**ЛАБОРАТОРНАЯ РАБОТА №10**

**РАЗРАБОТКА СИСТЕМЫ ОБРАБОТКИ СООБЩЕНИЙ**

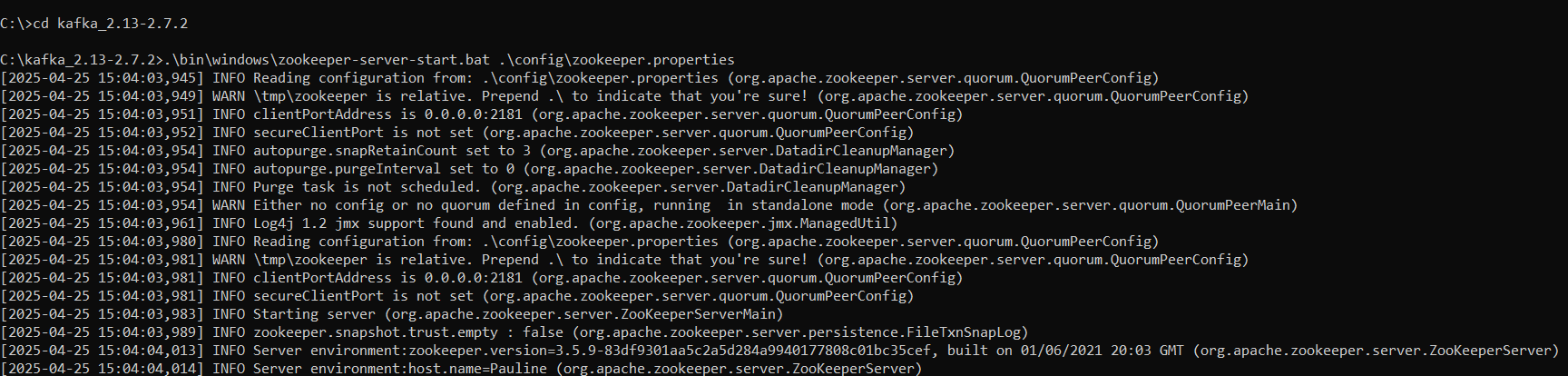
**С ИСПОЛЬЗОВАНИЕМ APACHE KAFKA**

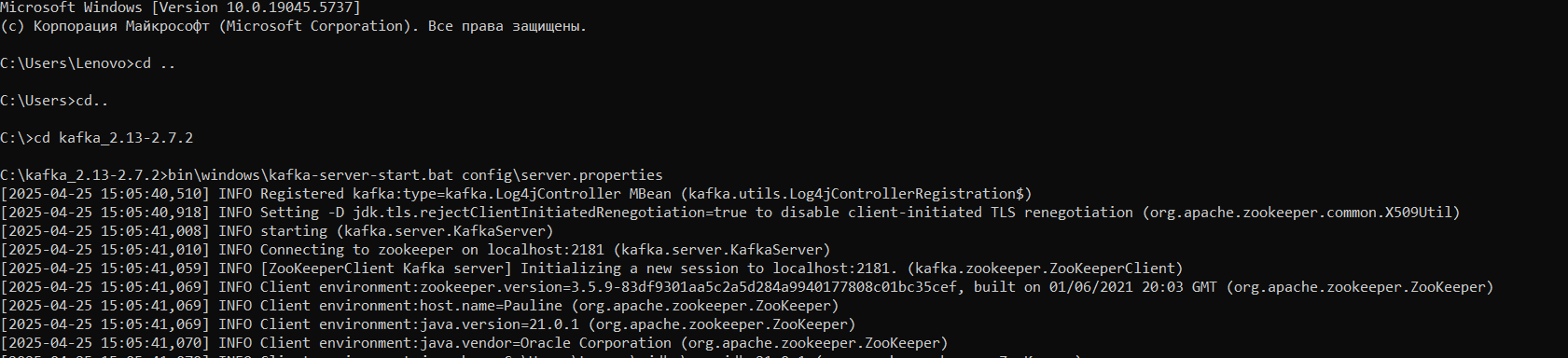
**Цель работы:** Разработка системы асинхронной обработки сообщений с использованием Apache Kafka, включая создание продюсеров, консьюмеров и настройку их взаимодействия.

**ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

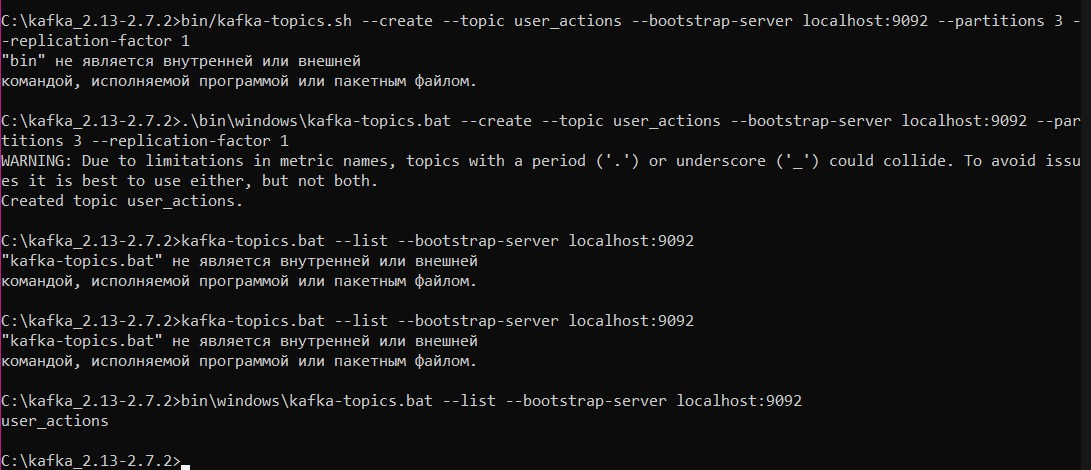
1. Установка и настройка Kafka

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



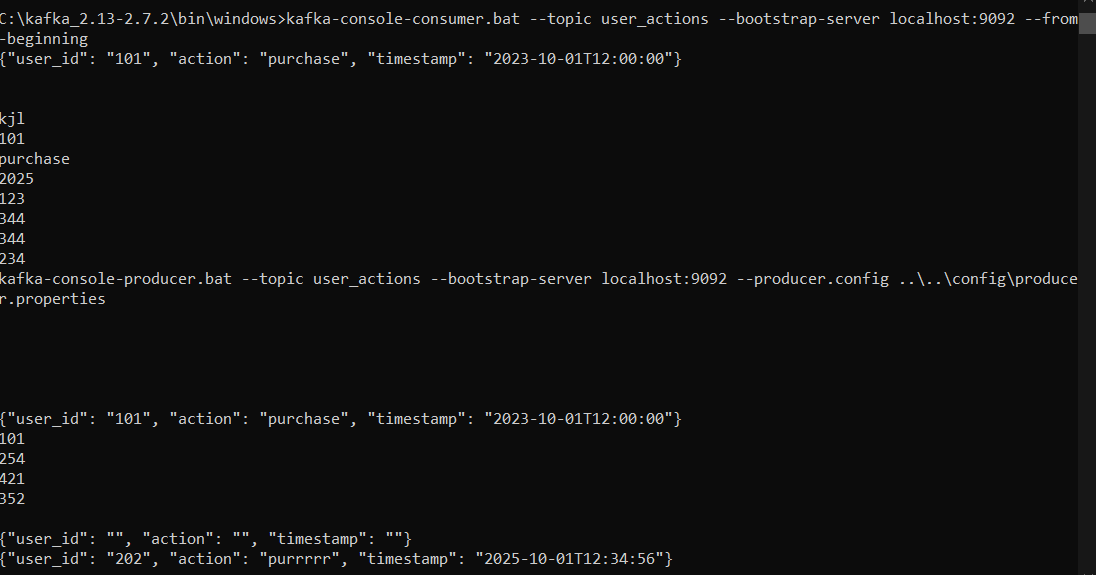


• Создайте топик user\_actions с 3 партициями.

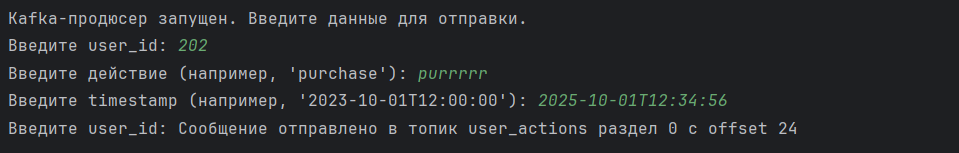


2. Разработка продюсера

• Напишите приложение (Java/Python), отправляющее JSON-сообщения в топик. Пример сообщения: {"user\_id": 101, "action": "purchase", "timestamp": "2023-10-01T12:00:00"}

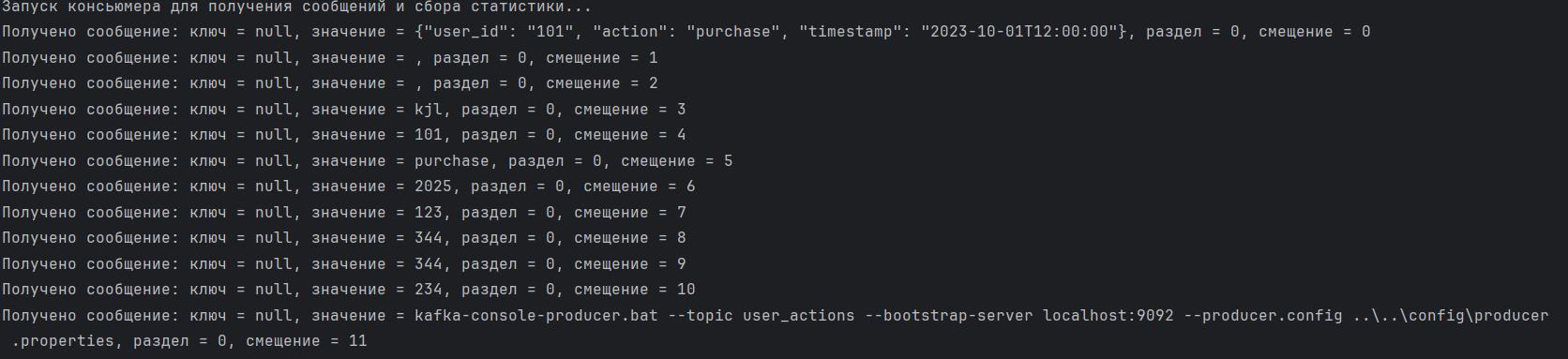


• Реализуйте ввод сообщений через консоль или чтение из файла.

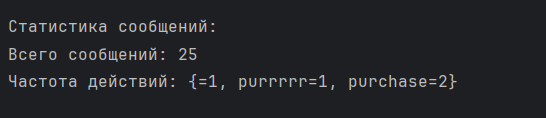


3. Разработка консьюмера

* Напишите консьюмер, обрабатывающий сообщения из топика: o Фильтрация по типу действия (например, "purchase").



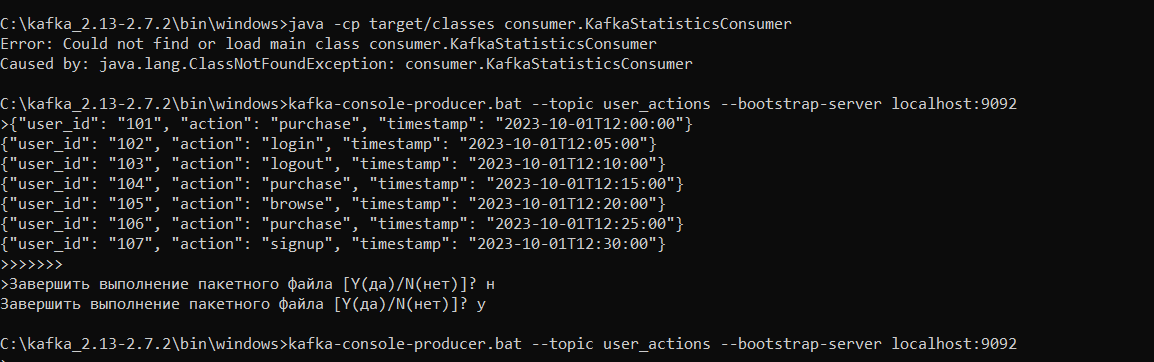
* Вывод статистики в консоль (количество сообщений, частые действия).



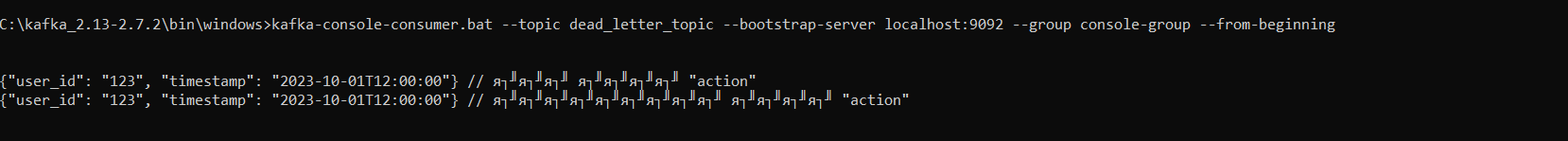
4. Обработка ошибок и масштабирование

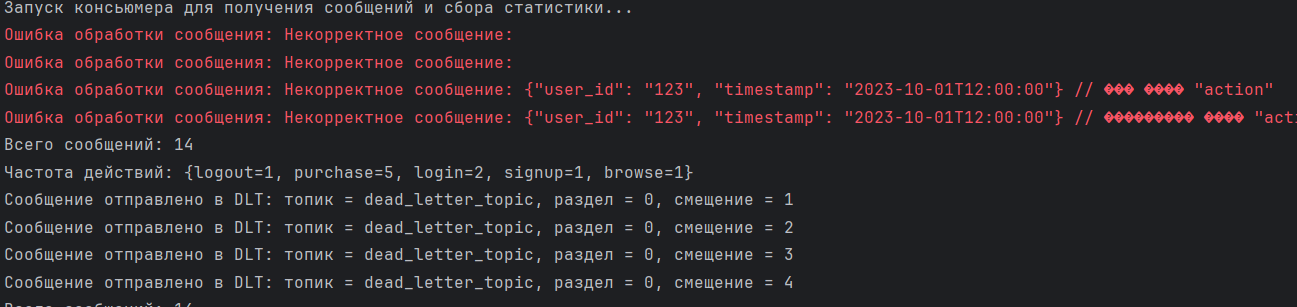
• Настройте Dead Letter Topic (DLT) для некорректных сообщений.

• Запустите 2 консьюмера в одной группе. Продемонстрируйте распределение сообщений между ними.



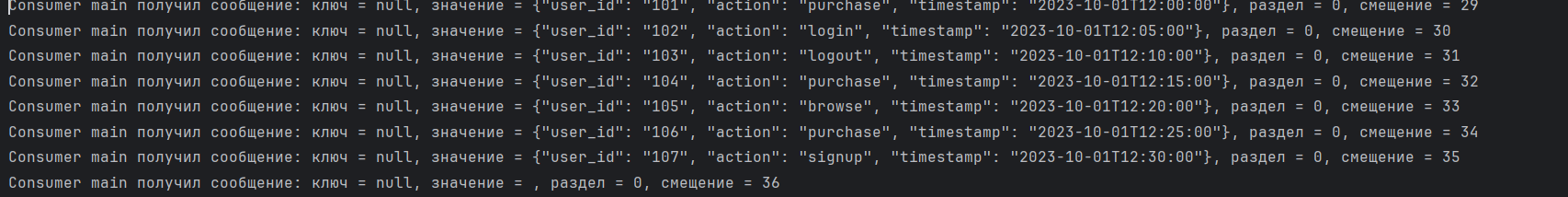
• Запустите 2 консьюмера в одной группе. Продемонстрируйте распределение сообщений между ними.

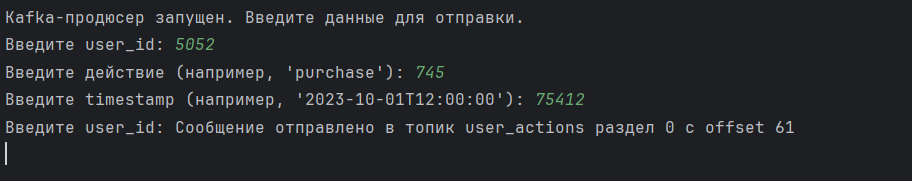


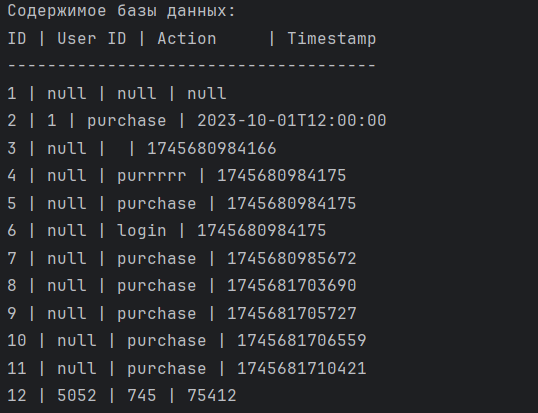


5. Интеграция с внешней системой

• Сохраняйте данные в PostgreSQL/MySQL.







**Преимущества Kafka**

1. **Масштабируемость**: Kafka обеспечивает высокую производительность и может обрабатывать миллионы сообщений в секунду благодаря архитектуре с разделами (partitions).
2. **Устойчивость**: Сообщения сохраняются на диск, что позволяет предотвратить их потерю при сбоях системы.
3. **Многопоточность**: Kafka поддерживает работу с несколькими потребителями и продюсерами, распределяя нагрузку между ними.
4. **Гибкость**: Подходит для самых разных задач: сбор логов, потоковая аналитика, обработка данных в реальном времени, интеграция микросервисов.
5. **Лёгкая интеграция**: Kafka поддерживает множество языков программирования и легко интегрируется с другими системами, такими как Spark, Hadoop, и базы данных.

**Возникшие сложности**

1. **Настройка инфраструктуры**: Установка и настройка Kafka требует понимания её архитектуры (ZooKeeper, брокеры, разделы).
2. **Работа с группами потребителей**: Необходима тщательная настройка, чтобы правильно распределять сообщения между консьюмерами и избегать конфликтов в смещении (offset).
3. **Обработка ошибок**: Настройка Dead Letter Topic (DLT) может быть сложной, если неправильно настроен продюсер или механизм отправки ошибок.
4. **Интеграция с внешними системами**: Например, подключение SQLite требует дополнительных зависимостей и продуманного подключения.
5. **Ребалансировка групп**: Частые ребалансировки могут вызывать временные задержки и потерю производительности.
6. **Парсинг данных**: Разбор JSON-сообщений внутри потребителя может вызвать трудности, если сообщения не стандартизированы.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Зачем Kafka использует ZooKeeper?

ZooKeeper используется для отслеживания состояния кластера Kafka, координации партиций между брокерами и обеспечения согласованности и отказоустойчивости.

2. Как обеспечить порядок сообщений в партиции?

Использовать один продюсер с одинаковым ключом для сообщений. Порядок сообщений гарантируется, если один продюсер отправляет сообщения с одинаковым ключом, так как все сообщения с этим ключом попадут в одну и ту же партицию и будут обработаны в порядке отправки.

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

Earliest начинает чтение с самого старого доступного сообщения, а latest — с самого нового. Когда смещение не задано, earliest позволяет потребителю начать чтение с первого доступного сообщения, тогда как latest начинает с сообщений, добавленных после подключения потребителя.

4. Как Kafka обеспечивает отказоустойчивость?

Использует репликацию партиций между брокерами. Репликация позволяет сохранить копии партиций на нескольких брокерах. В случае сбоя одного брокера, другой брокер с репликой партиции продолжает обработку.