**fully open-source, on-premises architecture** for monitoring and alerting on ISO 8583 transactions at **100,000–200,000 TPS**, designed for scalability, cost-efficiency, and real-time responsiveness:

**1. Data Collection & Ingestion**

**Tools**:

**Apache Kafka**: Distributed message broker for high-throughput ingestion (handles 200k+ TPS with horizontal scaling).

**Filebeat**: Lightweight agent to collect ISO 8583 logs from switches/terminals.

**Telegraf**: Collect system metrics (CPU, memory, disk I/O) from servers.

**Parsing**:

Use **jPOS** (Java) or **iso8583-python** to decode ISO 8583 messages and extract critical fields (MTI, response codes, PAN, amount).

**2. Stream Processing**

**Tools**:

**Apache Flink**: Stateful stream processing for real-time metrics (e.g., TPS, error rates, fraud detection).

**Kafka Streams**: Lightweight processing for enrichment (e.g., adding merchant details from a PostgreSQL DB).

**Key Workflows**:

Compute **success/error rates** per transaction type (MTI/processing code).

Detect **velocity-based fraud** (e.g., >5 transactions per PAN in 60 seconds).

Track **end-to-end latency** (request-to-response time).

**3. Storage**

**Time-Series Data**:

**VictoriaMetrics**: Prometheus-compatible, high-performance TSDB (handles 1M+ samples/sec on modest hardware).

**Apache Parquet + HDFS**: For cost-effective long-term retention of raw metrics.

**Logs & Traces**:

**Elasticsearch**: Store parsed ISO 8583 logs (retain 30 days hot storage, archive to **MinIO** for cold storage).

**PostgreSQL**: For reconciliation data (request/response matching) and reference tables (BIN ranges, merchant IDs).

**4. Visualization**

**Grafana**: Unified dashboards for:

Real-time TPS, error rates, and latency (sourced from VictoriaMetrics).

Geo-distribution of transactions (integrate with GeoIP in Elasticsearch).

**Kibana**: Investigate raw transaction logs (e.g., filter by response code 06 for "Error") and system health.

**5. Alerting**

**Prometheus Alertmanager**: For threshold-based alerts:

response\_code != 00 > 5% over 5m

http\_server\_requests\_seconds:percentile95 > 2s

**ElastAlert**: Detect anomalies in Elasticsearch logs (e.g., PAN velocity spikes).

**Grafana Alerts**: Notify teams via email/Slack for dashboard-based thresholds.

**6. Infrastructure Design**

**Hardware Requirements** (example for 200k TPS):

**Kafka Brokers**: 3 nodes (16 vCPU, 64GB RAM, NVMe SSDs) – handle message buffering.

**Flink Cluster**: 4 task managers (32 vCPU, 128GB RAM) – process streams.

**VictoriaMetrics**: 2 nodes (32 vCPU, 128GB RAM, 10TB SSD) – store metrics.

**Elasticsearch**: 5-node cluster (64 vCPU, 256GB RAM, 20TB NVMe) – hot storage.

**Networking**:

10 Gbps NICs for Kafka/Flink/Elasticsearch nodes.

Segment traffic: Isolate transaction processing from analytics.

**7. Security & Compliance**

**Data Masking**:

Use Flink to tokenize PANs (e.g., replace with SHA-256 hash) before storage.

**PostgreSQL pgcrypto** for encrypting sensitive fields at rest.

**Access Control**:

**Keycloak**: Integrate with Grafana/Kibana for RBAC.

**Auditbeat**: Log access to Elasticsearch and PostgreSQL.

**Encryption**:

TLS for Kafka/Flink/Elasticsearch inter-node communication.

LUKS disk encryption for VictoriaMetrics/Elasticsearch data.

**8. Scalability & Reliability**

**Kafka**:

Use 6+ partitions per topic to parallelize Flink processing.

Enable replication factor=3 for fault tolerance.

**Flink**:

Configure RocksDB state backend for checkpointing (fault tolerance).

Scale horizontally by adding task managers.

**Elasticsearch**:

Use time-based indices (e.g., transactions-2024-01-01) with 3 shards per index.

Deploy dedicated master nodes to prevent split-brain.

**9. Cost Optimization**

**MinIO**: Replace S3 for cold storage (deploy on HDDs for cost savings).

**ZFS Compression**: Enable on Elasticsearch/MinIO nodes to reduce storage costs.

**Vertical Scaling**: Start with smaller nodes and expand as traffic grows.

**10. Deployment Tools**

**Ansible**: Automate provisioning of Kafka/Flink/Elasticsearch nodes.

**Docker Compose**: For local testing (e.g., MinIO, Grafana).

**Prometheus Operator**: Manage VictoriaMetrics/Prometheus via Kubernetes (optional).

**11. Disaster Recovery**

**Backups**:

Daily Elasticsearch snapshots to MinIO.

VictoriaMetrics hourly snapshots to NFS.

**DR Site**:

Async Kafka mirroring to a secondary data center.