**ЛАБОРАТОРНАЯ РАБОТА**

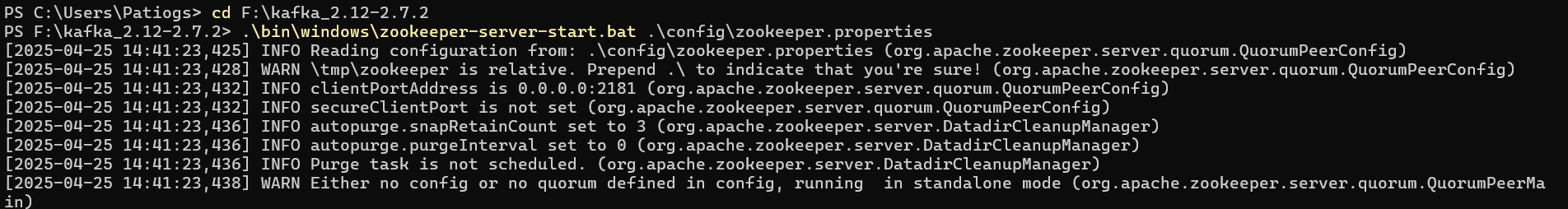
РАЗРАБОТКА СИСТЕМЫ ОБРАБОТКИ СООБЩЕНИЙ

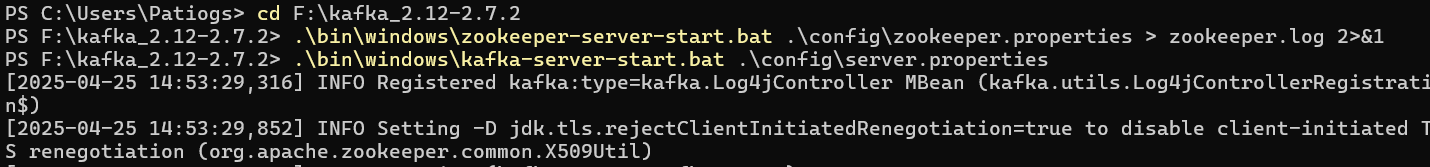
С ИСПОЛЬЗОВАНИЕМ APACHE KAFKA

**Цель работы**: Разработка системы асинхронной обработки сообщений с использованием Apache Kafka, включая создание продюсеров, консьюмеров и настройку их заимодействия.

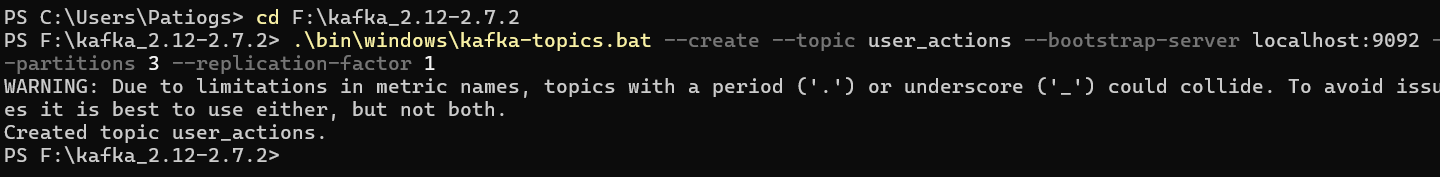
**1. Установка и настройка Kafka**

• Запустите ZooKeeper и Kafka-брокер (локально или через Docker).





• Создайте топик user\_actions с 3 партициями.



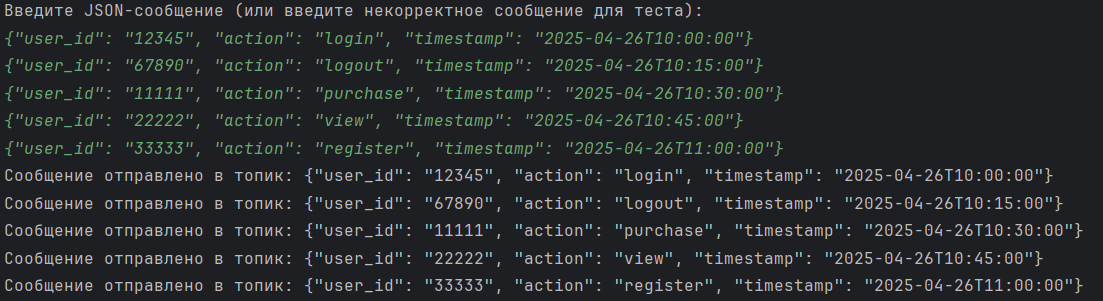
**2. Разработка продюсера**

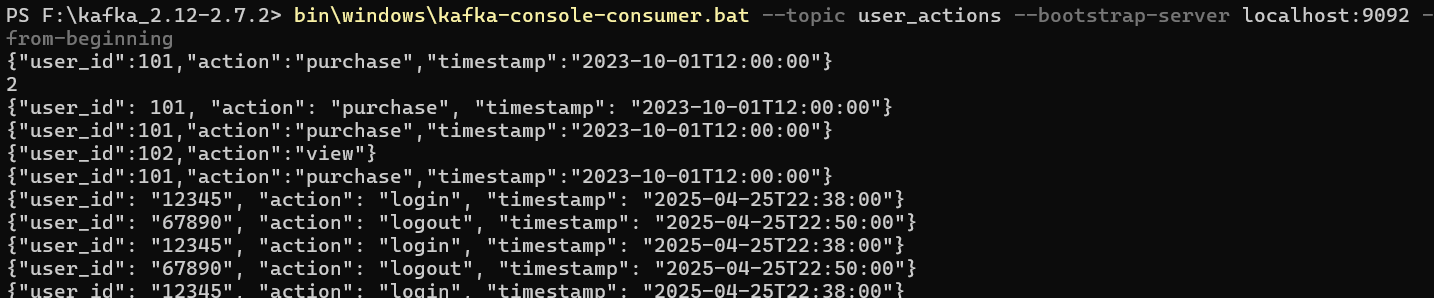
• Напишите приложение (**Java**/Python), отправляющее JSON-сообщения в

топик. Пример сообщения:

{"user\_id": 101, "action": "purchase", "timestamp": "2023-10-01T12:00:00"}

• Реализуйте ввод сообщений через консоль или чтение из файла.





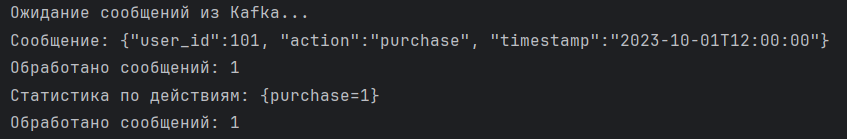
**3. Разработка консьюмера**

• Напишите консьюмер, обрабатывающий сообщения из топика:

- Фильтрация по типу действия (например, "purchase").

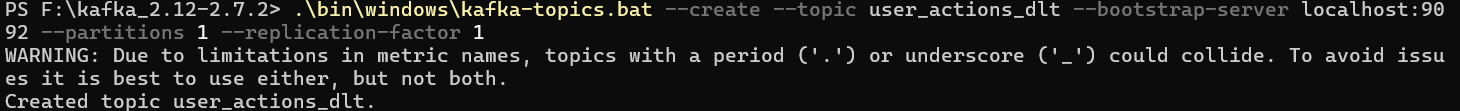
-Вывод статистики в консоль (количество сообщений, частые

действия).



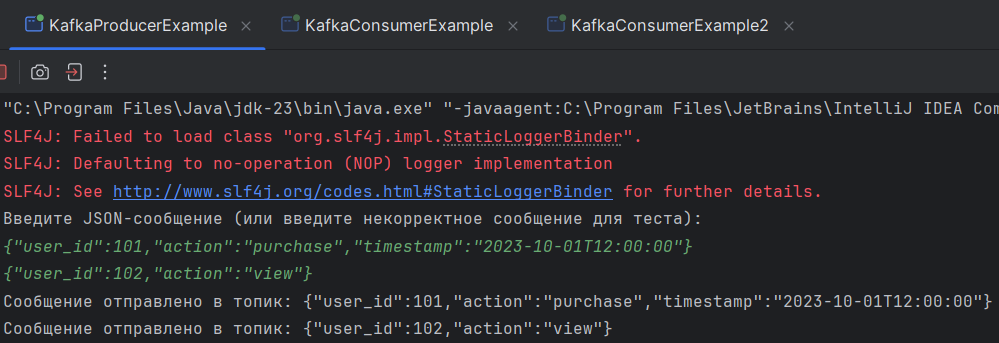
**4. Обработка ошибок и масштабирование**

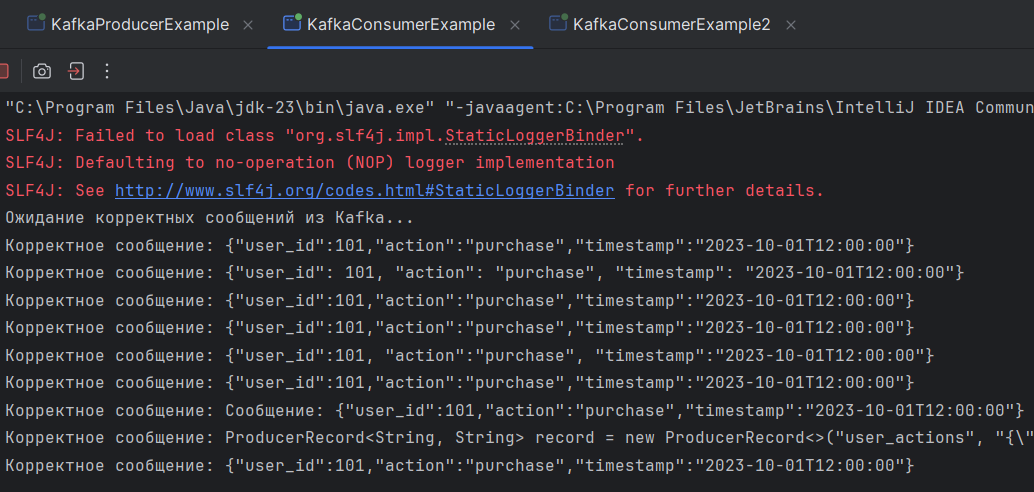
• Настройте Dead Letter Topic (DLT) для некорректных сообщений.

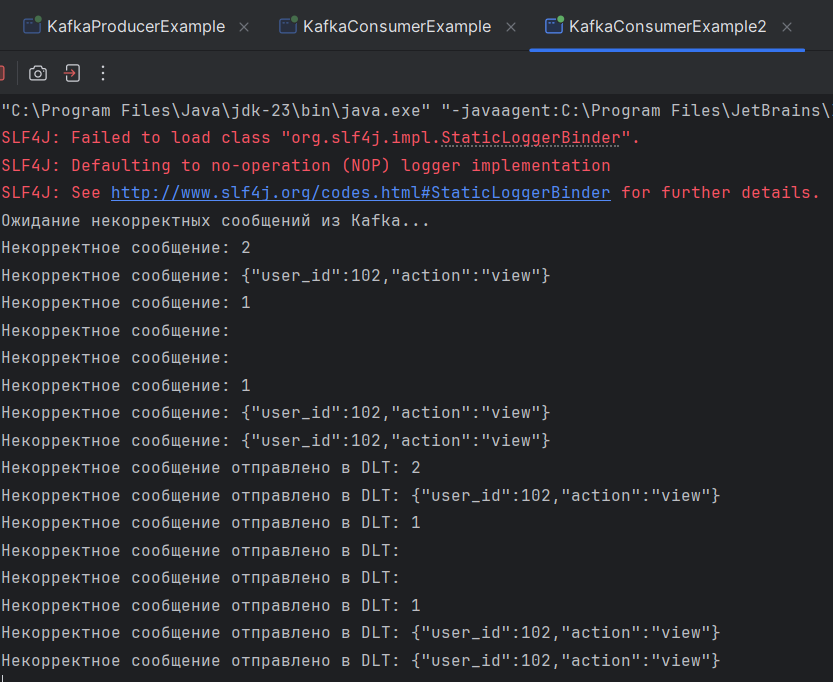


• Запустите 2 консьюмера в одной группе. Продемонстрируйте

распределение сообщений между ними.

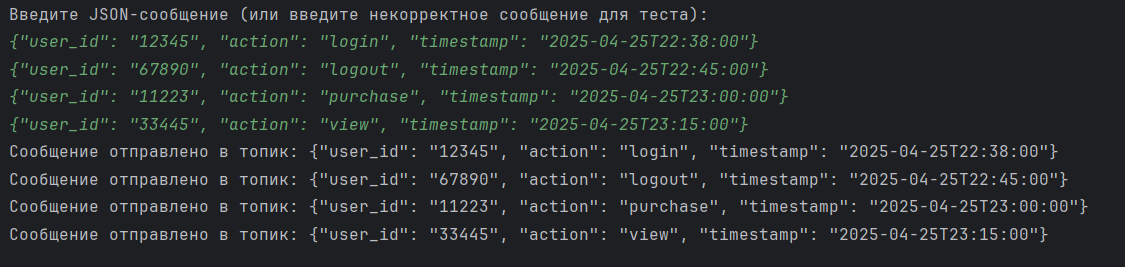


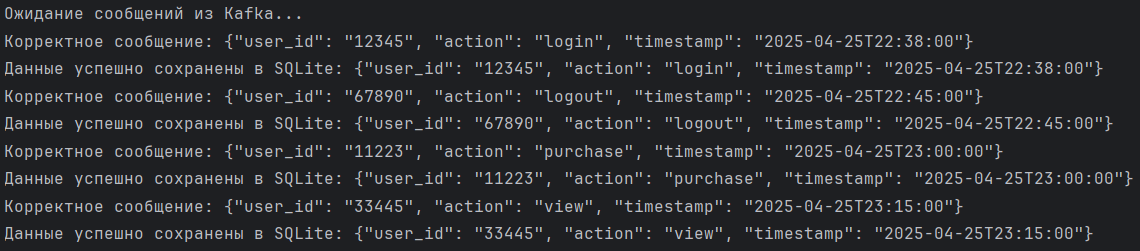


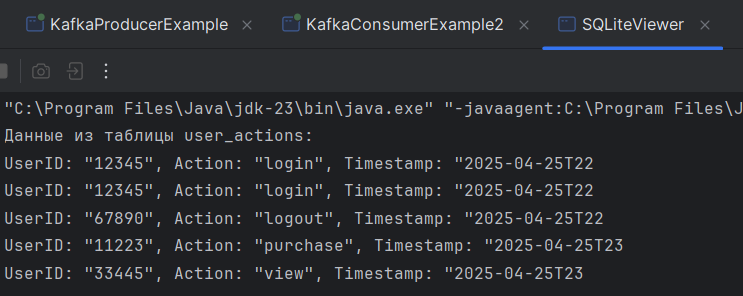


**5. Интеграция с внешней системой**

• Сохраняйте данные в PostgreSQL/MySQL (**SQLite**)







**Выводы**: Apache Kafka — мощная платформа для обработки и управления потоками данных, которая обладает рядом преимуществ, но в процессе её использования могут возникнуть сложности.

**Преимущества**:

1 - Высокая производительность и масштабируемость: Kafka поддерживает обработку огромных объёмов данных в режиме реального времени с минимальной задержкой. Она масштабируется горизонтально благодаря архитектуре распределённых брокеров и разделов (partitions).

2 - Надёжность данных: Kafka сохраняет данные на диске и имеет механизмы репликации, что обеспечивает устойчивость к сбоям и минимальный риск потери данных.

3 - Гибкость подключения: Поддерживаются многочисленные клиенты и интеграции с различными языками программирования (Java, Python, Go, и др.), а также популярными платформами, такими как Spark, Hadoop и Flink.

4 - Модель публикации/подписки: Kafka использует асинхронную модель публикации/подписки, что позволяет эффективную обработку потоков данных без непосредственной связи между отправителем и получателем.

5 - Поддержка событийной архитектуры: Благодаря Kafka, системы могут быть построены на основе событий, что упрощает реализацию микросервисной архитектуры.

**Возникшие сложности**:

1 - Ошибки при запуске команды, например, использование неправильного расширения на Windows (.bar вместо .bat).

2 - Сообщения из топика не отображаются.

3 - Недоступность удобных инструментов для управления базой, например, из-за отсутствия плагина Database Tools and SQL в используемой версии IntelliJ IDEA.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. **Зачем Kafka использует ZooKeeper?**

ZooKeeper необходим Kafka для управления кластером и координации между брокерами. Он обеспечивает отказоустойчивость, управляет метаданными, такими как список брокеров, распределение партиций и реплик, а также отвечает за выбор лидеров для каждой партиции. Кроме того, ZooKeeper помогает обнаруживать сбои брокеров и переназначать лидеров, что делает Kafka надёжной распределённой системой.

1. **Как обеспечить порядок сообщений в партиции?**

Чтобы гарантировать порядок сообщений в партиции, важно, чтобы все сообщения с одинаковым ключом отправлялись в одну и ту же партицию. Продюсер определяет ключи сообщений, что помогает Kafka выбрать целевую партицию. Кроме того, настройка параметра acks=all на стороне продюсера гарантирует, что сообщение будет зафиксировано всеми репликами перед подтверждением его записи.

1. **В чём разница между auto.offset.reset=earliest и latest?**

Параметр auto.offset.reset определяет, с какого момента начинать чтение сообщений, если текущее смещение (offset) потребителя отсутствует. Значение earliest заставляет потребителя читать все сообщения с самого начала топика. Напротив, latest позволяет обрабатывать только те сообщения, которые поступают после подключения потребителя.

1. **Как Kafka обеспечивает отказоустойчивость?**

Kafka реализует отказоустойчивость с помощью репликации данных. Каждая партиция хранится на нескольких брокерах, а одна из реплик назначается лидером. Если лидер выходит из строя, ZooKeeper автоматически переназначает лидера на другую реплику. Сообщения сохраняются на диске, что предотвращает их потерю. Механизм подтверждений записи (acks) гарантирует, что данные записаны на все реплики перед их использованием.