Kafka-Flink based Messaging Application – A Project Report for IT5100B: Stream Processing

1. Application Features:

The fundamental use-case of the application is to enable text-based messaging between one or more users through a console-based interface. **Each user will have to open two console windows** – one for sending text messages (consumer) and the other for receiving text messages from other users (producer).

To use the application, the user must supply a unique username which will be used for two scenarios – to identify the user sending messages and to indicate the user to send messages to. Following the username, the user can key in a text message of the format **(<username>) <text message>** which will be sent to the Kafka cluster for further processing.

The message will then be consumed by a Flink processor, which will perform two actions on it. Firstly, it will **censor the message based on profanity**, and for simplicity, the processor checks for three words in particular: "dick", "cunt" and "damn". Any presence of these words in the message (either as a whole word or part of a larger word) will be censored and changed into "d**k", "c**t" and "d**n" respectively.

The second action, which is the additional feature of the application, is the **spam detector**. The messages will go through a keyed Flink processor stage, which will check messages from each user and classify a message as "might be spam" or normal based on the language of the text message: a project reactor based processing is applied to the text message which reactively checks if any instance of words like "giveaway", "jackpot", "click", "activate" are detected, the message is annotated with the "might be spam" string along with to warn the user of a potential spam message.

2. Assumptions:

a. The name of the user is taken to be the key for Kafka events.
b. The user can exit the application anytime by only closing the console window.
c. The user can send messages to only one user at a time. (no broadcasts).
d. The user can only send alphanumeric text messages.
e. The user follows only the given format for sending messages, failing which the application prompts the user to resend the message in the right way.

3. Configuration:

The Kafka cluster will host one producer and consumer per user, and there are two topics configured – "***kafka-messenger-app***" will have the raw text messages sent from a user and "***kafka-messenger-censored-texts***" will have the Flink-processed messages ready to be consumed by the end user. The cluster will have **6 partitions** to perform parallel processing with a **replication factor of 3** which will ensure that the cluster is both faster and more reliable at the same time. The **retention time is set at 1 hour** so that all messages are retained in the cluster for an hour before they cease to exist. The Flink task manager is configured to have **6 task slots** so that it supports the 6 Kafka partitions working in parallel.

4. <u>Build and run steps:</u>

The project consists of two applications – one for the user interface that also takes care of the Kafka configurations and the other for Flink processing.

a. Run docker using the ***docker-compose.yaml*** attached in the zip (has been altered to allow 6 partitions in cluster)
b. Start the docker engine and create two topics with the following commands:
   kafka-topics.sh --create --topic kafka-messenger-app --partitions 6 --replication-factor 3 --config retention.ms=3600000 --bootstrap-server kafka0:9094
   kafka-topics.sh --create --topic kafka-messenger-censored-texts --partitions 6 --replication-factor 3 --config retention.ms=3600000 --bootstrap-server kafka0:9094
c. Perform maven clean package on the Flink application and create a jar file which can be used to create a new job on the Flink dashboard. The package where the main class resides is "***com.flink.messenger.FlinkCensor***"
d. Perform maven clean package on the Kafka application and create a jar file, which can be run using the command "***java -jar kafka-messenger-1.0.jar***". The application starts up and is ready to be used.

5. <u>Troubleshooting:</u>

In case the application faces intermittent issues in the cluster, stop the running job in Flink dashboard and use the commands:

kafka-topics.sh --delete --topic kafka-messenger-app --bootstrap-server localhost:9094
kafka-topics.sh --delete --topic kafka-messenger-censored-texts --bootstrap-server localhost:9094

to delete the Kafka topics and restart with the steps in point 4 to use the application again (skip the maven commands).

[**Sricharan Sriram** (A0279517N)]