Project Milestone 3: Data Storage Implementation - KV + Relational

- Sink connector
    - A type of connector that connects to a Kafka topic, where the data from that topic can be exported to a relational database
- Source connector
    - A type of connector that connects to a relational database, and imports data from that database into a Kafka topic

- Kafka connectors with data storage provides functionality of data retrieval and storage
    - This would allow users to have a “cold path” for those are not using real-time processing for their applications
    - It is a seamless integration of data storage and retrieval, and can design the system based on the amount of connectors that are needed
        - Also, these connectors can be designed individually based on their functionality
    - These connectors provide interfacing from Kafka with any external data sources and vice versa, and can be programmed in any way you like

- Kafka connectors maintain availability by interfacing with Kafka Connect
    - Since data flow through Kafka Connect, the connectors wouldn’t be congested with high amounts of data

- The popular Kafka converters for values are
    - JSON Convertor
        - The data can be serialized or deserialized through JSON, which is commonly used for formatting and grouping data that is highly used in common platforms
        - JSON files provide multiple advantages including compactness, simple hierarchical structure, and easy-to-read
    - Avro Convertor
        - Data can also be serialized or deserialized through Apache Avro, which is typically used in Apache Hadoop
        - Avro is more compact than JSON, better performance, and can also be read in parallel
    - Protobuf Convertor

        - Data can also be serialized or deserialized through Google Protobuf (short for Protocol Buffers)
        - Unlike the other convertors, Protobuf can be used in a non-compressed environment, which would take less time than the other convertors
        - Also, it provides backward compatibility when newer versions of Protobuf releases

- Key-Value Database
    - A type of a nonrelational database that uses a unique key (ID) to store data accordingly
    - The key that is associated with the value can then be used for various reasons
        - CRUD operations
        - Provide horizontal scaling
        - Can be highly partitional

- Advantages of Key-Value Database
    - It is very easy to implement and easy to use
    - Great performance, as they are quick to respond to simple querying
    - Highly scalable, and can be partitioned accordingly
    - Very reliable, as it has built-in redundancy so data that is duplicated can be taken place when the data is lost
- Disadvantages of Key-Value Database
    - The more complex the queries and data is, the more it affects the performance
    - Not efficient for searching/lookup, as it has to scan the whole collection to find the result
    - No access methods can be used as the data can only be accessed directly
- Popular KV Databases
    - Dynamo (Amazon)
    - Redis
    - NoSQL Database (Oracle)
    - BerkeleyDB