

Mysql - Kafka_Connector - SparkStreaming

- 크롤러

```
#!pip install pymysql
#!pip install sqlalchemy
#!pip install mysqlclient
#!pip install cryptography
```

```
import pymysql
from bs4 import BeautifulSoup
import urllib.request
from urllib.parse import quote
import pandas as pd
import MySQLdb
```

```
conn = MySQLdb.connect(
    user="mysqluser",
    passwd="mysqlpw",
    host="localhost",
    db="testdb",
    charset="utf-8"
)
print(type(conn))
# <class 'MySQLdb.connections.Connection'>
cursor = conn.cursor()
print(type(cursor))
# <class 'MySQLdb.cursors.Cursor'>
```

```
def get_movie_reviews(mcode, page_num):
    movie_review_df = pd.DataFrame(columns=("Title", "Score", "Review"))
    url = "https://movie.naver.com/movie/point/af/list.naver?st=mcode&sword=" + str(mcode)+"&target=after"
    idx = 0
    for _ in range(0, page_num):
        movie_page = urllib.request.urlopen(url).read()
        movie_page_soup = BeautifulSoup(movie_page, "html.parser")

        review_list = movie_page_soup.find_all("td", {"class": "title"})
        for review in review_list:
            title = review.find("a", {"class": "movie color_b"}).get_text()
            score = review.find("em").get_text()
            review_text = review.find("a", {"class": "report"}).get("onclick").split(",")[2]
            movie_review_df.loc[idx] = [title, score, review_text]

            cursor.execute(f"INSERT INTO accounts VALUES('{title}', '{score}', '{review_text}')"
            conn.commit()
            idx+=1

        print("#", end="")
    try:
        url = "https://movie.naver.com" + movie_page_soup.find("a", {"class": "pg_next"}).get("href")
    except:
        break
    return movie_review_df
```

```
movie_review_df = get_movie_reviews(193855, 3)
movie_review_df
```

- Kafka

1. Docker- compose를 이용하여 MySQL과 Kafka를 설치

```
version: '3'
services:
  mysql:
    image: mysql:8.0
    container_name: mysql
```

```

ports:
  - 3306:3306
environment:
  MYSQL_ROOT_PASSWORD: admin
  MYSQL_USER: mysqluser
  MYSQL_PASSWORD: mysqlpw
command:
  - --character-set-server=utf8mb4
  - --collation-server=utf8mb4_unicode_ci
volumes:
  - D:/mysql/data:/var/lib/mysql

zookeeper:
  container_name: zookeeper
  image: wurstmeister/zookeeper
  ports:
    - "2181:2181"

kafka:
  container_name: kafka
  image: wurstmeister/kafka
  depends_on:
    - zookeeper
  ports:
    - "9092:9092"
  environment:
    KAFKA_ADVERTISED_HOST_NAME: 127.0.0.1
    KAFKA_ADVERTISED_PORT: 9092
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock

```

2. MySQL 데이터 베이스 생성

```

mysql -u root -p

create database testdb;

use testdb;

CREATE TABLE accounts (
  Title VARCHAR(255),
  Score INT(10),
  Review VARCHAR(255),
  PRIMARY KEY (Review)
);

```

```

use mysql;
//mysqluser가 추가 되어 있는지 확인
select host, user from user;

//mysqluser 없으면 생성
CREATE USER 'mysqluser'@'%' IDENTIFIED BY 'mysqlpw';

// mysqluser 에게 권한 부여
GRANT ALL PRIVILEGES ON *.* TO 'mysqluser'@'%';

FLUSH PRIVILEGES;

```

```

https://debezium.io/releases/1.5/

// 다운로드
debezium-connector-mysql-1.5.4.Final-plugin.tar.gz

//위치에 압축 해제
/opt/kafka_2.13-2.7.1/connectors

```

3. 카프카 컨테이너에 접속, `/opt/kafka/config/connect-distributed.properties` 파일을 수정

```
// 원래 경로
#plugin.path=

// 수정 경로
plugin.path=/opt/kafka_2.13-2.8.1/connectors

// 카프카 컨테이너에서 실행
connect-distributed.sh /opt/kafka/config/connect-distributed.properties
```

```
// 8083 포트 열려있는지 확인
curl http://localhost:8083/

// 플러그인 조회
curl --location --request GET 'localhost:8083/connector-plugins'

//io.debezium.connector.mysql.MySqlConnector가 있어야함
```

4. REST API 호출하여 Connector를 생성

```
curl --location --request POST 'http://localhost:8083/connectors' \
--header 'Content-Type: application/json' \
--data-raw '{
  "name": "source-test-connector",
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "tasks.max": "1",
    "database.hostname": "mysql",
    "database.port": "3306",
    "database.user": "mysqluser",
    "database.password": "mysqlpw",
    "database.server.id": "184054",
    "database.server.name": "dbserver1",
    "database.allowPublicKeyRetrieval": "true",
    "database.include.list": "testdb",
    "database.history.kafka.bootstrap.servers": "kafka:9092",
    "database.history.kafka.topic": "dbhistory.testdb",
    "key.converter": "org.apache.kafka.connect.json.JsonConverter",
    "key.converter.schemas.enable": "true",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter.schemas.enable": "true",
    "transforms": "unwrap,addTopicPrefix",
    "transforms.unwrap.type": "io.debezium.transforms.ExtractNewRecordState",
    "transforms.addTopicPrefix.type": "org.apache.kafka.connect.transforms.RegexRouter",
    "transforms.addTopicPrefix.regex": "(.*)",
    "transforms.addTopicPrefix.replacement": "$1"
  }
}'
```

```
//커넥터의 Java 클래스
connector.class

//커넥터에 대해 생성되어야 할 태스크의 최대 수
task.max

//DB엔드포인트
database.hostname

//MySQL 인스턴스를 고유하게 식별하는데 사용하는 문자열
database.server.name

//지정한 서버에서 호스팅 하는 데이터베이스의 목록
database.include.list

//부트스트랩의 서버 주소
database.history.kafka.bootstrap.servers

// 데이터베이스 스키마 변경을 추적하기 위해 Debezium에서 내부적으로 사용하는 Kafka 주제
database.history.kafka.topic
```

```
# 목록 확인
kafka-topics.sh --list --bootstrap-server localhost:9092

## 분산 모드로 카프카 커넥트 실행 후 생성되는 토픽
```

```
__consumer_offsets
connect-configs
connect-offsets
connect-status

## 커넥터 생성 후 생성되는 토픽
dbhistory.testdb
dbserver1 // DDL 정보
dbserver1.testdb.accounts // 테이블 정보
```

5. 크롤러 작동 이후

```
mysql mysql:8.0
RUNNING

Logs Inspect Terminal Stats Open in external terminal

20 rows in set (0.00 sec)

mysql> select * FROM CRAW_TABLE
-> ;

+-----+-----+-----+
| Title                                     | Score | Review                                     |
+-----+-----+-----+
| 엔트맨과 와스프: 쿼텸매니아               | 4      | ''                                         |
| 엔트맨과 와스프: 쿼텸매니아               | 1      | '엔트맨의 매력을 다뎀 칸 비긴즈'         |
| 엔트맨과 와스프: 쿼텸매니아               | 7      | '양자영역 말고 그냥 도시에서 놀았으면 더 나았을듯' |
| 엔트맨과 와스프: 쿼텸매니아               | 8      | '비판도 이해는 되지만 일단 나는 재밌었음'   |
| 엔트맨과 와스프: 쿼텸매니아               | 2      | '마블.. 이제는 블랙팬서만 영화관에서 봐야겠다' |
| 엔트맨과 와스프: 쿼텸매니아               | 1      | '엔트맨 액션이 이렇게 잠들어야...'         |
| 엔트맨과 와스프: 쿼텸매니아               | 7      | ''                                         |
| 엔트맨과 와스프: 쿼텸매니아               | 10     | '어벤저스5를 기대하게 만드는 영화이다!'     |
| 엔트맨과 와스프: 쿼텸매니아               | 8      | '이게 코미디 인가요? 재밌는데 가볍게 보기 좋은 것 같은데뭔가 웃김 왜 웃기지 그냥 웃겨요그 젤리 같이 생긴 애 귀여움그리고 매독 해석에 감탄할 진짜 천재가?' |
| 엔트맨과 와스프: 쿼텸매니아               | 1      | '중평한다. 문화의 날 7천원으로 보는 것도 낭비다. 세트장에서 대부분 촬영하고 CG로 찍칠한 영화다. 이런 영화에 돈을 보태주지 말아야 이렇게 성의 없는 영화가 줄어들 것이다.' |
| 엔트맨과 와스프: 쿼텸매니아               | 8      | '재밌게 관람하였습니다'                 |
| 엔트맨과 와스프: 쿼텸매니아               | 7      | '양자 영역에 갇힌 칸과 마블'             |
| 엔트맨과 와스프: 쿼텸매니아               | 7      | '좀 아쉬웠습니다 중간에 졸리기도...'       |
| 엔트맨과 와스프: 쿼텸매니아               | 8      | '테란 vs 저그 +개미군단'                 |
```

BS4에서 넘어온 데이터 - MySQL

6. 스파크 스트리밍으로 읽기

```
from pyspark.sql import SparkSession

sc = SparkSession.builder.getOrCreate()

sc.sparkContext.setLogLevel('ERROR')

# Read stream
log = sc.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "dbserver1.testdb.accounts") \
    .option("startingOffsets", "earliest") \
    .load()

# Write stream - console
query = log.selectExpr("CAST(value AS STRING)") \
    .writeStream \
    .format("console") \
    .option("truncate", "false") \
    .start()

# Write stream - HDFS
query2 = log.selectExpr("CAST(value AS STRING)") \
```

```
.writeStream \
.format("parquet") \
.outputMode("append") \
.option("checkpointLocation", "/check") \
.option("path", "/test") \
.start()

query.awaitTermination()
query2.awaitTermination()
```

스파크에서 실행은 잘 되는데 데이터가 넘어오질 않았다.

```
[GroupCoordinator 1001]: Member connect-1-36b38210-1a82-4b6b-a61c-1d53dd2bb4e0 in group connect-cluster has failed, removing it from t
# 해결책
https://github.com/confluentinc/librdkafka/issues/2631
```

카프카에서 위와 같은 알람이 왔는데 해결책이 클러스터 늘리거나 컨슈머 늘리라고 한다.

그런데 여기서 도커 컨테이너 늘리면(Kafka, Zookepr 등) 컴퓨터가 버티질 못할것이 분명해보였다. (여기까지 오는데 블루스크린 6번, 재부팅 15번 실행하였다)

(이후 자금이 확보가 되면 AWS에서 다시한번 진행 하기로 함)

그러니 SparkStreaming 대신 카프카에서 Flatfile로 드랍하여 저장하는 방향으로 전환하였다.

7. 카프카에서 Flatfile Sink로 드랍

```
echo '{
  "name" : "Flatfile_Sink",
  "config" : {
    "connector.class" : "org.apache.kafka.connect.file.FileStreamSinkConnector",
    "file" : "/root/test.txt",
    "topics" : "dbserver1.testdb.accounts"
  }
}' | curl -X POST -d @- http://localhost:8083/connectors --header "content-Type:application/json"
```

[illegible]

Mysql에 저장되면 Kafka_Connector로 읽어와 토픽에 저장됨