使用 Apache Kafka 於 Java

使用 Apache Kafka 在 Java 中涉及在您的 Java 應用程式中設置 Kafka 客戶端以生產和消費訊息。Kafka 是一個分佈式流平台，其 Java 客戶端程式庫使其簡單地整合。以下，我將帶您了解基本步驟以開始。

首先，您需要設置您的環境。確保您已經在系統或伺服器上安裝並運行 Kafka。您可以從官方 Apache Kafka 網站下載它，並使用提供的腳本啟動 ZooKeeper 和 Kafka 伺服器。為了簡化，我假設您在本地以預設設置運行 Kafka（例如，`localhost:9092` 作為引導伺服器）。

接下來，將 Kafka 客戶端依賴項添加到您的 Java 專案中。如果您使用 Maven，請在 `pom.xml` 中包含以下內容：

```
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>3.6.0</version> <!-- 使用最新版本 -->
</dependency>
```

現在，讓我們編寫一些程式碼。我將向您展示如何創建簡單的生產者和消費者。

Kafka 生產者範例

生產者將訊息發送到 Kafka 主題。以下是基本範例：

```
import org.apache.kafka.clients.producer.*;
import java.util.Properties;

public class SimpleProducer {
    public static void main(String[] args) {
        // 配置生產者屬性
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092"); // Kafka 伺服器地址
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        // 創建生產者實例
        try (Producer<String, String> producer = new KafkaProducer<>(props)) {
            // 將訊息發送到名為 "test-topic" 的主題
            String topic = "test-topic";
            for (int i = 0; i < 10; i++) {
                String key = "key" + i;
                String value = "Hello, Kafka " + i;
                ProducerRecord<String, String> record = new ProducerRecord<>(topic, key, value);
```

```
                producer.send(record, (metadata, exception) -> {
                    if (exception == null) {
                        System.out.println("Sent message: " + value + " to partition " + metadata.partition());
                    } else {
                        exception.printStackTrace();
                    }
                });
            }
        }
    }
}
```

在這段程式碼中：- `bootstrap.servers` 指定 Kafka 運行的位置。- 串行化器定義如何將鍵和值（這裡都是字串）轉換為位元組。- `ProducerRecord` 表示訊息，`send()` 以非同步方式將其發送，並使用回調處理成功或失敗。

Kafka 消費者範例

消費者訂閱主題並讀取訊息。以下是範例：

```
import org.apache.kafka.clients.consumer.*;
import java.util.Collections;
import java.util.Properties;

public class SimpleConsumer {
    public static void main(String[] args) {
        // 配置消費者屬性
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("group.id", "test-group"); // 消費者群組 ID
        props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        props.put("auto.offset.reset", "earliest"); // 從主題的開始處開始

        // 創建消費者實例
        try (KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props)) {
            // 訂閱主題
            consumer.subscribe(Collections.singletonList("test-topic"));

            // 取消訊息
            while (true) {
```

```
                ConsumerRecords<String, String> records = consumer.poll(1000); // 以毫秒為單位的超時
                for (ConsumerRecord<String, String> record : records) {
                    System.out.printf("Received: key=%s, value=%s, partition=%d, offset=%d%n",
                            record.key(), record.value(), record.partition(), record.offset());
                }
            }
        }
    }
}
```

在這裡：- `group.id` 為協調的消費者群組。- 反串行化器將位元數據轉換回字串。- `poll()` 在循環中獲取訊息，`auto.offset.reset` 確保您從最早的訊息開始，如果沒有提交偏移量。

運行程式碼

1. **啟動 Kafka**：確保 ZooKeeper 和 Kafka 正在運行(例如，在類 Unix 系統上使用 `bin/zookeeper-server-start.sh config/zookeeper.properties` 和 `bin/kafka-server-start.sh config/server.properties`)。
2. **創建主題**：使用 Kafka 命令行工具創建 `test-topic`（如果不存在）：`bin/kafka-topics.sh --create --topic test-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1`。
3. **運行生產者**：執行 `SimpleProducer` 類以發送訊息。
4. **運行消費者**：運行 `SimpleConsumer` 類以查看消費的訊息。

提示

- **錯誤處理**：為生產代碼添加 try-catch 塊或記錄。
- **配置**：根據您的需求調整屬性，例如 `acks`（用於生產者可靠性）或 `max.poll.records`（用於消費者批次大小）。
- **程式庫**：對於更複雜的應用程式，請考慮使用 Spring Kafka，它簡化了配置和使用。

這是基本設置，讓您開始。如果您想深入了解特定功能，例如分區、錯誤處理或與更大應用程式的整合，請告訴我！