**CloudWalk Technical Case – Anti-Fraud Solution**

**Data Analyst - Risk Analyst I**

**Daniel Sousa**

**Architecture Diagram**

**Layer Descriptions & Technical Justification**

1. **Ingestion Layer**
   * **Function:** Receives each transaction event in real time via API or message queue (e.g., Kafka).
   * **Justification:** Guarantees low‑latency (< 100 ms) data capture and decouples upstream systems from processing logic.
2. **Feature Engineering**
   * **Function:** Enriches raw events with derived attributes:
     + **Velocity:** number of transactions per user\_id/device\_id in the last 10 minutes.
     + **AmountBin:** categorizes transaction\_amount into fixed intervals.
     + **Historical CBK Rate:** rolling average of chargeback rate per user.
     + **Device Risk Score:** fingerprinting and IP‑geolocation risk.
   * **Justification:** Produces meaningful predictors that distinguish genuine from fraudulent transactions.
3. **Risk Scoring**
   * **Rules Engine:** Applies configurable business rules (e.g., “> 3 transactions under R$50 in 10 min → flag”).
   * **ML Module:** Loads a pre‑trained supervised model (e.g., XGBoost) to compute a fraud probability.
   * **Score Combination:** Weighted blend (e.g., 50 % rules + 50 % ML) yields a unified risk score.
   * **Justification:** Hybrid approach uses human ability for clear‐cut cases and machine learning for complex patterns.
4. **Decision Engine**
   * **Logic:**
     + **Score ≥ 0.8 →** Decline at once.
     + **0.5 ≤ Score < 0.8 →** Step‑up authentication (3D Secure or OTP).
     + **Score < 0.5 →** Approve.
   * **Justification:** Balances fraud prevention with customer experience by escalating only uncertain cases.
5. **Feedback & Retraining**
   * **Function:** Captures dispute outcomes (stands for wins/losses) and continually retrains the ML model.
   * **Justification:** Ensures the system adapts to evolving fraud tactics and keeps high accuracy over time.

**Prototype Code Snippets**

from fastapi import FastAPI

import joblib

import pandas as pd

app = FastAPI()

model = joblib.load("fraud\_model.pkl")

# Example business rule: more than 3 small txns in 10 minutes

def rule\_velocity(features):

return features.velocity\_10min > 3

rules = [rule\_velocity]

@app.post("/score")

def score\_transaction(txn: dict):

# 1) Feature extraction

features = pd.DataFrame([txn])

features["velocity\_10min"] = compute\_velocity(txn)

features["amount\_bin"] = compute\_amount\_bin(txn["transaction\_amount"])

# …

# 2) Rules scoring

rule\_flags = [int(rule(features.iloc[0])) for rule in rules]

score\_rules = sum(rule\_flags) / len(rules)

# 3) ML scoring

score\_ml = model.predict\_proba(features)[:, 1][0]

# 4) Combined risk score

risk\_score = 0.5 \* score\_rules + 0.5 \* score\_ml

# 5) Decision logic

if risk\_score >= 0.8:

action = "decline"

elif risk\_score >= 0.5:

action = "step\_up"

else:

action = "approve"

return {"risk\_score": round(risk\_score, 3), "action": action}

* **FastAPI** serves a REST endpoint for real‑time scoring.
* **Business rules** are simple Python functions flagged as 0/1.
* **ML model** (e.g., XGBoost) is loaded via **joblib** for probability estimates.
* **Combined score** and **decision thresholds** drive the automated response.

**Evolution Plan**

1. **Monitoring & Alerting**
   * Instrument latency and error metrics (e.g., Prometheus + Grafana).
   * Set alerts if average scoring latency exceeds threshold or if fraud rate spikes.
2. **Rule Management UI**
   * Develop a lightweight dashboard to add, remove, and evaluate rules without deployment cycles.
3. **Model Governance**
   * Automate periodic retraining using a CI/CD pipeline (e.g., GitHub Actions).
   * Validate new models against a hold‑out set before production rollout.
4. **A/B Testing & Tuning**
   * Run parallel experiments comparing pure‑rule vs. hybrid vs. pure‑ML configurations.
   * Improve weightings and thresholds based on false‑positive/negative trade‑offs.
5. **Data Expansion**
   * Integrate new data sources (e.g., device fingerprinting service, external fraud feeds).
   * Continuously enrich feature store for improved detection fidelity.