Lab 7-2. Playing with Avro using Schema Registry

Schema Registry allows you to store Avro schemas for both producers and consumers. It also provides a RESTful interface for accessing this schema. It stores all the versions of Avro schema, and each schema version is assigned a schema ID.

When the producer sends a record to Kafka topic using Avro Serialization, it does not send an entire schema, instead, it sends the schema ID and record. The Avro serializer keeps all the versions of the schema in cache and stores data with the schemas matching the schema ID.

The consumer also uses the schema ID to read records from Kafka topic, wherein the Avro deserializer uses the schema ID to deserialize the record.

Here is an example of Avro schema and producer:

```
kafka-avro-console-producer \
 --broker-list localhost:9092 --topic test \
 --property value.schema='{"type":"record","name":"testrecord","fields":
[{"name":"country","type":"string"}]}'
```

Similarly, an example of Avro schema on consumer:

```
kafka-avro-console-consumer --topic test\
         --Zookeeper localhost:2181 \
         --from-beginning
```

Remember that if the Schema Registry is up and running, the consumer will be able to deserialize the record. Any attempt to push invalid or non-compatible records to Kafka topic will result in an exception.

The schema can also be registered using a REST request as follows:

```
 curl -X POST -H "Content-Type: application/vnd.schemaregistry.v1+json" \
 --data '{"schema": "{\"type\": \"string\"}"}' \
 http://localhost:8081/test/Kafka-test/versions
 {"id":1}
```

The ability of the Schema Registry to keep all the versions of the schema and configure their compatibility settings makes Schema Registry more special. Schema registry is very easy to use, and it also removes bottlenecks of the data format issue in a loosely coupled producer and consumer environment.