## Lab 2: Building a Real-Time User Activity Tracker with Kafka

### Scenario

You are working for a fintech startup that needs to track user interactions in real-time. Your task is to implement a Kafka-based activity tracking system using the Kafka Java API.

### Software Required

* Apache Kafka 3.0+ installed and running locally
* JDK 11+
* Maven or Gradle
* Eclipse, IntelliJ IDEA or any Java IDE

### Instructions

#### Step 1: Create a Kafka Topic

Open a Command Prompt and create a topic called user-interactions.

**For Windows:**

%KAFKA\_HOME%\bin\windows\kafka-topics.bat --create --topic user-interactions ^  
 --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

**Or use the provided script:**

cd "Lab Code\Lab2\scripts"  
create-topic.bat

#### Step 2: Create the Producer Application

Create a Java class UserActivityProducer.java and implement a Kafka producer that sends the following types of events:

* "login"
* "view\_profile"
* "transfer\_initiated"

Each event should contain a userId, action, and timestamp.

#### Step 3: Create the Consumer Application

Create a Java class UserActivityConsumer.java that consumes from the user-interactions topic and logs each event to the console.

#### Step 4: Run the Producer and Consumer

**Option A: Using Eclipse (Recommended)** 1. Start the consumer in Eclipse: Right-click UserActivityConsumer.java → Run As → Java Application 2. Run the producer in Eclipse: Right-click UserActivityProducer.java → Run As → Java Application 3. Observe the messages being consumed in real-time.

**Option B: Using Windows Scripts** 1. Start the consumer: Open Command Prompt → Navigate to Lab Code\Lab2\scripts → Run run-consumer.bat 2. Run the producer: Open another Command Prompt → Navigate to Lab Code\Lab2\scripts → Run run-producer.bat 3. Observe the messages being consumed in real-time.

**Option C: Manual Command Line** 1. Start the consumer in one Command Prompt window. 2. Run the producer in another Command Prompt window. 3. Observe the messages being consumed in real-time.

### Expected Output

When running the consumer, you should see output similar to:

Received message: key = 12345, value = {"userId": "12345", "action": "login", "timestamp": "2025-07-18T10:00:00Z"}, partition = 0, offset = 0  
Received message: key = 67890, value = {"userId": "67890", "action": "transfer\_initiated", "timestamp": "2025-07-18T10:01:00Z"}, partition = 0, offset = 1

## Complete Implementation

### Project Structure

The complete project is available in the Lab Code/Lab2/ folder with the following structure:

Lab2/  
├── pom.xml # Maven configuration  
├── README.md # Setup and usage guide  
├── ECLIPSE\_SETUP.md # Eclipse-specific setup  
├── src/  
│ └── main/java/com/fintech/  
│ ├── UserActivityProducer.java # Kafka Producer  
│ ├── UserActivityConsumer.java # Kafka Consumer  
│ └── UserActivityEvent.java # Event model class  
└── scripts/  
 ├── create-topic.bat # Windows script to create Kafka topic  
 ├── run-producer.bat # Windows script to run producer  
 ├── run-consumer.bat # Windows script to run consumer  
 ├── start-kafka.bat # Windows script to start Kafka services  
 └── stop-kafka.bat # Windows script to stop Kafka services

### File: UserActivityProducer.java

import org.apache.kafka.clients.producer.\*; import org.apache.kafka.common.serialization.StringSerializer; import java.util.Properties; import java.util.Random;

public class UserActivityProducer { public static void main(String[] args) throws InterruptedException { Properties props = new Properties(); props.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, “localhost:9092”); props.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName()); props.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());

Producer<String, String> producer = new KafkaProducer<>(props);  
  
 String[] actions = {"login", "view\_profile", "transfer\_initiated"};  
 Random rand = new Random();  
  
 for (int i = 0; i < 10; i++) {  
 String userId = String.format("%05d", rand.nextInt(100000));  
 String action = actions[rand.nextInt(actions.length)];  
 String timestamp = java.time.LocalDateTime.now().toString();  
  
 String json = String.format("{\"userId\": \"%s\", \"action\": \"%s\", \"timestamp\": \"%s\"}",  
 userId, action, timestamp);  
  
 ProducerRecord<String, String> record = new ProducerRecord<>("user-interactions", userId, json);  
 producer.send(record, (metadata, exception) -> {  
 if (exception != null) {  
 System.err.println("Error sending message: " + exception.getMessage());  
 } else {  
 System.out.printf("Sent: %s%n", json);  
 }  
 });  
  
 Thread.sleep(1000); // Simulate real-time delay  
 }  
  
 producer.close();  
}

}

---  
  
### File: `UserActivityConsumer.java`  
  
import org.apache.kafka.clients.consumer.\*;  
import org.apache.kafka.common.serialization.StringDeserializer;  
import java.time.Duration;  
import java.util.Collections;  
import java.util.Properties;  
  
public class UserActivityConsumer {  
 public static void main(String[] args) {  
 Properties props = new Properties();  
 props.put(ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092");  
 props.put(ConsumerConfig.GROUP\_ID\_CONFIG, "activity-consumer-group");  
 props.put(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());  
 props.put(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());  
 props.put(ConsumerConfig.AUTO\_OFFSET\_RESET\_CONFIG, "earliest");  
  
 Consumer<String, String> consumer = new KafkaConsumer<>(props);  
 consumer.subscribe(Collections.singletonList("user-interactions"));  
  
 try {  
 while (true) {  
 ConsumerRecords<String, String> records = consumer.poll(Duration.ofMillis(100));  
 for (ConsumerRecord<String, String> record : records) {  
 System.out.printf("Received: %s%n", record.value());  
 }  
 }  
 } finally {  
 consumer.close();  
 }  
 }  
}

## Windows-Specific Setup

### Prerequisites for Windows:

1. **Apache Kafka** installed and KAFKA\_HOME environment variable set
2. **JDK 11+** installed and JAVA\_HOME environment variable set
3. **Maven 3.6+** installed and added to PATH
4. **Eclipse IDE** with Maven plugin

### Quick Start for Windows:

1. **Start Kafka Services:**

* cd "Lab Code\Lab2\scripts"  
  start-kafka.bat

1. **Create Topic:**

* create-topic.bat

1. **Run Applications:**
   * **Consumer:** run-consumer.bat
   * **Producer:** run-producer.bat (in another Command Prompt)

### Environment Variables:

Set the following environment variables in Windows: - KAFKA\_HOME=C:\path\to\kafka - JAVA\_HOME=C:\Program Files\Java\jdk-11 - Add %KAFKA\_HOME%\bin\windows to PATH

### Troubleshooting Windows Issues:

1. **Kafka not starting:** Check if KAFKA\_HOME is set correctly
2. **Java not found:** Verify JAVA\_HOME and PATH settings
3. **Permission denied:** Run Command Prompt as Administrator
4. **Port conflicts:** Ensure ports 2181 (Zookeeper) and 9092 (Kafka) are available