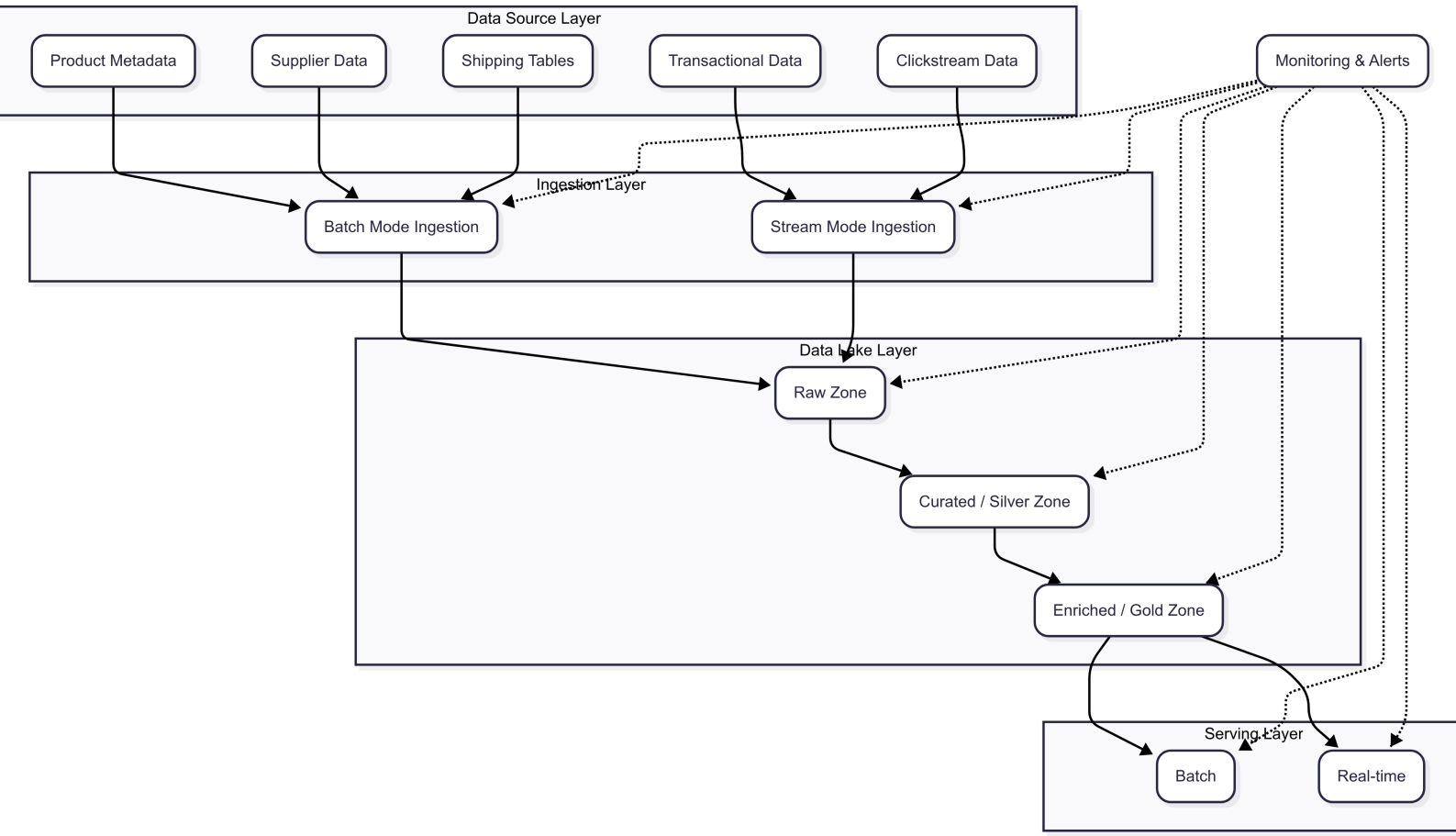
Avinash Devadiga
Data Platform Design
Data Engineering Challenge
22 July 2025

Architecture Diagram



Core Components :

1. Ingestion Layer (Batch and/or Streaming Data)

Our ingestion layer efficiently brings diverse data into our platform, handling both:

- **Batch Ingestion:** For periodic data like product metadata, supplier data, market data, and shipping tables. We'll use **Databricks** to ingest files from source into raw Delta Lake tables, orchestrated by **Apache Airflow**.
- **Streaming Ingestion:** For real-time data such as transactional data and clickstream events. **Apache Kafka** will serve as the message broker, with **Databricks Structured Streaming** consuming data to land it into raw Delta Lake tables.

2. Storage Design (using Delta Lake) and Partitioning Strategy

Our storage uses a multi-layered data lake with **Delta Lake** across all zones for ACID properties and quality.

- **RAW Layer (Bronze Zone):** Stores all ingested data immutably (e.g., product metadata, transactional, supplier, clickstream, market, shipping tables).
    - **Partitioning:** Primarily by **ingestion date** (`year/month/day`).
- **Cleansed Layer (Silver Zone):** Stores cleaned, de-duplicated, and standardized data.
    - **Processing: Databricks notebooks** transform data from Bronze.
    - **Partitioning Examples:** `Product Data` by `category`; `Transactional Data` by `customer_location_delivery_zone` and `transaction_date`.
- **Enriched Layer (Gold Zone):** Contains highly aggregated features and metrics optimized for models and serving.
    - **Processing:** Complex feature engineering via **Databricks notebooks**.
    - **Partitioning Examples:** `Dynamic Pricing Features` by `SKU` or `category`; `Shipping Estimation Features` by `delivery_zone`.

This structure ensures progressive data quality and optimized access for our pricing and shipping solutions.

3. Feature Engineering and Transformation Logic

This phase refines raw/cleansed data into usable features for models, primarily using **Databricks notebooks** (Spark SQL/PySpark).

- **Transformation:** Involves cleaning, aggregating, and joining data (e.g., standardizing `SKU` formats, summing `quantity` from transactional data).
- **Feature Engineering:** Creates new, impactful variables. Examples:
    - **Historical Conversion Rate:** From `Clickstream` and `Transactional data` for `SKU`/`category`.
    - **Anchor Product Score:** From `Product metadata`, `Transactional`, `Clickstream`.
    - **Profit Margin %:** From `Product metadata` and `Supplier data`.
    - **Volumetric Weight:** From `Product metadata` for shipping.
    - **Delivery Zone Mapping:** From `customer location` to standardized zones.

These features populate the **Enriched (Gold) Layer**, ready for dynamic pricing and shipping models.

4. Experimentation Support

Robust experimentation capabilities are essential to continuously optimize our dynamic pricing and shipping strategies, allowing us to test new models or rules without impacting all users.

- **A/B Testing Framework:** We will implement an A/B testing mechanism, likely using **feature flags** within the serving layer or the e-commerce application logic. This allows us to expose different pricing algorithms (e.g., a new dynamic pricing model vs. the existing rule-based system) or shipping logic (e.g., different free shipping thresholds) to specific user segments.
- **Detailed Logging:** Comprehensive logs will be captured for every pricing and shipping estimation request and response. These logs will include:
    - Input parameters (SKU, customer location, etc.)
    - Output (suggested price, estimated shipping cost)
    - The specific model/rule version used

    - Actual customer behavior (add-to-cart, conversion, abandonment rate) This detailed logging, stored in the Silver layer, forms the basis for evaluating experiment outcomes and model performance.

- **MLflow Integration:** For dynamic pricing models, **MLflow** will be used for experiment tracking. This enables us to compare different model runs, track hyperparameters, and monitor performance metrics. It will also serve as a model registry for versioning and managing deployed models.

5. Serving Layer for Real-time or Scheduled Access to Outputs

Our serving layer makes the calculated dynamic prices and shipping estimates available to the e-commerce platform and other consumers.

- **Real-time Serving (API):**
    - **Purpose:** Provide instant pricing and shipping estimates when a user views a product or proceeds to checkout.
    - **Tools: Databricks Model Serving Endpoints** will host our trained dynamic pricing and shipping estimation models for low-latency inference. An **API Gateway** (e.g., Azure API Management) will secure and route these requests.
- **Batch/Scheduled Serving:**
    - **Purpose:** For pre-calculating prices or generating shipping rate cards that don't require immediate updates.
    - **Tools:** Scheduled **Databricks Jobs** will write computed results to optimized **Delta Lake tables** in the Gold layer, which the e-commerce platform can then consume directly or via an operational database.

6. Monitoring and Alerting Setup for Reliability and Quality

We'll establish monitoring and alerting for system health, data quality, and model performance.

- **Data Pipeline Monitoring:** Track ingestion and data quality (e.g., schema drift) using **Databricks Monitoring** and cloud tools like Azure Monitor/AWS CloudWatch.
- **Model Performance Monitoring:** Observe dynamic pricing and shipping model accuracy using **MLflow** and custom dashboards.
- **System Health Monitoring:** Monitor infrastructure (Databricks clusters, API latency) with cloud-native tools.
- **Alerting:** Automated alerts (e.g., via PagerDuty) will notify teams of critical issues or deviations.

7. Where and How Databricks is Integrated

Databricks is central to our design: used for **ETL and data transformation** (notebooks, Spark SQL/PySpark), as the core of **Delta Lake storage**, for **MLflow model management**, **job orchestration**, and **real-time model serving**.

8. Reliability and Scalability Considerations

We ensure reliability and scalability via **cloud-managed services** (Databricks, Kafka), **Delta Lake's ACID properties**, **Spark's horizontal scaling**, and a **modular design**.

9. Where you would Leverage AI-Assisted Development Tools

AI/ML tools enable building core models for **Dynamic Product Pricing** (optimizing prices using historical data, market trends) and **Shipping Price Estimation** (refining estimates, anomaly detection). **MLflow** manages these AI models within the data pipeline.

10. What you would do Differently with More Time

With more time, I'd conduct **deeper research on requirements** to deliver a more effective and detailed solution. This would include exploring **advanced ML models** (e.g., reinforcement learning for pricing), implementing **direct carrier API integrations** for shipping, and building **predictive demand forecasting**.

11. Any Trade-offs You Made and Reasoning

- **Trade-off: Delta Lake across all layers.**
    - **Reasoning:** Minor ingestion overhead balanced by significant gains in data reliability, integrity, and simplified architecture.