# Project Report: Real-Time Fraud Detection Platform on GCP

Author: Sai Nihal Diddi
Date: August 7, 2025
Project Status: In Development / Conceptualized

## 1. Executive Summary

This project implements an end-to-end, real-time fraud detection platform built entirely on the Google Cloud Platform (GCP) and open-source technologies. The system is designed to ingest a high-throughput stream of financial transactions, apply a machine learning model to score each transaction for fraud in milliseconds, and raise immediate alerts. Its most unique feature is a **closed-loop MLOps pipeline** that uses human-in-the-loop feedback to automatically retrain and deploy improved models without system downtime. This project demonstrates a comprehensive understanding of modern data engineering principles, including stream processing, distributed computing, data warehousing, and automated machine learning operations, to solve a critical business problem.

## 2. Problem Statement

Financial fraud results in billions of dollars in losses annually. Traditional fraud detection systems often rely on batch processing, meaning fraudulent transactions are only discovered hours or even days after they occur, by which time the funds are often irrecoverable. Businesses require a solution that can:

1. **Detect fraud in real-time** as transactions happen.
2. **Scale** to handle millions of transactions per day.
3. **Adapt and learn** from new fraud patterns to stay ahead of malicious actors.
4. **Minimize false positives** to avoid disrupting legitimate customer transactions.

## 3. Solution Architecture

The solution is designed as two interconnected systems: a **Real-Time Processing Loop** for instant detection and an offline **Model Retraining Loop** for continuous improvement.

**The Real-Time Processing Loop** operates continuously to catch fraud as it happens. The flow is as follows:

1. A **Data Generator** script produces a constant stream of mock financial transactions.
2. Each transaction is published to an **Apache Kafka** topic, our real-time message bus.
3. A **Spark Streaming** application consumes these transactions, loads the current

ML model from **Google Cloud Storage (GCS)**, and scores each transaction for fraud.
4. If the score is high, a **Google Cloud Function** is triggered to send an immediate alert.
5. Regardless of the score, every transaction is logged in our **Google BigQuery** data warehouse for analysis and future training.

**The Model Retraining Loop** runs on a daily schedule to make our system smarter over time:

1. An **Analyst Dashboard** allows a human to review flagged transactions in BigQuery and label them as "Confirmed Fraud" or "Not Fraud".
2. An **Apache Airflow** pipeline triggers daily, querying BigQuery for this newly labeled data.
3. If enough new labels exist, a **retraining script** is executed to train a new, potentially more accurate, ML model.
4. The new model is evaluated, and if it performs better, it is versioned and deployed to **GCS**, replacing the old model. The live Spark application automatically picks up this new model without downtime.

### 3.1. Component Breakdown

- **Data Ingestion (Apache Kafka):** A high-throughput, distributed message queue acts as the entry point for all transaction data. This decouples the data producers from the consumers and ensures durability.
- **Stream Processing (Apache Spark):** A Spark Streaming application running in local mode consumes data from Kafka in micro-batches. It is responsible for applying the ML model to each transaction.
- **Model Storage (Google Cloud Storage - GCS):** A GCS bucket serves as the central repository for versioned machine learning models (e.g., model_v1.pkl, model_v2.pkl).
- **Alerting (Google Cloud Functions):** A lightweight, serverless function is triggered for high-risk transactions. It sends a real-time notification to a designated channel (e.g., Slack, email).
- **Data Warehouse (Google BigQuery):** This serverless, petabyte-scale warehouse is the single source of truth. It stores all raw and scored transactions, as well as the feedback labels from analysts, making it the foundation for both analytics and model retraining.
- **Orchestration (Apache Airflow):** Airflow, running locally via Docker, manages the entire retraining workflow. It schedules, executes, and monitors the tasks required to create and deploy a new model.

- **Analyst Feedback UI (Streamlit):** A simple web application built with Streamlit allows a human analyst to review flagged transactions and provide the crucial "ground truth" labels.

### 4. Technology Stack

| Category | Technology | Role in Project | Justification / No-Cost Strategy |
|---|---|---|---|
| **Messaging** | Apache Kafka | Real-time data ingestion bus. | Using **Confluent Cloud's free tier** for a managed, no-cost cluster. |
| **Processing** | Apache Spark (PySpark) | Real-time scoring of transactions. | Running in **local mode** on the development machine. Free and ideal for portfolio scale. |
| **Data Warehouse** | Google BigQuery | Central storage for all data; analytics and retraining source. | Utilizes GCP's generous **free tier** (1TB queries/month). |
| **Object Storage** | Google Cloud Storage | Storing versioned ML model files. | Utilizes GCP's **free tier**. |
| **Serverless** | Google Cloud Functions | Triggering real-time fraud alerts. | Utilizes GCP's **free tier** (2M invocations/month). |
| **Orchestration** | Apache Airflow | Scheduling and managing the model retraining pipeline. | Running **locally via Docker**. Free and provides full functionality. |
| **ML Framework** | Scikit-learn | Training the fraud detection model (e.g., Isolation Forest). | Open-source and runs locally. |
| **Feedback UI** | Streamlit | Simple web app for analyst feedback. | Open-source and runs locally. |

## 5. Implementation Details & Key Features

### 5.1. Real-Time Detection Loop

The core of the system is a Spark Streaming job that performs a continuous loop of Read -> Process -> Write. It reads from Kafka, enriches the data with a fraud score using the currently deployed model from GCS, and writes the result to BigQuery. The low latency of this loop (sub-second) is critical for preventing fraud before it completes.

### 5.2. MLOps Retraining Loop (The "Closed Loop")

This is the project's most unique feature. The Airflow DAG automates the entire model improvement process:

1. **Trigger:** Runs on a daily schedule.
2. **Extract:** Queries BigQuery for transactions that have been labeled by an analyst in the past 24 hours.
3. **Train & Evaluate:** A Python script is triggered, which uses the new data to retrain the model. It compares the new model's performance (e.g., AUC-ROC score) against the current production model.
4. **Deploy:** If the new model is superior, it is versioned and uploaded to the GCS bucket. The Spark Streaming application is designed to periodically check for a new model version and will "hot-swap" it without interrupting the live data stream.

### 5.3. Unique Technical Challenges

- **Hot-Swapping Models:** Implementing a mechanism in Spark Streaming to gracefully load a new model version without downtime.
- **State Management:** The system is designed to be stateless for simplicity, but could be extended to manage state (e.g., tracking a user's average transaction amount over time) within Spark.
- **Data Integrity:** Ensuring that every transaction is processed exactly once and that there is consistency between the real-time alerts and the data stored in BigQuery.

## 6. Business Impact & Outcomes

- **Reduced Financial Loss:** By detecting fraud in near real-time, the system can prevent fraudulent transactions from ever completing.
- **Improved Model Accuracy:** The automated retraining loop ensures the model constantly adapts to new fraud tactics, preventing model drift and improving accuracy over time.

- **Operational Efficiency:** Automates the entire model lifecycle, freeing up data scientists and engineers from manual retraining and deployment tasks.

**7. Future Enhancements**

- **Advanced Feature Engineering:** Incorporate real-time feature engineering within the Spark job, such as calculating rolling averages of transaction amounts or frequencies for each user.
- **A/B Testing Models:** Deploy multiple models (e.g., a new challenger vs. the current champion) simultaneously and route a fraction of traffic to each to evaluate performance in a live environment.
- **Graph Analytics:** Use a graph database to model relationships between users and merchants to uncover more complex fraud rings.