

MLOps

6강

Kafka

목차

01.Overview

02.Kafka Introduction

03.Producer & Consumer

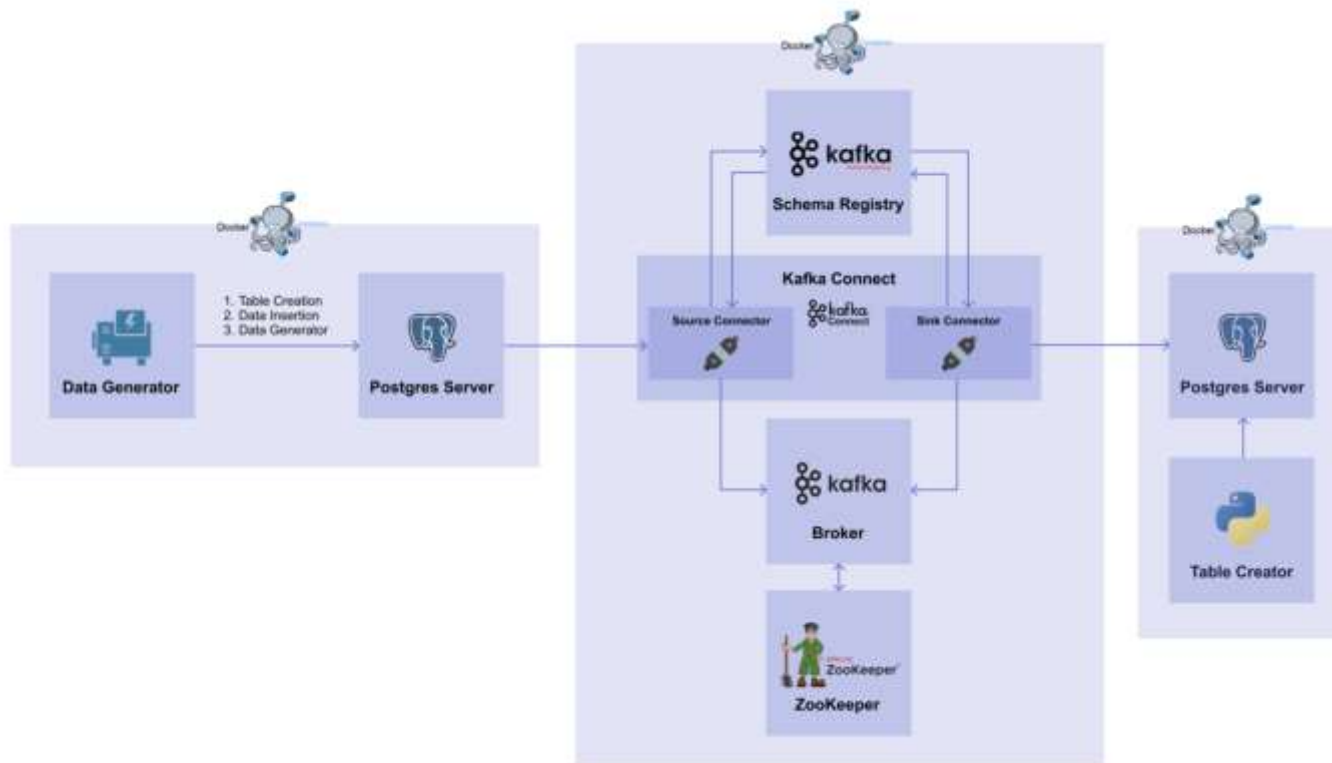
04.Connect & Connector

05.Kafka System

06.Source Connector

07.Sink Connector

Overview



Overview

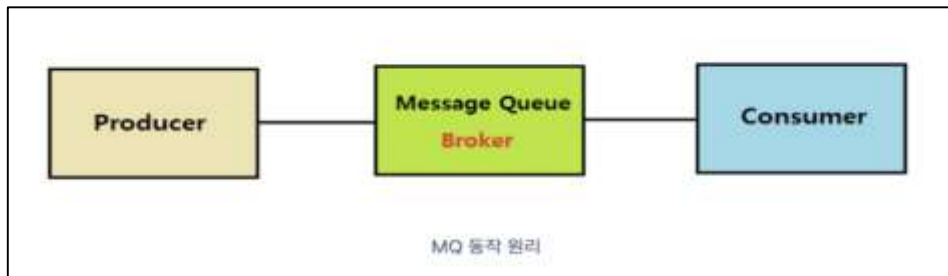
1.데이터 파이프라인

2.Source DB: 데이터가 계속해서 쌓이고 있는 외부 DB(01 - Database)

3.Target DB: 외부에서 가져온 데이터를 처리한 뒤 쌓이는 내부 DB(06 - Kafka)

Kafka Introduction

- 1.메시징 시스템: 서로 다른 애플리케이션끼리 정보를 교환(생성, 전송, 전달, 저장)
 - 메시지 생산자(message producer)와 메시지 소비자(message consumer) 사이에 **약한 결합성**
 - 높은 확장성, 통합성, 안정성
- 2.카프카: 파이프라인, 스트리밍 분석 등을 위한 **고성능 분산 이벤트 스트리밍 플랫폼**
 - Apache Kafka
 - Pub-Sub 모델의 메시지 큐 형태로 동작(분산처리)

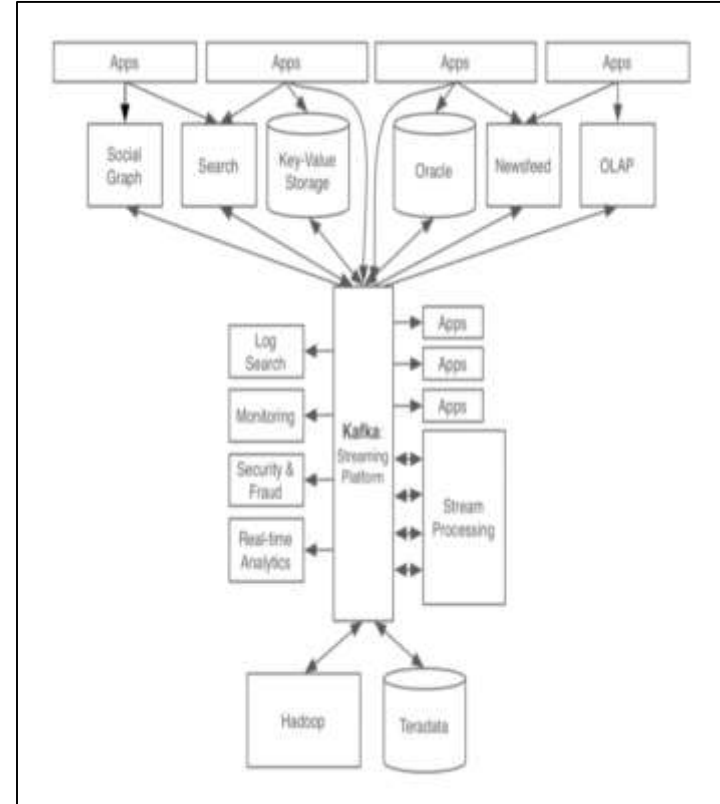


Kafka Introduction

3. 카프카 구성요소

- Producer: Kafka에 데이터를 제공하는 서버
- Consumer: Kafka로부터 데이터를 제공받아 사용하는 서버
- Broker: 메시징 서비스를 담당해주는 서버
- Kafka Cluster: 여러 개의 브로커로 이루어진 집합체
- Topic: Broker에서 event(데이터)를 저장하는 기준
- Partition: Topic에 존재하며, Producer로부터 전달된 데이터를 저장
- Zookeeper: 분산 처리를 관리하기 위한 오픈소스 서버

Kafka Introduction



Producer & Consumer

1. Zookeeper & Broker 설치(naive-docker-compose.yaml)

naive-docker-compose.yaml

```
# naive-docker-compose.yaml
version: "3"

services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.3.0
    container_name: zookeeper
    ports:
      - 2181:2181
    environment:
      ZOOKEEPER_SERVER_ID: 1
      ZOOKEEPER_CLIENT_PORT: 2181
  broker:
    image: confluentinc/cp-kafka:7.3.0
    container_name: broker
    depends_on:
      - zookeeper
    ports:
      - 9092:9092
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
```

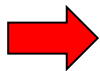
Producer & Consumer

2.Zookeeper & Broker 실행

```
$ docker compose -p class6 -f naive-docker-compose.yaml up -d
```

3.Zookeeper & Broker 이미지 확인

```
$ docker ps
```



```
$ docker ps
CONTAINER ID   IMAGE                                COMMAND
0a0599db4d96   confluentinc/cp-kafka:7.3.0        "/etc/confluent/dock..."
35cdc624ac34   confluentinc/cp-zookeeper:7.3.0    "/etc/confluent/dock..."
```

Producer & Consumer

4.Producer & Consumer 설치

- Topic 생성

```
$ docker compose -p class6-naive exec broker kafka-topics --create --  
topic topic-test --bootstrap-server broker:29092 --partitions 1 --replication-  
factor 1
```

- Topic 확인

```
$ docker compose -p class6-naive exec broker kafka-topics --describe --  
topic topic-test --bootstrap-server broker:29092
```

Producer & Consumer

4.Producer & Consumer 설치

```
문제 출력 터미널 포트
choemin-uui-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker kafka-topics --create --topic topic-test
--bootstrap-server broker:29092 --partitions 1 --replication-factor 1
Created topic topic-test.
choemin-uui-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker kafka-topics --describe --topic topic-test
--bootstrap-server broker:29092
Topic: topic-test      TopicId: 6ox5oHmMQq-I7QKiiIf0Yw PartitionCount: 1      ReplicationFactor: 1      Configs:
Topic: topic-test      Partition: 0      Leader: 1      Replicas: 1      Isr: 1
```

Producer & Consumer

4.Producer & Consumer 설치

- Consumer 접속(Broker Container)

```
$ docker compose -p class6-naive exec broker /bin/bash
```

- Consumer 실행

```
$ kafka-console-consumer --topic topic-test --bootstrap-server broker:29092
```

Producer & Consumer

4.Producer & Consumer 설치

```
문제  출력  터미널  포트
choemin-uui-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker kafka-topics --create --topic topic-test
--bootstrap-server broker:29092 --partitions 1 --replication-factor 1
Created topic topic-test.
choemin-uui-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker kafka-topics --describe --topic topic-test
--bootstrap-server broker:29092
Topic: topic-test      TopicId: 6ox5oHmMQq-I7QKiiIf0Yw PartitionCount: 1      ReplicationFactor: 1      Configs:
      Topic: topic-test      Partition: 0      Leader: 1      Replicas: 1      Isr: 1
choemin-uui-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker /bin/bash
[appuser@297f215d47ad ~]$ kafka-console-consumer --topic topic-test --bootstrap-server broker:29092
```

Producer & Consumer

4.Producer & Consumer 설치

- Producer 접속(Broker Container): 반드시 새로운 터미널에서 실행

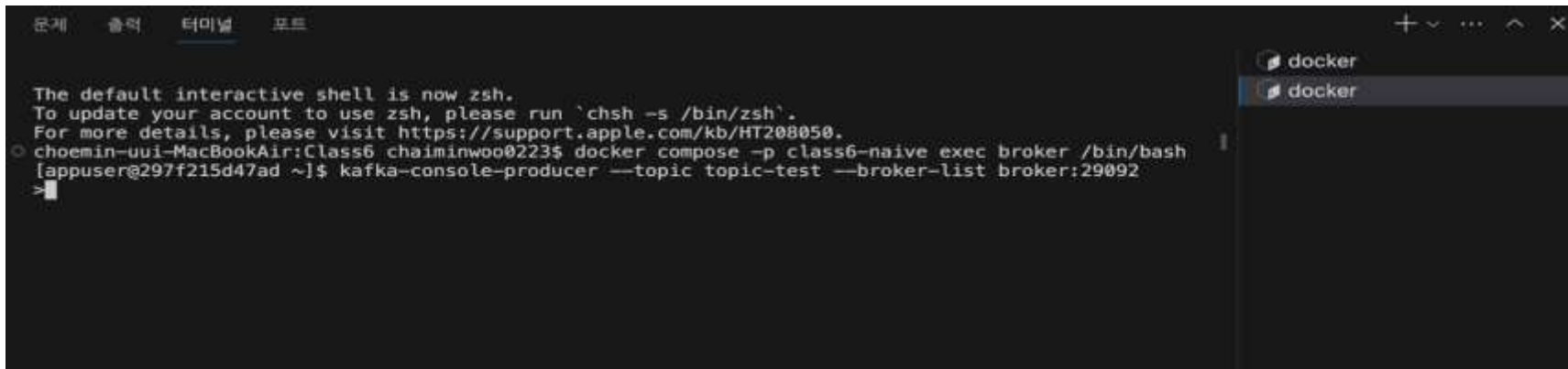
```
$ docker compose -p class6-naive exec broker /bin/bash
```

- Producer 실행

```
$ kafka-console-producer --topic topic-test --broker-list broker:29092
```

Producer & Consumer

4.Producer & Consumer 설치



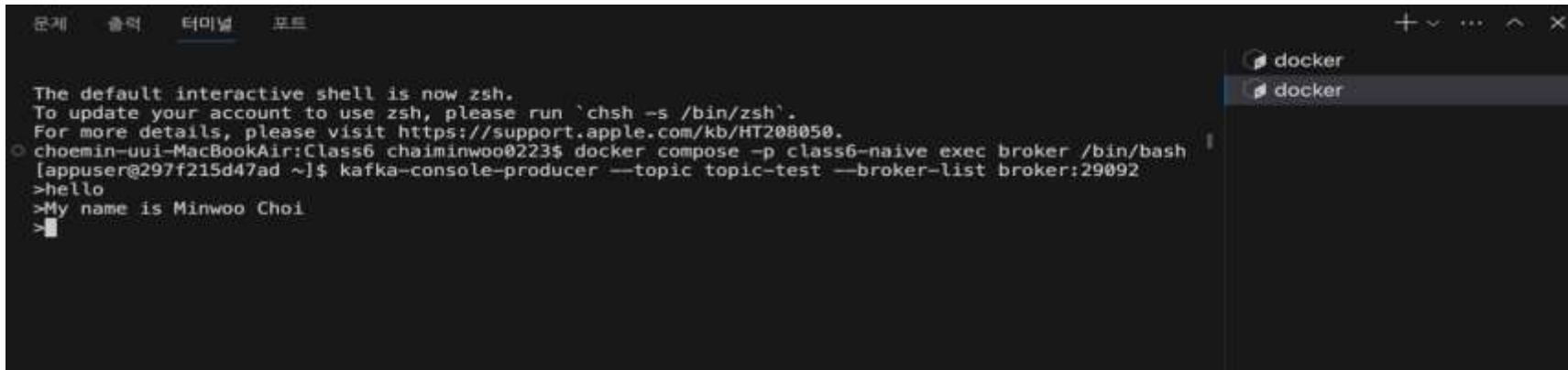
A terminal window with a dark background and light text. The window has tabs at the top labeled '문제', '출력', '터미널', and '포트'. The '터미널' tab is active. The terminal output shows a message about switching to zsh, followed by a command to run 'chsh' and a URL for more details. Then, a command is executed to run 'docker compose' to start a Kafka broker. Finally, the 'kafka-console-producer' command is run to create a topic named 'topic-test' with one broker at 'broker:29092'. The prompt returns to the user.

```
문제 출력 터미널 포트

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
choemin-uu1-MacBookAir:Class6 chainminwoo02223$ docker compose -p class6-naive exec broker /bin/bash
[appuser@297f215d47ad ~]$ kafka-console-producer --topic topic-test --broker-list broker:29092
>
```


Producer & Consumer

4.Producer & Consumer 설치



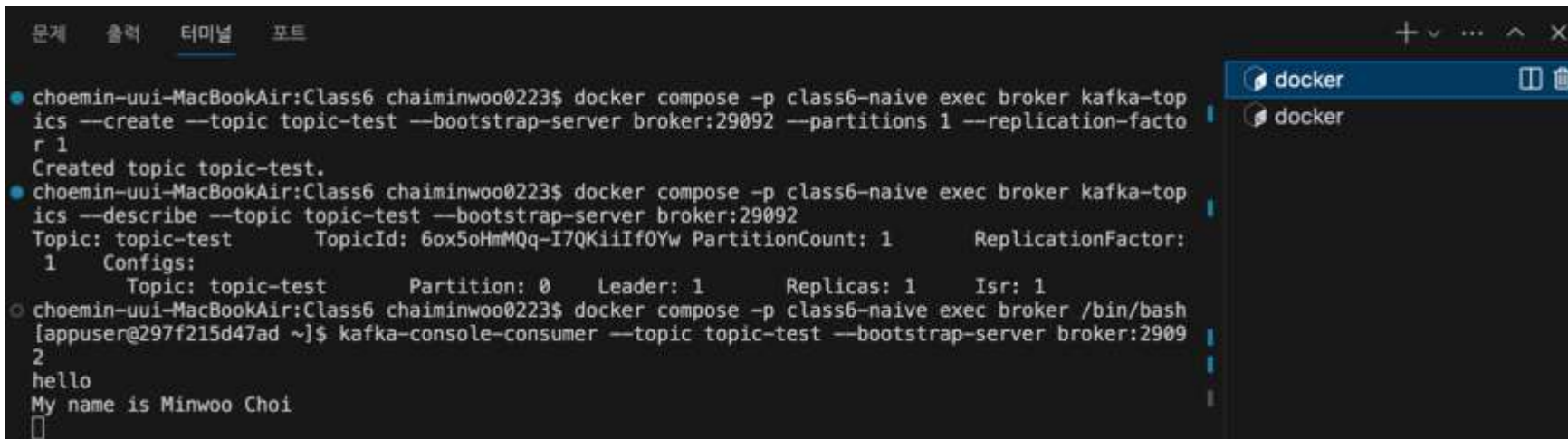
A terminal window with a dark background and light text. The window has tabs at the top labeled '문제', '출력', '터미널', and '포트'. The '터미널' tab is active. The terminal output shows the default shell being updated to zsh, followed by a command to run a Kafka broker in a Docker container. The user then enters 'hello' and 'My name is Minwoo Choi' into the Kafka console producer.

```
문제  출력  터미널  포트

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
choemin-uui-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker /bin/bash
[appuser@297f215d47ad ~]$ kafka-console-producer --topic topic-test --broker-list broker:29092
>hello
>My name is Minwoo Choi
>
```

Producer & Consumer

4.Producer & Consumer 설치



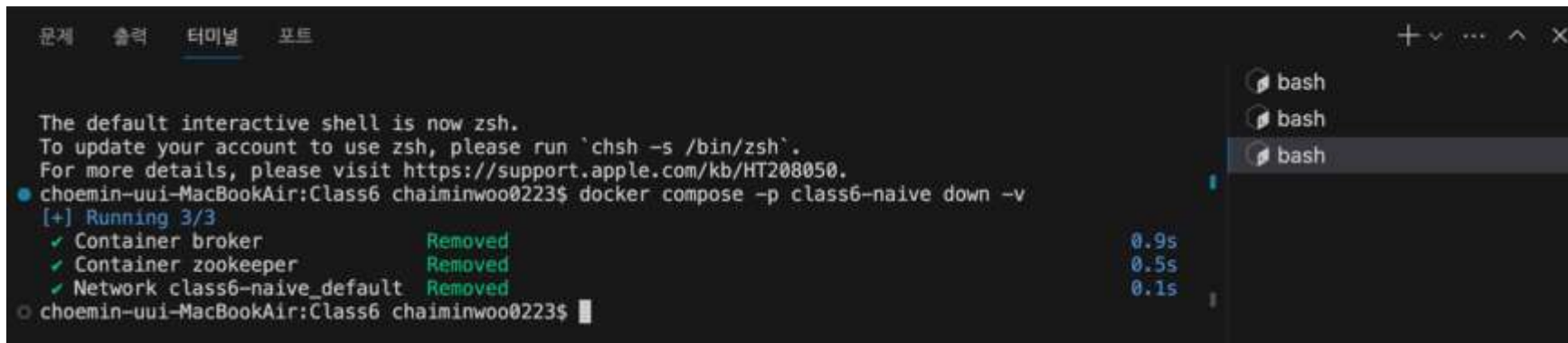
```
문제  출력  터미널  포트
choemin-uu1-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker kafka-topics --create --topic topic-test --bootstrap-server broker:29092 --partitions 1 --replication-factor 1
Created topic topic-test.
choemin-uu1-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker kafka-topics --describe --topic topic-test --bootstrap-server broker:29092
Topic: topic-test      TopicId: 6ox5oHmMQq-I7QKiiIf0Yw PartitionCount: 1      ReplicationFactor: 1
  Configs:
    Topic: topic-test      Partition: 0      Leader: 1      Replicas: 1      Isr: 1
choemin-uu1-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive exec broker /bin/bash
[appuser@297f215d47ad ~]$ kafka-console-consumer --topic topic-test --bootstrap-server broker:29092
hello
My name is Minwoo Choi
█
```

Producer & Consumer

4.Producer \$ Consumer 종료

- 반드시 새로운 터미널에서 실행

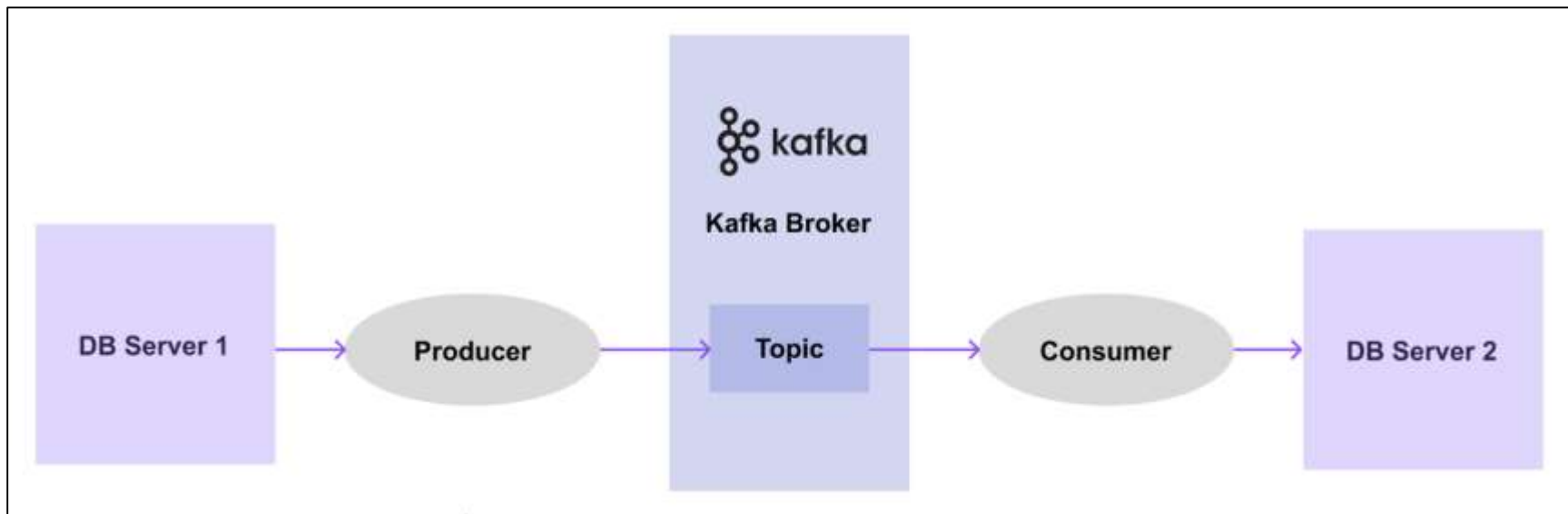
```
$ docker compose -p class6-naive down -v
```



```
문제 출력 터미널 포트
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
● choemin-uui-MacBookAir:Class6 chaiminwoo0223$ docker compose -p class6-naive down -v
[+] Running 3/3
  ✓ Container broker          Removed    0.9s
  ✓ Container zookeeper      Removed    0.5s
  ✓ Network class6-naive_default Removed    0.1s
○ choemin-uui-MacBookAir:Class6 chaiminwoo0223$
```

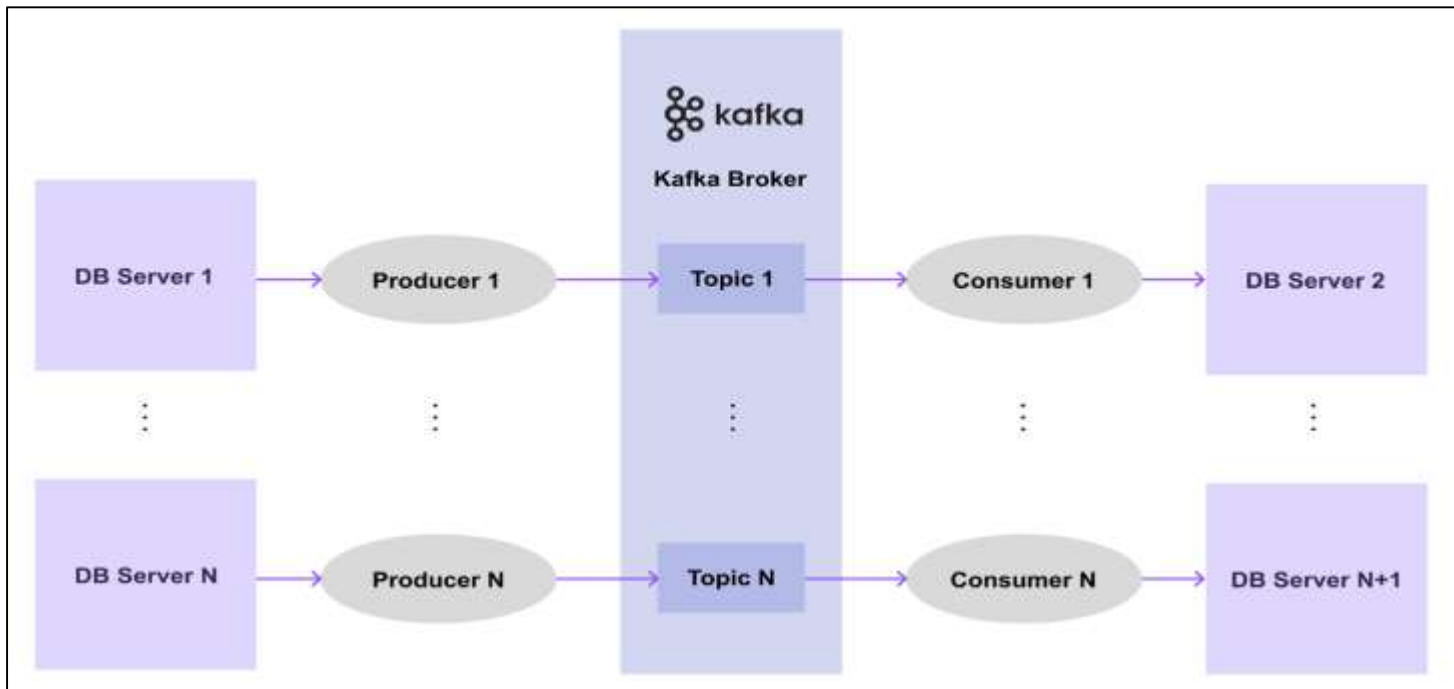
Connect & Connector

1.Producer & Consumer에서 데이터 파이프라인을 구축



Connect & Connector

2.한계



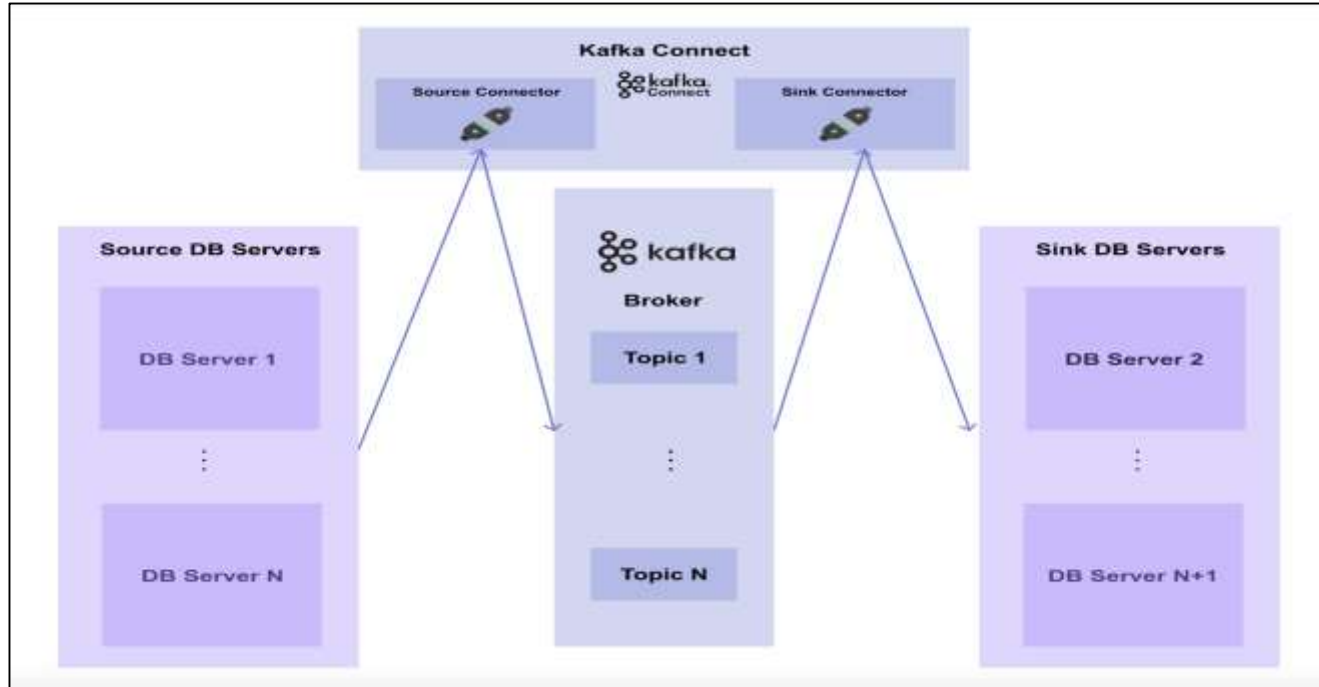
Connect & Connector

3.Connect: DB 서버와 Kafka 간의 데이터를 확장하고,
안전한 방법으로 데이터를 전송하기 위한 도구(프레임워크)

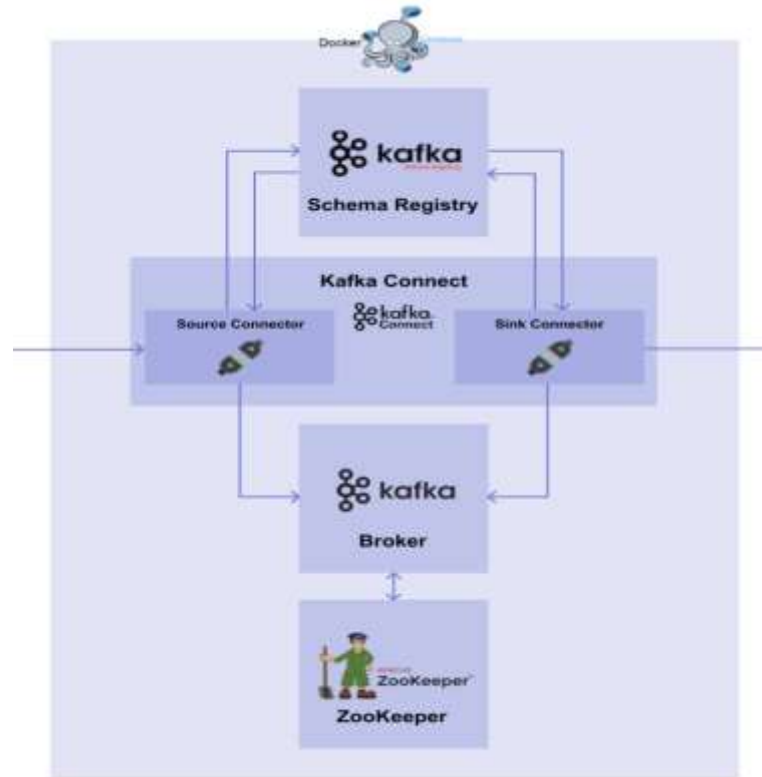
4.Connector: 데이터를 어디로부터 가져오는지, 어디에다가 전달해야 하는지 정의

- Source Connector: **Producer**의 역할을 하는 Connector
- Sink Connector: **Consumer**의 역할을 하는 Connector

Connect & Connector



Kafka System



Kafka System

1.Zookeeper & Broker & Schema Registry 설치(kafka-docker-compose.yaml)

kafka-docker-compose.yaml

```
version: "3"

services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.3.0
    container_name: zookeeper
    ports:
      - 2181:2181
    environment:
      ZOOKEEPER_SERVER_ID: 1
      ZOOKEEPER_CLIENT_PORT: 2181
  broker:
    image: confluentinc/cp-kafka:7.3.0
    container_name: broker
    depends_on:
      - zookeeper
    ports:
      - 9092:9092
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
```

Kafka System

2.Connect: 이미지를 build하기 위한 Dockerfile이 필요(connect.Dockerfile)

connect.Dockerfile

```
# connect.Dockerfile
```

```
FROM confluentinc/cp-kafka-connect:7.3.0
```

```
ENV CONNECT_PLUGIN_PATH="/usr/share/java,/usr/share/confluent-hub-components"
```

```
RUN confluent-hub install --no-prompt snowflakeinc/snowflake-kafka-connector:1.5.5 &&\
    confluent-hub install --no-prompt confluentinc/kafka-connect-jdbc:10.2.2 &&\
    confluent-hub install --no-prompt confluentinc/kafka-connect-json-schema-converter:7.3.
```

Kafka System

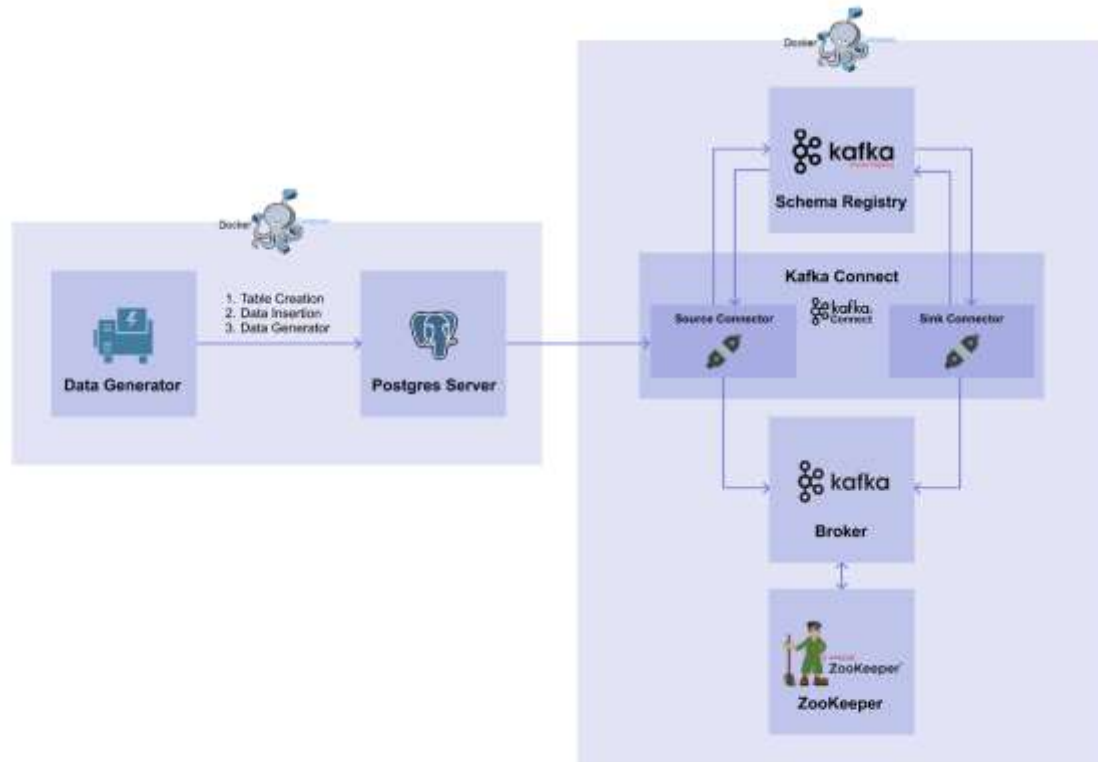
3.Kafka System 실행

```
$ docker compose -p class6-kafka -f kafka-docker-compose.yaml up -d
```

4.Kafka System 이미지 확인

```
$ docker ps
```

Source Connector



Source Connector

1.source_connector.json

source_connector.json

```
{
  "name": "postgres-source-connector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "connection.url": "jdbc:postgresql://postgres-server:5432/mydatabase",
    "connection.user": "myuser",
    "connection.password": "mypassword",
    "table.whitelist": "iris_data",
    "topic.prefix": "postgres-source-",
    "topic.creation.default.partitions": 1,
    "topic.creation.default.replication.factor": 1,
    "mode": "incrementing",
    "incrementing.column.name": "id",
    "tasks.max": 2,
    "transforms": "TimestampConverter",
    "transforms.TimestampConverter.type": "org.apache.kafka.connect.transforms.TimestampConverter",
    "transforms.TimestampConverter.field": "timestamp",
    "transforms.TimestampConverter.format": "yyyy-MM-dd HH:mm:ss.S",
    "transforms.TimestampConverter.target.type": "string"
  }
}
```

Source Connector

2. curl 명령어를 이용하여 POST method 로 Source Connector 를 생성

```
$ curl -X POST http://localhost:8083/connectors -H "Content-Type: application/json" -d @source_connector.json
```

```
● choemin-ui-MacBookAir:Class6 chaiminwoo0223$ curl -X POST http://localhost:8083/connectors -H "Content-Type: application/json" -d @source_connector.json
{"name":"postgres-source-connector","config":{"connector.class":"io.confluent.connect.jdbc.JdbcSourceConnector","connection.url":"jdbc:postgresql://postgres-server:5432/mydatabase","connection.user":"myuser","connection.password":"mypassword","table.whitelist":"iris_data","topic.prefix":"postgres-source-","topic.creation.default.partitions":"1","topic.creation.default.replication.factor":"1","mode":"incrementing","incrementing.column.name":"id","tasks.max":"2","transforms":"TimestampConverter","transforms.TimestampConverter.type":"org.apache.kafka.connect.transforms.TimestampConverter$Value","transforms.TimestampConverter.field":"timestamp","transforms.TimestampConverter.format":"yyyy-MM-dd HH:mm:ss.S","transforms.TimestampConverter.target.type":"string","name":"postgres-source-connector"},"tasks":[],"type":"source"}choemin-ui-MacBookAir:Class6 chaiminwoo0223$
```

Source Connector

3.kafkacat 설치

```
$ brew install kcat
```

```
choenin-uui-MacBookAir:Class6 chaininwoo0223$ brew install kcat
Running 'brew update --auto-update'...
=> Auto-updated Homebrew!
Updated 3 taps (homebrew/services, homebrew/core and homebrew/cask).
=> New Formulae
asmfmt      flowpipe    helm-docs   kin          sui
deadfinder  g-ls        icloudpd    ntm
=> New Casks
cleanuppuddy  easydevo    lunarbar    numuplayer   nrfutil

You have 16 outdated formulae and 1 outdated cask installed.

=> Downloading https://ghcr.io/v2/homebrew/core/kcat/manifests/1.7.0
##### 100.0%
=> Fetching dependencies for kcat: jansson, snappy, avro-c, lzlib, librdkafka, libserde andyajl
=> Downloading https://ghcr.io/v2/homebrew/core/jansson/manifests/2.14
##### 100.0%
=> Fetching jansson
=> Downloading https://ghcr.io/v2/homebrew/core/jansson/blobs/sha256:6652690ceed7b1425bc5f3ebb099
##### 100.0%
=> Downloading https://ghcr.io/v2/homebrew/core/snappy/manifests/1.1.10
##### 100.0%
=> Fetching snappy
=> Downloading https://ghcr.io/v2/homebrew/core/snappy/blobs/sha256:dbe6bca5814b986d91bf204c4b59d
##### 100.0%
=> Downloading https://ghcr.io/v2/homebrew/core/avro-c/manifests/1.11.3
##### 100.0%
=> Fetching avro-c
=> Downloading https://ghcr.io/v2/homebrew/core/avro-c/blobs/sha256:486572382a8323c7816b6244588ec
```

Source Connector

4. 모든 Topic 리스트 확인

```
$ kcat -L -b localhost:9092
```

```
•  
•  
•  
topic "postgres-source-iris_data" with 1 partitions:  
    partition 0, leader 1, replicas: 1, isrs: 1  
•  
•  
•
```

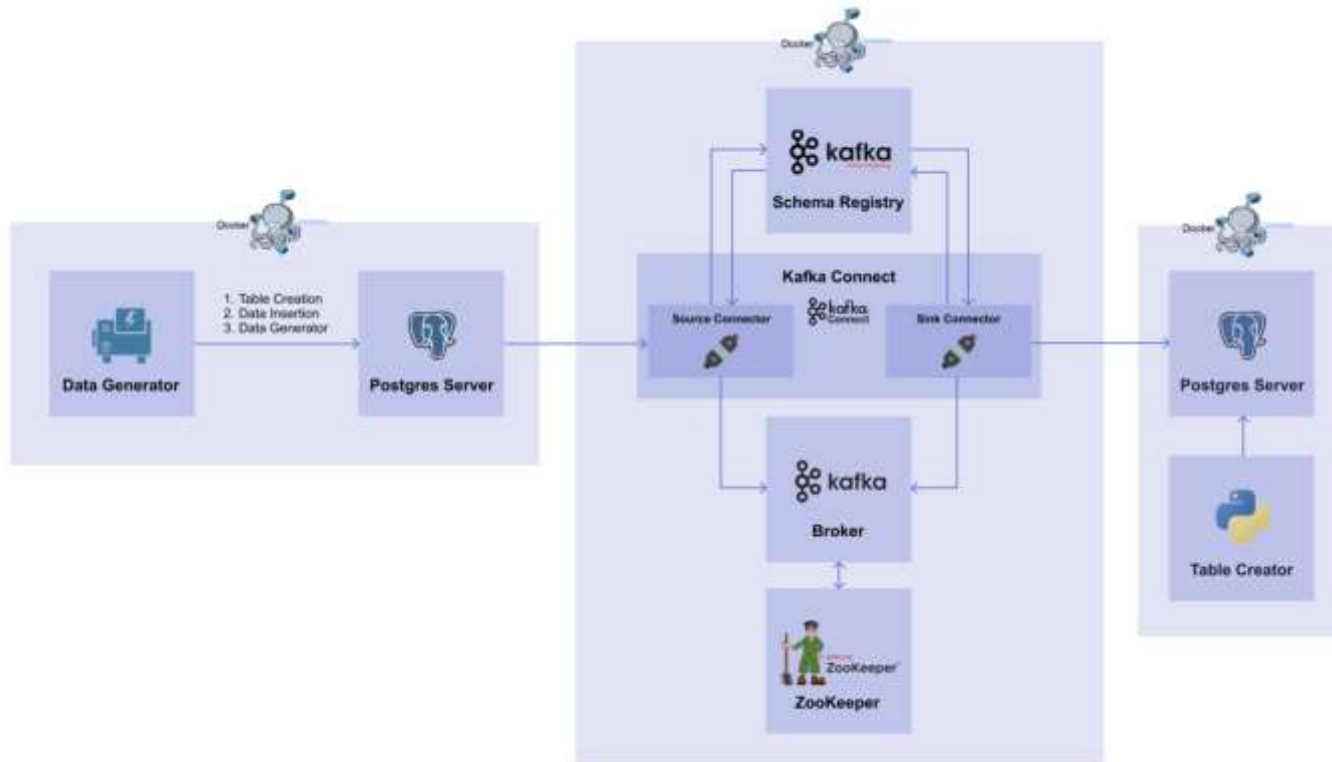

Source Connector

5.postgres-source-iris-data

```
$ kcat -b localhost:9092 -t postgres-source-iris_data
```

```
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}, {"type":"  
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}, {"type":"  
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}, {"type":"  
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}, {"type":"  
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}, {"type":"  
% Reached end of topic postgres-source-iris_data [0] at offset 191
```

Sink Connector



Sink Connector

1.create_table.py

```
# create_table.py
import psycopg2

def create_table(db_connect):
    create_table_query = """
    CREATE TABLE IF NOT EXISTS iris_data (
        id SERIAL PRIMARY KEY,
        timestamp timestamp,
        sepal_length float8,
        sepal_width float8,
        petal_length float8,
        petal_width float8,
        target int
    );"""
    print(create_table_query)
    with db_connect.cursor() as cur:
        cur.execute(create_table_query)
        db_connect.commit()

if __name__ == "__main__":
    db_connect = psycopg2.connect(
        user="targetuser",
        password="targetpassword",
        host="target-postgres-server",
        port=5432,
        database="targetdatabase",
    )
    create_table(db_connect)
```

Sink Connector

2.target.Dockerfile

target.Dockerfile

```
# target.Dockerfile
FROM amd64/python:3.9-slim

WORKDIR /usr/app

RUN pip install -U pip &&\
    pip install psycpg2-binary

COPY create_table.py create_table.py

ENTRYPOINT ["python", "create_table.py"]
```

Sink Connector

3.target-docker-compose.yaml

target-docker-compose.yaml

```
# target-docker-compose.yaml
version: "3"

services:
  target-postgres-server:
    image: postgres:14.0
    container_name: target-postgres-server
    ports:
      - 5433:5433
    environment:
      POSTGRES_USER: targetuser
      POSTGRES_PASSWORD: targetpassword
      POSTGRES_DB: targetdatabase
    healthcheck:
      test: ["CMD", "pg_isready", "-q", "-U", "targetuser", "-d", "targetdatabase"]
      interval: 10s
      timeout: 5s
      retries: 5

  table-creator:
    build:
      context: .
      dockerfile: target.Dockerfile
    container_name: table-creator
    depends_on:
      target-postgres-server:
        condition: service_healthy
```

Sink Connector

4.Target DB 서버와 Table Creator 생성

```
$ docker compose -p class6-target f target-docker-compose.yaml up -d
```

Sink Connector

5.sink_connector.json

sink_connector.json

```
{
  "name": "postgres-sink-connector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
    "connection.url": "jdbc:postgresql://target-postgres-server:5432/targetdatabase",
    "connection.user": "targetuser",
    "connection.password": "targetpassword",
    "table.name.format": "iris_data",
    "topics": "postgres-source-iris_data",
    "auto.create": false,
    "auto.evolve": false,
    "tasks.max": 2,
    "transforms": "TimestampConverter",
    "transforms.TimestampConverter.type": "org.apache.kafka.connect.transforms.Timestamp",
    "transforms.TimestampConverter.field": "timestamp",
    "transforms.TimestampConverter.format": "yyyy-MM-dd HH:mm:ss.S",
    "transforms.TimestampConverter.target.type": "Timestamp"
  }
}
```

Sink Connector

2. curl 명령어를 이용하여 POST method 로 Sink Connector 를 생성

```
$ curl -X POST http://localhost:8083/connectors -H "Content-Type: application/json" -d @sink_connector.json
```

```
● choemin-ui-MacBookAir:Class6 chaiminwoo0223$ curl -X POST http://localhost:8083/connectors -H "Content-Type: application/json" -d @source_connector.json
{"name":"postgres-source-connector","config":{"connector.class":"io.confluent.connect.jdbc.JdbcSourceConnector","connection.url":"jdbc:postgresql://postgres-server:5432/mydatabase","connection.user":"myuser","connection.password":"mypassword","table.whitelist":"iris_data","topic.prefix":"postgres-source-","topic.creation.default.partitions":"1","topic.creation.default.replication.factor":"1","mode":"incrementing","incrementing.column.name":"id","tasks.max":"2","transforms":"TimestampConverter","transforms.TimestampConverter.type":"org.apache.kafka.connect.transforms.TimestampConverter$Value","transforms.TimestampConverter.field":"timestamp","transforms.TimestampConverter.format":"yyyy-MM-dd HH:mm:ss.S","transforms.TimestampConverter.target.type":"string","name":"postgres-source-connector"},"tasks":[],"type":"source"}choemin-ui-MacBookAir:Class6 chaiminwoo0223$
```


Sink Connector

3. 데이터 확인

- psql 실행
- Target DB에 접속하고, 데이터 확인

PGPASSWORD=targetpassword psql -h localhost -p **5433** -U targetuser -d targetdatabase

Target DB Server

- **Image** : postgres:14.0
- **Container name** : target-postgres-server
- **POSTGRES_USER** : targetuser
- **POSTGRES_PASSWORD** : targetpassword
- **POSTGRES_DB** : targetdatabase
- **Port forwarding** : 5433:5432



```
chaiminwoo0223 ~ psql - runpsql.sh -- 80x24
last login: Sat Dec 30 17:34:19 on ttya000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
/Library/PostgreSQL/16/scripts/runpsql.sh; exit
chaiminwoo0223 ~ psql - runpsql.sh -- 80x24
Server [localhost]: PGPASSWORD=targetpassword psql -h localhost -p 5433 -U targetuser -d
mydatabase
Database [postgres]: postgres-server
Port [5432]: 5432
Username [postgres]: myuser
psql: warning: extra command-line argument "psql" ignored
psql: warning: extra command-line argument "postgres-server" ignored
Password for user myuser:
psql (16.1, server 14.0 (Debian 14.0-1.pgdg130+1))
Type "help" for help.

mydatabase=#
```

Sink Connector

3. 데이터 확인

- psql 실행
- Target DB에 접속하고, 데이터 확인

```
targetdatabase=# SELECT * FROM iris_data LIMIT 10;
```

참고자료

- <https://mlops-for-mle.github.io/tutorial/>
- <https://velog.io/@holicme7/apache-kafka-%EC%B9%B4%ED%94%84%EC%B9%B4%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80>
- ChatGPT 4

Thank You