|  |  |  |  |
| --- | --- | --- | --- |
|  | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji,  Informatyki i Elektrotechniki  **Zakład Systemów Teleinformatycznych** | |  |
| **Przedmiot** | Podstawy inżynierii danych | | |
| **Prowadzący** | prof. dr hab. inż. prof. PBŚ Piotr Cofta | | |
| **Temat** | Laboratorium 3 | | |
| **Student** | Cezary Tytko | | |
| **Ocena** |  | **Data oddania spr.** |  |

Zad 1. Instalacja plików

Skopiowałem pliki do odpowiednich katalogów zgodnie z instrukcją.

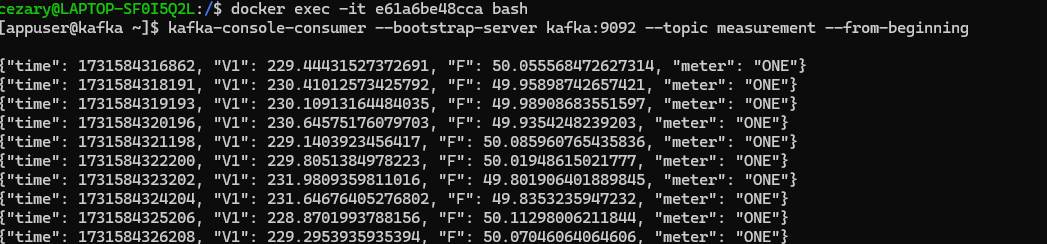
Zad 2. Instalacja generatora i brokera

Po uruchomieniu skryptu, utworzyły się dwa nowe obrazy, ale tylko broker uruchomił się poprawnie, kontener z generatorem nie działał, włączał się z powodu błędu związanego z brakiem biblioteki „six”. Jawnie wskazałem w skrypcie z jakiej wersji kafki ma korzystać python, oraz dodałem polecenie instalujące „six”, nie pomogło, rozwiązaniem było wskazanie wersji obrazu pythonowego na 3.8, a nie najnowszą ”latest”



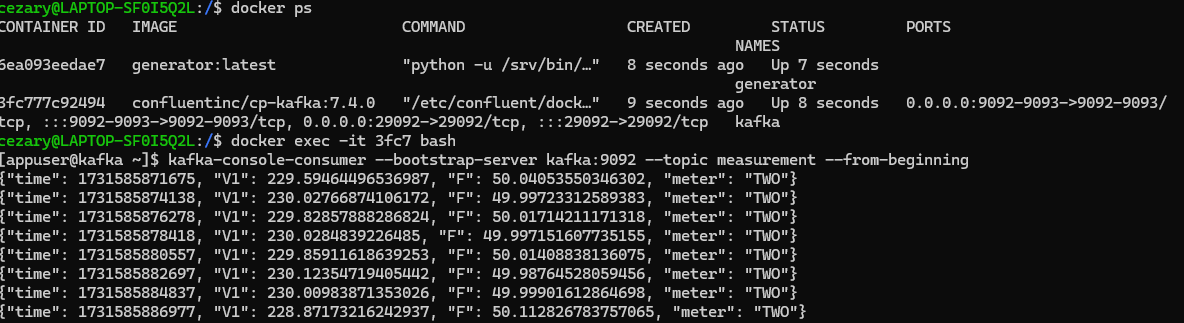
Zad 3. Sprawdzenie przepływu danych

Zadanie nie było problematyczne, wystarczyło wejść do kontenera kafki i utworzyć konsumenta na odpowiedni temat.



Zad 4. Modyfikacja generatora

Zmodyfikowałem parametry obecnego polecenia docker run uruchamiającego obraz generatora, podając SLEEP\_MS na 2137 i zmieniając METER\_ID na „TWO”



Jak widać kolejne rekordy różnią się parametrem time o mniej więcej zadaną zadany czas.

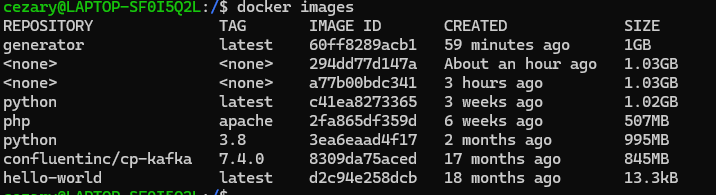
Zad 5. Wytworzenie obrazu generatora

Zapisywanie obrazu generatora było już umieszczone w skrypcie:

1. docker build -t "$GNAME:latest" .

2.

Powoduje to dodanie obrazu do lokalnego repozytorium



Zad 6. Uruchomienie dwóch generatorów

Zmodyfikowałem polecenie uruchamiające kontener z generatorem dodając kolejne polecenie „docker run” dla obrazu generator, zachowując pierwotny generator jaki był tworzony:

1. docker run \

2.   --hostname "$GNAME\_2" \

3.   --name "$GNAME\_2" \

4.   --network "$NNAME" \

5.   --env  BROKER\_BOOTSTRAP="$KNAME:9092" \

6.   --env  BROKER\_TOPIC='measurement' \

7.   --env  SLEEP\_MS='2137' \

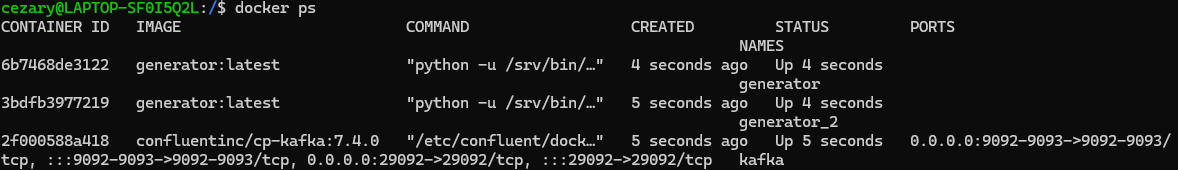
8.   --env  METER\_ID='TWO' \

9.   --detach \

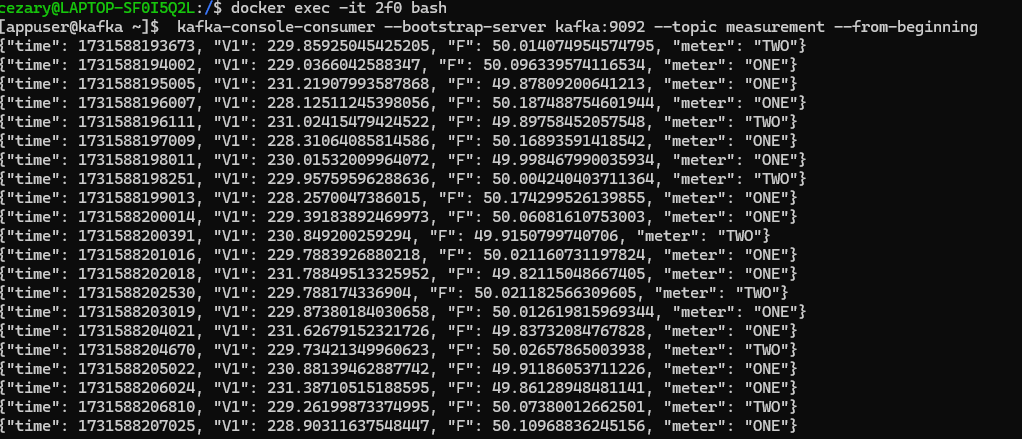
10.   "$GNAME:latest"

11.

Uzyskałem 3 kontenery:



Tak samo jak w poprzednim zadaniu w kontenerze kafki utworzyłem konsumenta na odpowiedni temat, przechwytywałem tym samym dane z obu generatorów:

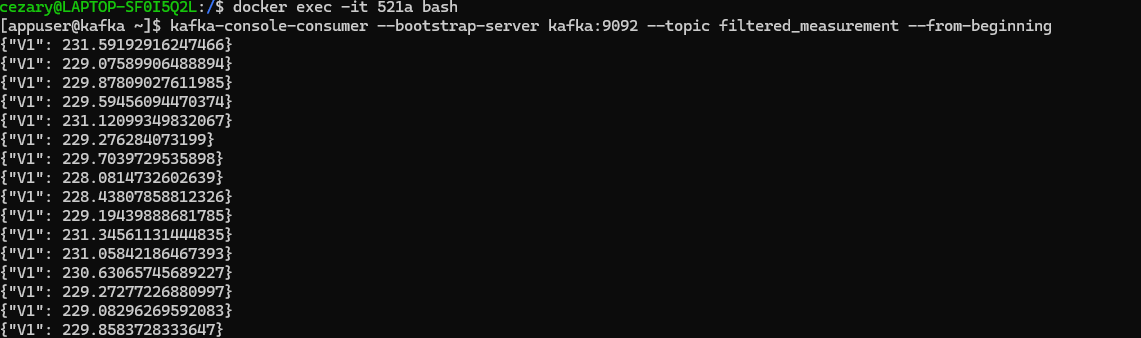


Zad 7. Zbudowanie filtru

Utworzyłem plik py zawierający kod definiujący filtr zgodnie z założeniami polecenia, następnie dodałem do skryptu polecenie uruchamiające filtr w nowym kontenerze. Otrzymałem aż 4 kontenery (broker, 2 generatory i filtr) :



Utworzyłem konsumenta na odpowiednim temacie z przefiltrowanymi danymi:



Kod Filtra:

1. from time import sleep

2. from json import dumps

3. from kafka import KafkaConsumer, KafkaProducer

4. from kafka.errors import NoBrokersAvailable

5. from kafka.errors import KafkaTimeoutError

6. from json import loads, dumps

7. import os

8.

9. input\_topic = os.environ.get("INPUT\_TOPIC", "input-topic")

10. output\_topic = os.environ.get("OUTPUT\_TOPIC", "output-topic")

11. broker = os.environ.get("BROKER\_BOOTSTRAP", "127.0.0.1:9092")

12. data\_type = os.environ.get("DATA\_TYPE", "V1")

13.

14. while True:

15.     try:

16.         consumer = KafkaConsumer(

17.             input\_topic,

18.             bootstrap\_servers=[broker],

19.             group\_id="filter-group",

20.             value\_deserializer=lambda x: loads(x.decode('utf-8'))

21.         )

22.

23.         producer = KafkaProducer(

24.             bootstrap\_servers=[broker],

25.             value\_serializer=lambda x: dumps(x).encode('utf-8')

26.         )

27.

28.         for message in consumer:

29.             data = message.value

30.

31.             if data\_type in data:

32.                 filtered\_data = {data\_type: data[data\_type]}

33.                 producer.send(output\_topic, value=filtered\_data)

34.                 print(f"Przesyłanie: {filtered\_data}")

35.

36.     except NoBrokersAvailable:

37.         sleep(5)

38.

39.     except KafkaTimeoutError:

40.         sleep(5)

41.

Wnioski:

Bardzo ciekawe i przydatne laboratorium, dzięki połączeniu kafki z generatorem i filtrem napisanym w pythonie, można było zobaczyć działanie w bardziej praktyczne. Podczas wykonywania ćwiczeń problematyczne były tylko pierwsze uruchomienie generatora z powodu błędu w wersjach, rozwiązaniem było wskazanie wersji pythona na 3.8. Napisanie własnego filtru było bardziej ciekawe niż problematyczne, zadziałało przy pierwszej próbie.

Do sprawozdania załączam zmodyfikowany kod skryptu oraz filtru.