Hi, thank you for taking the time to work on this case study. We really appreciate it! Please read thoroughly through the tasks, design and develop a solution which solves the problem. Please don‘t spend more than 6 hours working on it and email your completed solution (as tar or zip file) to grace.cako@lotto24.com within 2 days.

PROBLEM

Our organization processes a significant amount of real-time data from various sources for user activities. We need to build a scalable and efficient data processing pipeline that can handle streaming data, aggregate it, and store it for further analysis. The pipeline should be implemented using Python, Golang, or Kotlin, and it should incorporate technologies such as Kafka, a database for storing the data, and a visualization tool.

TASKS

Design and implement a real-time data processing pipeline that satisfies the following requirements:

1. Data source simulation
    a. Simulate a data source that produces streaming events related to a specific domain (e.g., user interactions, sensor readings, etc.).
    b. The data source should send events at irregular intervals.
2. Data transformation
    a. Use a language of your choice (Python, Golang, or Kotlin) to create a consumer that consumes events from the data source.
    b. Transform the raw data to extract relevant information or perform aggregations (e.g. hourly aggregations).
3. Message Broker (Kafka)
    a. Integrate Kafka as the message broker between the data source and the data processing components.
    b. Publish events to a Kafka topic from the data source.
    c. Consume events from the Kafka topic in the data processing component.
4. Data Storage
    a. Use a suitable database (e.g., PostgreSQL, MongoDB) to store the aggregated data.
    b. Design a schema for storing the aggregated data.
5. Observability
    a. Implement monitoring dashboards with basic KPIs to reason about the performance of the data processing pipeline (e.g. size of processed data)
    b. Implement a basic notification mechanism to be informed about the status of processing (e.g. create alerts in case of failures during processing)
    c. Use a tool like Grafana or any other of your choice.
6. Local Development
    a. The entire solution should be runnable locally on your machine.

    b. Use Docker images if necessary but ensure that the solution does not rely on cloud services (AWS, GCP, etc.).
7. Documentation
    a. Provide clear documentation on how to set up and run your solution including instructions for running each component and the data source simulation.
    b. Make assumptions where necessary and explain your solution and approach.