**Project Objectives**

This project aims to design an end-to-end **data engineering pipeline** on **Google Cloud Platform (GCP)** for processing **Avito Context Ad Clicks** datasets. The focus will be on using GCP services for **data ingestion**, **transformation**, and **storage** in a structured format, following the **Medallion Architecture** (Bronze, Silver, Gold). The final data will be visualized using **Google Data Studio** or **Looker** to derive insights.

**Tools & Tech Stacks to Involve:**

* **Python**
* **SQL**
* **Cloud Composer** (GCP's managed Airflow)
* **Google Cloud Functions**
* **Google Dataproc** (Apache Spark, PySpark)
* **Google BigQuery**
* **Google Cloud Storage (GCS)** for data lake
* **Terraform** (for infrastructure as code)
* **Bitbucket CI** (for Continuous Integration and Delivery)
* **Compute Engine (VM)**
* **Pub/Sub** (for event-driven architecture)
* **Cloud Monitoring & Logging**
* **Google Dataflow** (for streaming ETL)
* **Google IAM** (for access control and security)
* **Looker Studio / Google Data Studio** (for dashboards)

**Project Tasks**

1. **Set Up Infrastructure Using Terraform**:
   * Use **Terraform** to create GCP resources such as **GCS buckets**, **Dataproc clusters**, **BigQuery datasets**, and **IAM roles**.
2. **Ingest Raw Data into GCP**:
   * Sign up for **Google Cloud Free Tier** and create a **Compute Engine VM**.
   * Upload the raw datasets into the VM.
3. **Create Cloud Composer (Airflow) DAG**:
   * Set up **Cloud Composer** to run an **Airflow DAG** that orchestrates the pipeline from **VM** to **BigQuery** or **Cloud SQL** (MySQL/Postgres).
   * Use **Python operators** to automate the ingestion of raw files into **Cloud SQL**.
4. **Simulate Data Insertion via Python Script**:
   * Develop a **Python script** that continuously inserts simulated records (ad clicks, searches) into **Cloud SQL**. (Do this in bronze layer itself. Also simulate nested data)
5. **Event-Driven Data Movement Using Cloud Functions**:
   * Trigger **Google Cloud Functions** to capture new data inserts and move the data from **Cloud SQL** into **GCS** in **Parquet** or **Avro** format.
6. **ETL Pipeline Using Google Dataflow**:
   * Use **Google Dataflow** (or **Apache Beam**) to build a pipeline that extracts data from **Cloud SQL**, transforms it, and loads it into **GCS** for storage.
   * Define a **Dataflow job** to handle transformations and write the output to **GCS** as **Parquet** or **Avro**.
7. **Data Transformation with Google Dataproc and PySpark**:
   * Use **Google Dataproc** to run **PySpark** jobs that clean, transform, and enrich the data.
   * Perform transformations such as **joining tables**, **filtering data**, and **applying business logic**.
   * Output transformed data into the **Silver layer** in **GCS**.
8. **Aggregation in the Gold Layer**:
   * Define **aggregation use cases** (e.g., total ad clicks per category, region-wise performance) using **PySpark** on **Dataproc**.
   * Write aggregated data to the **Gold layer** in **GCS and also in CloudSQL**.
9. **Medallion Architecture in GCP**:
   * Implement a **Medallion Architecture** for the data lake in **GCS**: (Refer [this](https://saturam.atlassian.net/wiki/spaces/Training/pages/355565569) for samples)
     + **Bronze**: Raw data from **Cloud SQL** in its original form.
     + **Silver**: Transformed, cleaned data after applying joins and filtering.
     + **Gold**: Aggregated data for analytics and business reporting.
10. **Data Visualization with Google Data Studio / Looker**:
    * Use **Google Data Studio** or **Looker** to connect to the **Gold layer** in **BigQuery** or **GCS** for creating **dashboards** and visual reports.
    * Build dashboards that showcase **ad click metrics**, **CTR (Click-Through Rate)**, and **performance by category**.
11. **Monitoring and Logging via Cloud Monitoring**:
    * Set up **Cloud Monitoring** to track the pipeline’s health, log errors, and set up alerts for task failures.
    * Use **Cloud Logging** to monitor the logs from Cloud Functions, Dataflow, and Dataproc.
12. **CI/CD Pipelines with Bitbucket**:
    * Set up **Bitbucket CI** pipelines to automatically deploy infrastructure using **Terraform** and manage code updates to **Cloud Functions**, **Airflow DAGs**, and **PySpark scripts**.
    * Automate testing and deployment using the CI/CD pipeline to ensure the system is continuously maintained and updated.

**Notes**

***\*\* When you are not consuming cloud resources, kindly shutdown to not incur costs***