1. **How solution was built**

Solution of the code challenge is built on the Spring Boot framework wit Maven as a build tool. Kafka bootstrap address and group id are configured in application.properties file.

Kafka consumer and producer are configured with JSON serialized custom object values. Incoming messages are represented with Message.class, and outgoing messages are of type UserStats.class. Consumer reads from consumer-test topic, and producer sends messages to producer-test topic.

Incoming messages have only userid and timestamp mapped to custom object, since that is the only data needed for analysis. Timestamps from messages are converted to minutes(seconds are truncated) and user ids from messages are grouped by minute values in a map(minutes represent key in a map). User ids are saved in a Set since it does not allow duplicate values. With simple usage of size() method of Set collection, the number of unique users is returned for a given minute.

After every consumed message check is performed if message contains new userid. If user is new custom outgoing object is created. UserStats.class holds the minute timestamp, and the count of users for that minute. This object is sent by kafka producer to a new topic.

Since data could be historical and is not always strictly ordered, I decided to output data as it comes, after computing statistics for a message timestamp. So every time there is a change in user count, new message is sent out.

JSON was used since kafka has JSON serialization/deserialization support. JSON is faster/lighter(less memory consumption) than XML for example, and it does not need parsing like XML.

1. **Possible improvements**

For simplicity reasons data was kept in memory, but for more advanced solution database could be used, so the memory consumption would be decreased and data wouldn’t be lost on service restart.

Since one of the kafka message headers is offset number, it could be used for tracking progress of message consumption. Offset number could be saved, and if the app crashes, it could be configured to continue from the last consumed message, based on the offset value.