Домашна задача 1: Apache Flink апликација која процесира податоци од Apache Kafka

Овој проект имплементира real-time streaming pipeline за обработка на сензорски податоци користејќи Apache Flink и Apache Kafka. Системот чита JSON пораки од Kafka topic, ги обработува со временски прозорци, и генерира агрегирани резултати кои се испраќаат назад во Kafka.

Преглед на архитектурата
Проектот се состои од три главни компоненти:

1. Producer скрипта (*produce_messages.py*)
Генерира синтетички сензорски податоци и ги испраќа на Kafka topic sensors. Секоја порака содржи:

- key: Идентификатор на сензорот (A, B, C, или D)
- value: Случајна нумеричка вредност (0-1000)
- timestamp: Unix timestamp во милисекунди

Пораките се испраќаат на секои 0.5-2 секунди за симулирање на реален streaming сценарио.

2. Flink Streaming Job (*flink_job.py*)
Главната обработувачка логика која користи *PyFlink Table API* за:

- Конфигурација на прозорци:
    - Hopping Window со големина од 60 секунди
    - Slide интервал од 30 секунди
    - Watermark tolerance од 5 секунди за handling на late events
- Две паралелни query-ња:
    - Query 1 → results1 topic:
        - Пресметува број на пораки по клуч и по прозорец
        - Враќа: key, window_start, window_end, cnt
    - Query 2 → results2 topic:
        - Пресметува статистички агрегации:
            - Минимална вредност
            - Максимална вредност
            - Просечна вредност
            - Број на пораки

3. Consumer скрипта (*consume_messages.py*)
Чита резултати од двата output topics (results1 и results2) и ги прикажува во конзолата во реално време. Користи consumer group за да обезбеди at-least-once семантика.

Податочен Flow

- produce_messages.py → sensors topic (Kafka)

- flink_job.py чита од sensors → обработува со HOP windowing → пишува во results1 и results2
- consume_messages.py чита од results1 и results2 → прикажува резултати

Резултати добиени од целиот flow прикажани со consume_messages.py:

[results1] {"key":"D","window_start":1763845500000,"window_end":1763845560000,"cnt":11}
[results1] {"key":"A","window_start":1763845500000,"window_end":1763845560000,"cnt":10}
[results1] {"key":"C","window_start":1763845500000,"window_end":1763845560000,"cnt":4}
[results1] {"key":"B","window_start":1763845500000,"window_end":1763845560000,"cnt":5}
[results2]
{"key":"D","window_start":1763775840000,"window_end":1763775900000,"min_value":31,
"count":5,"average":444.0,"max_value":848}
[results2]
{"key":"A","window_start":1763775840000,"window_end":1763775900000,"min_value":327,
"count":3,"average":666.3333333333334,"max_value":875}
[results2]
{"key":"B","window_start":1763775840000,"window_end":1763775900000,"min_value":486,
"count":1,"average":486.0,"max_value":486}
[results2]
{"key":"C","window_start":1763775840000,"window_end":1763775900000,"min_value":354,
"count":4,"average":634.0,"max_value":882}

За изработка на овој проект беше употребен кодот симнат од репозиториумот https://github.com/stefanandonov/RNMP_homework1.git , со мали измени поради проблемот со некомпатабилноста на некои од сликите во Docker Compose датотеката, и надополнување на *consume_messages.py*, изработка на *flink_job.py* и поставување на виртуелни околини за извршување на кодот. Во *README.md* датотеката е детално објаснето како се извршува целата апликација.

Мила Ѓуровска 231116