



FraudShield 360

AI-Powered UEBA + Tamper-Proof Blockchain Logging, streamed with Kafka, containerized with Docker

1) TL;DR (non-tech, 30 sec)

- Today's fraud tools are **noisy** (too many false alarms), **rigid** (rules can't keep up), and **opaque** (can't explain decisions).
- **FraudShield 360** learns each customer's normal behaviour, flags unusual actions, **explains why**, and **locks every decision on a blockchain** so it can't be secretly changed.
- Built to **scale in real time** with **Kafka** and to **deploy anywhere** with **Docker**.

2) Cons of existing systems (layman's terms)

1. **Too many false alarms** → customers get blocked for legit activity.
2. **Slow to adapt** → fraudsters change tricks faster than rule updates.
3. **Tunnel vision** → only looks at transactions, ignores logins/devices/peer patterns.

4. **No clear reasons** → can't tell customers/RBI *why* something was flagged.
 5. **Logs can be edited** → weak trust with auditors.
 6. **Doesn't scale smoothly** → real-time spikes overwhelm legacy stacks.
-

3) How FraudShield 360 overcomes them

- **UEBA baselines** → learns "what's normal" per user/entity → **fewer false alarms**.
 - **Anomaly models** → catch **new** fraud methods without waiting for new rules.
 - **Signal fusion** → transactions + logins + device + geo + peer group → **full picture**.
 - **Explainability** → human reason codes (amount/time/device) and SHAP (optional).
 - **Blockchain ledger** → decisions are **immutable** (tamper-evident audit).
 - **Kafka streaming** → elastic, real-time ingestion and scoring at scale.
 - **Docker** → one-click local or cloud deployment.
-

4) Architecture (two views)

A) Human story (laymen terms)

1. We **watch all signals** (money movement, logins, devices, locations).
2. We **know your normal** habits.
3. If something looks **very unusual**, the AI raises risk.
4. We **act smartly**: allow, ask extra OTP/selfie, or hold/block.
5. We **explain clearly** why.
6. We **write in permanent ink** (blockchain) so no one can alter the record.
7. We **learn** from analyst feedback and get better every week.

B) Technical blueprint (engineers)

[Channels] Mobile • NetBanking • ATM • Card • CRM • IAM/SSO • Device Int
el



▼ (Events)

[API Gateway] → [Kafka Topics: logins, txns, devices, cases]



[Stream Processing] (Flink/Spark or lightweight consumer)

- ├─ enrich: geo, device reputation, peer z-score, session velocity
- ├─ update: Online profiles/baselines (Redis/Postgres)
- └─ publish: features → scoring topic



[Scoring Service] (FastAPI/Flask, autoscaled via Docker)

- ├─ Anomaly models (IsolationForest / Autoencoder)
- ├─ Rules engine (YAML policies)
- └─ Fusion risk score + reason codes



[Decisioning] (policy thresholds)

- ├─ allow / step-up auth / hold/block
- ├─ case create → Kafka "cases"
- └─ write immutable record → Blockchain/Notary



└─ [Analyst UI + Case Mgmt] (feedback)

└─ [Blockchain Ledger] (local hash-chain OR Hyperledg

er)



[Training Pipeline] (batch)

- ├─ feature store (offline)
- ├─ retrain + validate
- └─ model registry + rollout

5) End-to-end flow (diagram + JSON samples)

Customer Action → Kafka (events) → Feature Enrichment → UEBA + Rules → Risk + Reasons

↓

↓

Decision (allow/challenge/block) → Case Mgmt + Blockchain record

↓

Analyst feedback → Retrain nightly

Event (login/txn) JSON (example)

```
{
  "event_id": "e_924f",
  "user_id": "U18273",
  "channel": "netbanking",
  "action": "transfer",
  "amount_inr": 90000,
  "device_id_hash": "d9c1...",
  "ip": "49.205.x.y",
  "geo": {"lat": 9.98, "lon": 76.28},
  "ts": "2025-09-19T10:05:22+05:30"
}
```

Scoring response

```
{
  "risk": 0.87,
  "action": "HOLD",
  "reasons": [
    "amount 10x user median",
    "first-time device",
    "login time outside usual window"
  ],
}
```

```
"model_version": "iforest_v3.2",
"rules_triggered": ["new_device_high_amount"],
"ts": "2025-09-19T10:05:23+05:30"
}
```

Blockchain record (hashed)

```
{
  "decision_id": "d_657a",
  "prev_hash": "0000ab...",
  "payload_sha256": "5e2a...",
  "payload": {
    "event_id": "e_924f",
    "risk": 0.87,
    "action": "HOLD",
    "reasons": ["amount 10x", "new device", "odd hour"],
    "model_version": "iforest_v3.2",
    "ts": "2025-09-19T10:05:23+05:30"
  },
  "block_ts": "2025-09-19T10:05:24+05:30"
}
```

6) MVP scope for hackathon (fast but impressive)

- **Journey:** *Login → Add new payee → First high-value transfer.*
- **Signals:** time-of-day, device-seen-before, user typical amount, peer z-score, geo distance/velocity.
- **Model:** IsolationForest (unsupervised) + thin rules (YAML).
- **UI:** Analyst table with *risk*, *action*, *reasons*, buttons: "Mark fraud / not-fraud".
- **Blockchain:** local hash-chained JSON blocks; show tamper detection live.
- **Kafka:** 3 topics (events.features, events.scored, cases).

- **Goal: $\geq 30\%$ fewer false positives** at equal or higher detection vs rules-only baseline (on your synthetic demo set).
-

7) Tech stack (chosen for speed + credibility)

- **Ingestion/Streaming:** Apache Kafka (Redpanda works too).
 - **API/Scoring:** FastAPI or Flask + Gunicorn/Uvicorn.
 - **ML:** scikit-learn (IsolationForest/OneClassSVM), optional Keras AE; XGBoost later.
 - **Feature/Baselines:** Redis (online), Postgres/SQLite (profiles, cases).
 - **Explainability:** custom reason codes (real-time), SHAP (offline plots for slides).
 - **Blockchain/Notary:**
 - **MVP:** Python hash-chain (file/DB backed).
 - **Enterprise path:** Hyperledger Fabric channel "fraud-audit".
 - **Containers:** Docker + docker-compose (or K8s later).
 - **UI:** Flask/Jinja or React (depending on time).
 - **Observability:** Prometheus metrics + Grafana dashboard (stretch).
-

8) Folder structure (production-ish but hackathon-friendly)

```
fraudshield360/
├── docker-compose.yml
├── requirements.txt
├── run_local.sh
├── README.md
├── app/
│   ├── __init__.py
│   ├── routes/
│   └── api.py          # ingest events, score endpoint (fallback/manual)
```

```

| | | └─ dashboard.py      # analyst UI
| | └─ services/
| | | └─ kafka_consumer.py # consume events, produce features/scores/
cases
| | | └─ feature_engineering.py
| | | └─ baselines.py      # read/write user/entity baseline stats
| | | └─ anomaly_model.py  # train/load/predict
| | | └─ rules_engine.py   # YAML-driven policies
| | | └─ scorer.py         # fuse model + rules → risk + reasons
| | | └─ decisioning.py    # thresholds + actions
| | | └─ blockchain_logger.py # local hash-chain / fabric client
| | | └─ feedback_loop.py  # apply analyst labels
| | | └─ metrics.py        # Prometheus counters/histograms
| | └─ templates/
| | | └─ dashboard.html
| | └─ static/
| | | └─ app.css
| └─ config/
| | └─ rules.yml
| | └─ app.env.example
| | └─ kafka.yml
| └─ data/
| | └─ sample_events.jsonl
| | └─ seeds/
| └─ models/
| | └─ iforest_v1.pkl
| └─ scripts/
| | └─ seed_kafka.py
| | └─ retrain_offline.py

```

9) Docker & Compose (ready to run locally)

docker-compose.yml (concept)

```
version: "3.9"
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.6.1
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
  kafka:
    image: confluentinc/cp-kafka:7.6.1
    depends_on: [zookeeper]
    environment:
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
  api:
    build: .
    command: uvicorn app.routes.api:app --host 0.0.0.0 --port 8000
    env_file: config/app.env.example
    depends_on: [kafka]
    ports: ["8000:8000"]
  worker:
    build: .
    command: python -m app.services.kafka_consumer
    env_file: config/app.env.example
    depends_on: [kafka, api]
  dashboard:
    build: .
    command: flask --app app.routes.dashboard run --host 0.0.0.0 --port 8050
    env_file: config/app.env.example
    depends_on: [api]
    ports: ["8050:8050"]
```

Quick start

1) Build & run

```
docker compose up --build -d
```

2) Create topics (if not auto-created)

```
docker exec -it fraudshield360-kafka-1 \
```

```
kafka-topics --create --topic events.raw --bootstrap-server kafka:9092
```

```
docker exec -it fraudshield360-kafka-1 \
```

```
kafka-topics --create --topic events.features --bootstrap-server kafka:9092
```

```
docker exec -it fraudshield360-kafka-1 \
```

```
kafka-topics --create --topic events.scored --bootstrap-server kafka:9092
```

```
docker exec -it fraudshield360-kafka-1 \
```

```
kafka-topics --create --topic cases --bootstrap-server kafka:9092
```

3) Seed demo events

```
docker exec -it fraudshield360-api-1 python scripts/seed_kafka.py
```

10) Rules policy (YAML snippet)

hard_stops:

- id: new_device_high_amount

- if: "not features.device_seen_before and features.amount_inr >= 75000"

- action: "HOLD"

soft_signals:

- id: night_owl

- weight: 0.2

- if: "features.hour in [0,1,2,3,4]"

thresholds:

- allow_below: 0.35

- stepup_between: [0.35, 0.7]

- block_above: 0.7

11) Blockchain choices (pick one for demo)

- **Option A — Local Notary (fastest):** Append-only “blocks” in a DB/file, each storing `sha256(prev_block || payload)`. Provide a *Verify* button that recomputes the chain and shows “**TAMPERED / OK**”.
 - **Option B — Hyperledger Fabric (enterprise path):** Write decision payload hash to a Fabric chaincode on a dedicated **fraud-audit** channel. Show a CLI/query to prove immutability.
-

12) Metrics & success criteria

- **Alert Precision↑:** fewer false alarms vs rules-only.
 - **TPR↑:** equal or better true-positive rate.
 - **Mean explain length ≤ 3 reasons** (clear).
 - **Latency p95 < 100ms** per scoring call (demo target).
 - **Tamper check = OK** across N decisions.
-

13) Demo script (3 minutes)

1. **Simulate normal** transfers → show “ALLOW” with no friction.
 2. **Run a risky** event (new device + ₹90k at 2:03am) → “HOLD” + **reasons** pop up.
 3. **Open blockchain viewer** → last block displays hash + payload.
 4. **(Optional) Tamper test:** edit a past decision in DB → click *Verify* → shows **TAMPERED**.
 5. **Analyst marks** case as fraud → run retrain script → new model version deployed (print v#).
 6. Re-run similar risky event → see **better calibrated risk** (learns).
-

14) Risks & mitigations

- **Cold start profiles:** no history for new users → fall back to **peer-group norms** + tighter rules for first N actions.
 - **Data drift:** behaviour shifts after product launches → weekly retrain + drift monitors.
 - **PII & privacy:** hash device IDs, coarse geo bins; role-based access to cases.
 - **Over-blocking:** keep **step-up auth** tier between allow & block.
-

15) What to say to judges (1-minute pitch)

FraudShield 360 is a real-time, explainable fraud defense.

It **learns each customer's normal**, cuts **false alarms**, and **catches new scams**.

Every decision is **written to a blockchain**—so no one can quietly alter the past.

We stream with **Kafka** and ship in **Docker**, so banks can **scale fast** and **deploy anywhere**.