**Project - 5**

**Real-Time Data Processing Pipeline on Google Cloud**

**Objective**

To build a real-time data processing pipeline using Google Cloud services that can ingest, process, analyze, and archive streaming data. This solution enables low-latency insights and scalable data ingestion for real-time use cases.

**🌐 Architecture Overview**

The architecture is designed to simulate and ingest real-time streaming data using Cloud Pub/Sub, process it using Dataflow (Apache Beam), store analytical output in BigQuery, and archive raw data in Cloud Storage for backup or future batch processing.

**🧩 Services Used**

* **Cloud Pub/Sub** – Ingest simulated real-time messages
* **Dataflow (Apache Beam)** – Stream processing (ETL) in real time
* **BigQuery** – Storage and querying of processed data
* **Cloud Storage** – Backup and archival of raw data for audit or recovery

**🛠️ Step-by-Step Implementation Tasks**

**Part 1: Create and Configure Services**

* Create a **Pub/Sub topic** (stream-topic) to receive streaming messages.
* Create a **Cloud Storage bucket** for storing raw data backups.
* Create a **BigQuery dataset and table** with schema for processed output.

**Part 2: Simulate Streaming Data**

* Use a Python script to publish JSON messages to the stream-topic every few seconds.
  + Example fields: **user\_id**, **action**, **timestamp**

from google.cloud import pubsub\_v1

import json, time, random

publisher = pubsub\_v1.PublisherClient()

topic\_path = publisher.topic\_path("your-project-id", "stream-topic")

while True:

data = {

"user\_id": f"user\_{random.randint(1,100)}",

"action": random.choice(["click", "purchase", "view"]),

"timestamp": time.strftime("%Y-%m-%dT%H:%M:%SZ")

}

publisher.publish(topic\_path, json.dumps(data).encode("utf-8"))

time.sleep(2)

**Part 3: Stream Processing with Dataflow (Apache Beam)**

* Develop a Dataflow pipeline (Python or Java) using Apache Beam SDK:
  + Read messages from Pub/Sub
  + Parse JSON and apply transformations
  + Write processed output to BigQuery
  + Archive raw messages to Cloud Storage

# Example pseudocode

p | ReadFromPubSub()

| ParseJSON()

| WriteToBigQuery()

| WriteToCloudStorage()

* Use PipelineOptions to specify project, region, temp location, etc.
* Deploy pipeline via command-line or Dataflow UI.

**Part 4: Monitoring and Observability**

* Use **Dataflow job monitoring** to view performance, throughput, and latency.
* Verify:
  + Processed data is arriving in **BigQuery**.
  + Raw messages are being written to **Cloud Storage**.
* Set up **logging** and use **Stackdriver Error Reporting** for diagnostics.

**Part 5: Optional – Scheduling and Resilience**

* Use **Dataflow Flex Templates** for reusability and automation.
* Configure retry policies and dead-letter topics for message failures.

**📄 Expected Deliverables**

* Python Publisher Script (publisher.py)
* Apache Beam Dataflow pipeline script (dataflow\_pipeline.py)
* BigQuery dataset and schema documentation
* Cloud Storage backup path (URI)
* Architecture diagram (PDF or PNG)
* Sample messages and BigQuery output (screenshots or CSV export)

**📘 Bonus / Optional Extensions**

* Add schema validation and alerting for malformed messages
* Use Cloud Composer (Airflow) to orchestrate the pipeline
* Integrate with Data Studio for live dashboards
* Add windowing and aggregation (e.g., count actions every minute)