Kafka

W ramach zestawów zadań dotyczących platformy Esper napisaliśmy własny generator strumienia danych. Formatem danych, który wówczas wykorzystywaliśmy był JSON.

Generator EventsScore

W ramach tego zestawu stworzymy analogiczny generator strumienia sztucznych danych, który będzie wprowadzał dane do wskazanego tematu Kafki. Wykorzystamy do tego celu, podobnie jak w wówczas bibliotekę *Datafaker* (`https://www.datafaker.net/`).

1. Korzystając z **technicznego** terminala SSH pobierz początkową wersję naszego programu.

```
wget http://jankiewicz.pl/bigdata/KafkaFakerProducer.java
```

2. Pobierz bibliotekę *Datafaker* w odpowiedniej wersji dla wykorzystywanej wersji Javy. Przykładowo wersja 1.4 jest kompatybilna z Javą w wersji 1.8.

```
wget https://repo1.maven.org/maven2/net/datafaker/datafaker/1.4.0/datafaker-1.4.0.jar
```

3. Otwórz w edytorze tekstowym zawartość pobranego programu. Przeanalizuj ją. Na razie niczego nie zmieniaj.

```
nano KafkaFakerProducer.java
```

4. Wyjdź z edytora. Dokonamy teraz kompilacji naszego programu, a następnie go uruchomimy.

```
mkdir -p build
javac -cp /usr/lib/kafka/libs/*:datafaker-1.4.0.jar KafkaFakerProducer.java -d build
jar -cvf KafkaFakerProducer.jar -C build/ .
```

5. Jak można było to wyczytać z treści programu, ma on następujące parametry:
   - Adres serwera Kafki
   - Nazwa tematu Kafki
   - Format w którym będą tworzone nasze dane (csv lub json)
   - Liczba wiadomości wysyłana w ciągu sekundy
   - Długość czasu wysyłki w sekundach

   Na tą chwilę program niczego nie wysyła, a jedynie wyświetla zawartość rekordów przeznaczonych do wysyłki. Sprawdzimy teraz czy obecna wersja naszego programu działa zgodnie z założeniami.

```
CLUSTER_NAME=$(/usr/share/google/get_metadata_value attributes/dataproc-cluster-name)
java -cp /usr/lib/kafka/libs/*:datafaker-1.4.0.jar:KafkaFakerProducer.jar \
      KafkaFakerProducer ${CLUSTER_NAME}-w-0:9092 kafka-input json 2 5
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ java -cp /usr/lib/kafka/libs/*:datafaker-1.4.0.jar:KafkaFakerProducer.jar \
>       KafkaFakerProducer ${CLUSTER_NAME}-w-0:9092 kafka-input json 2 5
{"house": "Horned Serpent", "character": "Wilhelmina Grubbly-Plank", "score": "9", "ts": "2022-08-11 10:39:20.678"}
{"house": "Wampus", "character": "Bill Weasley", "score": "8", "ts": "2022-08-11 10:39:25.741"}
{"house": "Slytherin", "character": "Charlie Weasley", "score": "3", "ts": "2022-08-11 10:39:17.282"}
{"house": "Wampus", "character": "Ronan", "score": "6", "ts": "2022-08-11 10:39:24.299"}
```

6. Jeśli wszystko zadziałało, usuń katalog `build` oraz wynikowy plik jar.

```
rm -rf build
rm KafkaFakerProducer.jar
```

7. Otwórz ponownie plik zawierający treść naszego programu, a następnie wprowadź do niego poprawki zgodnie z komentarzami TODO. Skorzystaj z materiałów wykładowych oraz dokumentacji
   `https://kafka.apache.org/23/javadoc/index.html?org/apache/kafka/clients/producer/KafkaProducer.html`

```
nano KafkaFakerProducer.java
```

8. Ponownie dokonaj kompilacji Twojego programu oraz utwórz plik jar.

```
mkdir -p build
javac -cp /usr/lib/kafka/libs/*:datafaker-1.4.0.jar KafkaFakerProducer.java -d build
jar -cvf KafkaFakerProducer.jar -C build/ .
```

9. Czas na sprawdzenie naszych dokonań. W **technicznym** terminalu SSH utwórz temat Kafki do którego dane będzie zapisywał nasz producent.

```
kafka-topics.sh --create \
     --zookeeper localhost:2181 \
     --replication-factor 2 --partitions 3 --topic kafka-input
```

10. W terminalu **odbiorczym** uruchom konsumenta „z konsoli”

```
CLUSTER_NAME=$(/usr/share/google/get_metadata_value attributes/dataproc-cluster-name)
kafka-console-consumer.sh --group my-consumer-group \
     --bootstrap-server ${CLUSTER_NAME}-w-0:9092 \
     --topic kafka-input --from-beginning
```

11. W terminalu **nadawczym** rozpocznijmy nadawanie

```
CLUSTER_NAME=$(/usr/share/google/get_metadata_value attributes/dataproc-cluster-name)
java -cp /usr/lib/kafka/libs/*:datafaker-1.4.0.jar:KafkaFakerProducer.jar \
      KafkaFakerProducer ${CLUSTER_NAME}-w-0:9092 kafka-input json 2 5
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ kafka-console-consumer.sh --group my-consumer-group      --bootstrap-server
 ${CLUSTER_NAME}-w-0:9092      --topic kafka-input --from-beginning
{"house": "Hufflepuff", "character": "Ted Tonks", "score": "3", "ts": "2022-08-11 11:23:25.631"}
{"house": "Pukwudgie", "character": "Helga Hufflepuff", "score": "4", "ts": "2022-08-11 11:23:35.15"}
{"house": "Ravenclaw", "character": "Xenophilius Lovegood", "score": "2", "ts": "2022-08-11 11:23:34.575"}
{"house": "Gryffindor", "character": "Everard", "score": "1", "ts": "2022-08-11 11:23:36.011"}
{"house": "Pukwudgie", "character": "Yaxley", "score": "2", "ts": "2022-08-11 11:23:20.637"}
```

12. Jeśli wszystko dobrze działa to sprawdźmy jak wygląda wypełnienie naszego tematu. Wyjdź z **konsumenta** za pomocą *Ctrl+C* i wykonaj poniższe polecenie

```
kafka-run-class.sh kafka.tools.GetOffsetShell \
     --broker-list ${CLUSTER_NAME}-w-0:9092 \
     --topic kafka-input --time -1
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ kafka-run-class.sh kafka.tools.GetOffsetShell \
>      --broker-list ${CLUSTER_NAME}-w-0:9092 \
>      --topic kafka-input --time -1
kafka-input:0:5
kafka-input:1:4
kafka-input:2:3
```

13. Spróbujemy teraz trochę agresywniej skorzystać z tego tematu. W **nadawczym** terminalu SSH uruchom ten sam program z nieco innymi wartościami parametrów.

```
java -cp /usr/lib/kafka/libs/*:datafaker-1.4.0.jar:KafkaFakerProducer.jar \
      KafkaFakerProducer ${CLUSTER_NAME}-w-0:9092 kafka-input csv 20000 5
```

14. Po zakończeniu wykonywania programu ponownie wykonaj poniższe polecenie

```
kafka-run-class.sh kafka.tools.GetOffsetShell \
     --broker-list ${CLUSTER_NAME}-w-0:9092 \
     --topic kafka-input --time -1
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ kafka-run-class.sh kafka.tools.GetOffsetShell \
>      --broker-list ${CLUSTER_NAME}-w-0:9092 \
>      --topic kafka-input --time -1
kafka-input:0:44961
kafka-input:1:30042
kafka-input:2:45009
```

Z powyższego wyniku możemy wysnuć wniosek, że wysyłka 20000 wiadomości na sekundę nawet przy tak małym klastrze jest możliwa.

Mamy już nasze źródło strumienia danych (nasz program), oraz mechanizm dostarczania tych strumieni (*Klaster Apache Kafka*) do systemów, które mogą je przetwarzać. Przykładami takich systemów mogą być *Spark Structured Streaming*, *Kafka Streams* oraz *Flink*. W spróbujemy z nich w przyszłości skorzystać.

Twój własny generator

15. Czas na twórczość własną. Napisz własny generator sztucznych danych. Skorzystaj z generatora, który opracowałeś w ramach zestawów zadań z systemem Esper.