**Задание №1.1**

Развернуть у себя kafka (можно локально, можно в докере), запустить и создать топик. С ним будет вестись дальнейшая работа в рамках пунктов 1.2 и 1.3.

Для выполнения задания понадобится вручную поднять Kafka и Zookeeper. Инструкции в интернете в помощь. Чтобы просмотреть очередь сообщений в кафке, можно воспользоваться Kafka Tool.

Развернуть инстанс PostgreSQL, создать в нем произвольную таблицу со столбцами msgUuid, head, timeRq.

**Задание №1.2**

Задание требует выполненного пункта №1.1, т.к. в нем потребуется нагрузить кафку, которую мы ранее настраивали. Желательно использовать инструмент jMeter (опционально допускается любой другой инструмент НТ), чтобы написать тестовый скрипт и сценарий нагрузки:

1. Скрипт должен отправлять в кафку сообщение, каждый раз используя случайный msg\_uuid в теле запроса. У каждого 10 сообщения параметр head должен иметь значение false.

*Пример сообщения в топик Kafka: { "msg\_uuid": "e7b8f190-051a-11ee-9e98-63e09b591d3a", "head": true, "method": "POST", "uri": "/post-message" }*

1. Необходимо также реализовать сценарий нагрузки:
   1. Нагрузку подавать ступенями по 5 минут;
   2. Всего 4 ступени;
   3. На первой ступени нагрузки подаем запросы со скоростью 5 оп/с. На второй ступени 10 оп/с, на третьей – 12 оп/с, на четвертой – 14 оп/с.

**Задание №1.3**

Разработать заглушку на Java (Spring), которая обеспечивает обработку поступающих kafka-запросов и запись данных в подготовленную в п.1.1 БД.

1. Заглушка постоянно в ходе своей работы слушает топик kafka, созданный в п.1.1
2. Получив сообщение, заглушка должна достать из тела запроса msg\_uuid и head, и на их основе сформировать запись в БД, вписав их данные, соответственно в столбцы msgUuid, head. В столбец timeRq записать время (в формате UNIX) когда заглушка вычитала сообщение из топика;
3. Добавить многопоточную вычитку сообщений в заглушку
4. Заглушка выводит в лог событие о вычитке сообщения из топика Kafka и о созданной записи в БД

*Пример лога:*   
*YYYY-MM-DD HH:MM:SS.SSS – [Read from Kafka] { "msg\_uuid": "e7b8f190-051a-11ee-9e98-63e09b591d3a", "head": true, "method": "POST", "uri": "/post-message" }*

*YYYY-MM-DD HH:MM:SS.SSS – [Write to DB] { "msgUuid": "e7b8f190-051a-11ee-9e98-63e09b591d3a", "head": true, "timeRq": "1543622400" }*

Чтобы просмотреть очередь сообщений в кафке, можно воспользоваться Kafka Tool. Для контроля работы БД можно воспользоваться любым клиентом, например, DBeaver.

**Задание №2**

* Развернуть локально Grafana
* Подключить java spring actuator к заглушке, разработанной в п. 1.3
* Вывести JVM метрики заглушки в Grafana. Можно использовать любые связки (Prometheus, Micrometer). Дашборд можно любой настроить, например JVM (Micrometer); Micrometer Spring Throughput. Почитать про отображаемые метрики.
* Вывести метрики PostgreSQL в Grafana. Попробовать через postgres prometheus exporter, дашборд также любой, можно воспользоваться более-менее стандартным PostgreSQL Database. Почитать про отображаемые метрики.
* Вывести метрики Jmeter в Grafana с использованием любого стандартного дашборда с сайта Grafana Labs.

**Задание №3\***

* Для заглушки реализовать функционал задержки (например, создание записи в БД происходит с задержкой 1000мс после вычитки его из топика Kafka)
* Для заглушки реализовать функционал динамического изменения времени задержки (например, изменить задержку с 1000мс до любого значения, 500мс, 3000мс и т.д. не останавливая и не перезапуская заглушку)