KAFKA

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Copyright 2020 Confluent Inc.
#
# Licensed under the Apache License, Version 2.0 (the
"License"); # you may not use this file except in compliance
with the License.
# You may obtain a copy of the License
at #
#
http://www.apache.org/licenses/LICENSE-
2.0 #
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied. # See the License for the specific language
governing permissions and # limitations under the License.


# A simple example demonstrating use of JSONSerializer.

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 = "restaurant_orders.csv"
columns=['Order Number', 'Order Date', 'Item Name', 'Quantity', 'Product Price', 'Total
products']

API_KEY = 'WLOKHCU6PDMZNHVC'
ENDPOINT_SCHEMA_URL  = 'https://psrc-30dr2.us-central1.gcp.confluent.cloud'
API_SECRET_KEY =
'Fs8lJPMAQhYjxfDawy7jVUI/fO/rOPBexWQeZauwMLQyY44zW1MWg0X20K6FGnMe'
BOOTSTRAP_SERVER = 'pkc-lzvrd.us-west4.gcp.confluent.cloud:9092'
SECURITY_PROTOCOL = 'SASL_SSL'
SSL_MACHENISM = 'PLAIN'
SCHEMA_REGISTRY_API_KEY = 'HT3IX7DIHATKMTWC'
SCHEMA_REGISTRY_API_SECRET =
'IT9SiyCqaEZCFttPscGYbJ7C4LvJ1cvrKGFb0qX7oATGd2SQMtRsnnR1142qBMMh'
```

KAFKA

```
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[:,0:]     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_str = """
    {
  "$id": "http://example.com/myURI.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
```

KAFKA


```
  "additionalProperties": false,
  "description": "Sample schema to help you get started.",
"properties": {
    "Order Number": {
      "description": "The type(v) type is used.",
      "type": "number"
    },
    "Order Date": {
      "description": "The type(v) type is used.",
      "type": "string"
    },
    "Item Name": {
      "description": "The type(v) type is used.",
      "type": "string"
    },
    "Quantity": {
      "description": "The type(v) type is used.",
      "type": "number"
},
    "Product Price": {
      "description": "The type(v) type is used.",
      "type": "number"
    },
    "Total products": {
```

KAFKA

```
    "description": "The type(v) type is used.",
    "type": "number"
   }
  },
  "title": "SampleRecord",
  "type": "object"
}    """
   schema_registry_conf = schema_config()
   schema_registry_client = SchemaRegistryClient(schema_registry_conf)
   #reading the latest version of schema and schema_str from schema registry and useing it for data serialization.     schema_name = schema_registry_client.get_schema(100004).schema_str

   string_serializer = StringSerializer('utf_8')
   json_serializer = JSONSerializer(schema_name, 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 idx,car in
enumerate(get_car_instance(file_path=FILE_PATH)):

         print(car)          producer.produce(topic=topic, # publishing data in Kafka Topic one by one
and use dynamic key                    key=string_serializer("dynamic_key", car_to_dict),
value=json_serializer(car, SerializationContext(topic, MessageField.VALUE)),
on_delivery=delivery_report)
         if idx==5:      # index value is used for testing 1 or 2 records
break    except KeyboardInterrupt:
      pass    except
ValueError:
      print("Invalid input, discarding record...")
pass

   print("\nFlushing records...")
producer.flush() main("restaurent-take-away-
data")
```

KAFKA

```
import argparse

from confluent_kafka import Consumer from
confluent_kafka.serialization import SerializationContext,
MessageField from confluent_kafka.schema_registry.json_schema
import JSONDeserializer from confluent_kafka.schema_registry
import SchemaRegistryClient


API_KEY = 'WLOKHCU6PDMZNHVC'
ENDPOINT_SCHEMA_URL  = 'https://psrc-30dr2.us-central1.gcp.confluent.cloud'
API_SECRET_KEY =
'Fs8lJPMAQhYjxfDawy7jVUI/fO/rOPBexWQeZauwMLQyY44zW1MWg0X20K6FGnMe'
BOOTSTRAP_SERVER = 'pkc-lzvrd.us-west4.gcp.confluent.cloud:9092'
SECURITY_PROTOCOL = 'SASL_SSL'
SSL_MACHENISM = 'PLAIN'
SCHEMA_REGISTRY_API_KEY = 'HT3IX7DIHATKMTWC'
SCHEMA_REGISTRY_API_SECRET =
'IT9SiyCqaEZCFttPscGYbJ7C4LvJ1cvrKGFb0qX7oATGd2SQMtRsnnR1142qBMMh'


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)
```

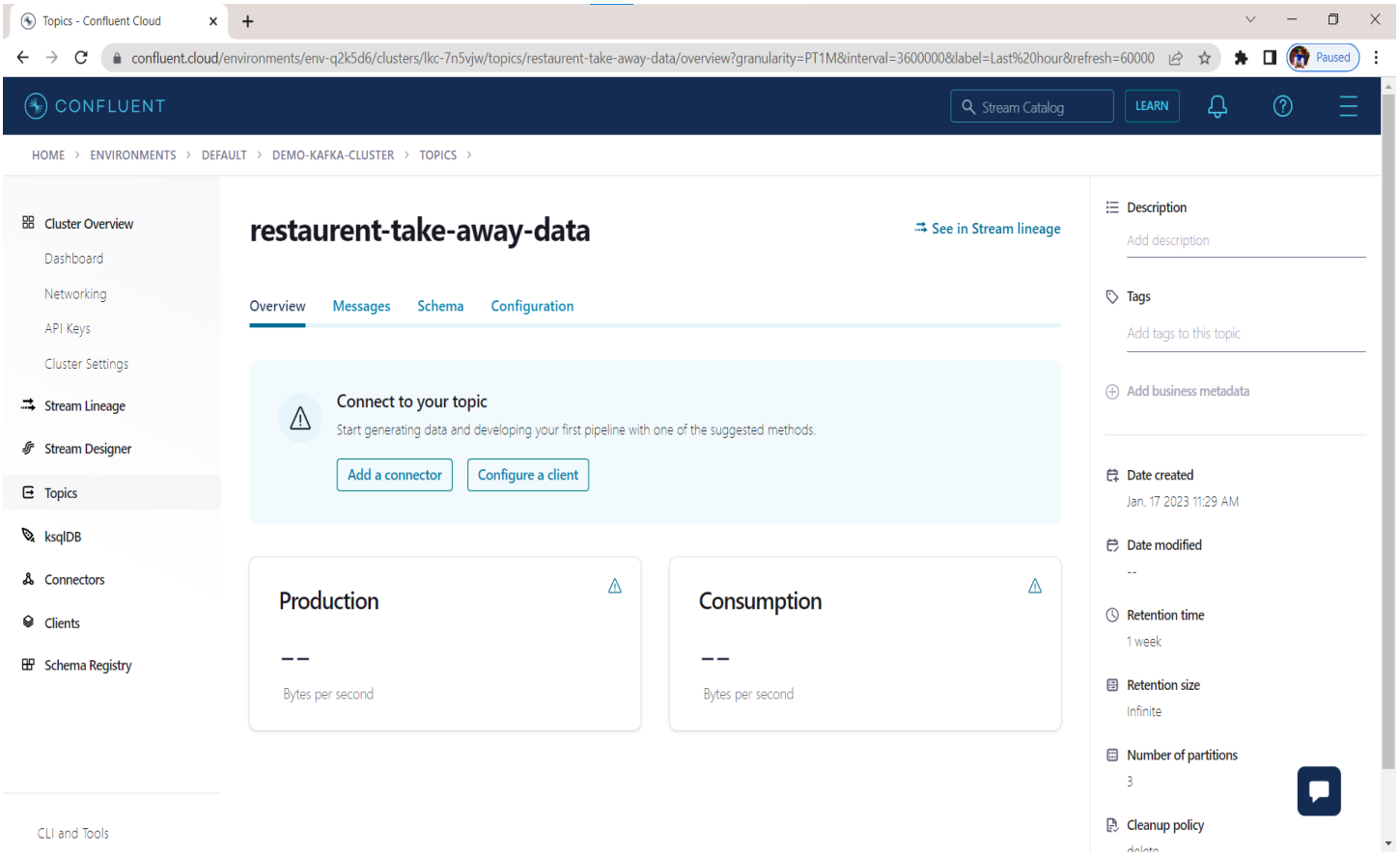
```
self.record=record

    @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#",
  "additionalProperties": false,
  "description": "Sample schema to help you get started.",
"properties": {
    "Order Number": {
      "description": "The type(v) type is used.",
      "type": "number"
    },
    "Order Date": {
      "description": "The type(v) type is used.",
      "type": "string"
    },
    "Item Name": {
      "description": "The type(v) type is used.",
      "type": "string"     },
```

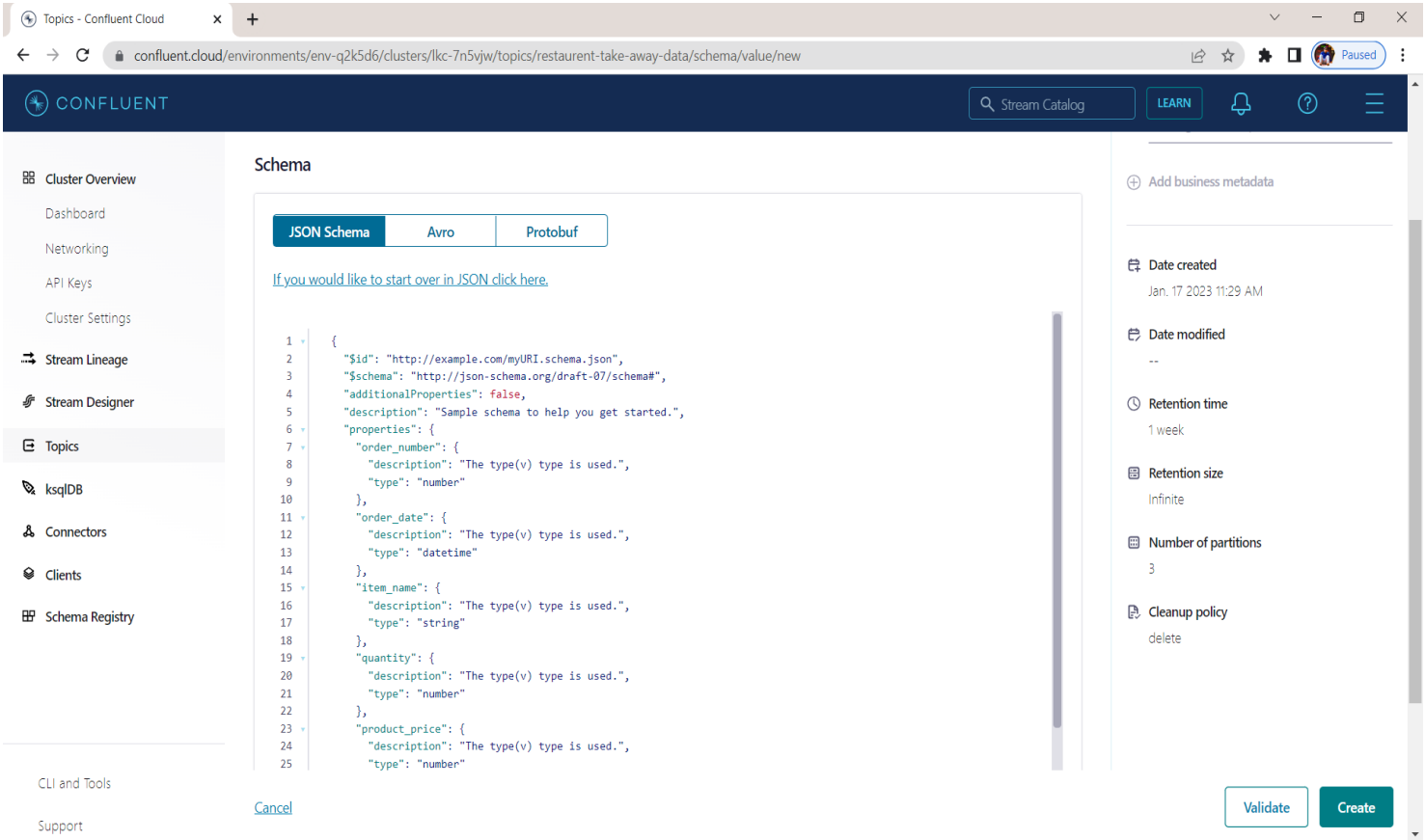
```
    "Quantity": {
      "description": "The type(v) type is used.",
      "type": "number"
},
    "Product Price": {
      "description": "The type(v) type is used.",
      "type": "number"
    },
    "Total products": {
      "description": "The type(v) type is used.",
      "type": "number"
    }
  },
  "title": "SampleRecord",
  "type": "object"
}
    """
   #reading the latest version of schema and schema_str from schema registry and useing it for data deserialization.
    schema_registry_conf = schema_config()
    schema_registry_client = SchemaRegistryClient(schema_registry_conf)
schema_name = schema_registry_client.get_schema(100004).schema_str
json_deserializer = JSONDeserializer(schema_name,
                                     from_dict=Car.dict_to_car)

    consumer_conf = sasl_conf()
consumer_conf.update({
                 'group.id': 'group1',
                 'auto.offset.reset': "earliest"})

    consumer = Consumer(consumer_conf)
consumer.subscribe([topic])


    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))       except
KeyboardInterrupt:          break
```

KAFKA

```
consumer.close() main("restaurent-take-
away-data")
```

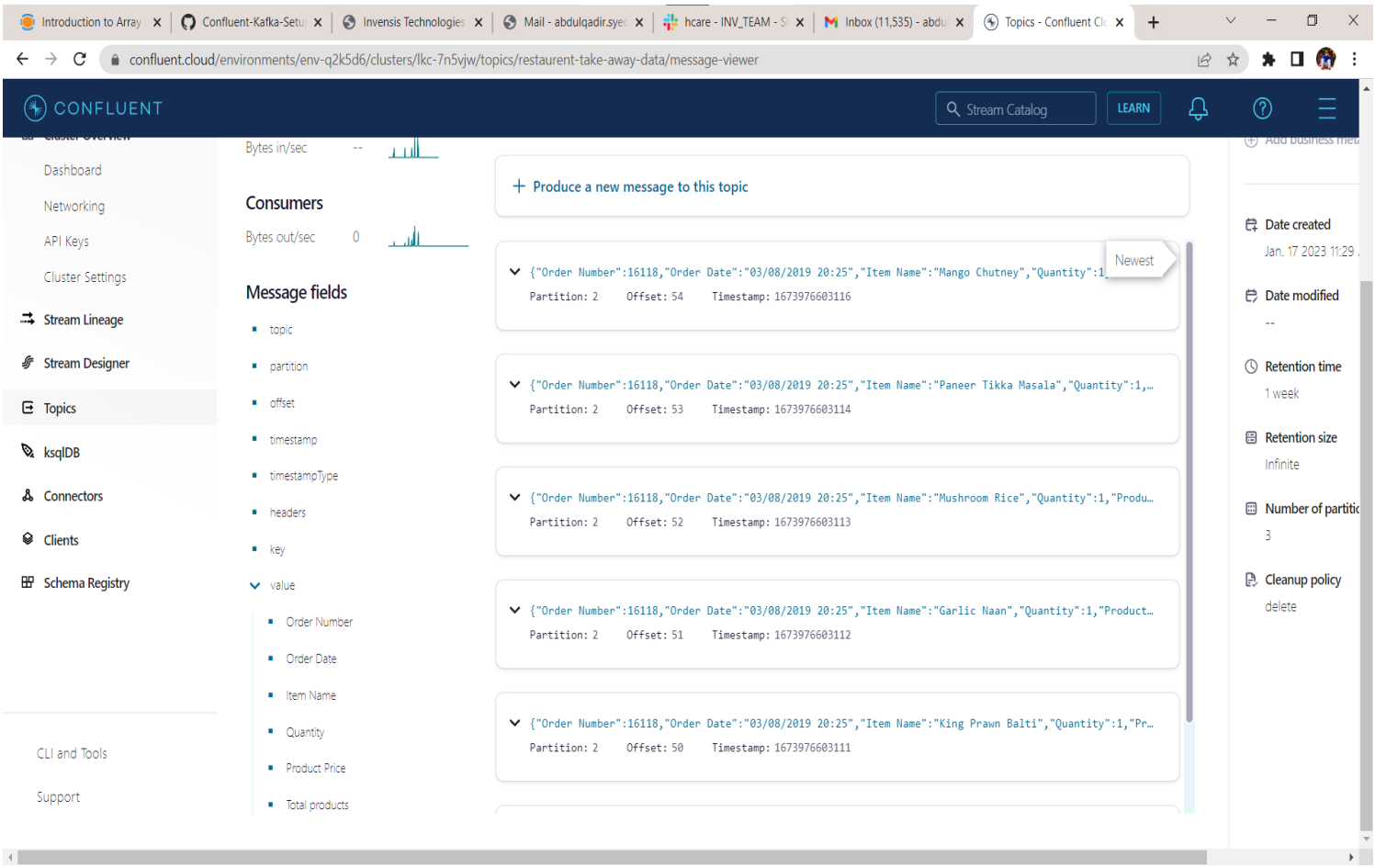
KAFKA


Topics - Confluent Cloud
confluent.cloud/environments/env-q2k5d6/clusters/lkc-7n5vjw/topics/restaurent-take-away-data/overview?granularity=PT1M&interval=3600000&label=Last%20hour&refresh=60000
CONFLUENT
Stream Catalog
LEARN
HOME › ENVIRONMENTS › DEFAULT › DEMO-KAFKA-CLUSTER › TOPICS ›
Cluster Overview
Dashboard
Networking
API Keys
Cluster Settings
Stream Lineage
Stream Designer
Topics
ksqlDB
Connectors
Clients
Schema Registry
CLI and Tools
restaurent-take-away-data
See in Stream lineage
Overview
Messages
Schema
Configuration
Connect to your topic
Start generating data and developing your first pipeline with one of the suggested methods.
Add a connector
Configure a client
Production
--
Bytes per second
Consumption
--
Bytes per second
Description
Add description
Tags
Add tags to this topic
Add business metadata
Date created
Jan. 17 2023 11:29 AM
Date modified
--
Retention time
1 week
Retention size
Infinite
Number of partitions
3
Cleanup policy
delete


Topics - Confluent Cloud
confluent.cloud/environments/env-q2k5d6/clusters/lkc-7n5vjw/topics/restaurent-take-away-data/schema/value/new
CONFLUENT
Stream Catalog
LEARN
Cluster Overview
Dashboard
Networking
API Keys
Cluster Settings
Stream Lineage
Stream Designer
Topics
ksqlDB
Connectors
Clients
Schema Registry
CLI and Tools
Support
Schema
JSON Schema
Avro
Protobuf
If you would like to start over in JSON click here.
{
"$id": "http://example.com/myURI.schema.json",
"$schema": "http://json-schema.org/draft-07/schema#",
"additionalProperties": false,
"description": "Sample schema to help you get started.",
"properties": {
"order_number": {
"description": "The type(v) type is used.",
"type": "number"
},
"order_date": {
"description": "The type(v) type is used.",
"type": "datetime"
},
"item_name": {
"description": "The type(v) type is used.",
"type": "string"
},
"quantity": {
"description": "The type(v) type is used.",
"type": "number"
},
"product_price": {
"description": "The type(v) type is used.",
"type": "number"
Cancel
Validate
Create
Add business metadata
Date created
Jan. 17 2023 11:29 AM
Date modified
--
Retention time
1 week
Retention size
Infinite
Number of partitions
3
Cleanup policy
delete

KAFKA

```
Anaconda Prompt (Anaconda3)

(base) C:\Users\Abdul Qadir>d:

(base) D:\>cd D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain

(base) D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain>python kafka_json_producer_check.py
Producing user records to topic restaurent-take-away-data. ^C to exit.
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Plain Papadum', 'Quantity': 2, 'Product Price': 0.8, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'King Prawn Balti', 'Quantity': 1, 'Product Price': 12.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Garlic Naan', 'Quantity': 1, 'Product Price': 2.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mushroom Rice', 'Quantity': 1, 'Product Price': 3.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Paneer Tikka Masala', 'Quantity': 1, 'Product Price': 8.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mango Chutney', 'Quantity': 1, 'Product Price': 0.5, 'Total products': 6}

Flushing records...
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 49
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 50
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 51
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 52
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 53
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 54

(base) D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain>
```



KAFKA

Anaconda Prompt (Anaconda3)

```
(base) C:\Users\Abdul Qadir>d:

(base) D:\>cd D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain

(base) D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain>python kafka_json_producer_check.py
Producing user records to topic restaurent-take-away-data. ^C to exit.
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Plain Papadum', 'Quantity': 2, 'Product Price': 0.8, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'King Prawn Balti', 'Quantity': 1, 'Product Price': 12.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Garlic Naan', 'Quantity': 1, 'Product Price': 2.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mushroom Rice', 'Quantity': 1, 'Product Price': 3.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Paneer Tikka Masala', 'Quantity': 1, 'Product Price': 8.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mango Chutney', 'Quantity': 1, 'Product Price': 0.5, 'Total products': 6}

Flushing records...
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 49
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 50
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 51
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 52
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 53
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 54

(base) D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain>python kafka_json_producer_check.py
Producing user records to topic restaurent-take-away-data. ^C to exit.
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Plain Papadum', 'Quantity': 2, 'Product Price': 0.8, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'King Prawn Balti', 'Quantity': 1, 'Product Price': 12.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Garlic Naan', 'Quantity': 1, 'Product Price': 2.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mushroom Rice', 'Quantity': 1, 'Product Price': 3.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Paneer Tikka Masala', 'Quantity': 1, 'Product Price': 8.95, 'Total products': 6}
{'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mango Chutney', 'Quantity': 1, 'Product Price': 0.5, 'Total products': 6}

Flushing records...
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 55
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 56
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 57
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 58
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 59
User record b'dynamic_key' successfully produced to restaurent-take-away-data [2] at offset 60

(base) D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain>
```

Anaconda Prompt (Anaconda3) - python kafka_json_consumer_1.py

```
(base) C:\Users\Abdul Qadir>d:

(base) D:\>cd D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain

(base) D:\SSD_DEVICE_STORAGE\Desktop\Kafkademo\ConfluentKafkaSetupmain>python kafka_json_consumer_1.py
User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Plain Papadum', 'Quantity': 2, 'Product Price': 0.8, 'Total product
s': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'King Prawn Balti', 'Quantity': 1, 'Product Price': 12.95, 'Total pr
oducts': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Garlic Naan', 'Quantity': 1, 'Product Price': 2.95, 'Total products
': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mushroom Rice', 'Quantity': 1, 'Product Price': 3.95, 'Total produc
ts': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Paneer Tikka Masala', 'Quantity': 1, 'Product Price': 8.95, 'Total
products': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mango Chutney', 'Quantity': 1, 'Product Price': 0.5, 'Total product
s': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Plain Papadum', 'Quantity': 2, 'Product Price': 0.8, 'Total product
s': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'King Prawn Balti', 'Quantity': 1, 'Product Price': 12.95, 'Total pr
oducts': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Garlic Naan', 'Quantity': 1, 'Product Price': 2.95, 'Total products
': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mushroom Rice', 'Quantity': 1, 'Product Price': 3.95, 'Total produc
ts': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Paneer Tikka Masala', 'Quantity': 1, 'Product Price': 8.95, 'Total
products': 6}

User record b'dynamic_key': car: {'Order Number': 16118, 'Order Date': '03/08/2019 20:25', 'Item Name': 'Mango Chutney', 'Quantity': 1, 'Product Price': 0.5, 'Total product
s': 6}
```