AutoML FraudShield – V3 (Revised): SRS + TDD + Deployment Manifest

0. Project Overview

- **Project Name:** AutoML FraudShield V3 (The "Real-Time Velocity" Upgrade)
- **Role:** Principal AI Architect / Streaming Data Engineer
- **V2 Recap:** Delivered a production-grade batch system with Explainability (SHAP), Drift Monitoring, and Auto-Retraining using supervised XGBoost.
- **V3 Goal:** Transition from "Reactive Batch" to "Proactive Real-Time" detection. We will implement the streaming infrastructure to catch velocity attacks and unsupervised anomalies in real-time, while establishing the data structures for future multi-tenancy.
- **Philosophy:** "Velocity First." We prioritize the correctness of the streaming data path and hybrid scoring before adding complex network intelligence (GNNs) in V4.

Primary Architecture Changes (V2 $\to$ V3)

1. **Ingestion:** Batch CSV $\to$ **Streaming Pub/Sub + Dataflow**.
2. **Features:** Daily Aggregates $\to$ **Sliding Window Aggregates (10m)**.
3. **Model:** Single XGBoost $\to$ **Hybrid Ensemble (XGBoost + Isolation Forest)**.
4. **Tenancy:** Single Tenant $\to$ **Schema-Level Multi-Tenancy**.

---

1. Scope & Requirements (SRS)

1.1 In-Scope (V3)

1. **Streaming Infrastructure:** Deploy a Dataflow (Apache Beam) pipeline to ingest raw events and compute windowed features.
2. **Real-Time Feature Store:** Materialize sliding window aggregates (e.g., txn_count_10m) to Vertex AI Feature Store (Online) with strict time semantics.
3. **Hybrid Model Serving:** Implement a Custom Prediction Routine (CPR) container that runs XGBoost (Supervised) and Isolation Forest (Unsupervised) sequentially.
4. **Schema-Level Multi-Tenancy:** Add tenant_id to all APIs, logs, storage, and feature lookups to prepare for SaaS scale.

1.2 Out-of-Scope (Deferred to V4)

- **Graph Neural Networks (GNN):** No graph embeddings or neighbor aggregations in this phase.
- **Active Feedback Loop:** No "Confirm Fraud" dashboard write-backs; retraining remains batch-based on historical data.
- **Hard Infrastructure Isolation:** No separate GCP projects or IAM namespaces per tenant.

1.3 Functional Requirements

1.3.1 Real-Time Data (FR-S)

- **FR-S1 – Ingestion:** The system MUST accept JSON transaction events via fraudshield-raw-events Pub/Sub topic.
- **FR-S2 – Windowing:** A Dataflow job MUST compute **Sliding Windows** (Size: 10m, Period: 1m) for velocity features.
- **FR-S3 – Time Semantics:** The pipeline MUST handle late data with an **Allowed Lateness of 5 minutes**. Data arriving after this threshold is dropped.

1.3.2 Hybrid Intelligence (FR-I)

- **FR-I1 – Unsupervised Detection:** The training pipeline MUST train an Isolation Forest on non-fraud data to learn "normality."
- FR-I2 – Ensemble Logic: The Scoring API MUST return a weighted score: $$FinalScore = (w_1 \times P_{XGB}) + (w_2 \times S_{IsoForest})$$ (Default weights: $w_1=0.8, w_2=0.2$, normalized to 0-1).

1.3.3 Operations (FR-O)

- **FR-O1 – Schema Isolation:** Every API request and database row MUST contain a tenant_id. Feature Store entity IDs MUST be constructed as tenant_id#entity_id.
- **FR-O2 – Multi-Model Drift:** The monitoring system MUST track:
    - **PSI** for XGBoost input features.
    - **Mean Anomaly Score** for Isolation Forest (to detect global behavior shifts).

1.4 Non-Functional Requirements (NFR)

- **NFR-1 – Latency Budget (P95):** Total API response time $\le$ **300ms**.
    - Network: 20ms
    - Feature Fetch: 50ms
    - Ensemble Inference: 80ms
    - Overhead/Buffer: 150ms
- **NFR-2 – Data Freshness:** Streaming features MUST be available in the Online Store $\le$ **60 seconds** after the event is published to Pub/Sub.

---

2. High-Level Architecture (System View)

2.1 Component Diagram

2.2 Data Flow (The "Fast Path")

1. **Event:** Client publishes transaction JSON (inc. tenant_id) to Pub/Sub.
2. **Process:** Dataflow job:
    - Reads stream.
    - Assigns Event Time timestamps.
    - Groups by tenant_id + card_id.

    - Calculates count, sum over 10m sliding window.
3. **Materialize:** Writes result to Vertex AI Feature Store (Online).
4. **Score:** Client calls API /v1/score.
5. **Fetch:** API fetches fresh txn_count_10m + static features.
6. **Predict:** Vertex Endpoint runs CPR (XGB + IsoForest).
7. **Respond:** Returns Score + Risk Band.

---

3. Detailed Design (TDD)

3.1 Streaming Pipeline (Dataflow / Apache Beam)

- **Input:** fraudshield-raw-sub
- **Window Configuration:**
    - Type: SlidingWindows(size=600s, period=60s)
    - Trigger: AfterWatermark(early=AfterProcessingTime(delay=10s), late=AfterCount(1))
    - Allowed Lateness: 300s (5 minutes)
- **Deduplication:**
    - Key: transaction_id
    - State: Bloom filter or StateId within the window to prevent double-counting retries.
- **Output Sink:** Vertex AI Feature Store (write_feature_values).

3.2 Hybrid Model Container (CPR)

We replace the standard XGBoost container with a **Custom Prediction Routine**.

- **Artifacts:**
    - model.bst (XGBoost)
    - isolation_forest.joblib (Sklearn)
- **Predictor Logic (predictor.py):**
  Python
  def predict(self, instances):
- # 1. Preprocess
- # 2. Thread A: XGBoost Probability
- prob_xgb = self.xgb_model.predict_proba(instances)[:, 1]
-
- # 3. Thread B: Isolation Forest Score (Normalized)
- # IsoForest returns -1 (anomaly) to 1 (normal). We invert and normalize to 0-1.
- raw_iso = self.iso_model.decision_function(instances)
- prob_iso = self.normalize_anomaly_score(raw_iso)
-
- # 4. Weighted Ensemble
- final_score = (0.8 * prob_xgb) + (0.2 * prob_iso)

- return final_score.tolist()
-

3.3 Training Pipeline Updates

The training pipeline now produces *two* artifacts.

- **Step 1:** Load Data (Filter by tenant_id if specific model, or global).
- **Step 2:** Split Data.
  - *Supervised Set:* Labeled transactions.
  - *Unsupervised Set:* All transactions (ignoring labels) for IsoForest training.
- **Step 3:** Train XGBoost (Optimize for AUC).
- **Step 4:** Train Isolation Forest (Optimize for outlaw density).
- **Step 5:** Upload both to Vertex Model Registry as a single Model resource (using the Custom Container).

3.4 API Schema Updates

- **Request Header:** X-Tenant-ID (Required).
- **Feature Lookup:**
  - Key Construction: entity_id = f"{tenant_id}#{customer_id}"
  - This ensures Tenant A cannot access Tenant B's feature history.

---

4. Deployment Manifest (DM)

4.1 Infrastructure Resources (Terraform)

- **Pub/Sub:**
  - fraudshield-raw-events (Topic)
  - fraudshield-raw-sub (Subscription, Ack Deadline: 60s)
- **Dataflow:**
  - Service Account: fraudshield-dataflow-sa
  - IAM: roles/dataflow.worker, roles/aiplatform.user (for Feature Store write)
  - Bucket: gs://fraudshield-dataflow-temp-{env}
- **Vertex AI:**
  - Feature Store: Add txn_count_10m (INT64), txn_sum_10m (FLOAT).
  - Endpoint: fraudshield-hybrid-endpoint

4.2 Repository Structure (V3)

Plaintext

- fraudshield-v3/
- ├── streaming/ # [NEW]
- │ ├── pipeline.py # Beam Logic
- │ ├── setup.py # Worker dependencies

- │ └── run.sh # Submission script
- ├── models/
- │ ├── ensemble_cpr/ # [NEW]
- │ │ ├── predictor.py # Hybrid Logic
- │ │ ├── requirements.txt
- │ │ └── Dockerfile
- ├── api/ # Updated for Tenant ID
- ├── infra/ # Updated Terraform
- └── pipelines/ # Training (produces 2 artifacts)

4.3 Deployment Strategy

1. **Infrastructure:** terraform apply to create Pub/Sub and Feature Store updates.
2. **Streaming:** Submit Dataflow job. Verify watermark progression and Feature Store writes.
3. **Training:** Run pipeline to generate initial XGB+IsoForest artifacts.
4. **Serving:** Deploy CPR container to Vertex Endpoint.
5. **Cutover:** Update API to point to new Endpoint and require tenant_id.

---

5. Roadmap Check

- **V3 (Current):** Streaming Velocity, Hybrid Model, Schema Multi-tenancy.
- **V4 (Next):** Graph Neural Networks, Active Feedback Loop, Hard Isolation.