Lab 8. Building ETL Pipelines Using Kafka

In this lab, we will look into creating an ETL pipeline using these tools and Kafka Connect use cases and examples. In this lab, we will cover Kafka Connect in detail.

Using Kafka Connect

Kafka Connect provides us with various Connectors, and we can use the Connectors based on our use case requirement. It also provides an API that can be used to build your own Connector. We will go through a example in this section.

Note: Confluent services (started in lab 7) should be stopped before proceeding:

Stop Confluent Platform

When you are done working with the local install, you can stop Confluent Platform. Open new terminal and go to following directory:

```
cd /headless/kafka-advanced/confluent-6.1.1/bin
```

1. Stop Confluent Platform using the Confluent CLI.

   ```
   ./confluent local services stop
   ```

2. Destroy the data in the Confluent Platform instance with the confluent local destroy command.

   ```
   ./confluent local destroy
   ```

Start Kafka, Zookeeper and Schema Registry

Open new terminal and start zookeeper:

```
~/kafka-advanced/run-zookeeper.sh
```

Open new terminal and start kafka:

```
~/kafka-advanced/run-kafka.sh
```

Open new terminal and start schema registry:

```
~/kafka-advanced/run-schema_registry.sh
```

JDBC Sink Connector

The JDBC sink connector allows you to export data from Kafka topics to any relational database with a JDBC driver. By using JDBC, this connector can support a wide variety of databases without requiring a dedicated connector for each one. The connector polls data from Kafka to write to the database based on the topics subscription. It is possible to achieve idempotent writes with upserts. Auto-creation of tables, and limited auto-evolution is also supported.

Task

To see the basic functionality of the connector, we'll be copying Avro data from a single topic to a local SQLite database. This example assumes you are running Kafka and Schema Registry locally on the default ports.

```
cd /headless/kafka-advanced/confluent-6.1.1
```

Let's create a configuration file for the connector. This file is included with the connector in `./etc/kafka-connect-jdbc/sink-quickstart-sqlite.properties` and contains the following settings:

```
name=test-sink
connector.class=io.confluent.connect.jdbc.JdbcSinkConnector
tasks.max=1
topics=orders
connection.url=jdbc:sqlite:test.db
auto.create=true
```

The first few settings are common settings you'll specify for all connectors, except for `topics` which is specific to sink connectors like this one. The `connection.url` specifies the database to connect to, in this case a local SQLite database file. Enabling `auto.create` allows us to rely on the connector for creating the table.

Now we can run the connector with this configuration.

```
$ cd /headless/kafka-advanced/confluent-6.1.1

$ ./bin/connect-standalone etc/schema-registry/connect-avro-standalone.properties
etc/kafka-connect-jdbc/sink-quickstart-sqlite.properties
```

Now, we will produce a record into the orders topic. Open new terminal and execute following command:

```
$ cd /headless/kafka-advanced/confluent-6.1.1

$ bin/kafka-avro-console-producer \
--broker-list localhost:9092 --topic orders \
--property value.schema='{"type":"record","name":"myrecord","fields":
[{"name":"id","type":"int"},{"name":"product", "type": "string"}, {"name":"quantity",
"type": "int"}, {"name":"price",
"type": "float"}]}'
```

The console producer is waiting for input. Copy and paste the following record into the terminal:

```
{"id": 999, "product": "foo", "quantity": 100, "price": 50}
```

Now if we query the database, we will see that the orders table was automatically created and contains the record.

```
$ sqlite3 test.db

sqlite> select * from orders;
```

Output: `foo|50.0|100|999`

Summary

In this lab, we learned about Kafka Connect in detail. In the next lab, you will learn about Kafka Stream in detail, and we will also see how we can use Kafka stream API to build our own streaming application. We will explore the Kafka Stream API in detail and focus on its advantages.