## **Shivani Sinha, IUBH**

### **Project Overview**

The aim of this project is to build a batch-processing-based data-intensive application using Google Cloud Platform (GCP) services. The application will ingest a large time-series dataset, perform batch processing, store the data securely, and deliver aggregated insights to a visualization dashboard. The application architecture emphasizes modularity, scalability, maintainability, and data governance.

### **Chosen Dataset**

* **Dataset Name**: [Kaggle - NYC Taxi Trip Duration](https://www.kaggle.com/competitions/nyc-taxi-trip-duration/data)
* **Justification**:  
  + Contains over 1.5 million rows (satisfies the volume requirement).
  + Includes time-referenced data (pickup datetime).
  + Real-world use case (transportation/carbon tracking possible).

### **Microservices and Responsibilities**

1. **Data Ingestion Microservice**
   * **Technology**: Cloud Functions (Python), triggered via Cloud Scheduler.
   * **Task**: Load CSV data from a public bucket or uploaded file into a GCS bucket.
   * **Justification**: Lightweight, event-driven, suitable for small data uploads in free tier.
2. **Data Storage Microservice**
   * **Technology**: Google Cloud Storage (raw), BigQuery (processed).
   * **Task**: Store raw uploaded CSV in GCS; load processed and aggregated data into BigQuery.
   * **Justification**: BigQuery supports massive datasets and SQL-based analytics with free-tier capabilities.
3. **Data Preprocessing and Aggregation Microservice**
   * **Technology**: Cloud Functions (or Cloud Run), written in Python.
   * **Task**: Clean data, filter for relevant columns, compute aggregates (e.g., average trip time, CO₂ emissions).
   * **Justification**: Cloud Functions offer stateless, on-demand processing and are cost-effective for batch jobs.
4. **Data Delivery Microservice**
   * **Technology**: Looker Studio connected to BigQuery.
   * **Task**: Display trends, summaries, and emissions estimates in an interactive dashboard.
   * **Justification**: Looker Studio is free, and native integration with BigQuery is seamless.

### **Reliability, Scalability, and Maintainability**

* **Reliability**: Cloud Functions auto-retry on failure; BigQuery guarantees durable storage.
* **Scalability**: GCP services (Cloud Functions, BigQuery) scale automatically with data volume.
* **Maintainability**: Modular components and version-controlled Cloud Function code using GitHub.

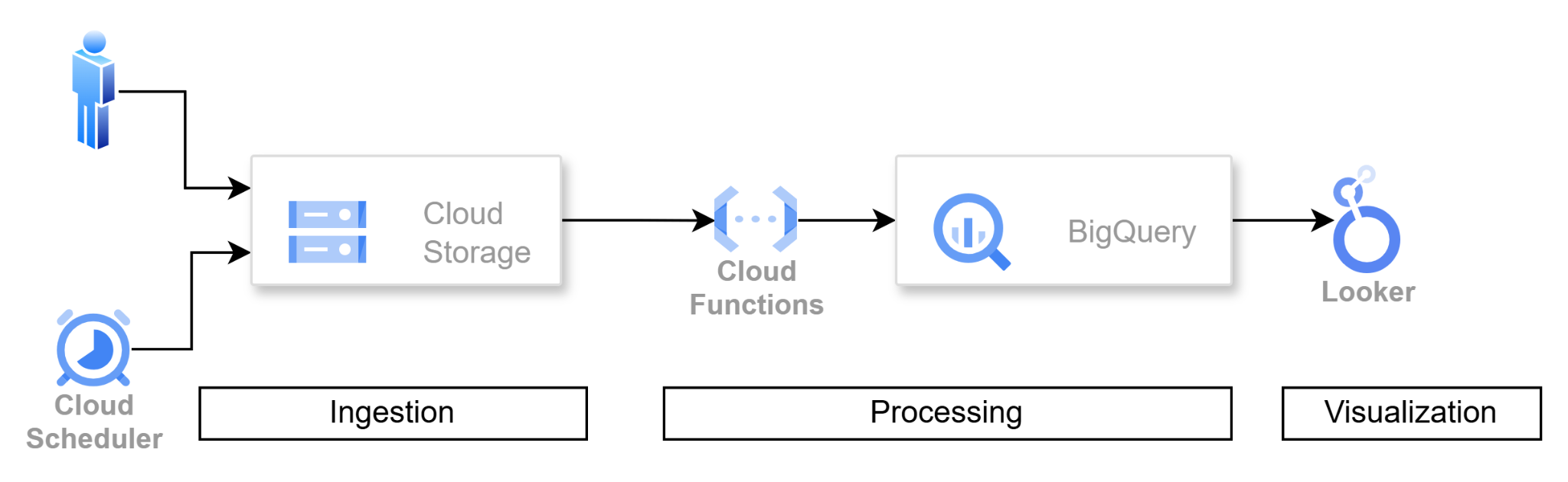
### **Security, Governance, and Protection**

* **IAM Roles**: Use least-privilege principles for access (Cloud Function, BigQuery, GCS permissions).
* **Data Encryption**: All data encrypted at rest and in transit by default in GCP.
* **Audit Logging**: Enable Cloud Audit Logs for traceability.
* **Governance**: Separate storage buckets for raw and processed data; use data retention policies in GCS.

### **Data Processing Frequency**

* **Ingestion**: Weekly or monthly via Cloud Scheduler.
* **Processing**: Immediately after ingestion using event-based trigger or daily via scheduler.
* **Aggregation**: On each run; processed data is appended or replaced in BigQuery.
* **Delivery**: Dashboard is updated automatically as new data flows into BigQuery.

### **Visual Architecture Description**



*Image created through* [*draw.io*](http://draw.io) *software*

### **Advantages of This Conceptual Architecture**

* **Cost-effective**: All services used are within the GCP free tier.
* **Reproducible**: Code and infrastructure managed via GitHub; can be deployed using Terraform or gcloud CLI (optional).
* **Modular**: Each component is independent and easy to maintain or upgrade.