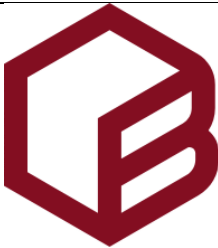



|   |  |                          |   |
|---|--|--------------------------|---|
|  | Politechnika Bydgoska im. J. J. Śniadeckich<br>Wydział Telekomunikacji,<br>Informatyki i Elektrotechniki<br><b>Zakład Systemów Teleinformatycznych</b> |                          |  |
| <b>Przedmiot</b>  | Podstawy inżynierii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

### 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”

```

cezary@LAPTOP-SF015Q2L:/$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS
5ec7acbaad23   generator:latest                    "python -u /srv/bin/..." About a minute ago Up 59 seconds   generator
e61a6be48cca   confluentinc/cp-kafka:7.4.0        "/etc/confluent/dock..." 2 minutes ago    Up 2 minutes    0.0.0.0:9092-9093
-->9092-9093/tcp, :::9092-9093->9092-9093/tcp, 0.0.0.0:29092->29092/tcp, :::29092->29092/tcp   kafka
cezary@LAPTOP-SF015Q2L:/$

```

### Zad 3. Sprawdzenie przepływu danych

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

```

cezary@LAPTOP-SF015Q2L:/$ docker exec -it e61a6be48cca bash
lappuser@kafka: ~]$ kafka-console-consumer --bootstrap-server kafka:9092 --topic measurement --from-beginning

{"time": 1731584316862, "V1": 229.44431527372691, "F": 50.055568472627314, "meter": "ONE"}
{"time": 1731584318191, "V1": 230.41012573425792, "F": 49.95898742657421, "meter": "ONE"}
{"time": 1731584319193, "V1": 230.10913164484035, "F": 49.98908683551597, "meter": "ONE"}
{"time": 1731584320196, "V1": 230.64575176079703, "F": 49.9354248239203, "meter": "ONE"}
{"time": 1731584321198, "V1": 229.1403923456417, "F": 50.085960765435836, "meter": "ONE"}
{"time": 1731584322200, "V1": 229.8051384978223, "F": 50.01948615021777, "meter": "ONE"}
{"time": 1731584323202, "V1": 231.9809359811016, "F": 49.801906401889845, "meter": "ONE"}
{"time": 1731584324204, "V1": 231.64676405276802, "F": 49.8353235947232, "meter": "ONE"}
{"time": 1731584325206, "V1": 228.8701993788156, "F": 50.11298006211844, "meter": "ONE"}
{"time": 1731584326208, "V1": 229.2953935935394, "F": 50.07046064064606, "meter": "ONE"}

```

## Zad 4. Modyfikacja generatora

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

```
cezary@LAPTOP-SF0I5Q2L:/$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
6ea093eedae7   generator:latest   "python -u /srv/bin/..." 8 seconds ago   Up 7 seconds   generator
3fc777c92494   confluentinc/cp-kafka:7.4.0   "/etc/confluent/dock..." 9 seconds ago   Up 8 seconds   0.0.0.0:9092-9093->9092-9093/tcp, ::9092->9092/tcp, ::29092->29092/tcp   kafka
cezary@LAPTOP-SF0I5Q2L:/$ docker exec -it 3fc7 bash
lappuser@kafka ~]$ kafka-console-consumer --bootstrap-server kafka:9092 --topic measurement --from-beginning
{"time": 1731585871675, "V1": 229.59464496536987, "F": 50.04053550346302, "meter": "TWO"}
{"time": 1731585874138, "V1": 230.02766874106172, "F": 49.99723312589383, "meter": "TWO"}
{"time": 1731585876278, "V1": 229.82857888286824, "F": 50.01714211171318, "meter": "TWO"}
{"time": 1731585878418, "V1": 230.0284839226485, "F": 49.997151607735155, "meter": "TWO"}
{"time": 1731585880557, "V1": 229.85911618639253, "F": 50.01408838136075, "meter": "TWO"}
{"time": 1731585882697, "V1": 230.12354719405442, "F": 49.98764528059456, "meter": "TWO"}
{"time": 1731585884837, "V1": 230.00983871353026, "F": 49.99901612864698, "meter": "TWO"}
{"time": 1731585886977, "V1": 228.87173216242937, "F": 50.112826783757065, "meter": "TWO"}
```

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

```
cezary@LAPTOP-SF0I5Q2L:/$ docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
generator        latest       60ff8289acb1   59 minutes ago 1GB
<none>           <none>       294dd77d147a   About an hour ago 1.03GB
<none>           <none>       a77b00bdc341   3 hours ago    1.03GB
python           latest       c41ea8273365   3 weeks ago    1.02GB
php              apache       2fa865df359d   6 weeks ago    507MB
python           3.8         3ea6eaad4f17   2 months ago   995MB
confluentinc/cp-kafka 7.4.0       8309da75aced   17 months ago 845MB
hello-world      latest       d2c94e258dcb   18 months ago 13.3kB
```

## 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:

```
cezary@LAPTOP-SF0ISQ2L:/$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS
6b7468de3122   generator:latest     "python -u /srv/bin/..." 4 seconds ago   Up 4 seconds   generator
3bdfb3977219   generator:latest     "python -u /srv/bin/..." 5 seconds ago   Up 4 seconds   generator_2
2f000588a418   confluentinc/cp-kafka:7.4.0 "/etc/confluent/dock..." 5 seconds ago   Up 5 seconds   0.0.0.0:9092-9093->9092-9093/
tcp, :::9092-9093->9092-9093/tcp, 0.0.0.0:29092->29092/tcp, :::29092->29092/tcp   kafka
```

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

```
cezary@LAPTOP-SF0ISQ2L:/$ docker exec -it 2f0 bash
lappuser@kafka ~]$ kafka-console-consumer --bootstrap-server kafka:9092 --topic measurement --from-beginning
{"time": 1731588193673, "V1": 229.85925045425205, "F": 50.014074954574795, "meter": "TWO"}
{"time": 1731588194002, "V1": 229.0366042588347, "F": 50.096339574116534, "meter": "ONE"}
{"time": 1731588195005, "V1": 231.21907993587868, "F": 49.87809200641213, "meter": "ONE"}
{"time": 1731588196007, "V1": 228.12511245398056, "F": 50.187488754601944, "meter": "ONE"}
{"time": 1731588196111, "V1": 231.02415479424522, "F": 49.89758452057548, "meter": "TWO"}
{"time": 1731588197009, "V1": 228.31064085814586, "F": 50.16893591418542, "meter": "ONE"}
{"time": 1731588198011, "V1": 230.01532009964072, "F": 49.998467990035934, "meter": "ONE"}
{"time": 1731588198251, "V1": 229.95759596288636, "F": 50.004240403711364, "meter": "TWO"}
{"time": 1731588199013, "V1": 228.2570047386015, "F": 50.174299526139855, "meter": "ONE"}
{"time": 1731588200014, "V1": 229.39183892469973, "F": 50.06081610753003, "meter": "ONE"}
{"time": 1731588200391, "V1": 230.849200259294, "F": 49.9150799740706, "meter": "TWO"}
{"time": 1731588201016, "V1": 229.7883926880218, "F": 50.021160731197824, "meter": "ONE"}
{"time": 1731588202018, "V1": 231.78849513325952, "F": 49.82115048667405, "meter": "ONE"}
{"time": 1731588202530, "V1": 229.788174336904, "F": 50.021182566309605, "meter": "TWO"}
{"time": 1731588203019, "V1": 229.87380184030658, "F": 50.012619815969344, "meter": "ONE"}
{"time": 1731588204021, "V1": 231.62679152321726, "F": 49.83732084767828, "meter": "ONE"}
{"time": 1731588204670, "V1": 229.73421349960623, "F": 50.02657865003938, "meter": "TWO"}
{"time": 1731588205022, "V1": 230.88139462887742, "F": 49.91186053711226, "meter": "ONE"}
{"time": 1731588206024, "V1": 231.38710515188595, "F": 49.86128948481141, "meter": "ONE"}
{"time": 1731588206810, "V1": 229.26199873374995, "F": 50.07380012662501, "meter": "TWO"}
{"time": 1731588207025, "V1": 228.90311637548447, "F": 50.10968836245156, "meter": "ONE"}
```

## Zad 7. Zbudowanie filtra

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) :

```
cezary@LAPTOP-SF0ISQ2L:/$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS
8749c0a1fd68   filter:latest        "python -u /srv/bin/..." 3 seconds ago   Up 3 seconds   filter
aa5ec57a5438   generator:latest     "python -u /srv/bin/..." 8 seconds ago   Up 8 seconds   generator
0332bc1bf28a   generator:latest     "python -u /srv/bin/..." 9 seconds ago   Up 8 seconds   generator_2
521a9d9ee4e3   confluentinc/cp-kafka:7.4.0 "/etc/confluent/dock..." 9 seconds ago   Up 9 seconds   0.0.0.0:9092-9093->9092-9093/
tcp, :::9092-9093->9092-9093/tcp, 0.0.0.0:29092->29092/tcp, :::29092->29092/tcp   kafka
cezary@LAPTOP-SF0ISQ2L:/$
```

Utworzyłem konsumenta na odpowiednim temacie z przefiltrowanymi danymi:

```
cezary@LAPTOP-SF015Q2L:/$ docker exec -it 521a bash
[appuser@kafka ~]$ kafka-console-consumer --bootstrap-server kafka:9092 --topic filtered_measurement --from-beginning
{"V1": 231.59192916247466}
{"V1": 229.07589906488894}
{"V1": 229.87809027611985}
{"V1": 229.59456094470374}
{"V1": 231.12099349832067}
{"V1": 229.276284073199}
{"V1": 229.7039729535898}
{"V1": 228.0814732602639}
{"V1": 228.43807858812326}
{"V1": 229.19439888681785}
{"V1": 231.34561131444835}
{"V1": 231.05842186467393}
{"V1": 230.63065745689227}
{"V1": 229.27277226880997}
{"V1": 229.08296269592083}
{"V1": 229.8583728333647}
```

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 filtra było bardziej ciekawe niż problematyczne, zadziałało przy pierwszej próbie.

Do sprawozdania załączam zmodyfikowany kod skryptu oraz filtra.