ECE 5984
ZOOM

30 August 2023

Pipelines and Apache Kafka

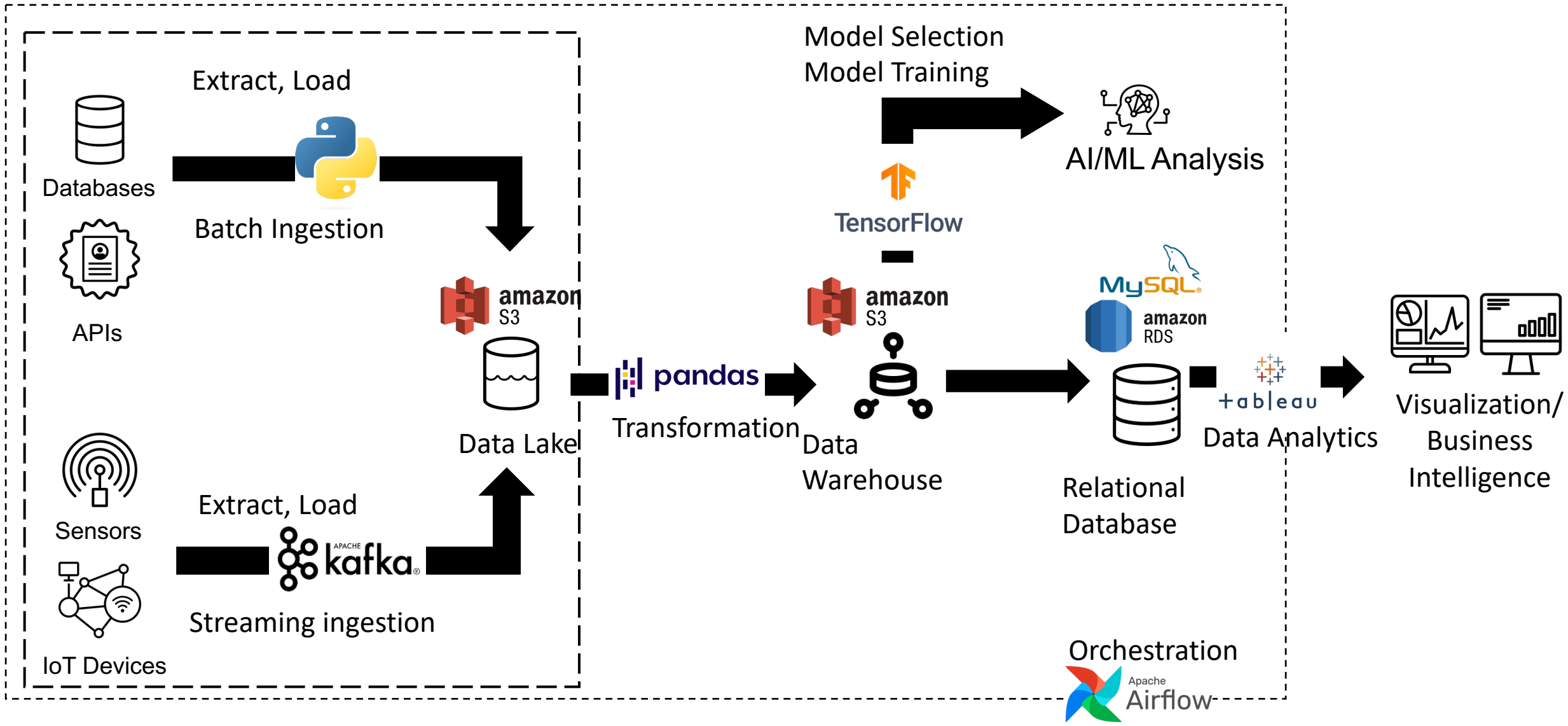
Nektaria Tryfona, PhD
Electrical and Computer Engineering
Virginia Tech

VIRGINIA TECH

Outline

- Apache Kafka introduction
- Apache Kafka as a Message system
  - Messaging system
- How does it work?
- The role of Zookeeper

Custom Data Engineering Pipeline



Introduction to Apache Kafka

An event streaming platform

- Usage
  - to build real-time streaming data pipelines and applications that adapt to the data streams
- Distributed
  - relies on multiple servers
  - data is replicated over multiple locations
    - if some servers fail, we're still fine
- Scalable
  - as many servers as needed
  - start small and add more servers as the system grows
- High-throughput
  - handles trillions of messages per day
  - ending up in petabytes of data persistently stored over disks

…more on Apache Kafka

- Originally developed by **LinkedIn in 2010**, and later it was donated to the Apache Software Foundation
- Written in Scala and Java
- In the year **2011** Kafka was made public
- It is maintained by **Confluent** under Apache Software Foundation

Streaming data

Two main challenges

- How to collect large volume of data
- How to analyze the collected data

To overcome those challenges, we Kafka uses a messaging system

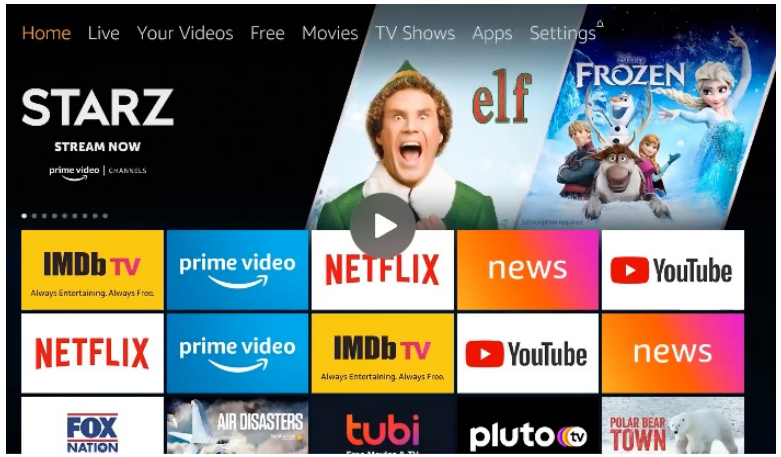
In particular, the publishing & subscribing messaging system

Is this something new?

Publish-and-Subscribe Messaging System

A real-life example is Amazon Fire TV

- publishes different channels like sports, movies, music, etc.,
- anyone can subscribe to channels and get sports, movies, etc whenever their subscribed channels are available



This is based on the publish-and-subscribe messaging system

Publish-and-Subscribe Messaging System

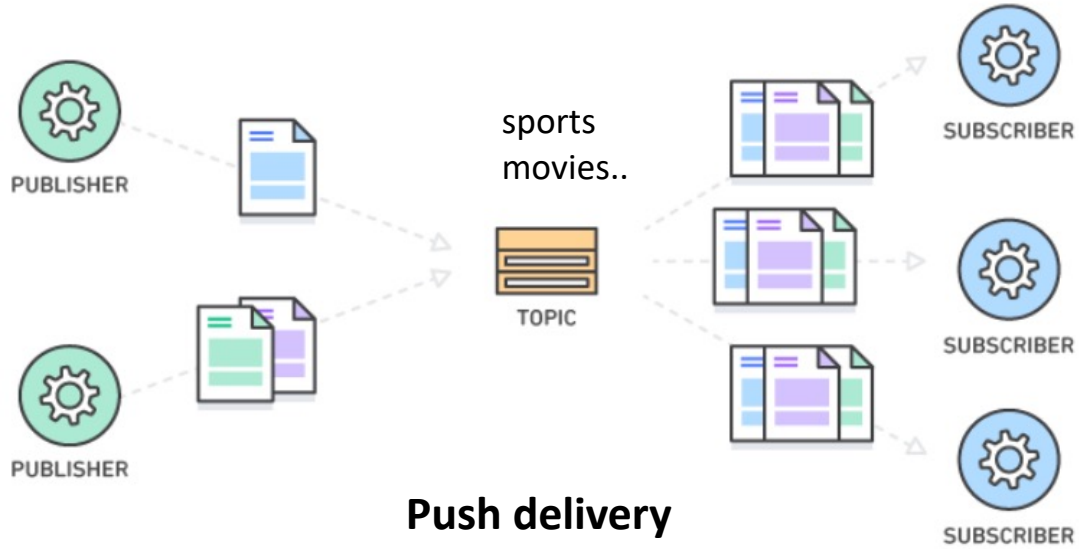
A message is communication data sent from sender to receiver. Message data types can be anything from strings to complex objects representing text, video, sensor data, audio, or other digital content

Every message has a topic associated with it. The system maintains a list of receivers who are interested in messages about that topic

Message producers are called publishers and message consumers are called subscribers

Consumers can subscribe to one or more topics and consume all the messages in that topic

Topics are what tables are in databases



Push delivery

Messages are events

Apache Kafka & the messaging pub/sub system

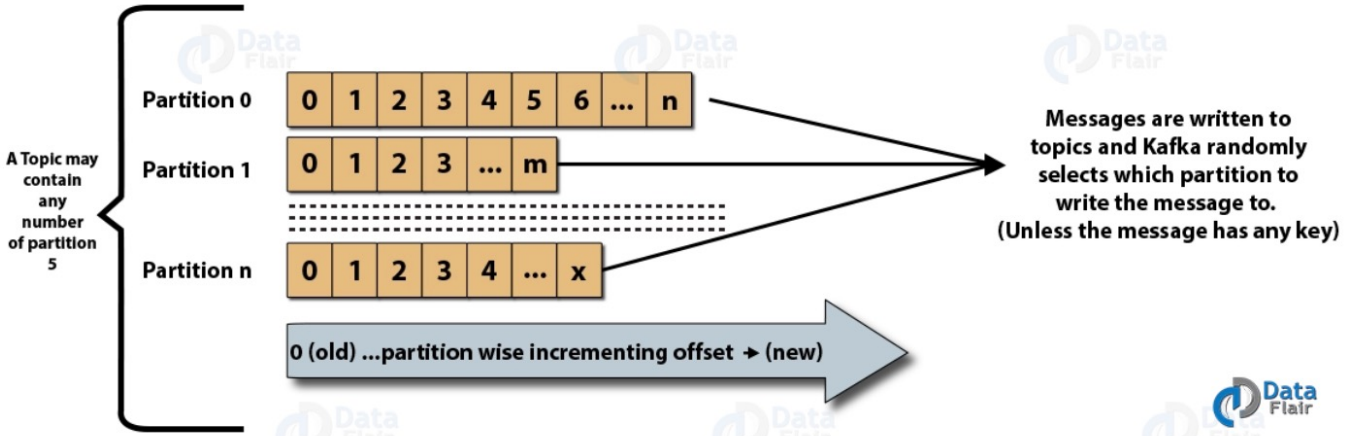
Kafka works by receiving and transmitting data as events, known as messages.

Messages are

- organized into topics
- published by data producers
- consumed by subscribers to these topics

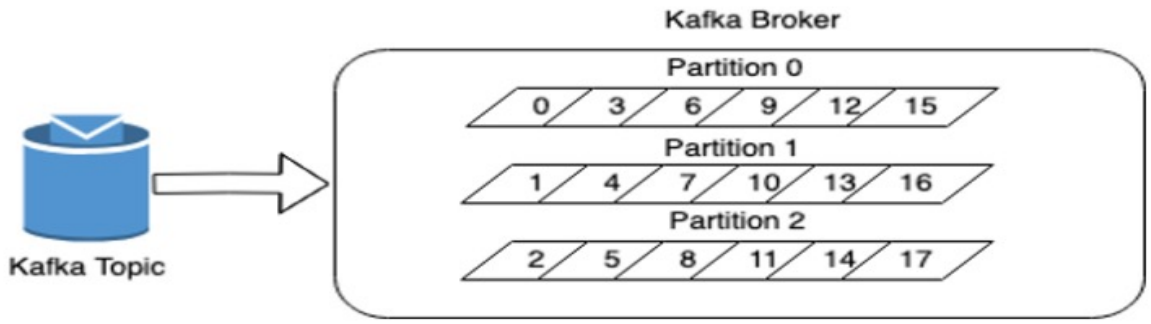
This structure is based on the principle of 'publishing' & 'subscribing'

Kafka Topic & Partitions



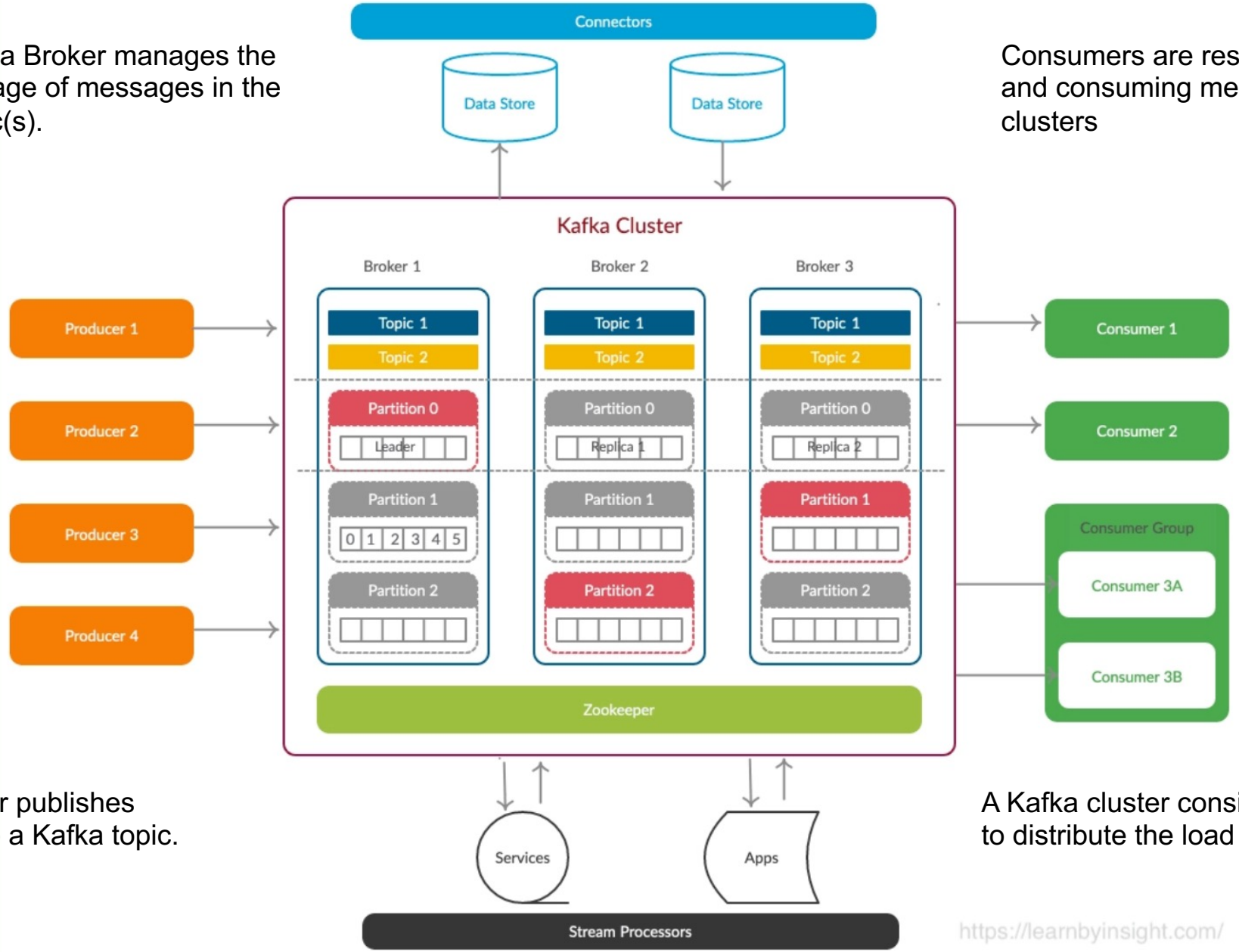
- Kafka Broker manages the storage of messages in the topic(s)
- If Kafka has more than one broker, that is what we call a Kafka cluster

Single Broker With Multiple Partitions



Kafka Broker manages the storage of messages in the topic(s).

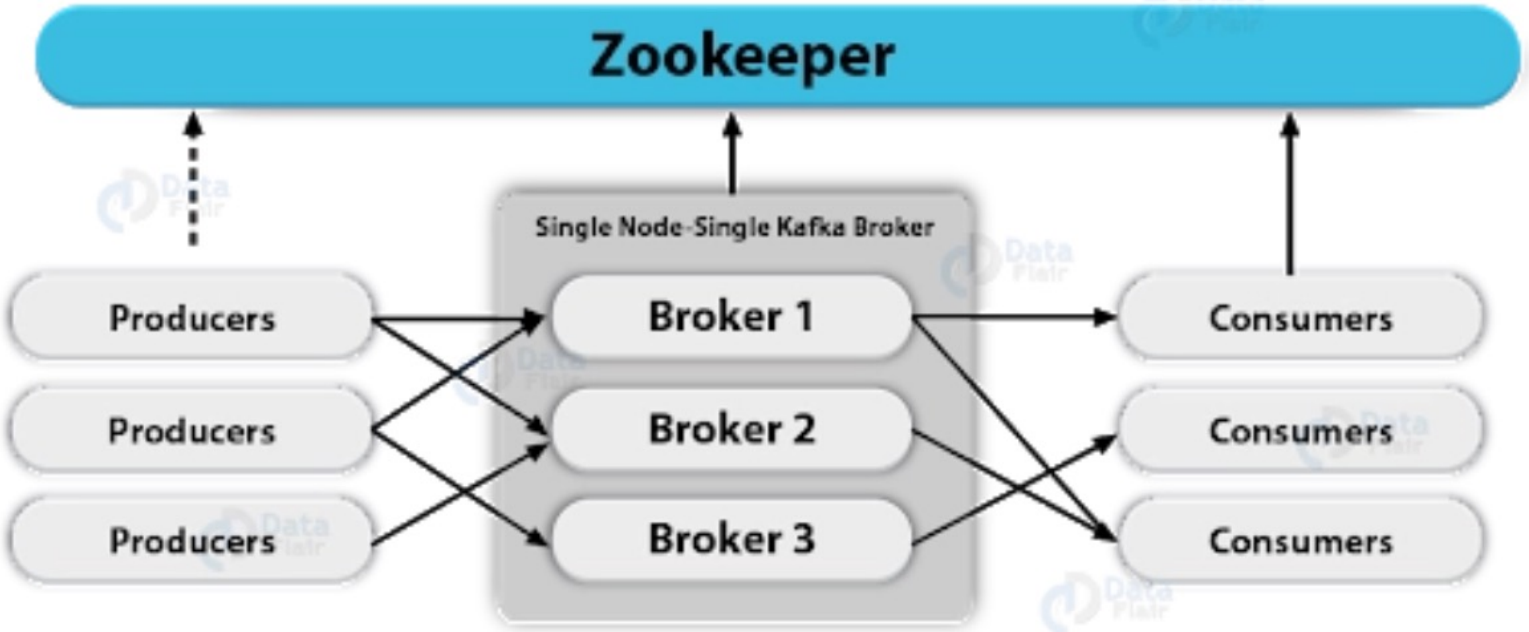
Consumers are responsible for reading and consuming messages from Kafka clusters



The producer publishes messages to a Kafka topic.

A Kafka cluster consists of multiple brokers to distribute the load and ensure availability.

One more component: Zookeeper



Kafka Components

Topics and Partitions

Kafka Producer

It publishes messages to a Kafka topic. The producer is responsible for choosing which record to assign to which partition within the topic.

Kafka Consumer

This component subscribes to a topic(s), reads and processes messages from the topic(s).

Kafka Broker

Kafka Broker manages the storage of messages in the topic(s). If Kafka has more than one broker, that is what we call a Kafka cluster.

Kafka Zookeeper

To offer the brokers with metadata about the processes running in the system and to facilitate health checking and managing and coordinating, Kafka uses Kafka zookeeper.

Apache Kafka Workflow

Following is the step-wise workflow of the Pub-Sub Messaging

- Producers send message to a topic at regular intervals
- Kafka broker stores all messages in the partitions configured for that particular topic. It ensures the messages are equally shared between partitions. If the producer sends two messages and there are two partitions, Kafka will store one message in the first partition and the second message in the second partition
- Consumer subscribes to a specific topic
- Once the consumer subscribes to a topic, Kafka will provide the current offset of the topic to the consumer and also saves the offset in the Zookeeper
- Consumer will request the Kafka in a regular interval (like 100 Ms) for new messages
- Once Kafka receives the messages from producers, it forwards these messages to the consumers
- Consumer will receive the message and process it
- Once the messages are processed, consumer will send an acknowledgement to the Kafka broker
- Once Kafka receives an acknowledgment, it changes the offset to the new value and updates it in the Zookeeper. Since offsets are maintained in the Zookeeper
- This above flow will repeat until the consumer stops the request
- Consumer has the option to rewind/skip to the desired offset of a topic at any time and read all the subsequent messages

https://www.tutorialspoint.com/apache_kafka/apache_kafka_workflow.htm

Apache Kafka Capabilities

Kafka combines 3 key capabilities for event streaming end-to-end with a single solution:

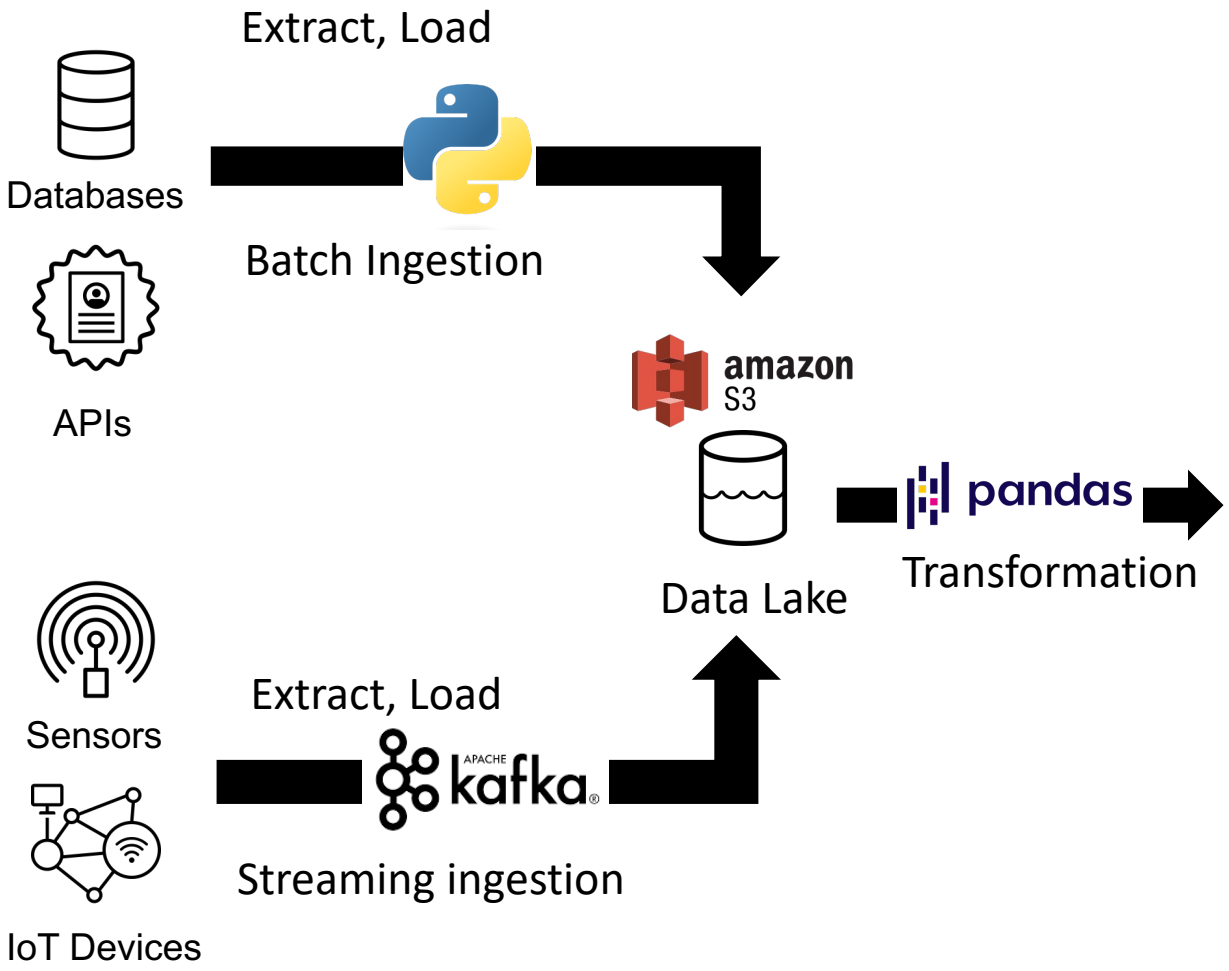
1. To **publish** (write) and **subscribe to** (read) streams of events, including continuous import/export of your data from other systems
2. To **store** streams of events durably and reliably for as long as you want
3. To **process** streams of events as they occur or retrospectively

This is a publish-subscribe messaging system, which allows exchanging of data between applications, servers, and processors as well

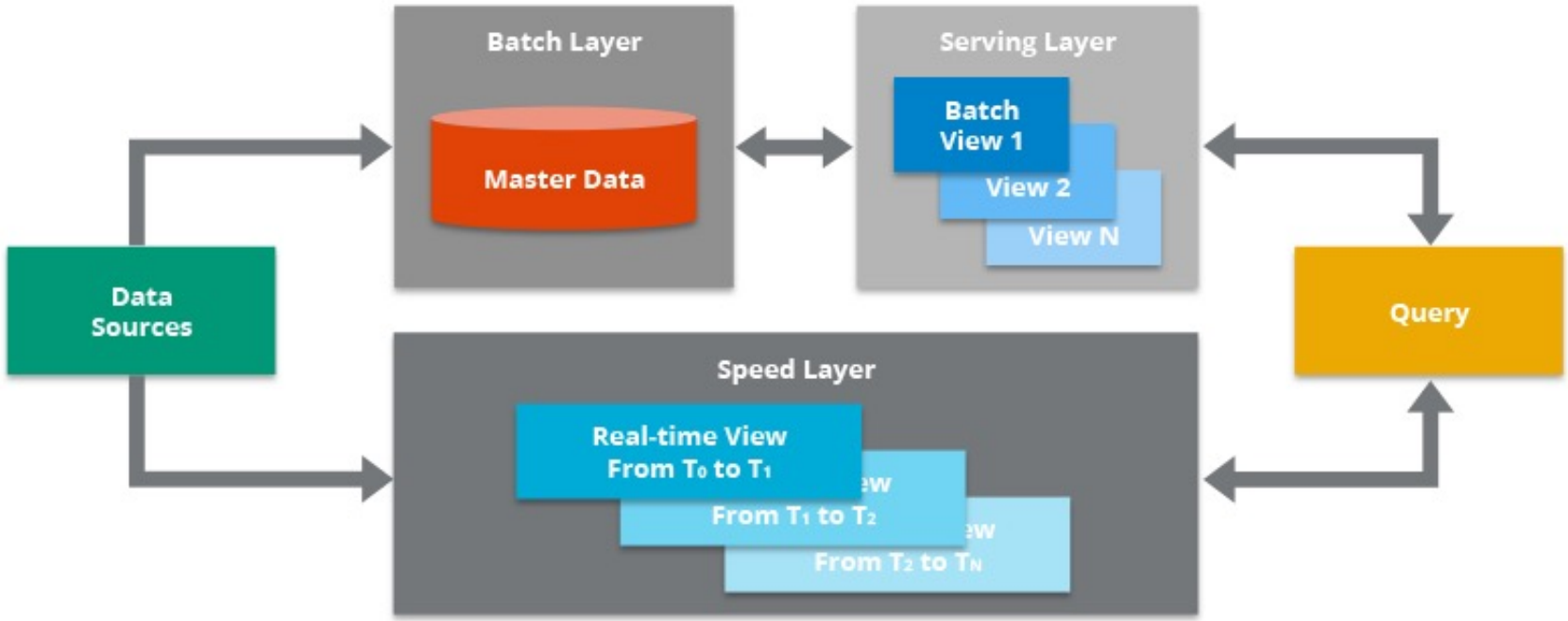
(some of…) Benefits

- Real-time Data Streaming
- Scalability & Fault Tolerance
- Data Integration
- High Throughput & Low Latency
- Durability & Persistence

Is this all? What's next?



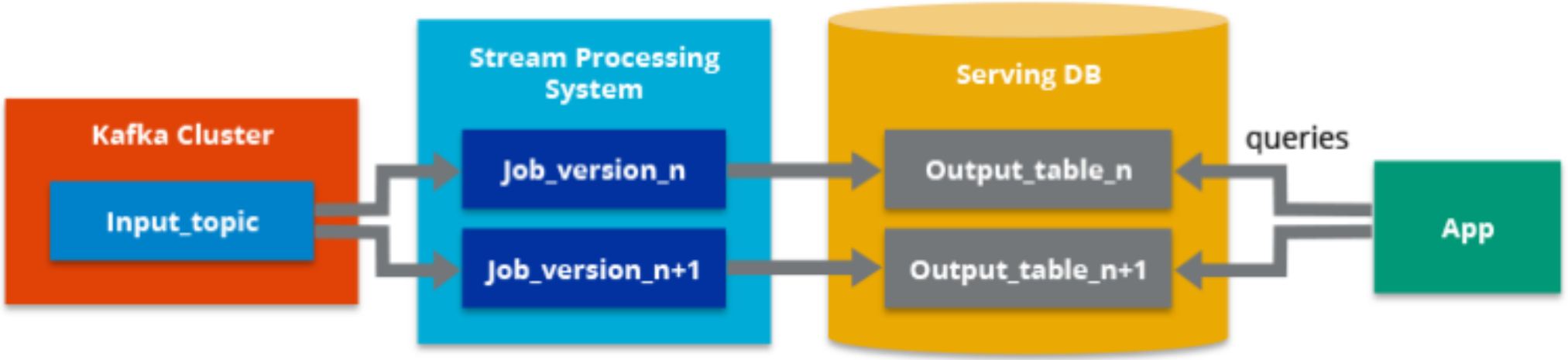
Lambda Architecture



- A deployment model for data processing that combines
  - a traditional batch pipeline
  - with a fast real-time stream pipeline for data access

- A common architecture model in IT and development organizations' toolkits
- Support data-driven and event-driven solutions

Kappa Architecture



- A deployment model that supports (near) real-time analytics when the data is read and transformed immediately after it is inserted into the messaging engine

- All data is treated as if it were a stream
  - the stream processing engine acts as the sole data transformation engine