**Title: *Scalable Data Pipeline for a Multi-Department Pharmaceutical Company***

**Author: Aman Singh**

**Date: 16 Sept, 2025**

Table of Contents

|  |  |  |
| --- | --- | --- |
| S.No. | Contents | Page |
| 1 | Executive Summary | 1 |
| 2 | Business Context & Requirements | 2 |
| 3 | Architecture Overview | 3-5 |
| 4 | Design Pattern Choice (Hybrid: Lambda + Kappa) | 5 |
| 5 | Data Flow & Processing Stages | 6 |
| 6 | Implementation Strategy   * Tools & Technologies * Data Schemas * Sample Code (Python, SQL, Airflow) * Scalability Planning | 6-9 |
| 7 | Data Quality & Error Handling | 9 |
| 8 | Monitoring & Governance Strategy | 9-10 |
| 9 | Data Lineage & Audit Trails | 10 |
| 10 | GDPR & Compliance Considerations | 10-11 |
| 11 | Roadmap & Scalability Outlook | 11 |
| 12 | Conclusion  **Appendix:** Sample code references and snippets | 11-12 |

**1. Executive Summary**

Pharmaceutical companies today operate in one of the most data-intensive and compliance-driven industries. Every department — from sales and manufacturing to clinical research and regulatory affairs — produces high-value data that is essential for both daily operations and long-term innovation.

Sales executives rely on **real-time ERP data** to manage orders, monitor revenue, and make rapid commercial decisions. Manufacturing plants depend on **IoT sensor data** to maintain quality and safety, often requiring near-real-time alerts to avoid costly shutdowns. Clinical researchers handle **longitudinal patient data**, where integrity and reproducibility can determine the success of drug approvals. Marketing and pharmacovigilance teams monitor **social sentiment**, spotting adverse reactions early. Finance teams close books every day, reconciling vast amounts of structured data. Finally, regulatory affairs manage **monthly compliance reports** that, if mishandled, could trigger audits, fines, or even product recalls.

Currently, the organization processes **1 million records per day**, but projections estimate a surge to **10 million per day** within two years, fueled by digital expansion and increased IoT adoption. Scaling this volume with existing fragmented processes is unsustainable.

The solution is a **hybrid pipeline** that blends the strengths of **Kappa (streaming-first)** and **Lambda (batch + stream)** architectures. At its core, the design leverages:

* **Apache Kafka** as the ingestion backbone.
* **Apache Airflow** to orchestrate six separate DAGs aligned to departmental frequencies.
* **Spark/Flink** for real-time ERP and IoT processing.
* **SQL/Spark batch jobs** for Clinical, Regulatory, Finance, and Social Media.
* A **three-zone storage model** (staging → cleansed → presentation).
* Governance, lineage, and GDPR compliance integrated from the ground up.

This pipeline not only ensures timely, reliable insights but also guarantees **regulatory-grade reproducibility**, a must in the pharmaceutical industry.

**2. Business Context & Requirements**

**ERP Sales**

ERP data captures customer orders, product demand, and revenue in real time. In today’s competitive pharmaceutical landscape, sales managers demand dashboards that reflect the **latest transactions instantly**. If it takes even five minutes to surface an order, opportunities may be lost during a critical drug launch campaign.

**IoT Manufacturing Sensors**

Factories run 24/7 and generate thousands of sensor readings every minute. Monitoring every single raw reading is excessive, but ignoring this data could risk quality issues or regulatory violations. A **15-minute aggregation cadence** provides the right balance — enough to detect anomalies before they escalate, while keeping system load manageable.

**Clinical Trials**

Clinical trial data is among the most sensitive datasets in the company. Integrity, accuracy, and **audit trails** are more important than latency. Weekly extracts provide scientists with reproducible data snapshots while meeting regulatory expectations under **FDA 21 CFR Part 11** and **EMA Annex 11**.

**Social Media Sentiment**

Pharmaceutical brands face scrutiny not just from regulators but also from patients and the public. A single viral tweet about adverse side effects can damage reputation. By ingesting and analyzing **daily sentiment data** from APIs, the company strengthens pharmacovigilance and brand monitoring.

**Regulatory Compliance Reports**

Regulatory departments submit structured reports on a **monthly basis**. These must be ingested securely, stored immutably, and tracked with lineage. A misfiled or altered report could trigger penalties, making auditability paramount.

**Financial Systems**

Finance departments require **daily closing data** to reconcile ERP sales with expenses and banking systems. Accuracy is non-negotiable, as discrepancies can affect shareholder reporting and compliance with accounting standards.

**Summary of Requirements**

The pipeline must:

1. Handle diverse data velocities (real-time, near real-time, batch).
2. Provide a unified backbone for all six sources.
3. Ensure governance, lineage, and compliance with GDPR and pharma regulations.
4. Scale from **1M → 10M+ records/day**.
5. Support both **BI dashboards** and **advanced ML pipelines**.

**3. Architecture Overview**

The architecture follows a layered design to separate concerns and simplify scalability.

**Ingestion Layer**

* Six producers push events into Kafka topics (erp\_sales, iot\_sensors, clinical\_trials, social\_media, reg\_reports, finance\_data).
* Producers are written in Python using Faker for synthetic data. In production, these would be API connectors, IoT gateways, or CDC (Change Data Capture) tools like Debezium.

**Processing & Orchestration Layer**

* Apache Airflow DAGs orchestrate consumers on departmental schedules.
* Stream processing (ERP, IoT): Spark/Flink jobs provide sub-minute latency.
* Batch processing (Clinical, Finance, Regulatory, Social): Spark SQL/ETL jobs transform scheduled loads.

**Storage Layer**

* Staging zone: Raw JSON/Parquet files with minimal validation.
* Cleansed zone: Deduplicated, validated, pseudonymized, and standardized data stored in Delta Lake/Hudi.
* Presentation zone: Aggregated data optimized for BI/ML queries, stored in Snowflake/BigQuery/Redshift.

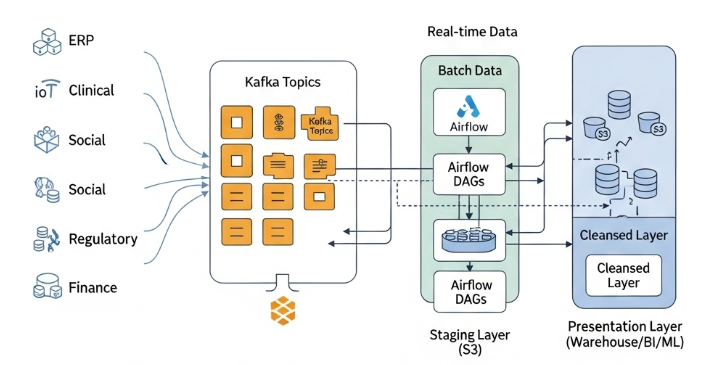
**Serving Layer**

* BI dashboards (Power BI, Tableau).
* ML feature store for predictive models (e.g., predictive maintenance, sales forecasting).
* Compliance reporting systems for regulators.

**Governance & Monitoring Layer**

* Catalog: Amundsen/DataHub.
* Lineage: OpenLineage + Airflow.
* Data Quality: Great Expectations.
* Monitoring: Prometheus + Grafana, Airflow SLA alerts.
* Alerting: Slack, PagerDuty integration.

Architecture Diagram (to reproduce in draw.io):



**4. Design Pattern Choice**

Lambda Architecture

* Pros: Reproducibility, reliability, ability to backfill from batch layer.
* Cons: Two code paths increase complexity, slower for real-time use cases.

Kappa Architecture

* Pros: Single stream-first approach, excellent for real-time analytics.
* Cons: Overkill for monthly/weekly datasets; complexity in handling batch backfills.

Hybrid Approach

Our design combines the best of both:

* Kappa-style for ERP and IoT — where speed is critical.
* Lambda-style for Clinical, Regulatory, Finance, and Social — where reproducibility is key.

This ensures every department gets the right balance of speed vs. reliability.

**5. Data Flow & Processing Stages**

**Record lifecycle (ERP example):**

1. ERP system (or Python producer) emits a JSON message to erp\_sales.
2. Kafka persists the message.
3. ERP DAG (Airflow) either triggers a short consumer job (event-driven to approach ~30s latency) or runs scheduled micro-batches (1 minute recommended where sub-minute cron isn’t feasible).
4. The message is written to the staging zone with metadata: source, ingest\_ts, topic\_offset, raw\_payload.
5. Cleansing step: schema validation, deduplication (ROW\_NUMBER), masking/pseudonymization of PII, enrichment (customer\_id mapping).
6. Curated records written to Delta tables. Aggregations populate presentation tables (daily revenue, region summaries).
7. Consumers (BI/ML) query presentation layer for dashboards and models.

**IoT specifics:** windowed streaming aggregates (15-minute tumbling windows), watermarks for late data, and output retention for downstream analytics.

**Batch flows:** Clinical/Regulatory/Finance are staged as files/parquet and processed in scheduled Airflow jobs with full audit logging.

**6. Implementation Strategy**

**Tools & Technologies (what we used / recommend)**

* **Ingestion & Messaging:** Apache Kafka (topics per source), Kafka Connect for DB CDC (Debezium in production).
* **Orchestration:** Apache Airflow — one DAG per topic for clarity and isolation.
* **Stream Processing:** Spark Structured Streaming or Flink for ERP and IoT transformations.
* **Batch Processing:** Spark / SQL jobs for weekly/monthly sources.
* **Storage:** S3/GCS for landing; Delta Lake/Hudi for cleansed tables; Snowflake/BigQuery/Redshift as presentation layer.
* **Quality & Governance:** Great Expectations, OpenLineage, Amundsen/DataHub.
* **Monitoring:** Prometheus + Grafana, ELK/OpenSearch for logs.
* **Secrets/Encryption:** Vault/KMS, TLS for Kafka, AES-256 at rest.

**Data Schemas (examples)**

**Staging (ERP):**

order\_id STRING,

customer\_name STRING,

amount DECIMAL(12,2),

currency STRING,

source\_ts TIMESTAMP,

ingest\_ts TIMESTAMP,

raw\_payload STRING

**Cleansed (ERP):**

order\_id STRING PRIMARY KEY,

customer\_id STRING, -- pseudonymized

amount DECIMAL(12,2),

currency STRING,

order\_ts TIMESTAMP,

ingest\_ts TIMESTAMP,

is\_high\_value BOOLEAN

**IoT 15-min aggregate:**

machine\_id INT, window\_start TIMESTAMP, window\_end TIMESTAMP, avg\_temp DOUBLE, avg\_pressure DOUBLE

**Sample Code (high-level)**

* **Producer snippet (ERP):**

sale = {...}

producer.send("erp\_sales", sale)

time.sleep(2) # near-real-time generation in our test setup

* **Airflow DAG (ERP):** event-driven or 1-min micro-batch; consume N messages and commit offsets.
* **SQL dedupe (ERP):**

INSERT INTO erp\_cleansed

SELECT order\_id, sha2(customer\_name,256) as customer\_id, amount, currency, source\_ts

FROM (

SELECT \*, ROW\_NUMBER() OVER (PARTITION BY order\_id ORDER BY ingest\_ts DESC) rn FROM stg\_erp\_sales

) t WHERE rn = 1;

* **PySpark (IoT aggregation):** Spark Structured Streaming reading iot\_sensors topic, groupBy window and machine\_id, write to Delta.

**Scalability Planning**

* Partition Kafka topics (ERP: 30+ partitions).
* Autoscale streaming clusters (K8s + Flink/Spark operator).
* Use compaction for topics with idempotent keys.
* Warehouse compute auto-scaling (Snowflake).
* Run load tests using synthetic producers at 2x/5x target rates.

**7. Data Quality & Error Handling**

**Validation rules (implemented):**

* ERP amount >= 0, currency in ISO list.
* IoT temperature and pressure in expected ranges with anomaly scoring.
* Clinical trial record completeness (patient\_id, trial\_id, visit\_date).
* Finance balances non-negative and reconciled.

**Error handling pattern:**

* Schema validation at ingestion (fail-fast).
* Retries: connectors and Airflow tasks retry 3 times with exponential backoff.
* DLQ: malformed or rule-breaking messages land in \*\_dlq topics and S3 folders, annotated with error reason and ingest metadata.
* Reprocessing: DLQ reprocess job that attempts fix-and-retry with human-in-the-loop review where necessary.

**8. Monitoring & Governance Strategy**

**Key monitoring signals:**

* Kafka: throughput, consumer lag, partition health.
* Streaming jobs: input rate, processed rate, processing latency, checkpoint age.
* Airflow: DAG runtime, success/failure counts, SLA misses.
* Data quality: rule pass/fail counts, DLQ growth.

**Dashboards & Alerts:**

* Grafana dashboards per topic/job.
* PagerDuty for P1 events (consumer lag > threshold, DLQ spike).
* Slack alerts for P2 (minor anomalies).

**Governance:**

* Metadata catalog (Amundsen) with dataset owners, glossary, and sensitivity tags.
* Lineage generation for every Airflow task (OpenLineage).
* Audit table: job\_id, dag\_id, start\_ts, end\_ts, records\_in, records\_out, status, error\_summary.

**9. Data Lineage & Audit Trails**

Lineage is captured from source topic → staging file → cleansed table → presentation table. Airflow tasks emit OpenLineage events so automated lineage is visible in the catalog. Audit trails include per-job logs, file manifests, and Kafka offsets to ensure traceability and reproducibility for regulatory review.

**10. GDPR & Compliance Considerations**

Key controls:

* **Pseudonymization**: customer and patient identifiers hashed (SHA-256 with salt) for analytics; reversible tokenization only if required (stored in a secure vault with strict access).
* **Right-to-be-forgotten**: process that accepts a deletion token and removes/pseudonymizes data across staging, cleansed, and presentation layers; audit log of deletion events retained as proof.
* **Retention policies**: raw (90 days), cleansed (3 years), audit (7 years).
* **Encryption & Access**: TLS in transit, AES-256 at rest, role-based access, principle of least privilege.

**11. Roadmap & Scalability Outlook**

Phase 1 (0–3 months): Harden ingestion, finalize topics and partitions, deploy Airflow DAGs and staging.

Phase 2 (3–6 months): Implement cleansing, Delta tables, and initial governance (catalog + lineage).

Phase 3 (6–12 months): Scale streaming clusters, optimize partitioning, and implement DLQ reprocessing.

Phase 4 (12–24 months): Full ML pipelines, feature store, and scale to >10M/day through autoscaling and partitioning improvements.

**12. Conclusion**

The hybrid data pipeline outlined in this report is not merely a technology initiative but a **business transformation strategy** that positions the pharmaceutical company for long-term success. By tailoring ingestion and processing approaches to the unique rhythms of each department — real-time for ERP and IoT, scheduled batch for clinical, finance, social, and regulatory — the architecture ensures that every business function receives data in the form and cadence that best supports its objectives. The adoption of a layered model — staging for raw capture, cleansed for trusted integration, and presentation for actionable insights — provides a clear separation of concerns, making the pipeline easier to scale, govern, and audit. With Kafka enabling high-throughput ingestion, Airflow orchestrating workflows, and modern lakehouse and cloud warehouse technologies supporting analytics, the company can confidently scale from **1M to 10M+ daily records** while maintaining reliability and control. More importantly, the pipeline weaves in compliance and governance from the ground up: GDPR safeguards, audit trails, lineage, and pseudonymization practices ensure that sensitive clinical and customer data is both secure and trustworthy. This foundation not only protects the organization from regulatory risks but also creates the trust required to innovate with advanced analytics and AI, from predictive maintenance on manufacturing lines to clinical trial outcome forecasting and real-time commercial dashboards. With a phased roadmap that delivers immediate operational benefits and a clear path toward advanced capabilities, this design provides the company with a sustainable and future-proof data backbone. In short, this pipeline equips the organization with the **timely, accurate, and governed insights** needed to compete in a rapidly evolving industry where agility, compliance, and innovation must go hand in hand.

**Appendix — Sample code references (already present in repo)**

* erp\_producer.py — ERP producer (Faker-based) → erp\_sales topic.
* iot\_producer.py — IoT producer → iot\_sensors.
* clinical\_producer.py, social\_producer.py, regulatory\_producer.py, finance\_producer.py.
* multi\_consumer.py — debug consumer for all 6 topics.
* erp\_dag.py, iot\_dag.py, clinical\_dag.py, social\_dag.py, regulatory\_dag.py, finance\_dag.py — Airflow DAGs per source.
* SQL sample: deduplication and pseudonymization (shown in the body).
* PySpark sample: IoT 15-minute aggregation (available on request as a full job file).