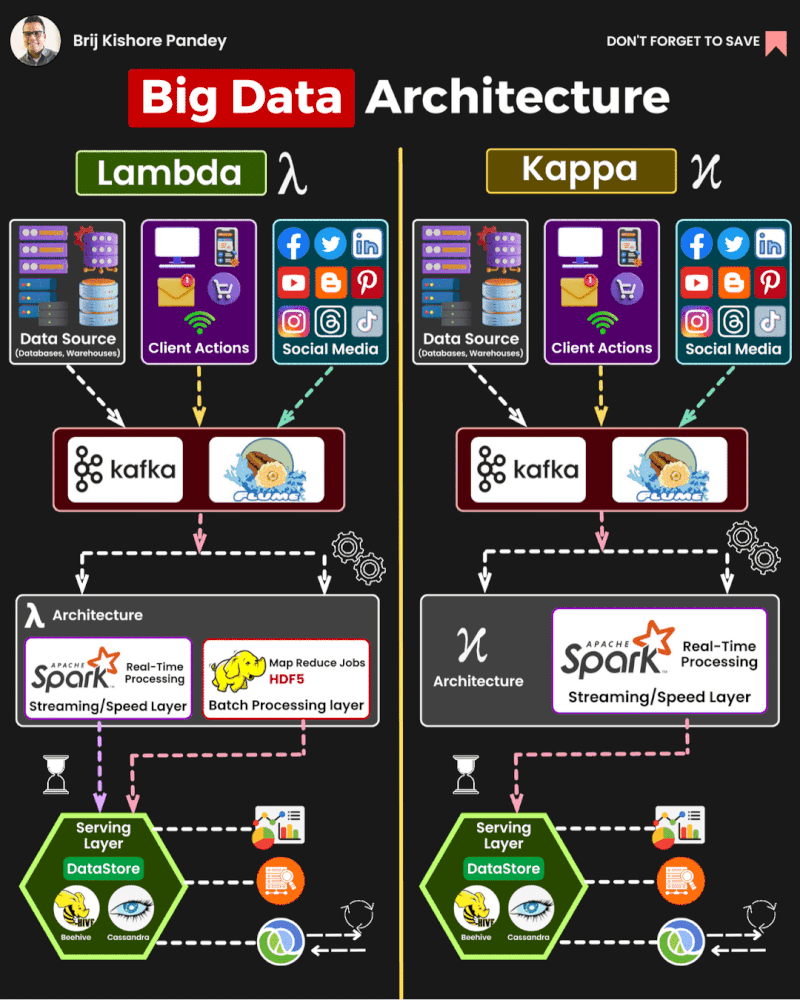
What is Big Data Architecture?

  
At its core, Big Data Architecture refers to the logical and physical structure designed to handle the ingestion, processing, and analysis of data sets too large and complex for traditional database systems.   
Lambda Architecture  
  
Lambda Architecture, introduced by Nathan Marz, is designed to handle massive quantities of data by leveraging both batch and stream-processing methods. It's particularly useful for systems that require both real-time and in-depth batch analysis.  
  
Key Components of Lambda Architecture:  
  
1. 𝗜𝗻𝗴𝗲𝘀𝘁𝗶𝗼𝗻 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Serves as the entry point for raw, immutable data.  
 - Considerations: Must handle high throughput and various data formats.  
  
2. 𝗕𝗮𝘁𝗰𝗵 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Performs resource-intensive computations on the complete dataset.  
 - Technologies: Apache Hadoop, Apache Spark  
 - Output: Creates pre-computed batch views.  
  
3. 𝗦𝗽𝗲𝗲𝗱 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Processes data in real-time to provide low-latency, approximate results.  
 - Technologies: Apache Storm, Apache Flink, Kafka Streams  
 - Output: Generates real-time views.  
  
4. 𝗦𝗲𝗿𝘃𝗶𝗻𝗴 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Combines outputs from batch and speed layers.  
 - Technologies: NoSQL databases (e.g., Apache Cassandra, Apache HBase)  
 - Characteristic: Provides a unified interface for querying results.  
  
Kappa Architecture  
  
Proposed by Jay Kreps, Kappa Architecture simplifies Lambda by treating both real-time and batch processing as stream processing. It's particularly effective for use cases where real-time processing is a primary requirement.  
  
Key Components of Kappa Architecture:  
  
1. 𝗜𝗻𝗴𝗲𝘀𝘁𝗶𝗼𝗻 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Similar to Lambda, ingests raw data.  
 - Technology: Often utilizes log-based systems like Apache Kafka.  
  
2. 𝗦𝘁𝗿𝗲𝗮𝗺 𝗣𝗿𝗼𝗰𝗲𝘀𝘀𝗶𝗻𝗴 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Processes all data as real-time streams.  
 - Technologies: Apache Kafka Streams, Apache Flink  
 - Characteristic: Reprocessing is achieved by replaying the stream.  
  
3. 𝗦𝗲𝗿𝘃𝗶𝗻𝗴 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Stores and serves processed data.  
 - Technologies: Similar to Lambda (e.g., Cassandra, HBase)  
  
The Service Layer: A Critical Component in Both Architectures  
  
While not always explicitly mentioned, the Service Layer plays a crucial role in both Lambda and Kappa architectures:  
  
1. 𝗔𝗣𝗜 𝗘𝗻𝗱𝗽𝗼𝗶𝗻𝘁𝘀:   
 - Function: Exposes data to external systems and applications.  
 - Considerations: RESTful or GraphQL APIs for flexibility.  
  
2. 𝗗𝗮𝘁𝗮 𝗔𝗰𝗰𝗲𝘀𝘀 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Translates API requests into specific database queries.  
 - Importance: Abstracts database complexity from API consumers.  
  
3. 𝗦𝗲𝗰𝘂𝗿𝗶𝘁𝘆 𝗟𝗮𝘆𝗲𝗿:   
 - Function: Manages authentication, authorization, and data governance.  
 - Considerations: Compliance with regulations like GDPR, CCPA.