1-Month Expert Architecture Training Plan

Week 1, Day 3: Real-Time Ingestion and Reliable Storage

Prepared by: Sami Mbarki
Solution Architect, Java Expert
ISYGO Consulting Services 18 September 2025

ISYGO Consulting Services
Delivering Advanced Architectural Training for Enterprise Solutions

Document ID: d3d09d59-7b6d-4ba2-b4de-7cbc28a8b12b
Version: 1.0
Confidential: For Internal Training Use Only

1-Month Expert Training in Microservices and Polyglot Persistence

Sami Mbarki
Solution Architect, Java Expert
ISYGO Consulting Services

18 September 2025

Contents

1 Preface: Streaming and Storage in Big Data

This detailed chapter turns Day 3 into a student textbook, with definitions (e.g., "stream processing"), terminologies (e.g., "micro-batch"), clarifications on concepts like watermarking, technologies (Spark, Delta Lake), and alternatives (Flink, Iceberg). Include exercises to apply ideas.

1.1 How to Use This Book

- Use equations for performance calculations. - Explore alternatives for informed choices.

2 Introduction: Processing Data in Motion

Why process data in real-time rather than batches? Structured Streaming in Spark enables continuous applications. Historical: Spark started as batch (2010), evolved to streaming with DStreams (2013), then Structured (2016). Alternatives: Apache Flink for true continuous processing.

2.1 Learning Objectives

Including sub-objectives like calculating watermark thresholds.

3 Core Theory: Structured Streaming

3.1 Evolution from DStreams

Definition: DStreams are discretized streams of RDDs. Clarification: Structured Streaming uses DataFrames for unified batch/streaming.

3.2 Micro-Batch Execution Model

Definition: Processes data in small batches at triggers.

Terminology: "Unbounded table" - logical view of growing data.

Clarification: Trigger types - processing time, once.

Equation: Batch size trigger interval * ingress rate.

Alternatives: Flink's windowing for lower latency.

Pitfalls: Long micro-batches increase latency; tune triggers.

Code Example:

3.3 Watermarking for Late Data

Definition: Watermark is a threshold for late events, based on event time.

Terminology: "Event time" vs "processing time".

Clarification: Watermark = $\max_e vent_t ime - delay_t hreshold$.

Equation: Drop if event$_time < watermark$.

Example: .withWatermark("timestamp", "10 minutes")

Advanced: Handling multiple watermarks in joins.

Pitfalls: Too aggressive watermark drops valid data.

4 The Medallion Architecture

Definition: Layered data refinement - Bronze (raw, Parquet), Silver (cleansed, validated), Gold (aggregated, business-ready).

Terminology: "Data lakehouse" - lake storage with warehouse features.

Clarification: Bronze for ingestion, Silver for cleaning (e.g., dedup), Gold for analytics.

Alternatives: Use Apache Iceberg for schema evolution over Delta Lake.

Diagram:

Figure 1: Layers with Data Flow and Transformations

Best Practices: Use Delta Lake for ACID in lakehouse.

Case Study: Databricks' use in healthcare for compliant data.

5 Delta Lake: Reliable Storage Layer

Definition: Open-source storage adding transaction log to Parquet.

Terminology: "ACID" - Atomicity, Consistency, Isolation, Durability.

Clarification: Time travel via versions; schema enforcement prevents bad data.

Features: - Transactions: Concurrent writes. - Upserts: MERGE INTO. - Deletion: VACUUM for GDPR.

Alternatives: Apache Hudi for upsert-heavy workloads, Iceberg for multi-engine support.

Code:

Pitfalls: Ignoring optimization (Z-Ordering) leads to slow queries.

6 Spark Optimization Techniques

6.1 Data Partitioning

Definition: Divides data into directories by column values.

Terminology: "Partition pruning" - skips irrelevant partitions.

Clarification: Predicate pushdown optimizes filters.

Equation: Query time data scanned; pruning reduces by factor of partitions.

Code: .partitionBy("tenant$_i d$")

Alternatives: Bucketing for joins.

6.2 Caching and Persistence

Definition: Caches DataFrames in memory/disk.

Terminology: "Persistence levels" - MEMORY$_O NLY, DISK_O NLY$.

Clarification: Use .cache() for repeated access.

Pitfalls: Caching large data causes OOM; use .persist(StorageLevel.MEMORY$_A ND_D ISK$).

Advanced: Checkpointing for fault recovery in streaming.

7 Lab: Building a Real-Time Pipeline

Detailed with basic/advanced tracks, code templates, troubleshooting.

8 Wrap-Up: Socratic Discussion

Discuss: How does Delta Lake solve data lake problems? What alternative to Spark for streaming?

9 Student Exercises and Review Questions

1. Implement a watermark in code. 2. Compare Delta Lake and Iceberg.

10 Glossary

Expanded:

- **Micro-Batch**: Discrete processing interval.
- **Watermark**: Late data threshold.
- **ACID Transactions**: Reliability guarantees.
- **Predicate Pushdown**: Filter optimization.
- **Z-Ordering**: Multi-dimensional clustering.

11 References and Further Reading

Books:

- Zaharia, Matei, et al. *Learning Spark*. O'Reilly, 2020.
- Armbrust, Michael, et al. *Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores*. VLDB 2020.

Online:

- Spark Docs: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html.
- Delta Lake: https://delta.io.