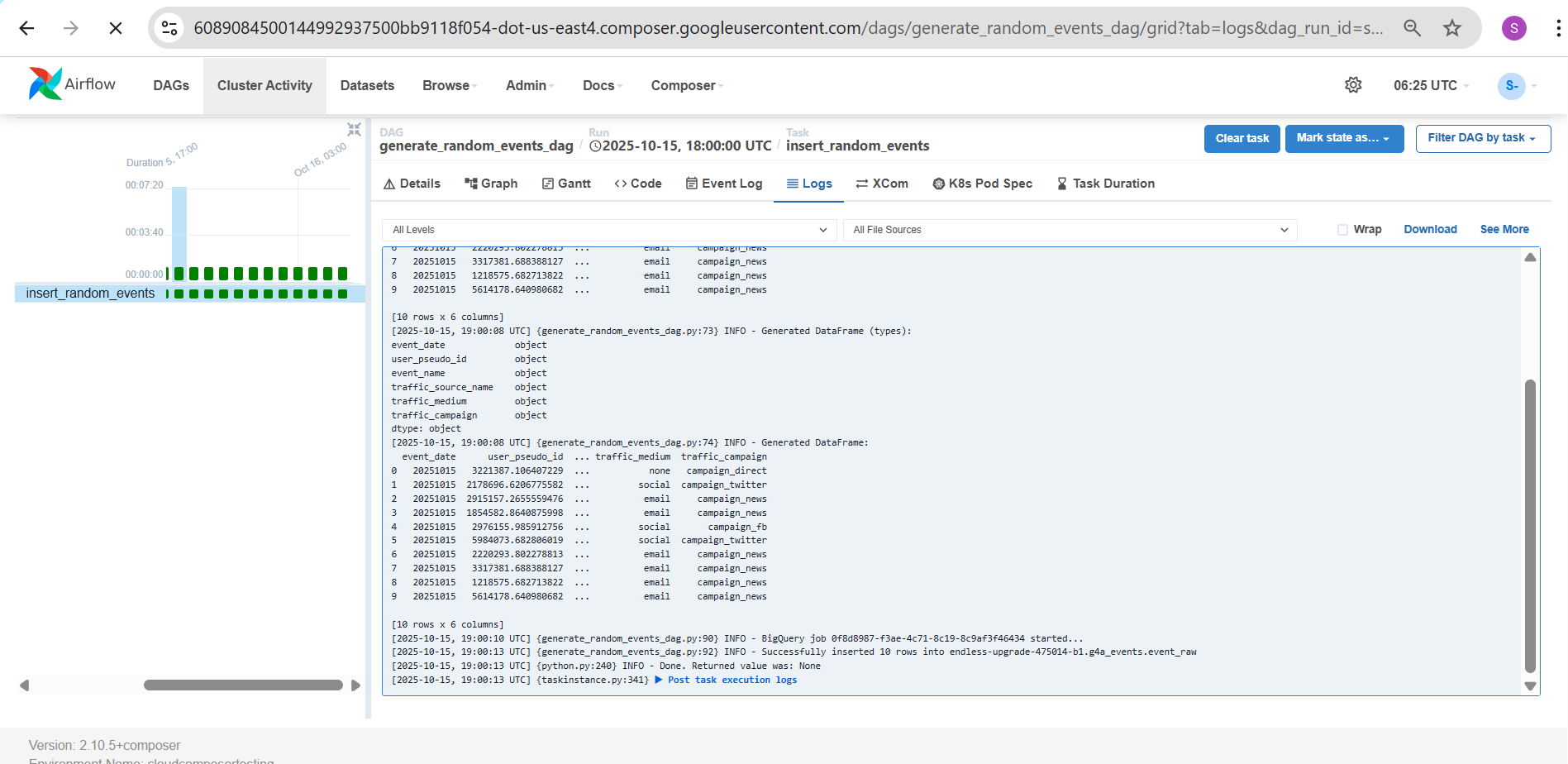
### 

### **3. Data Simulation via DAG**

To emulate a streaming data source, a custom DAG has been developed using Apache Airflow. The DAG generates and inserts synthetic data into the **raw table** using randomly selected values for predefined fields from the public dataset.

* **Reference Dataset**:  
   The logic and schema for the generated data are inspired by the public dataset:  
   **bigquery-public-data.ga4\_obfuscated\_sample\_ecommerce.events\_\***
* **Record Insertion**:  
   On each DAG run, **10 synthetic records** are inserted into the **event\_raw** table.
* **Scheduling**:  
   Due to free-tier resource limitations, the DAG is scheduled to run **once every hour**.

**Generate\_random\_events\_dag.py**  - data streaming



## 

### **4. Data Processing Logic**

The transformation flow is orchestrated through an **Airflow DAG** that executes a BigQuery routine that performs data deduplication, staging, and user-level attribution enrichment.

* **Raw to Staging:**  
   The procedure loads data from event\_raw into event\_stg using a MERGE statement. Duplicate events are avoided by matching on event\_date, user\_pseudo\_id, event\_name, event\_time, and traffic source fields, ensuring only new and unique events are inserted into the staging table.
* **Attribution Flags:**

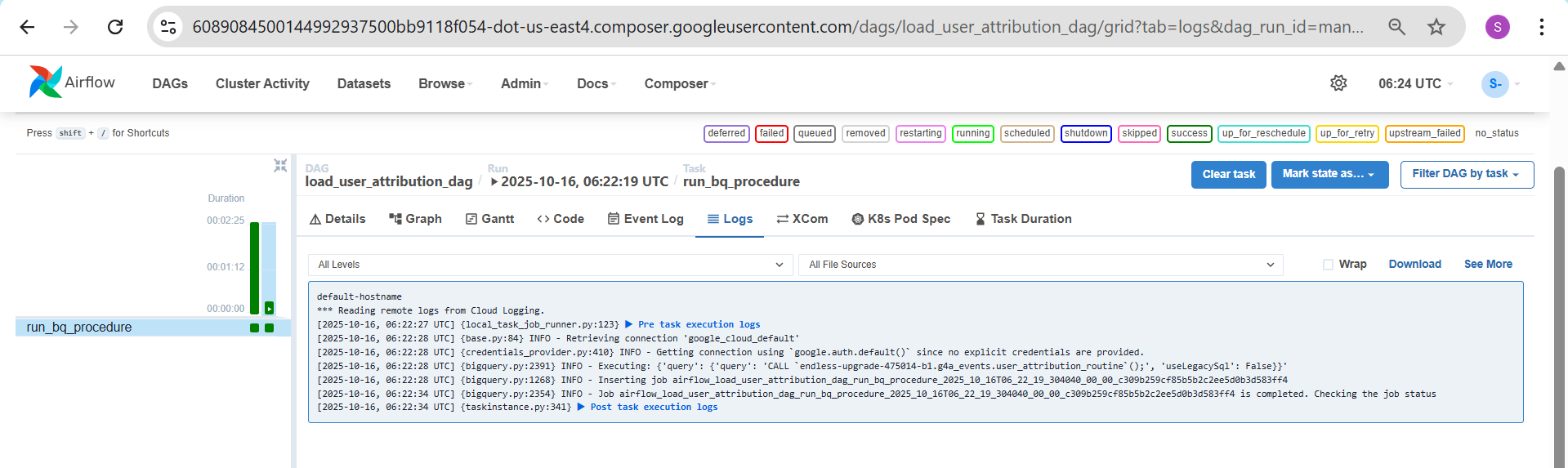
Within the same routine, first-click and last-click events are identified for each user\_pseudo\_id.

1. first\_click\_flag is set to TRUE for the event with the earliest first\_visit timestamp.
2. last\_click\_flag is set to TRUE for the event with the latest timestamp.

All other events for the user have the flags set to FALSE.  
**load\_user\_attribution\_dag.py**  - data transformation

* **Staging to Final Table:**  
   The enriched dataset, including event details and attribution flags, is inserted into user\_attribution. This ensures accurate first/last click attribution while preserving all event-level information.
* **Airflow Integration:** The DAG triggers this routine using the **BigQueryExecuteQueryOperator**, ensuring retry logic, logging, and scheduling within Cloud Composer.

This modular setup ensures data accuracy, easier maintenance, and scalability for near real-time attribution analytics.



### **5. Deployment Instructions**

1. Create a table and dataset using the provided sql file in the repository.

<https://github.com/sathyasankar23/Realtime_dashboard_public_dataset/blob/main/table_and_routine_creation.sql>

To deploy the DAG:

1. Locate the DAG file named:

**load\_user\_attribution\_dag.py**  - data transformation

**Generate\_random\_events\_dag.py**  - data streaming

<https://github.com/sathyasankar23/Realtime_dashboard_public_dataset/blob/main/DAGS.zip>

1. Upload the DAG from the ZIP file in the github to your Cloud Composer environment’s DAGs bucket.
2. Cloud Composer will automatically detect and schedule the DAG based on the defined interval.

### **6. Dashboard**

Due to access limitations with Looker, I utilized Power BI to create the dashboard for visualizing processed data.

* Data Source:  
   The dashboard pulls data directly from the final table as a exported file.  
   **Endless-upgrade-475014-b1.g4a\_events.user\_attribution**

[**https://github.com/sathyasankar23/Realtime\_dashboard\_public\_dataset/blob/main/sample\_data\_user\_attribution.csv**](https://github.com/sathyasankar23/Realtime_dashboard_public_dataset/blob/main/sample_data_user_attribution.csv)

* **Visualizations:**
  + Key user engagement metrics
  + Attribution breakdown by source, medium, and campaign
  + Time-based trends and event distributions

Power BI proved effective for interactive exploration and quick visualization of insights derived from the transformed data.

