Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра системного проектування

Оброблення надвеликих масивів даних

Лабораторна робота №5

Виконала:

студентка групи ДА-11мп

Молчанова В.С.

Київ – 2021 р.

### Завдання

1. Запустити Apache Kafka у середовищі контейнеризації, як приклад

Docker.

2. Написати «Виробника» даних, який передаватиме дані через Apache

Kafka.

3. Написати «Споживача» даних, який зчитуватиме дані з Apache Kafka.

4. Дати відповіді на контрольні запитання.

### Текст програми

Варіант =13 %9 = 4

Файл docker-compose.yml

version: '2'

services:

  zookeeper:

    image: wurstmeister/zookeeper

    ports:

     - "2181:2181"

  kafka:

    image: wurstmeister/kafka

    ports:

     - "9092:9092"

    expose:

     - "9093"

    environment:

      KAFKA\_ADVERTISED\_LISTENERS: INSIDE://kafka:9093,OUTSIDE://localhost:9092

      KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT

      KAFKA\_LISTENERS: INSIDE://0.0.0.0:9093,OUTSIDE://0.0.0.0:9092

      KAFKA\_INTER\_BROKER\_LISTENER\_NAME: INSIDE

      KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

      KAFKA\_CREATE\_TOPICS: "produced:2:1,shipped:2:1,sold:2:1"

    volumes:

     - /var/run/docker.sock:/var/run/docker.sock

Файл producer.py

from random import randint

from kafka import KafkaProducer

from json import dumps

types = ['blue', 'red', 'light', 'mint']

shops = ['silpo', 'atb', 'auchan', 'novus']

producer = KafkaProducer(

   value\_serializer=lambda m: dumps(m).encode('utf-8'),

   bootstrap\_servers=['localhost:9092'])

# production

produced = {}

for type in types:

    amount = randint(100, 1000)

    produced[type] = amount

    print(f'Produced {amount} {type} packs.')

    producer.send('produced', {type:amount})

# shipping

shipped = {}

for shop in shops:

    data = dict(map(lambda x: (x, randint(100, produced[x])), types))

    shipped[shop] = data

    print(f'Shipped to {shop}: {data}.')

    producer.send('shipped', {shop:data})

# selling

sold = {}

for shop in shops:

    data = dict(map(lambda x: (x, randint(100, shipped[shop][x])), types))

    sold[shop] = data

    print(f'Sold in {shop}: {data}.')

    producer.send('sold', {shop:data})

producer.flush()

Файл consumer.py

from typing import Dict

from kafka import KafkaConsumer

from json import loads

produced = {}

shipped = {}

sold = {}

def process\_produced(msg: Dict[str, int]):

    produced.update(msg)

    print(f'Saved production data: {msg}')

def process\_shipped(msg: Dict[str, Dict[str, int]]):

    shipped.update(msg)

    print(f'Saved shipment data: {msg}')

def process\_sold(msg: Dict[str, Dict[str, int]]):

    sold.update(msg)

    print(f'Saved selling data: {msg}')

consumer = KafkaConsumer(

    'produced', 'shipped', 'sold',

    auto\_offset\_reset='earliest',

    enable\_auto\_commit=True,

    group\_id='my\_group',

    value\_deserializer=lambda m: loads(m.decode('utf-8')),

    bootstrap\_servers=['localhost:9092'])

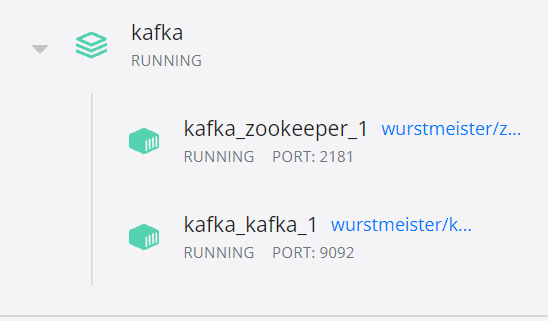
func\_dict = { 'produced': process\_produced, 'shipped': process\_shipped, 'sold': process\_sold}

for msg in consumer:

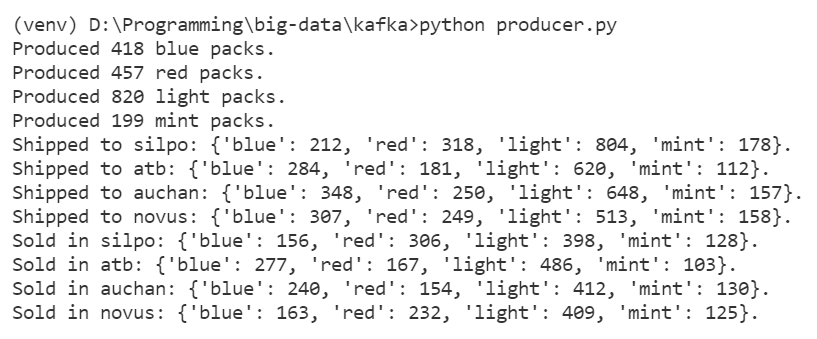
    func\_dict[msg.topic](msg.value)

### Результат виконання

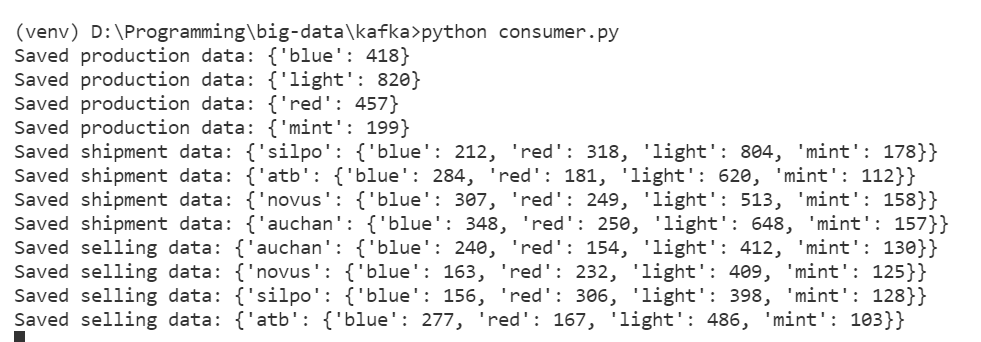
Запущені контейнери:



Результат роботи виробника:



Результат роботи споживача:



### Контрольні питання

*1. Що таке Apache Kafka?*

Apache Kafka — це фреймворкова реалізація програмної шини з використанням потокової обробки. Це програмна платформа з відкритим вихідним кодом, розроблена Apache Software Foundation, написана на Scala та Java. Проект спрямований на створення уніфікованої високопродуктивної платформи з низькою затримкою для обробки каналів даних у реальному часі.

*2. Типи API у Apache Kafka?*

Kafka включає п'ять основних apis:

* API Producer дозволяє програмам надсилати потоки даних темам у кластері Kafka.
* Consumer API дозволяє програмам зчитувати потоки даних з тем у кластері Kafka.
* API Streams дозволяє перетворювати потоки даних з вхідних тем у вихідні теми
* API Connect дозволяє реалізувати конектори, які постійно витягують з певної вихідної системи або програми в Kafka або передають з Kafka в якусь систему або додаток.
* API адміністратора дозволяє керувати та перевіряти теми, брокерів та інші об’єкти Kafka.

*3. Структура запису у потоці даних Apache Kafka?*

Ключ, значення та час

*4. На якому протоколі реалізована комунікація між вузлами системи*

*Apache Kafka?*

Kafka — це розподілена система, що складається з серверів і клієнтів, які спілкуються через високопродуктивний мережевий протокол TCP.

*5. Як досягається підвищений рівень відмовостійкості при обробці даних?*

Щоб зробити дані відмовостійкими та високодоступними, кожну тему можна реплікувати, щоб завжди було кілька брокерів, які мають копію даних на випадок, якщо щось піде не так. Звичайним виробничим налаштуванням є коефіцієнт реплікації 3, тобто завжди буде три копії даних. Ця реплікація виконується на рівні розділів теми.

*6. У якій послідовності «Споживачі» можуть обробляти дані?*

В залежності від часу повідомлення та в залежності від його ключа

*7. Яку роль виконують “Partitions of a Topic” Apache Kafka? 8. Як реалізована розподіленість Apache Kafka?*

Теми розділені на розділи, що означає, що тема розподілена на кілька «бакетів», розташованих у різних брокерів Kafka. Таке розподілене розміщення даних дуже важливо для масштабованості, оскільки воно дозволяє клієнтським програмам як читати, так і записувати дані від/до багатьох брокерів одночасно. Коли нова подія публікується до теми, вона фактично додається до одного з розділів теми. Події з однаковим ключем події (наприклад, ідентифікатор клієнта або автомобіля) записуються в той самий розділ, і Kafka гарантує, що будь-який споживач даного розділу теми завжди читатиме події цього розділу точно в тому ж порядку, в якому вони були записані.

*9. Як відбувається синхронізація серверів Apache Kafka, «виробників» та*

*«споживачів»?*

За допомогою ZooKeeper — розподіленої ієрархічної файлової системи, яка полегшує погане з'єднання між клієнтами та забезпечує надзвичайно послідовне представлення своїх znodes, які подібні до файлів і каталогів у традиційній файловій системі. Він забезпечує основні операції, такі як створення, видалення та перевірка існування znodes. Він забезпечує модель, керовану подіями, в якій клієнти можуть спостерігати за змінами в певних вузлах, наприклад, якщо до існуючого вузла додається новий дочірній вузол. ZooKeeper досягає високої доступності, запускаючи кілька серверів ZooKeeper, які називаються ансамблем, при цьому кожен сервер містить у пам'яті копію розподіленої файлової системи для обслуговування запитів на читання клієнта.

*10. Що таке принцип відсутності копіювання (англ. zero-copy)? Як він*

*використовується у Apache Kafka.*

Щоб забезпечити високу швидкість операцій запису на диск та читання, Kafka використовує не класичну передачу даних, а технологію під назвою "Zero-copy", коли ЦП не копіює дані з однієї області пам'яті в іншу, а працює з прямим доступом до пам'яті (DMA, direct memory access) та відображенням у пам'яті (memory mapping), а також зі сторінковим кешем.

*11.Сфери застосування Apache Kafka?*

Kafka добре працює як заміна більш традиційного брокера повідомлень, для відстеження активності на веб-сайті, оперативного моніторингу даних.

Багато користувачів Kafka обробляють дані в конвеєрах обробки, що складаються з кількох етапів, де вихідні дані споживаються з тем Kafka, а потім агрегуються, збагачуються або іншим чином перетворюються на нові теми для подальшого використання або подальшої обробки.