Proposed Solution

1. Platform Architecture

- **Databricks** will act as the central tool for:
    - Data ingestion
    - Transformation
    - Orchestration
    - Sharing data for reporting and machine learning
- Integration with AWS services:
    - **Amazon S3**: Central data lake for raw, processed, and curated data.
    - **Amazon Kinesis/ DLT**: For real-time data ingestion from microservices and CDC pipelines.
    - **AWS Lambda**: Trigger-based processing for lightweight ETL tasks or pipeline orchestration.
    - **Amazon Redshift**: Optional for serving data marts for analysts.

2. Pipeline Development in Databricks

- Leverage **Delta Lake** as the storage format for ingesting and managing datasets on S3.
- Define modular **notebooks** for:
    1. Real-time data ingestion (Kinesis).
    2. Batch data ingestion (SFTP, scheduled exports).
    3. Transformation workflows (Delta Tables, SQL transformations).
    4. Data export pipelines (curated data to Redshift or downstream APIs).
    5. Connection to RDBMS sources for read and write.
- Use **Databricks Workflows** to orchestrate these notebooks into end-to-end pipelines.
- Monitor with **Databricks Job API**, for logging and alerts.

Implementation Components

A. Ingestion Tools

1. **Real-time Data**:
    - Use **Kinesis Data Streams** integrated with Databricks for ingestion.
    - Leverage structured streaming in Databricks to process real-time data using DLT, delta live tables.
2. **Batch Data**:

  - Configure scheduled Databricks jobs to pull data from:
    - SFTP sources connection.
    - Relational databases using JDBC drivers.
    - Cloud exports stored on S3.

B. Transformation & Enrichment

- Use Delta Lake to store data incrementally with ACID transactions.
- Standardize transformation logic in PySpark or SQL within Databricks notebooks.
- Automate schema evolution with Delta Lake for dynamic data sources.

C. Data Orchestration

- Use **Databricks Workflows** with triggers from Databricks Workflows Scheduler.

D. Data Sharing & Access

- Provide analysts and data engineers access to transformed datasets using:
  - SQL Analytics in Databricks.

Automation and DevOps

1. **Infrastructure as Code**:
   - Use **Terraform** for provisioning AWS resources and Databricks clusters, jobs, and workflows.
   - Maintain reproducible configurations for environments (dev, staging, prod).
2. **CI/CD Pipelines**:
   - Implement CI/CD pipelines using tools like GitHub Actions:
     - Automated deployment of Databricks notebooks and workflows.
     - Test transformations and data quality using Great Expectations.
3. **Version Control**:
   - Use Git for managing notebook versions and pipeline code.

Advantages of Using Databricks with AWS

1. **Scalability**: Seamless scaling for batch and streaming workloads.
2. **Cost Efficiency**: Use Spot Instances for transient clusters.
3. **Unified Workspace**: Combine data engineering, analytics, and machine learning workflows.
4. **Open-Source Compatibility**: Support for Delta Lake, Spark, and MLflow.