Resource Book

Site:	AmaliTech LMS
Course:	DEM10: Kafka, Streaming, and Event-Driven
Book:	Resource Book

Printed by:	Ernest Kwisanga
Date:	Saturday, 14 February 2026, 7:09 PM

Table of contents

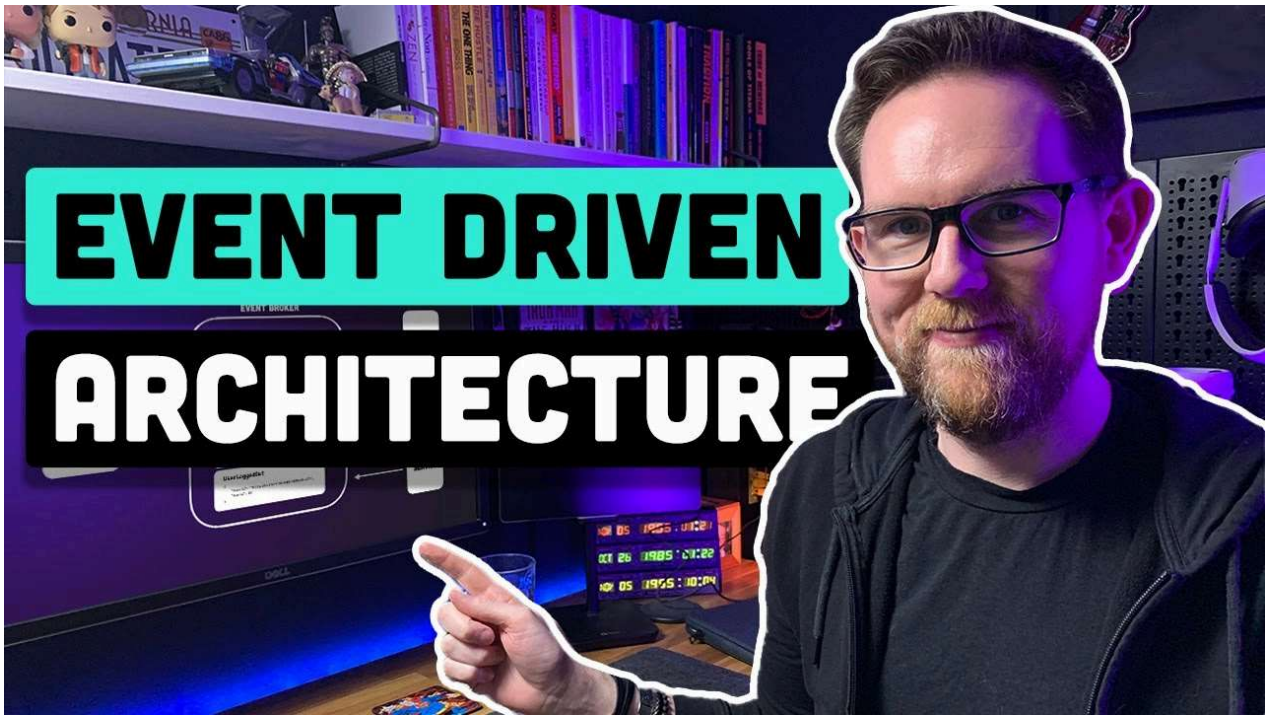
1. 1. Apache Kafka Deep Dive – Core Concepts & Setup ⚙️
2. 2. Kafka Producers & Consumers – Python Integration
3. 3. Kafka Stream Processing – PySpark Structured Streaming
4. 4. Kafka + Data Engineering Workflows 🔗
5. Discussion Prompt

1. 1. Apache Kafka Deep Dive – Core Concepts & Setup

Understand the fundamentals of event-driven systems and the core architecture of Apache Kafka.

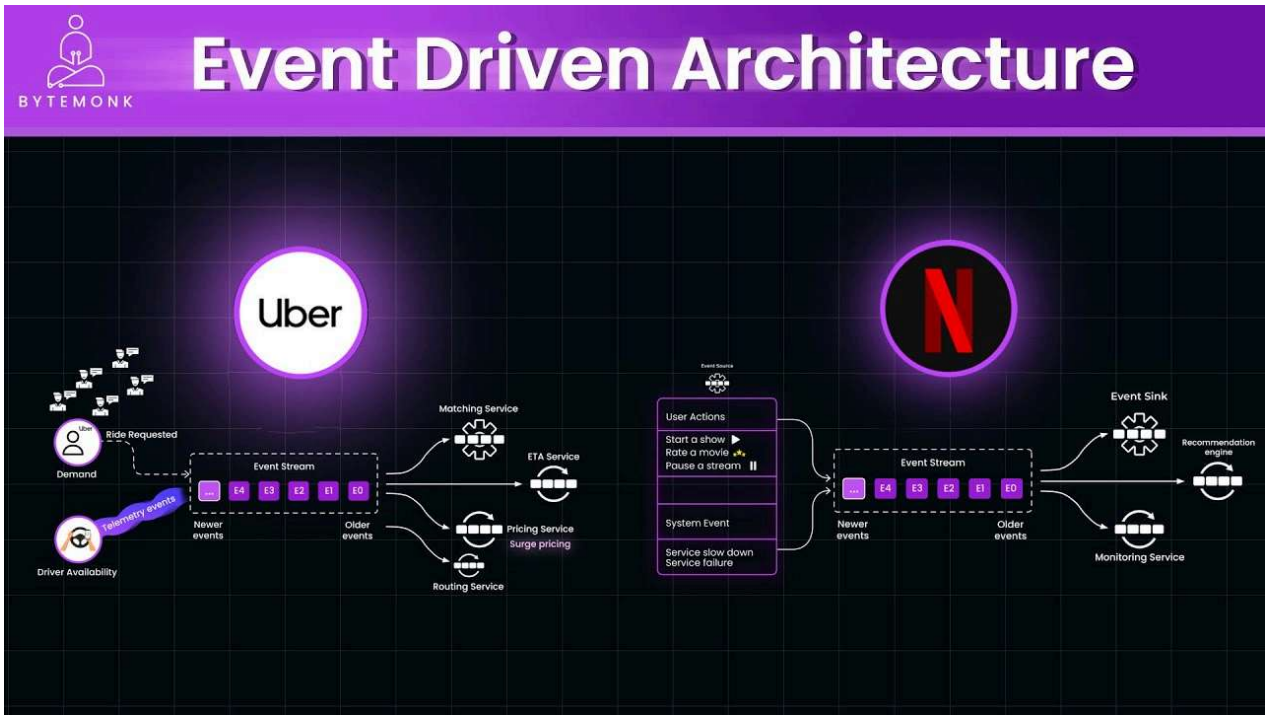
Intro to Event-Driven Architecture:

- **Articles:**
  - **Event-Driven Architecture Explained:** Introduces the core concepts and patterns of event-driven systems. link
  - **Benefits and Challenges of Event-Driven Systems:** Discusses the advantages and potential difficulties of implementing event-driven architectures. link
  - **Microservices and Event-Driven Architecture:** Explores the relationship between microservice design and event-driven patterns. link
- **Videos:**
  - **Event-Driven Architecture Fundamentals (Playlist):** Covers the basic principles and benefits of event-driven design.

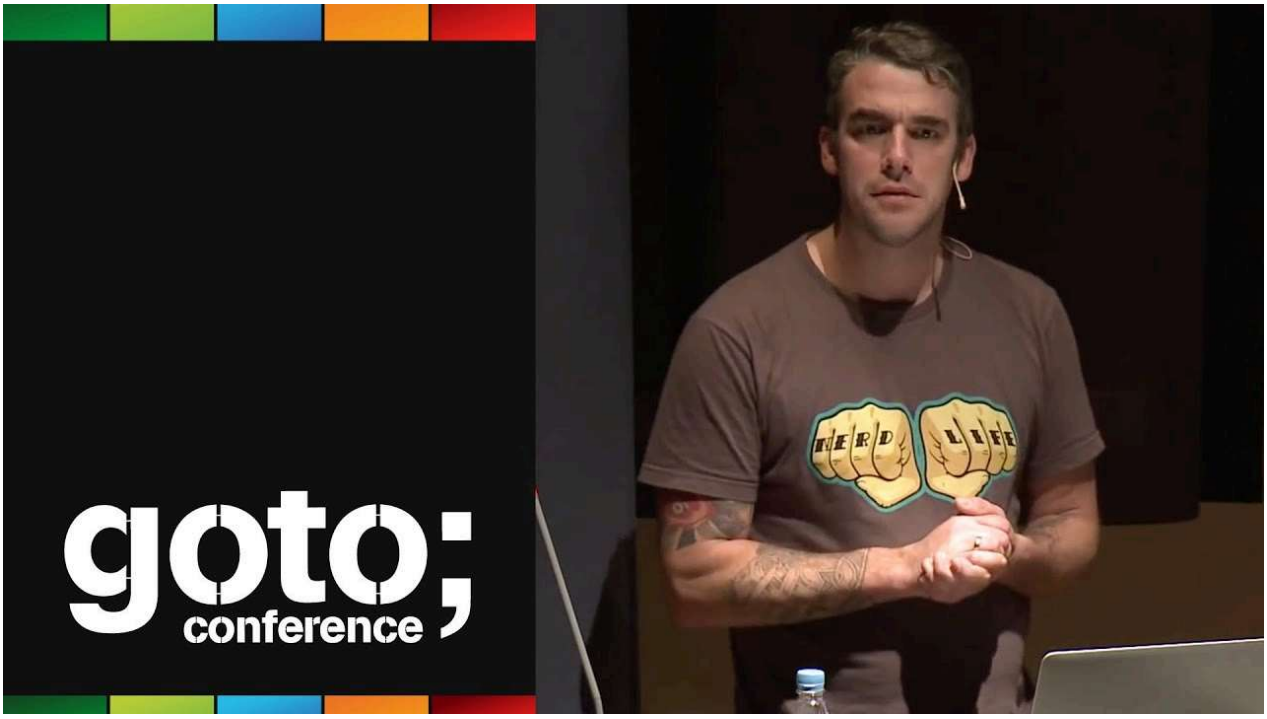


,

- **Building Event-Driven Microservices (10 min):** Focuses on implementing microservices using event-driven communication.



- **Event Sourcing Patterns (40 min):** Explains the event sourcing pattern often used in event-driven systems.

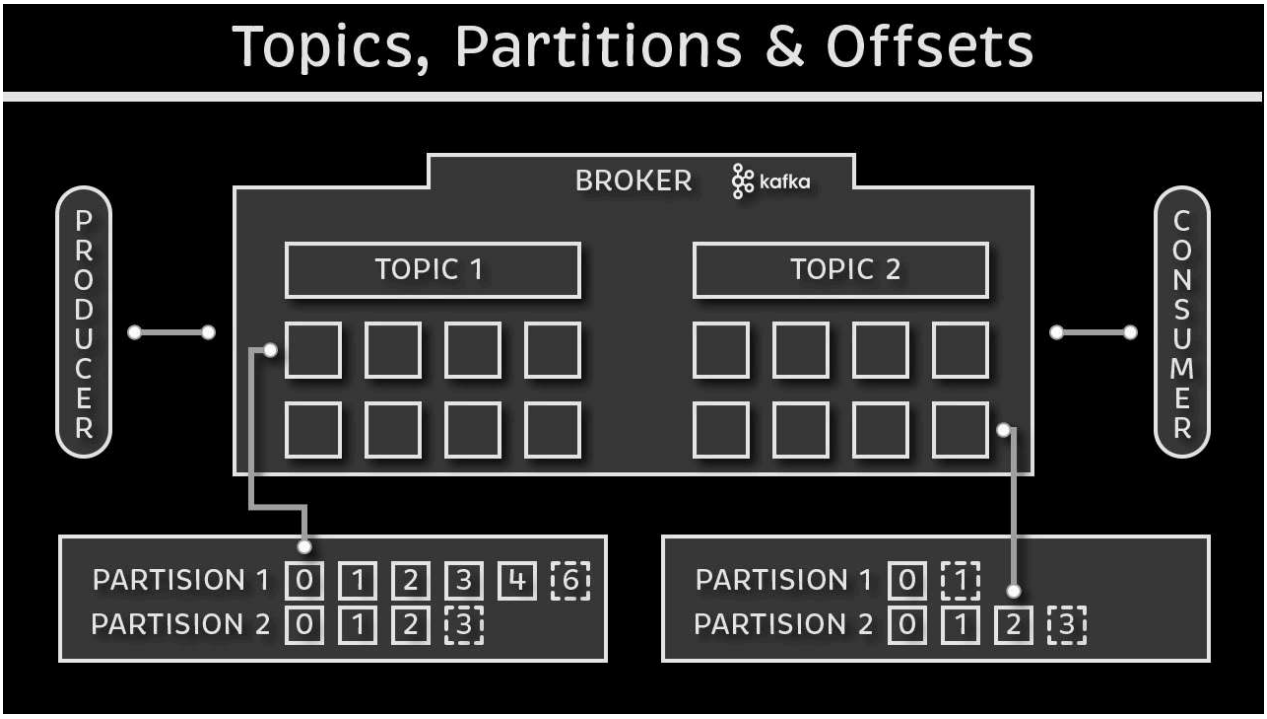


Kafka Internals: Topics, Brokers, Partitions:

- **Articles:**
  - **Kafka Architecture Deep Dive:** Official documentation detailing Kafka's core architectural components. link *(Link to general documentation)*
  - **Understanding Kafka Topics and Partitions:** Explains how data is organized into topics and distributed across partitions for scalability. link
  - **Kafka Replication and Durability:** Discusses how Kafka achieves fault tolerance and data durability through replication. link *(Link to general documentation)*
- **Videos:**
  - **Kafka Architecture Explained (25 min):** A video walkthrough of Kafka's key components and how they interact.

- **Kafka Internals Deep Dive (45 min):** A more detailed exploration of Kafka's internal workings.

- **Partition Strategy Design (15 min):** Discusses strategies for partitioning Kafka topics effectively.


Topics, Partitions & Offsets
BROKER
kafka
PRODUCER
CONSUMER
TOPIC 1
TOPIC 2
PARTISION 1 0 1 2 3 4 6
PARTISION 2 0 1 2 3
PARTISION 1 0 1
PARTISION 2 0 1 2 3

2. 2. Kafka Producers & Consumers – Python Integration

Learn how to send data to (produce) and read data from (consume) Kafka topics using Python.

Producer and Consumer Configs:

- **Articles:**
  - **Kafka Producer Configuration:** Official documentation covering configuration options for Kafka producers. link *(Link to general documentation)*
  - **Kafka Consumer Configuration:** Official documentation covering configuration options for Kafka consumers and consumer groups. link *(Link to general documentation)*
  - **Producer and Consumer Best Practices:** Provides tips and recommended practices for building reliable producers and consumers. link 1, link 2
- **Videos:**
  - **Kafka Producer Deep Dive (20 min):** Focuses on the details of configuring and using Kafka producers.
  - **Kafka Consumer Patterns (6 mins):** Explains common patterns for consuming data from Kafka topics.

- **Performance Tuning Producers/Consumers (35 min):** Discusses strategies for optimizing the performance of Kafka producers and consumers.

Python Integration with kafka-python:

- **Articles:**
  - **kafka-python Library Guide:** Documentation for the popular `kafka-python` library. link
  - **Building Kafka Applications in Python:** A guide to getting started with Kafka and Python using Confluent's library. link *(Note: Uses confluent-kafka-python, another popular library)*

- **Videos:**
    - **Python Kafka Tutorial (Playlist - 50 min total):** A tutorial series demonstrating how to use Python with Kafka.
    - **Use case (85 min):** A longer video potentially showcasing a real-world use case involving Python and Kafka.

3. 3. Kafka Stream Processing – PySpark Structured Streaming ⚡

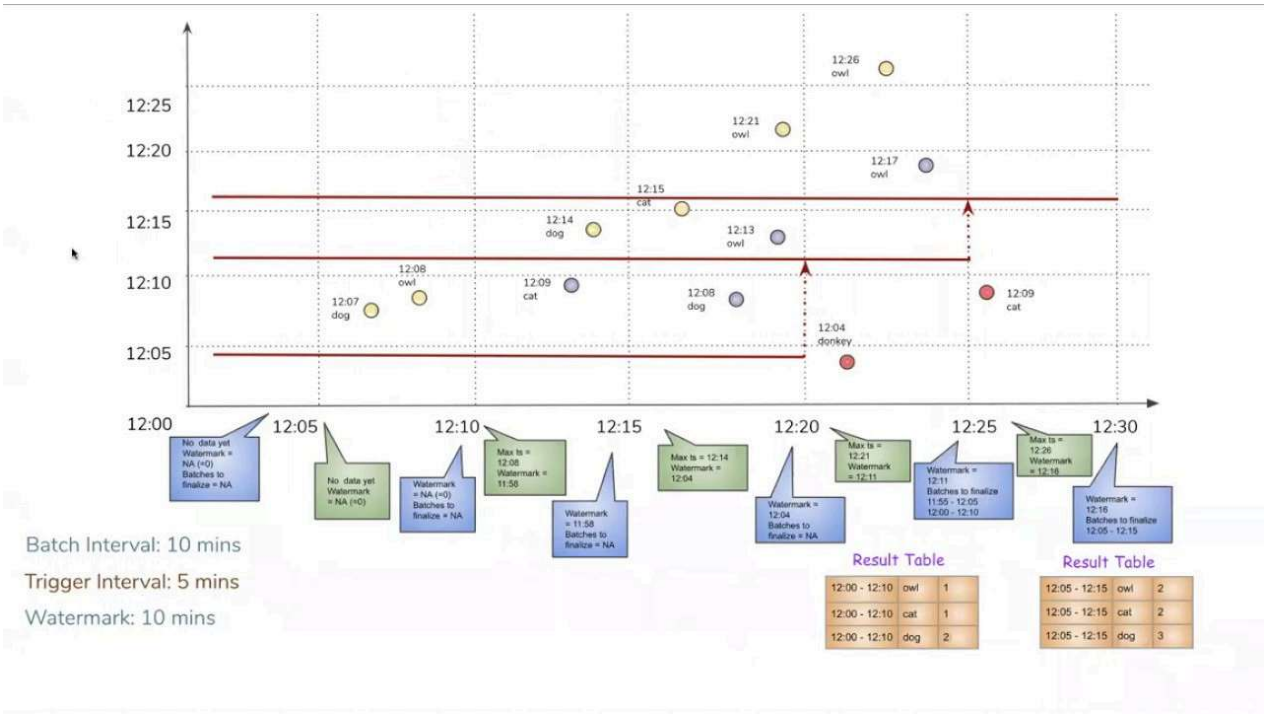
Integrate Kafka with Apache Spark for powerful, scalable real-time data processing.

PySpark Structured Streaming Primer:

- **Articles:**
  - **Structured Streaming + Kafka Integration Guide:** Official Spark documentation on reading from and writing to Kafka using Structured Streaming. link
  - **Stream Processing with Kafka and Spark:** Blog post explaining the integration and benefits. link
  - **Watermarking and Late Data:** Revisits the concept of watermarking for handling event time and late data, crucial in Kafka streams. link
- **Videos:**
  - **Kafka + Spark Streaming (60 min):** A tutorial demonstrating the integration of Kafka and Spark Structured Streaming.

  - **Real-time Analytics Pipeline (Playlist - 120 min total):** A series likely covering the build-out of a real-time pipeline using Spark and potentially Kafka.

- **Watermarking Strategies (20 min):** Focuses specifically on implementing watermarking in Spark Structured Streaming.



4. 4. Kafka + Data Engineering Workflows 🔗

Explore how Kafka fits into broader data engineering pipelines using tools like Kafka Connect.

Kafka Connect and Sinks Overview:

- **Articles:**
  - **Kafka Connect Architecture:** Confluent documentation explaining Kafka Connect for streaming data between Kafka and other systems. link
  - **Building Custom Kafka Connectors:** Guide on developing custom connectors if pre-built ones aren't available. link
- **Videos:**
  - **Kafka Connect Deep Dive (40 min):** A detailed video explaining Kafka Connect concepts and usage.

  - **Connect (Kafka Summit Talk):** A presentation likely covering Kafka Connect features and use cases.


Kafka Connect
Data Source
Kafka Connect
Cluster
KAFKA 101

5. Discussion Prompt

Event-driven architecture decouples services by having them react to events (messages) rather than making direct requests to each other. How does Apache Kafka facilitate this decoupling, and what are the main advantages of this approach compared to traditional request/response architectures?