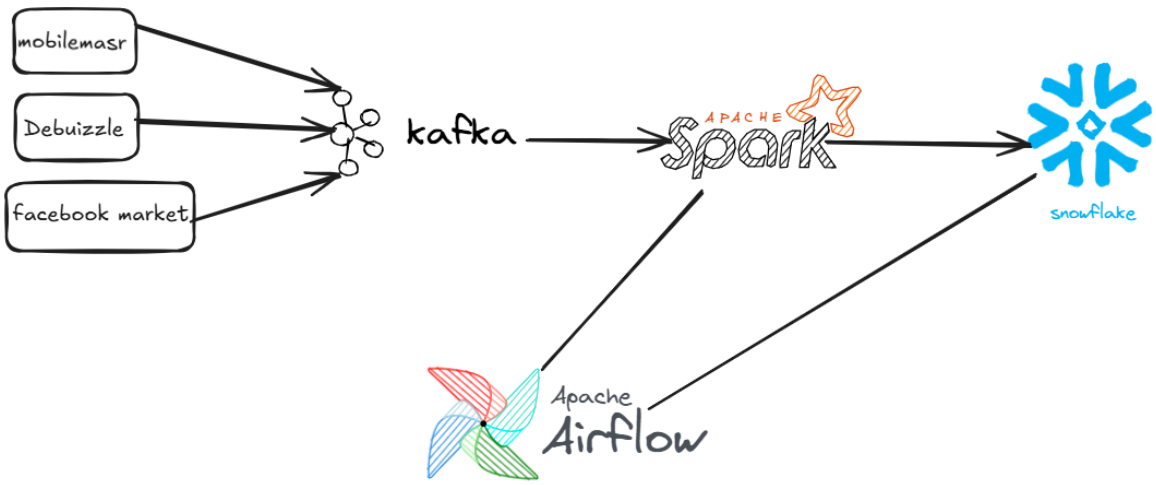
Streaming Architecture Diagram



---

1. Why each component was chosen:

- **Kafka:** Provides fast and reliable ingestion from multiple sources including web scraping, internal sales transactions, app logs, AI engine outputs, and manual market data. It decouples sources from consumers and ensures no data loss.
- **Spark Streaming:** Ideal for real-time streaming data; it cleans, transforms, and standardizes incoming data, making it analytics-ready. Spark also supports batch processing for historical or reprocessing needs.
- **Snowflake:** Columnar, scalable, and highly organized storage suitable for analytics. Works efficiently with Spark for both real-time and batch loads. Supports schema evolution and partitioning for performance.
- **Airflow:** Orchestrates tasks, monitors job execution, handles retries on failures, and ensures dependencies between tasks are respected (e.g., Spark jobs run after Kafka ingestion, Snowflake load runs after Spark processing).

---

2. How real-time and batch data are handled:

- **Real-time:**
  - New data from all sources enters **Kafka topics** continuously.
  - **Spark Streaming** consumes data from Kafka in near real-time, applies **cleaning, validation, and transformation**, and generates analytics-ready datasets.

    - Processed data is then loaded into **Snowflake** for immediate analysis and reporting.
- **Batch:**
    - Historical data or reprocessing tasks can be handled using **Spark batch jobs** on stored data (either in Kafka staging or intermediate storage).
    - Batch jobs apply the same transformations and validations, then load the results into **Snowflake**, optionally using **incremental load** to avoid overwriting unchanged data.

---

3. How data quality and schema evolution are managed:

- **Data quality:**
    - Spark validates incoming data before loading, including checking for missing values, data type correctness, duplicates, and out-of-range or inconsistent values.
    - Data cleansing steps like standardizing dates, trimming text, and correcting anomalies ensure high-quality datasets.
    - Airflow monitors job execution and can trigger alerts for any failed validations or rejected records.
- **Schema evolution:**
    - Snowflake allows adding or modifying columns without breaking existing analyses.
    - Spark handles schema changes by mapping new fields and providing **default values** for older records.
    - Versioning and audit logs help track schema changes and ensure lineage of data transformations.

---

4. Partitioning and performance considerations in the warehouse:

- **Partitioning:** Data can be partitioned by date, source, or other relevant dimensions to speed up queries and reduce scanning overhead.
- **Performance:** Snowflake's **clustering keys, indexing, and columnar storage** improve query performance, especially for large datasets.
- Efficient incremental loading and optimized transformations in Spark reduce unnecessary processing and enhance overall pipeline performance.

---

3. Schema Design