**Техническое задание: Развертывание Kafka и PostgreSQL, тестирование нагрузки и разработка заглушки**

**Задание №1.1: Развертывание Kafka и PostgreSQL**

**Подготовка тестовой среды**

**1. Установка Kafka:**

/opt/kafka/bin/kafka-topics.sh --version

3.9.0

**2. Установка Zookeeper:**

echo srvr | nc localhost 2181 | grep "Zookeeper version"

Zookeeper version: 3.8.4-9316c2a7a97e1666d8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC

**3. Запуск Zookeeper:**

cd /opt/kafka

bin/zookeeper-server-start.sh -daemon config/zookeeper.properties

**4. Проверка работы Zookeeper:**

Информация о сервере и другие метрики

echo srvr | nc localhost 2181

Состояние и структура данных

bin/zookeeper-shell.sh localhost:2181 ls /

**5. Запуск Kafka:**

/opt/kafka/bin/kafka-server-start.sh -daemon config/server.properties

**6. Проверка созданных топиков:**

/opt/kafka/bin/kafka-topics.sh --list --bootstrap-server localhost:9092

Вывод:

Franz\_Kafka

\_\_consumer\_offsets

test-topic

**7. Проверка нагрузки топика:**

bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic Franz\_Kafka --from-beginning

**Установка и запуск PostgreSQL**

Запуск PostgreSQL:

1. verakolkutina@Air-Vera ~ % brew services start postgresql

Проверка версии PostgreSQL:

1. verakolkutina@Air-Vera ~ % psql --version

psql (PostgreSQL) 17.2

Подключение к БД:

1. psql -h localhost -U verakolkutina -d antares\_db

Проверка существующих таблиц:

1. antares\_db=# \dt

Проверка структуры таблицы message\_record:

1. antares\_db=# \d message\_record;

### Задание №1.2: Тестирование нагрузки Kafka

**Функциональные требования**

* Используется **JMeter**.
* Скрипт Kafka\_Axen.jmx отправляет сообщения в Kafka с рандомным msg\_id.

**1. Проверка топиков Kafka:**

bin/kafka-topics.sh --list --bootstrap-server localhost:9092

Franz\_Kafka

\_\_consumer\_offsets

my\_topic

test-topic

**2. Чтение сообщений из топика Franz\_Kafka:**

bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic Franz\_Kafka --from-beginning

**Вывод сообщений:**

{"msg\_id": "0729148312", "timestamp": "1736696289419", "method": "POST", "uri": "/post-message"}

{"msg\_id": "1273105705", "timestamp": "1736696289761", "method": "POST", "uri": "/post-message"}

^CProcessed a total of 3888 messages

**3. Сценарий нагрузки:**

* 4 ступени по 5 минут.
* Первая ступень: 0.5 запросов/сек.
* Вторая ступень: 1 запрос/сек.
* Третья ступень: 2 запроса/сек.
* Четвертая ступень: 3 запроса/сек.

**Задание №1.3: Разработка заглушки на Java (Spring)**

**1. Функциональные требования**

* Заглушка постоянно слушает топик Kafka Franz\_Kafka.
* Извлекает msg\_id и timestamp из сообщения. Сохраняет их в БД antares\_db , таблицу message\_record

#### 2. Реализация

* **Файл заглушки:** kafkalistener-0.0.1-SNAPSHOT.jar
* **Проверка структуры БД:**

antares\_db=# \dt

**Вывод:**

Schema | Name | Type | Owner

--------+----------------+-------+---------------

public | kafka\_message | table | verakolkutina

public | kafka\_messages | table | verakolkutina

**public | message\_record | table | verakolkutina**

public | messages | table | verakolkutina

public | my\_table | table | verakolkutina

(5 rows)

**2. Проверка структуры таблицы message\_record:**

antares\_db=# \d message\_record;

**Вывод:**

Table "public.message\_record"

Column | Type | Collation | Nullable | Default

---------+------------------------+-----------+----------+---------

msg\_id | character varying(255) | | not null |

time\_rq | character varying(255) | | |

Indexes:

"message\_record\_pkey" PRIMARY KEY, btree (msg\_id)

**3. Проверка записанных данных:**

antares\_db=# SELECT \* FROM message\_record;

**Вывод:**

msg\_id | time\_rq

------------+---------------

2038602471 | 1736620108734

1854489227 | 1736620110876

0764923568 | 1736620112886

1788240451 | 1736620114895

2013079895 | 1736620116904

#### (и далее)

#### Мониторинг Kafka

Для мониторинга используется **Kafka Tool (Offset Explorer)**.

**Задание №1.4: Установка pg\_stat\_statements и анализ запросов**

**1. Проверка установки pg\_stat\_statements**

antares\_db=# SELECT \* FROM pg\_extension WHERE extname = 'pg\_stat\_statements';

Вывод:

oid | extname | extowner | extnamespace | extrelocatable | extversion | extconfig | extcondition

------+--------------------+----------+--------------+----------------+------------+-----------+--------------

24589 | pg\_stat\_statements | 10 | 2200 | t | 1.9

статистика собирается:

antares\_db=# SELECT count(\*) FROM pg\_stat\_statements;

count

-------

31

(1 row)

**2. Анализ top-запросов**

antares\_db=# SELECT query, calls, total\_exec\_time AS total\_time, mean\_exec\_time AS mean\_time, rows

FROM pg\_stat\_statements

ORDER BY mean\_exec\_time DESC;

SELECT query, calls, total\_exec\_time AS total\_time, mean\_exec\_time AS mean\_time, rows

FROM pg\_stat\_statements

ORDER BY calls DESC, mean\_exec\_time DESC;

Вывод: файл DB.txt

**Вывод**

В ходе выполнения технического задания были развернуты и настроены локально Kafka, Zookeeper и PostgreSQL. Проведено тестирование Kafka с помощью JMeter, разработана заглушка на Java (Spring) и выполнен анализ производительности SQL-запросов с помощью расширения pg\_stat\_statements.