**Kafka\_Assignment**

1.Setup Confluent Kafka Account

Ans:The Kafka Account has been set in confluent kafka using mail id

2. Create one kafka topic named as " restaraurent-take-away-data" with 3 partitions

Ans: I have created the topic with different name as “topic 11” with 3 partions

3. Setup key (string) & value (json) schema in the confluent schema registry

Ans:

{

"$id": "http://example.com/myURI.schema.json",

"$schema": "http://json-schema.org/draft-07/schema#",

"additionalProperties": false,

"description": "Sample schema to help you get started.",

"properties": {

"Item\_Name": {

"description": "The type(v) type is used.",

"type": "string"

},

"Order\_Date": {

"description": "The type(v) type is used.",

"type": "string"

},

"Order\_Number": {

"description": "The type(v) type is used.",

"type": "number"

},

"Product\_Price": {

"description": "The type(v) type is used.",

"type": "number"

},

"Quantity": {

"description": "The type(v) type is used.",

"type": "number"

},

"Total\_products": {

"description": "The type(v) type is used.",

"type": "number"

}

},

"title": "SampleRecord",

"type": "object"

}

4. Write a kafka producer program (python or any other language) to read data records from restaurent data csv file, make sure schema is not hardcoded in the producer code, read the latest version of schema and schema\_str from schema registry and use it for data serialization.

Ans:

import argparse  
from uuid import uuid4  
from six.moves import input  
from confluent\_kafka import Producer  
from confluent\_kafka.serialization import StringSerializer, SerializationContext, MessageField  
from confluent\_kafka.schema\_registry import SchemaRegistryClient  
from confluent\_kafka.schema\_registry.json\_schema import JSONSerializer  
# from confluent\_kafka.schema\_registry import \*  
import pandas as pd  
from typing import List  
  
FILE\_PATH = "/Users/nagar/OneDrive/Desktop/kafka classes/restuarant\_orders.txt"  
columns = ['Order\_Number','Order\_Date','Item\_Name','Quantity','Product\_Price','Total\_products']  
  
API\_KEY = 'ITBTATIBWRXJAZ5G'  
ENDPOINT\_SCHEMA\_URL = 'https://psrc-mw731.us-east-2.aws.confluent.cloud'  
API\_SECRET\_KEY = '+sKT3+XHHvhgzBsSRAKOgSyVLhtN4ktdmnIJPS1nKlesfyQK6Z+vXGorYdvhpJwp'  
BOOTSTRAP\_SERVER = 'pkc-6ojv2.us-west4.gcp.confluent.cloud:9092'  
SECURITY\_PROTOCOL = 'SASL\_SSL'  
SSL\_MACHENISM = 'PLAIN'  
SCHEMA\_REGISTRY\_API\_KEY = 'LP52E5CGC2IX7U75'  
SCHEMA\_REGISTRY\_API\_SECRET = 'ju7bR9JBZVDGQaTW9pujWnE4BenY3l1ZImNsuIPzNBAa6etD/ikDXXskZsP4xDK5'  
  
  
def sasl\_conf():  
 sasl\_conf = {'sasl.mechanism': SSL\_MACHENISM,  
 # Set to SASL\_SSL to enable TLS support.  
 # 'security.protocol': 'SASL\_PLAINTEXT'}  
 'bootstrap.servers': BOOTSTRAP\_SERVER,  
 'security.protocol': SECURITY\_PROTOCOL,  
 'sasl.username': API\_KEY,  
 'sasl.password': API\_SECRET\_KEY  
 }  
 return sasl\_conf  
  
  
def schema\_config():  
 return {'url': ENDPOINT\_SCHEMA\_URL,  
  
 'basic.auth.user.info': f"{SCHEMA\_REGISTRY\_API\_KEY}:{SCHEMA\_REGISTRY\_API\_SECRET}"  
  
 }  
  
  
class Car:  
 def \_\_init\_\_(self, record: dict):  
 for k, v in record.items():  
 setattr(self, k, v)  
  
 self.record = record  
  
 @staticmethod  
 def dict\_to\_car(data: dict, ctx):  
 return Car(record=data)  
  
 def \_\_str\_\_(self):  
 return f"{self.record}"  
  
  
def get\_car\_instance(file\_path):  
 df = pd.read\_csv(file\_path)  
 df = df.iloc[:, :]  
 cars: List[Car] = []  
 for data in df.values:  
 car = Car(dict(zip(columns, data)))  
 cars.append(car)  
 yield car  
  
  
def car\_to\_dict(car: Car, ctx):  
 *"""  
 Returns a dict representation of a User instance for serialization.  
 Args:  
 user (User): User instance.  
 ctx (SerializationContext): Metadata pertaining to the serialization  
 operation.  
 Returns:  
 dict: Dict populated with user attributes to be serialized.  
 """* # User.\_address must not be serialized; omit from dict  
 return car.record  
  
  
def delivery\_report(err, msg):  
 *"""  
 Reports the success or failure of a message delivery.  
 Args:  
 err (KafkaError): The error that occurred on None on success.  
 msg (Message): The message that was produced or failed.  
 """* if err is not None:  
 print("Delivery failed for User record {}: {}".format(msg.key(), err))  
 return  
 print('User record {} successfully produced to {} [{}] at offset {}'.format(  
 msg.key(), msg.topic(), msg.partition(), msg.offset()))  
  
  
def main(topic):  
 schema\_registry\_conf = schema\_config()  
 schema\_registry\_client = SchemaRegistryClient(schema\_registry\_conf)  
 subjects = schema\_registry\_client.get\_subjects()  
 print(subjects)  
 for subject in subjects:  
 if subject=='topic\_11-value':  
 schema = schema\_registry\_client.get\_latest\_version(subject)  
 print(schema.version)  
 print(schema.schema\_id)  
 value\_schema=schema.schema.schema\_str  
 print(value\_schema)  
  
 string\_serializer = StringSerializer('utf\_8')  
 json\_serializer = JSONSerializer(value\_schema, schema\_registry\_client, car\_to\_dict)  
  
 producer = Producer(sasl\_conf())  
  
  
  
 print("Producing user records to topic {}. ^C to exit.".format(topic))  
 # while True:  
 # Serve on\_delivery callbacks from previous calls to produce()  
 producer.poll(0.0)  
 try:  
 for car in get\_car\_instance(file\_path=FILE\_PATH):  
 print(car)  
 producer.produce(topic=topic,  
 key=string\_serializer(str(uuid4()), car\_to\_dict),  
 value=json\_serializer(car, SerializationContext(topic, MessageField.VALUE)),  
 on\_delivery=delivery\_report)  
 #break  
 except KeyboardInterrupt:  
 pass  
 except ValueError:  
 print("Invalid input, discarding record...")  
 pass  
  
 print("\nFlushing records...")  
 producer.flush()  
  
  
main("topic\_11")

5. From producer code, publish data in Kafka Topic one by one and use dynamic key while publishing the records into the Kafka Topic

Ans : Yes 74817 records has been published into topic after running the producer application

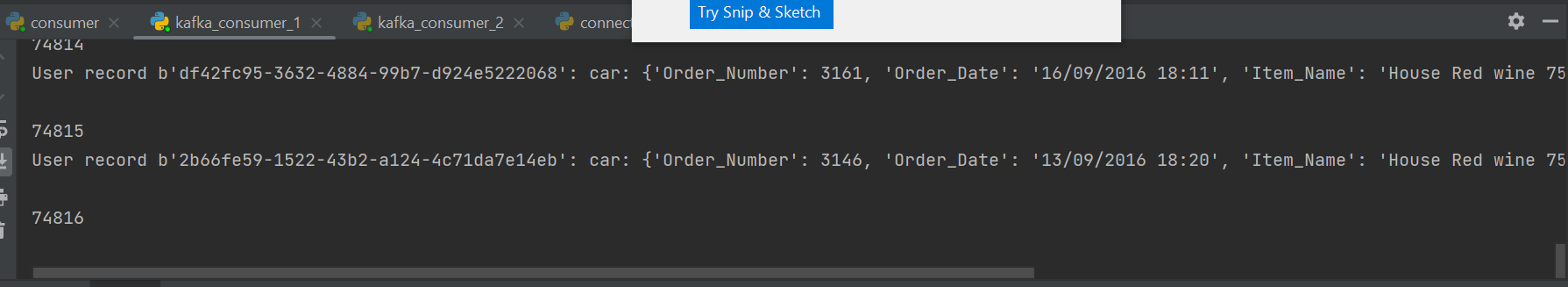
6. Write kafka consumer code and create two copies of same consumer code and save it with different names (kafka\_consumer\_1.py & kafka\_consumer\_2.py),again make sure lates schema version and schema\_str is not hardcoded in the consumer code, read it automatically from the schema registry to desrialize the data.

Now test two scenarios with your consumer code:

1. Use "group.id" property in consumer config for both consumers and mention different group\_ids in kafka\_consumer\_1.py & kafka\_consumer\_2.py,apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consumer consumed and printed on the terminal

Ans : I have mention the different group.id and just given code for one kafka\_consumer\_1.py as it is the same code for the other and the both consumers with different group id consumed equal number of records that is 74816 and I have used a COUNTER to calculate the number of records, I am attaching the screen shot of out put as well please find it below this code

import argparse  
  
from confluent\_kafka import Consumer  
from confluent\_kafka.schema\_registry import SchemaRegistryClient  
from confluent\_kafka.serialization import SerializationContext, MessageField  
from confluent\_kafka.schema\_registry.json\_schema import JSONDeserializer  
  
API\_KEY = 'ITBTATIBWRXJAZ5G'  
ENDPOINT\_SCHEMA\_URL = 'https://psrc-mw731.us-east-2.aws.confluent.cloud'  
API\_SECRET\_KEY = '+sKT3+XHHvhgzBsSRAKOgSyVLhtN4ktdmnIJPS1nKlesfyQK6Z+vXGorYdvhpJwp'  
BOOTSTRAP\_SERVER = 'pkc-6ojv2.us-west4.gcp.confluent.cloud:9092'  
SECURITY\_PROTOCOL = 'SASL\_SSL'  
SSL\_MACHENISM = 'PLAIN'  
SCHEMA\_REGISTRY\_API\_KEY = 'LP52E5CGC2IX7U75'  
SCHEMA\_REGISTRY\_API\_SECRET = 'ju7bR9JBZVDGQaTW9pujWnE4BenY3l1ZImNsuIPzNBAa6etD/ikDXXskZsP4xDK5'  
  
  
def sasl\_conf():  
 sasl\_conf = {'sasl.mechanism': SSL\_MACHENISM,  
 # Set to SASL\_SSL to enable TLS support.  
 # 'security.protocol': 'SASL\_PLAINTEXT'}  
 'bootstrap.servers': BOOTSTRAP\_SERVER,  
 'security.protocol': SECURITY\_PROTOCOL,  
 'sasl.username': API\_KEY,  
 'sasl.password': API\_SECRET\_KEY  
 }  
 return sasl\_conf  
  
  
def schema\_config():  
 return {'url': ENDPOINT\_SCHEMA\_URL,  
  
 'basic.auth.user.info': f"{SCHEMA\_REGISTRY\_API\_KEY}:{SCHEMA\_REGISTRY\_API\_SECRET}"  
  
 }  
  
  
class Car:  
 def \_\_init\_\_(self, record: dict):  
 for k, v in record.items():  
 setattr(self, k, v)  
  
 self.record = record  
  
 @staticmethod  
 def dict\_to\_car(data: dict, ctx):  
 return Car(record=data)  
  
 def \_\_str\_\_(self):  
 return f"{self.record}"  
  
  
def main(topic):  
# schema\_str = """  
# {  
# "$id": "http://example.com/myURI.schema.json",  
# "$schema": "http://json-schema.org/draft-07/schema#",  
  
 schema\_registry\_conf = schema\_config()  
 schema\_registry\_client = SchemaRegistryClient(schema\_registry\_conf)  
 subjects = schema\_registry\_client.get\_subjects()  
 print(subjects)  
 for subject in subjects:  
 if subject=='topic\_11-value':  
 schema = schema\_registry\_client.get\_latest\_version(subject)  
 print(schema.version)  
 print(schema.schema\_id)  
 value\_schema=schema.schema.schema\_str  
 print(value\_schema)  
 json\_deserializer = JSONDeserializer(value\_schema,  
 from\_dict=Car.dict\_to\_car)  
  
 consumer\_conf = sasl\_conf()  
 consumer\_conf.update({  
 'group.id': '1',  
 'auto.offset.reset': "earliest"})  
  
 consumer = Consumer(consumer\_conf)  
 consumer.subscribe([topic])  
  
  
 count = 0  
  
 while True:  
 try:  
 # SIGINT can't be handled when polling, limit timeout to 1 second.  
 msg = consumer.poll(1.0)  
 if msg is None:  
 continue  
  
 car = json\_deserializer(msg.value(), SerializationContext(msg.topic(), MessageField.VALUE))  
  
 if car is not None:  
 print("User record {}: car: {}\n"  
 .format(msg.key(), car))  
  
 count = count + 1  
 print(count)  
  
 except KeyboardInterrupt:  
 break  
  
 consumer.close()  
  
  
main("topic\_11")



1. Use "group.id" property in consumer config for both consumers and mention same group\_ids in kafka\_consumer\_1.py & kafka\_consumer\_2.py,apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consumer consumed and printed on the terminal

Ans : When I used same group.id first consumer consumed 25182 and second consumer consumed 49634 which means consumers with same group id will share the partitions and read from them for example if I have 4 partitions 2 consumers each of them read from 2 different set of partitions means they will read different messages

7. Once above questions are done, write another kafka consumer to read data from kafka topic and from the consumer code create one csv file "output.csv” and append consumed records output.csv file

Ans: I have used **pandas data frame** and **generator** concept to get the records one by one into the csv file “output.csv” please find the code below

from confluent\_kafka import Consumer  
from confluent\_kafka.schema\_registry import SchemaRegistryClient  
from confluent\_kafka.serialization import SerializationContext, MessageField  
from confluent\_kafka.schema\_registry.json\_schema import JSONDeserializer  
import pandas as pd  
import csv  
  
API\_KEY = 'ITBTATIBWRXJAZ5G'  
ENDPOINT\_SCHEMA\_URL = 'https://psrc-mw731.us-east-2.aws.confluent.cloud'  
API\_SECRET\_KEY = '+sKT3+XHHvhgzBsSRAKOgSyVLhtN4ktdmnIJPS1nKlesfyQK6Z+vXGorYdvhpJwp'  
BOOTSTRAP\_SERVER = 'pkc-6ojv2.us-west4.gcp.confluent.cloud:9092'  
SECURITY\_PROTOCOL = 'SASL\_SSL'  
SSL\_MACHENISM = 'PLAIN'  
SCHEMA\_REGISTRY\_API\_KEY = 'LP52E5CGC2IX7U75'  
SCHEMA\_REGISTRY\_API\_SECRET = 'ju7bR9JBZVDGQaTW9pujWnE4BenY3l1ZImNsuIPzNBAa6etD/ikDXXskZsP4xDK5'  
  
  
def sasl\_conf():  
 sasl\_conf = {'sasl.mechanism': SSL\_MACHENISM,  
 # Set to SASL\_SSL to enable TLS support.  
 # 'security.protocol': 'SASL\_PLAINTEXT'}  
 'bootstrap.servers': BOOTSTRAP\_SERVER,  
 'security.protocol': SECURITY\_PROTOCOL,  
 'sasl.username': API\_KEY,  
 'sasl.password': API\_SECRET\_KEY  
 }  
 return sasl\_conf  
  
  
def schema\_config():  
 return {'url': ENDPOINT\_SCHEMA\_URL,  
  
 'basic.auth.user.info': f"{SCHEMA\_REGISTRY\_API\_KEY}:{SCHEMA\_REGISTRY\_API\_SECRET}"  
  
 }  
  
  
class Car:  
 def \_\_init\_\_(self, record: dict):  
 for k, v in record.items():  
 setattr(self, k, v)  
  
 self.record = record  
  
 @staticmethod  
 def dict\_to\_car(data: dict, ctx):  
 return Car(record=data)  
  
 def \_\_str\_\_(self):  
 return f"{self.record}"  
  
  
def main(topic):  
  
 schema\_registry\_conf = schema\_config()  
 schema\_registry\_client = SchemaRegistryClient(schema\_registry\_conf)  
 subjects = schema\_registry\_client.get\_subjects()  
 print(subjects)  
 for subject in subjects:  
 if subject=='topic\_11-value':  
 schema = schema\_registry\_client.get\_latest\_version(subject)  
 print(schema.version)  
 print(schema.schema\_id)  
 value\_schema=schema.schema.schema\_str  
 print(value\_schema)  
 json\_deserializer = JSONDeserializer(value\_schema,  
 from\_dict=Car.dict\_to\_car)  
  
 consumer\_conf = sasl\_conf()  
 consumer\_conf.update({  
 'group.id': '1',  
 'auto.offset.reset': "earliest"})  
  
 consumer = Consumer(consumer\_conf)  
 consumer.subscribe([topic])  
  
  
 count = 0  
  
  
 while True:  
 try:  
 # SIGINT can't be handled when polling, limit timeout to 1 second.  
 msg = consumer.poll(1.0)  
 if msg is None:  
 continue  
  
 car = json\_deserializer(msg.value(), SerializationContext(msg.topic(), MessageField.VALUE))  
  
 if car is not None:  
 print("User record {}: car: {}\n"  
 .format(msg.key(), car))  
 fieldnames = sorted(car.\_\_dict\_\_.keys()) # needs to be the first line in correct order  
 fieldnames.pop()  
 del car.\_\_dict\_\_['record']  
 yield car.\_\_dict\_\_  
 count = count + 1  
 if count==10:  
 break  
 print(count)  
  
 except KeyboardInterrupt:  
 break  
  
 consumer.close()  
  
  
re=main("topic\_11")  
  
  
dfList=[]  
  
for row in re:  
 dfList.append(row)  
print(dfList)  
pd.DataFrame(dfList).to\_csv('output.csv', index=False)