**Week4**

**Objective**: to save streaming data (movie ticket sales events) to database

The stream has three fields:

1. title, the name of the movie;
2. sale\_ts, the time at which the ticket was sold; and
3. ticket\_total\_value, the price paid for the ticket.

**Example of data**

**INSERT** **INTO** MOVIE\_TICKET\_SALES (title, sale\_ts, ticket\_total\_value) **VALUES** ('Aliens', '2019-07-18T10:00:00Z', 10);

**INSERT** **INTO** MOVIE\_TICKET\_SALES (title, sale\_ts, ticket\_total\_value) **VALUES** ('Die Hard', '2019-07-18T10:00:00Z', 12);

**INSERT** **INTO** MOVIE\_TICKET\_SALES (title, sale\_ts, ticket\_total\_value) **VALUES** ('Die Hard', '2019-07-18T10:01:00Z', 12);

**Part 1: Initial process**

1. **Create a kafka cluster via Docker**
   1. Three brokers and one zoo keeper
2. **Create a topic (movie)**
   1. Two partitions and three replicas
3. **Create a producer (movie\_producer.py)**
   1. **pip install pymongo (for bson)**

To collect and send individual movie ticket sales events

from confluent\_kafka import Producer

import json

import random

import time

from datetime import datetime

from bson import json\_util

def acked(err, msg):

if err is not None:

print("Failed to deliver message: %s: %s" % (str(msg), str(err)))

else:

print("Message produced: %s" % (msg.value().decode()))

p = Producer({'bootstrap.servers':'localhost:8097'})

titles = ["ET", "Hulk", "Spiderman"]

prices=[12,24,36]

for i in range(10):

current\_time = datetime.now().strftime("%Y/%m/%d %H:%M:%S")

data = {

'title': random.choice(titles),

'sale\_ts' : current\_time,

'ticket\_total\_value' : random.choice(prices)

}

p.produce('movie',

key="key",

value=json.dumps(data, default=json\_util.default).encode('utf-8'),

callback=acked)

p.poll(1)

time.sleep(random.randint(1,5))

1. **Create a consumer (movie\_consumer.py)**

from confluent\_kafka import Consumer

import json

import pandas as pd

running = True

def basic\_consume\_loop(c, topics):

try:

c.subscribe(topics)

while running:

msg = c.poll(timeout=1.0)

if msg is None: continue

if msg.error():

if msg.error().code() == KafkaError.\_PARTITION\_EOF:

# End of partition event

sys.stderr.write('%% %s [%d] reached end at offset %d\n' %

(msg.topic(), msg.partition(), msg.offset()))

elif msg.error():

raise KafkaException(msg.error())

else:

results = json.loads(msg.value().decode())

print(results)

print(results['title'])

finally:

# Close down consumer to commit final offsets.

c.close()

def shutdown():

running = False

c = Consumer({'bootstrap.servers':'localhost:8097',

'group.id':'group1',

'auto.offset.reset':'earliest'})

basic\_consume\_loop(c,['movie'])

1. **Start running** 
   1. python movie\_consumer.py (Terminal 1)
   2. pyhton movie\_producer.py (Terminal 2)

**Part 2: Consumer saves data to MySQL**

**1. Create a MySQL database via Docker**

* 1. Download and unzip file “docker-mysql.zip”
  2. Use CMD for accessing the “docker-mysql” folder
  3. Start mysql docker by typing
     1. docker-compose up -d (\*\*This command will only work if the ‘docker-compose.yml’ file exists in the same path)

1. **Create the table “movie\_tb” in mysql db**
   1. Enter the mysql terminal (How?)

mysql -uroot -p

enter password confluent2

show databases;

use connect\_test2;

create table movie\_tb (

title varchar(100) not null,

sale\_ts varchar(100) not null,

ticket\_total\_value int not null

);

1. **Test the DB by inserting a sample data and then select it.**

**INSERT** **INTO** movie\_tb (title, sale\_ts, ticket\_total\_value) **VALUES** ('Aliens', '2019-07-18T10:00:00Z', 10);

1. **Modify the consumer (movie\_consumer.py) and named it as (movie\_consumer2.py) and pip install mysql.connector**

from confluent\_kafka import Consumer

import json

import pandas as pd

running = True

import mysql.connector

connection = mysql.connector.connect(host="localhost",database="connect\_test2",user="confluent2",password="confluent2", port=3307)

def basic\_consume\_loop(c, topics):

try:

c.subscribe(topics)

while running:

msg = c.poll(timeout=1.0)

if msg is None: continue

if msg.error():

if msg.error().code() == KafkaError.\_PARTITION\_EOF:

# End of partition event

sys.stderr.write('%% %s [%d] reached end at offset %d\n' %

(msg.topic(), msg.partition(), msg.offset()))

elif msg.error():

raise KafkaException(msg.error())

else:

results = json.loads(msg.value().decode())

print(results)

insert\_cmd = f"insert into movie\_tb (title, sale\_ts, ticket\_total\_value) values

( '{results['title']}',

'{results[‘sale\_ts’]}',

‘{results[‘ticket\_total\_value’]}’)"

print(insert\_cmd)

cursor = connection.cursor() #Should this line be called once?

cursor.execute(insert\_cmd)

connection.commit()

finally:

# Close down consumer to commit final offsets.

c.close()

def shutdown():

running = False

c = Consumer({'bootstrap.servers':'localhost:8097',

'group.id':'group1',

'auto.offset.reset':'earliest'})

basic\_consume\_loop(c,['movie'])

1. **Start running** 
   1. movie\_consumer2.py
   2. movie\_producer.py