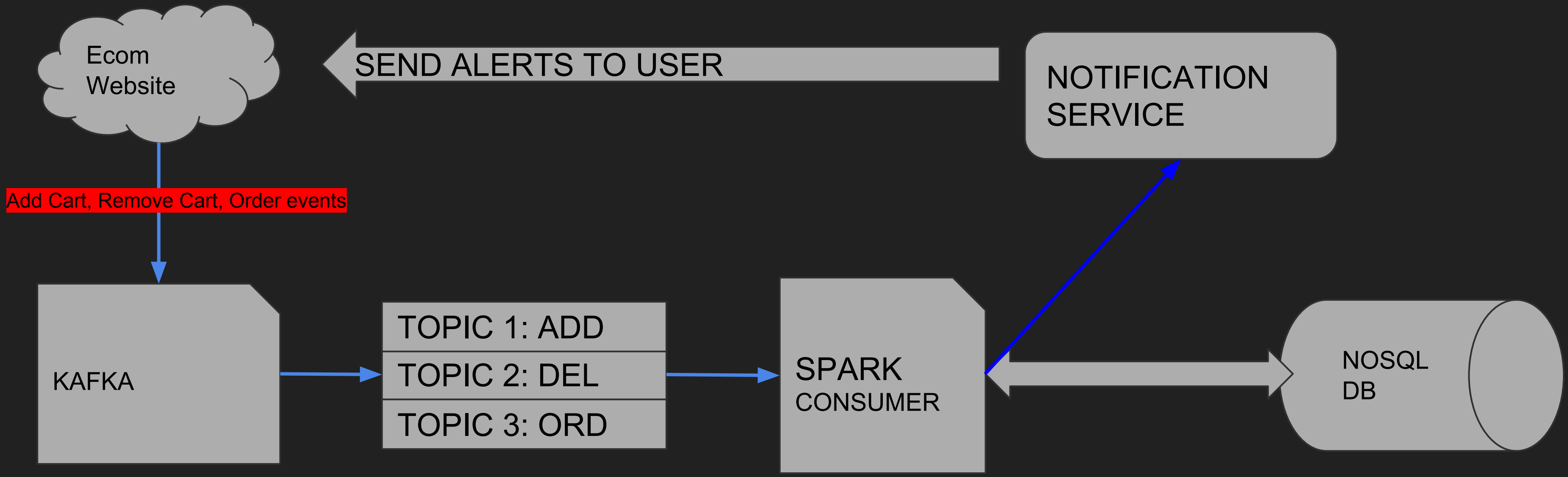
SuperMart Ecommerce Application

HIGH LEVEL DATA INGESTION



DATA INGESTION EXPLAINED

- Website
  - Captures customer event like products Added to Cart, removed from cart, details of order placed along with userid, timestamp of event
  - Each of the events captured is pushed to a Kafka

- KAFKA
  - Each of the events received by Kafka is categorized into three topics - Add, Del and Order
  - Messages are queued up Until a Kafka consumer module reads the messages
  - A SPARK job is used to trigger Kafka consumer to read the messages

DATA INGESTION EXPLAINED

- SPARK (Executed every 2 Hrs)
    - Reads the incoming messages from Kafka producer and creates a Data frame tempCartDF
    - Reads the Cart data from NoSQLDB(Mongo) and creates cartDF
    - Reads the Order Data from NoSQLDB and creates orderDF
    - tempCartDF is ranked using timestamp in reverse grouping by UserID, Products
    - tempCartDF is then filtered on Rank (where RNK = 1)
    - The resultant DF is then compared with cartDF on following criteria
        - If the event is ADD then cartDF is checked for possible duplicates.
            - A list of UserIds and duplicate product list is created and sent to notification service which will send alerts to user.
            - If there no dups then record is updated/Inserted in the NoSQL DB

DATA INGESTION EXPLAINED

- SPARK (Contd)
  -
    - If the Event is DEL, then record is deleted from NoSQLDB
    - If Event is order then similar records then Spark will check against similar order in last 5 orders. A similar check will be performed for ADD/DEL event comparing with ordersDF
      - If a similar order is found, the order details along with user id is sent to notification service which alerts the users.
      - If order is not similar then the order table in NoSQLDb is updated. Also corresponding entries in CART table in NoSQLDB is deleted.

DESIGN CONSIDERATIONS

- Why KAFKA for Streaming Weblog data
  - Spark streaming has following limitations
    - Not Suitable for low latency requirements
    - There are many parameters to tune
  - Kafka on other hand
    - Good for event based processing and can handle low latency
    - Doesnt require a dedicated cluster
    - Easy to onfigure and use. Messages can be configured to consume by different Systems
- Why SPARK as Consumer
  - There are many sequence of events to be performed once the message is read
  - Spark is better in performing comparison of data between different datasets
  - Spark provides efficient libraries to write the data back to the target system.
- Considerations for the Data Storage
  - Prefer using MONGODB
    - Ease of handling data in json format
    - Replacement for RDBMS
    - Highly scalable
    - Faster read/write capability