CloudWalk – Tx Monitoring PoC (Step 3.2)

Purpose: a tiny FastAPI service that receives per-minute transaction counts (failed/denied/reversed), compares them to a historical baseline (mean/std) calculated from a CSV, and flags anomalies using a Z-score threshold. Optionally, it posts an alert message to Slack via an incoming webhook.

What you can demo quickly:

- POST /alert with the minute and counts → service returns whether to alert for each status.
- Baseline is loaded once at startup from CSV (path in .env).
- Threshold K (in std devs) is configurable in .env (default 3.0).
- If SLACK_WEBHOOK_URL is set, anomalies trigger a Slack notification.

Repository structure: app.py · .env.example · README.md · data/transactions.csv (local)

How it Works – Data Flow

1) Baseline build (at startup):

- Read CSV → normalize status → group per minute → pivot to columns (failed/denied/reversed).
- For each column c, compute mean[c] and std[c] across the whole CSV window.
- Save a per-status threshold thr[c] = mean[c] + K * std[c].

2) Real-time check (per POST /alert call):

- Receive JSON with {minute, failed, denied, reversed}.
- For each status c: compute z = (value - mean[c]) / std[c] (if std[c] > 0).
- If value > thr[c] → alert_c = true (or use |z| ≥ 2/3 for other policies).

Configuration (via .env):

TX_CSV_PATH=path to the CSV with historical per-minute counts

ALERT_K=3.0

SLACK_WEBHOOK_URL=<optional webhook URL>

API – minimal contract & example

```
POST /alert
Content-Type: application/json

{
  "minute": "2025-07-12 13:45",
  "failed": 50,
  "denied": 50,
  "reversed": 50
}
```

Response (fields per status): value, mu, sigma, threshold, k, z, alert.


```
{
  "minute": "2025-07-12 13:45",
  "result": {
    "failed":  {"value": 50, "mu": 0.06, "sigma": 1.49, "threshold": 4.63, "k": 3, "z": 33.2, "alert": true},
    "denied":  {"value": 50, "mu": 6.95, "sigma": 3.54, "threshold": 17.59, "k": 3, "z": 12.1, "alert": true},
    "reversed":{"value": 50, "mu": 2.19, "sigma": 1.64, "threshold": 7.11, "k": 3, "z": 29.3, "alert": true}
  },
  "any_alert": true
}
```


If SLACK_WEBHOOK_URL is set, a concise message is posted when any_alert=true.

Quick Start (local)

```
# 1) Create env & install deps
python -m venv .venv
.\.venv\Scripts□ctivate
pip install -r requirements.txt

# 2) Configure environment
copy .env.example .env
# (edit .env if needed)

# 3) Run
uvicorn app:app --reload --port 8000

# 4) Test (Swagger)
http://127.0.0.1:8000/docs
```

Or send a raw request with curl:

```
curl -X POST http://127.0.0.1:8000/alert ^
  -H "Content-Type: application/json" ^
  -d "{"minute":"2025-07-12 13:45","failed":50,"denied":50,"reversed":50}"
```

Reference – SQL we used in DuckDB

This is the exact shape we explored while building the baseline in Python. You can run it in DuckDB to preview the minute-level series from the CSV:

```
-- 1) Parse & normalize, then pivot to one row per minute
WITH base AS (
  SELECT
    date_trunc('minute', TRY_CAST(timestamp AS TIMESTAMP)) AS ts_minute,
    CASE
      WHEN status IN ('reversed','backend_reversed','refunded') THEN 'reversed'
      WHEN status IN ('approved','failed','denied') THEN status
      ELSE 'other'
    END AS stat,
    CAST(count AS BIGINT) AS cnt
  FROM read_csv_auto('data/transactions.csv', HEADER=TRUE)
),
wide AS (
  SELECT
    ts_minute AS minute,
    SUM(CASE WHEN stat='failed'   THEN cnt ELSE 0 END) AS failed,
    SUM(CASE WHEN stat='denied'   THEN cnt ELSE 0 END) AS denied,
    SUM(CASE WHEN stat='reversed' THEN cnt ELSE 0 END) AS reversed
  FROM base
  GROUP BY 1
)
SELECT * FROM wide ORDER BY minute LIMIT 20;
```

From this table we compute mean/std per column to form the alert thresholds.