|  |
| --- |
| #!/usr/bin/env python |
|  | # |
|  | # 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. |
|  | # |
|  |  |
|  | # ============================================================================= |
|  | # |
|  | # Consume messages from Confluent Cloud |
|  | # Using Confluent Python Client for Apache Kafka |
|  | # |
|  | # ============================================================================= |
|  |  |
|  | from confluent\_kafka import Consumer |
|  | import json |
|  | import ccloud\_lib |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  |  |
|  | # Read arguments and configurations and initialize |
|  | args = ccloud\_lib.parse\_args() |
|  | config\_file = args.config\_file |
|  | topic = args.topic |
|  | conf = ccloud\_lib.read\_ccloud\_config(config\_file) |
|  |  |
|  | # Create Consumer instance |
|  | # 'auto.offset.reset=earliest' to start reading from the beginning of the |
|  | # topic if no committed offsets exist |
|  | consumer\_conf = ccloud\_lib.pop\_schema\_registry\_params\_from\_config(conf) |
|  | consumer\_conf['group.id'] = 'python\_example\_group\_1' |
|  | consumer\_conf['auto.offset.reset'] = 'earliest' |
|  | consumer = Consumer(consumer\_conf) |
|  |  |
|  | # Subscribe to topic |
|  | consumer.subscribe([topic]) |
|  |  |
|  | # Process messages |
|  | total\_count = 0 |
|  | try: |
|  | while True: |
|  | msg = consumer.poll(1.0) |
|  | if msg is None: |
|  | # No message available within timeout. |
|  | # Initial message consumption may take up to |
|  | # `session.timeout.ms` for the consumer group to |
|  | # rebalance and start consuming |
|  | print("Waiting for message or event/error in poll()") |
|  | continue |
|  | elif msg.error(): |
|  | print('error: {}'.format(msg.error())) |
|  | else: |
|  | # Check for Kafka message |
|  | record\_key = msg.key() |
|  | print(str(record\_key), str(record\_key) == "0") |
|  | if(str(record\_key) == '0'): |
|  | record\_value = msg.value() |
|  | data = json.loads(record\_value) |
|  | print("Consumed record with key {} and value {}, \ |
|  | and updated total count to {}" |
|  | .format(record\_key, record\_value, total\_count)) |
|  | total\_count = total\_count+1 |
|  | print(f"total count = {total\_count}") |
|  | else: |
|  | print(str(record\_key) + " is this") |
|  | except KeyboardInterrupt: |
|  | pass |
|  | finally: |
|  | # Leave group and commit final offsets |
|  | consumer.close() |