**Kafka Connect**

* Kafka Connect를 이용한 CDC(Change Data Capture)를 통해 데이터 동기화를 실습한다.
* Connect는 Connector를 실행시켜주는 서버로 DB동기화시, 벤더사가 만든 Connector, 또는 OSS(Debezium, Confluent) 계열의 Connector를 사용한다.
* Lab에서는 경량의 h2 DB를 사용한다.

**Connector, H2 database 다운로드**

* H2 DB와 Kafka Connect를 위한 JDBC 드라이브를 다운로드한다.

**git clone https://github.com/acmexii/kafka-connect.git**

**cd kakka-connect**

* h2-database 아카이브를 압축해제한다.

**unzip h2.zip**

**H2 데이터베이스 실행**

* bin 폴더로 이동해 h2 database를 서버모드로 실행한다.

**cd bin**

**chmod 755 h2.sh**

**./h2.sh -webPort 8087 -tcpPort 9099**

* 지정한 webPort로 Client WebUI가 접근가능하며, h2 database는 9099포트(default 9092)로 실행된다.

**Kafka JDBC Connector 설치**

* Jdbc Connector는 설치된 Kafka 서버에 등록하고 사용한다.
* Connector를 설치할 폴더를 생성한다.

**cd $kafka\_home**

**mkdir connectors**

**cd connectors**

* 다운받은 confluentinc-kafka-connect-jdbc-10.2.5.zip을 복사 후 unzip 한다.

**cp /home/project/advanced-connect/kafka-connect/confluentinc-kafka-connect-jdbc-10.2.5.zip ./**

**unzip confluentinc-kafka-connect-jdbc-10.2.5.zip**

**Connect 서버에 Connector 등록**

* kafka Connect에 설치한 Confluent jdbc Connector를 등록한다.
* $kafka\_home/config 폴더로 이동 후 connect-distributed.properties 파일 오픈하고,

**cd $kafka\_home/config**

**vi connect-distributed.properties**

* 마지막 행을 plugin.path=/usr/local/kafka/connectors 로 편집하고 저장종료한다.

**Kafka Connect 실행**

* $kafka\_home에서 connect를 실행한다.

**cd $kafka\_home**

**bin/connect-distributed.sh config/connect-distributed.properties**

* Kafka Connect는 default 8083 포트로 실행이 된다.
* Labs > 포트확인 메뉴를 통해 실행중 서비스를 확인한다.

**root@labs-315390334:/home/project# netstat -lntp | grep :808**

**tcp 0 0 0.0.0.0:8083 0.0.0.0:\* LISTEN 23597/java**

**tcp 0 0 0.0.0.0:8087 0.0.0.0:\* LISTEN 21885/java**

**root@labs-315390334:/home/project#**

* Kafka topic을 확인해 본다.

**/usr/local/kafka/bin/kafka-topics.sh --bootstrap-server localhost:9092 --list**

* Connect를 위한 토픽이 추가되었다.

connect-configs, connect-offsets, connect-status

**Source Connector 설치**

* Kafka connect의 REST API를 통해 Source 및 Sink connector를 등록한다.

**curl -i -X POST -H "Accept:application/json" \**

**-H "Content-Type:application/json" http://localhost:8083/connectors/ \**

**-d '{**

**"name": "h2-source-connect",**

**"config": {**

**"connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",**

**"connection.url": "jdbc:h2:tcp://localhost:9099/./test",**

**"connection.user":"sa",**

**"connection.password":"passwd",**

**"mode":"incrementing",**

**"incrementing.column.name" : "ID",**

**"table.whitelist" : "ORDER\_TABLE",**

**"topic.prefix" : "CONNECT\_",**

**"tasks.max" : "1"**

**}**

**}'**

Connector 등록시, ‘No suitable driver’ 오류가 발생할 경우, Classpath에 h2 driver를 설정해 준다.  
h2/bin에 있는 JDBC 드라이브를 $kafka\_home/lib에 복사하고 다시 Connect를 실행한다.

* 등록한 Connector를 확인한다.

**http localhost:8083/connectors**

**Order 마이크로서비스 설정**

* 주문 서비스를 서버모드로 실행한 h2 Database에 연결한다.
* Order의 application.yml을 열어 default profile의 datasource를 수정한다.

**datasource:**

**url: jdbc:h2:tcp://localhost:9099/./test**

**username: sa**

**password: passwd**

**driverClassName: org.h2.Driver**

**소스 테이블에 Data 입력**

* order 마이크로서비스를 기동하고 소스 테이블에 데이터를 생성한다.

**cd order**

**mvn spring-boot:run**

**http POST :8081/orders message="1st OrderPlaced."**

**http POST :8081/orders message="2nd OrderPlaced."**

* Kafka topic을 확인해 본다.

**/usr/local/kafka/bin/kafka-topics.sh --bootstrap-server localhost:9092 --list**

* ‘CONNECT\_ORDER\_TABLE’ 토픽이 추가되어 목록에 나타난다.

Kafka Connect는 테이블 단위로 토픽이 생성되어 Provider와 Consumer간 데이터를 Sync합니다.

**$kafka\_home/bin/kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic CONNECT\_ORDER\_TABLE --from-beginning**

**Sink Connector 설치**

**curl -i -X POST -H "Accept:application/json" \**

**-H "Content-Type:application/json" http://localhost:8083/connectors/ \**

**-d '{**

**"name": "h2-sink-connect",**

**"config": {**

**"connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",**

**"connection.url": "jdbc:h2:tcp://localhost:9099/./test",**

**"connection.user":"sa",**

**"connection.password":"passwd",**

**"auto.create":"true",**

**"auto.evolve":"true",**

**"delete.enabled":"false",**

**"tasks.max":"1",**

**"topics":"CONNECT\_ORDER\_TABLE"**

**}**

**}'**

**타겟 테이블 Data 확인**

* Sync대상 테이블을 조회한다.

**cd order**

**mvn spring-boot:run**

**http GET :8081/syncOrders**

Sink Connector를 통해 syncOrders 테이블에 복제된 데이터가 조회된다

* 다시한번 Orders 테이블에 데이터를 입력하고 확인해 본다.

**http POST :8081/orders message="3rd OrderPlaced."**

**http GET :8081/syncOrders**