

Kafka connector Source and Sink

📍 상태	3rd party
🕒 작성일시	@2022년 3월 11일 오후 5:30

구성 솔루션

Azure Event Hub
Apache Kafka connector
Debezium, confluent
oracle 21c
Azure database for Postgre

참고문헌

Apache Kafka Connect와의 통합 - Azure Event Hubs - Azure Event Hubs

Apache Kafka Connect 는 Kafka 클러스터를 통해 MySQL, HDFS 같은 외부 시스템 및 파일 시스템에 연결하고 데이터를 가져오는/내보내는 프레임워크입니다. 이 자습서에서는 Event hubs와 함께 Kafka Connect 프레임워크를 사용하는 방법을 안내합니다. 경고 Apache Kafka 연결 프레임 워크와 커넥터의 사용은 Microsoft Azure <https://docs.microsoft.com/ko-kr/azure/event-hubs/event-hubs-kafka-connect-tutorial>

변경 데이터 캡처를 위해 Azure Event Hubs의 Apache Kafka Connect를 Debezium과 통합 - Azure Event Hubs

CDC(변경 데이터 캡처) 는 만들기, 업데이트 및 삭제 작업에 대한 응답으로 데이터베이스 테이블의 행 수준 변경 내용을 추적하는 데 사용되는 기술입니다. Debezium은 다양한 데이터베이스에서 사용할 수 있는 변경 데이터 캡처 기능(예: PostgreSQL의 논리적 디코딩)을 토대로 구축되는 분산 플랫폼입니다. 데이터베이스 테이블에서 행 수준 변경 내용을 이벤트 스트림으로 변환한 후 <https://docs.microsoft.com/ko-kr/azure/event-hubs/event-hubs-kafka-connect-debezium>

Logical replication and logical decoding - Azure Database for PostgreSQL - Flexible Server

Azure Database for PostgreSQL - Flexible Server supports the following logical data extraction and replication methodologies: Logical replication Using PostgreSQL native logical replication to replicate data objects. Logical replication allows fine-grained control over the data replication, including table-level <https://docs.microsoft.com/en-us/azure/postgresql/flexible-server/concepts-logical>

<https://debezium.io/documentation/reference/1.9/connectors/oracle.html>

<https://github.com/edenhill/kcat>

<https://oracle-base.com/articles/21c/oracle-db-21c-installation-on-oracle-linux-8>

<https://www.confluent.io/hub/confluentinc/kafka-connect-jdbc>

<https://www.oracle.com/kr/database/technologies/appdev/jdbc-downloads.html>

<https://debezium.io/documentation/reference/1.9/connectors/oracle.html>

Kafka Connector Server Setting [Cent OS 8.5]

💡 명령어는 모두 root 기준입니다.

⚠️ Cent OS 8 는 EOL 되었습니다.

1. Kafka release / Java 설치

```
# Kafka release / Java 설치
wget https://dlcdn.apache.org/kafka/3.2.0/kafka_2.12-3.2.0.tgz

tar -xzf kafka_2.12-3.2.0.tgz
cd kafka_2.12-3.2.0

dnf -y install java-11-openjdk
dnf -y install java-11-openjdk-devel
```

Java 버전이 여러개 깔려 있다면 아래 명령어로 변경 (하기 문서내용은 java8 에서 실패)

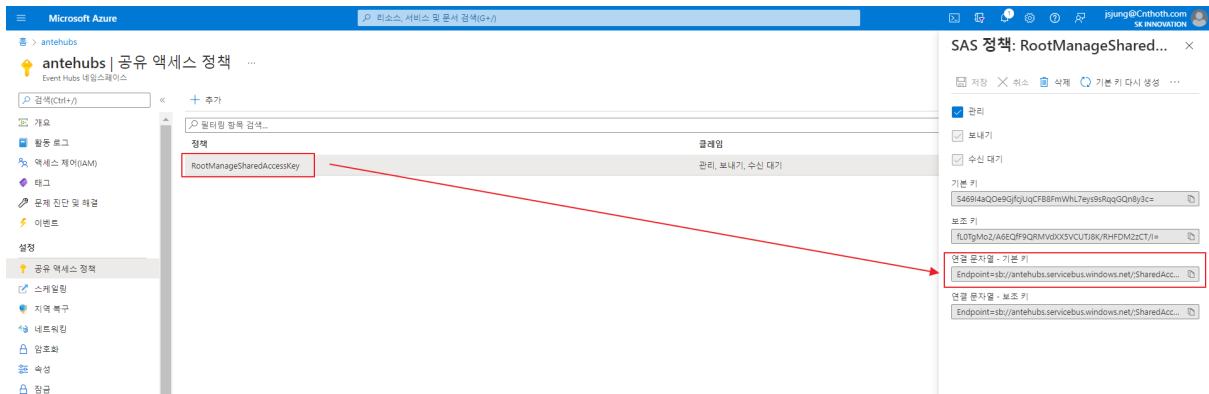
```
alternatives --config java
alternatives --config javac
```

2. Event Hub Namespace 생성

① 참고

Kafka용 Event Hubs는 기본 계층에서 지원되지 않습니다.

- 이 시나리오에서는 기본 제공하는 SAS 인 RootManageSharedAccessKey 로 사용해야함.



3. Kafka connector 설정

```
vi ./config/connect-distributed.properties
```

```
# EventHub namespace info
bootstrap.servers=ehants.servicebus.windows.net:9093
group.id=connect-cluster-group

config.storage.topic=connect-cluster-configs
offset.storage.topic=connect-cluster-offsets
status.storage.topic=connect-cluster-status

config.storage.replication.factor=1
offset.storage.replication.factor=1
status.storage.replication.factor=1

rest.advertised.host.name=connect
offset.flush.interval.ms=10000

key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
internal.key.converter=org.apache.kafka.connect.json.JsonConverter
internal.value.converter=org.apache.kafka.connect.json.JsonConverter
```

```

internal.key.converter.schemas.enable=false
internal.value.converter.schemas.enable=false

# Password = EventHub ConnectionString
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="$ConnectionString" password="Endpoint=sb://ehants.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=ILZSbJNszyGisX3D7LX0j0b9fM48zTm48Dwaley1jzM=";

producer.security.protocol=SASL_SSL
producer.sasl.mechanism=PLAIN
producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="$ConnectionString" password="Endpoint=sb://ehants.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=ILZSbJNszyGisX3D7LX0j0b9fM48zTm48Dwaley1jzM=";

consumer.security.protocol=SASL_SSL
consumer.sasl.mechanism=PLAIN
consumer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="$ConnectionString" password="Endpoint=sb://ehants.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=ILZSbJNszyGisX3D7LX0j0b9fM48zTm48Dwaley1jzM=";

# Kafka release location
plugin.path=/root/kafka_2.12-3.2.0/libs

```

4. Kafka Connector server 실행

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

INFO Finished starting connectors and tasks 가 보이면 준비 됨.

Debezium Setting

Postgresql Source Connector

1. Postgresql Source Connector plugin 다운로드 및 설정

```

cd ~
wget https://repo1.maven.org/maven2/io/debezium/debezium-connector-postgres/1.9.2.Final/debezium-connector-postgres-1.9.2.Final-plugin.tar.gz

#jar 파일들을 Kafka Connector 설정 파일에 구성된 plugin.path 에 넣는다.
tar -zxvf debezium-connector-postgres-1.9.2.Final-plugin.tar.gz
cp ./debezium-connector-postgres/*.jar /root/kafka_2.12-3.2.0/libs/

```

2. pg-source-connector.json 생성

```

{
  "name": "pgs-connector",
  "config": {
    "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
    "database.hostname": "poant.postgres.database.azure.com",
    "database.port": "5432",
    "database.user": "ant",
    "database.password": "QWERasdf123!",
    "database.dbname": "pgdbant",
    "database.server.name": "antns",
    "table.include.list": "public.STABLES",
    "plugin.name": "pgoutput"
  }
}

```

name : 임의의 Connector 이름
database.hostname : postgres 접속 URL
database.dbname : postgres 의 DB 이름
database.server.name : DB가 속한 **schema**(alias namespace) - 적으면 알아서 생성
table.include.list : 수집할 테이블 이름
plugin.name : 논리적 디코딩 출력 플러그인

데이터 스키마 레지스트리 기본 설정은 avro format 임.

PostgreSQL Setting

heidisql 로 접속 (Linux 환경에서는 Psql 사용)

Postgre 서버에 DB 생성

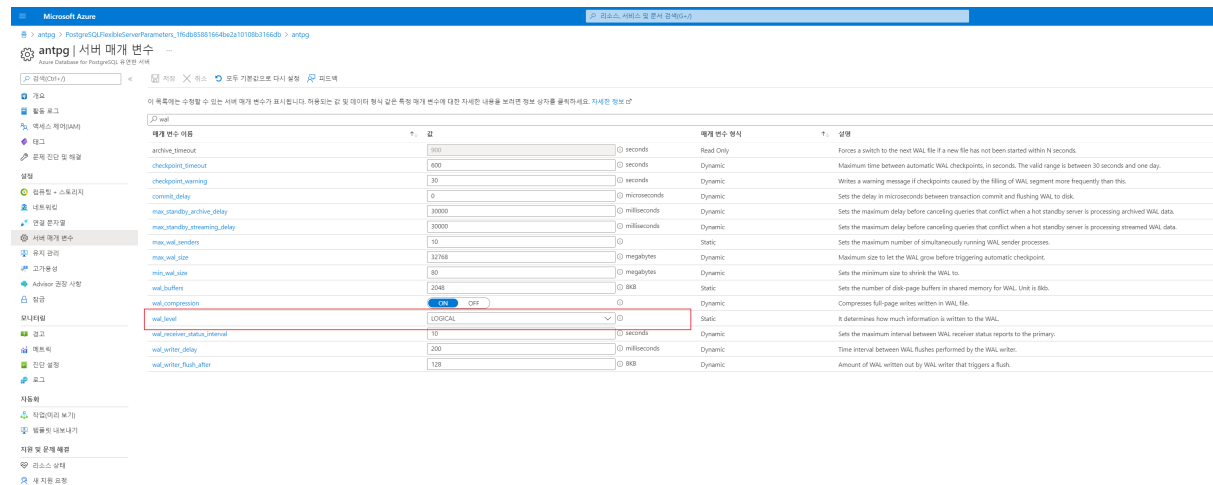
```
create database antpostgres;
#확인
SELECT datname FROM pg_database WHERE datistemplate=FALSE;
```

table 생성 및 데이터 입력 ("" 는 값들을 대문자로 유지하기 위해 사용)

```
CREATE TABLE "STABLES" ("ID" SERIAL, "DESCRIPTION" VARCHAR(50), "TODO_STATUS" VARCHAR(12), PRIMARY KEY("ID"));

INSERT INTO "STABLES" ("DESCRIPTION", "TODO_STATUS") VALUES ('setup postgresql on azure', 'complete');
INSERT INTO "STABLES" ("DESCRIPTION", "TODO_STATUS") VALUES ('setup kafka connect', 'complete');
INSERT INTO "STABLES" ("DESCRIPTION", "TODO_STATUS") VALUES ('configure and install connector', 'in-progress');
INSERT INTO "STABLES" ("DESCRIPTION", "TODO_STATUS") VALUES ('start connector', 'pending');
```

Azure portal의 서버 매개 변수에서 wal_level 을 logical 로 변경 (서버 재시작됨)



Postgresql 에서 복제 권한 부여

```
ALTER ROLE ant WITH REPLICATION;
```

Source Connector 생성

Postgre source Connector 생성

```
#위에서 작성한 pg-source-connector.json 를 사용합니다.
curl -X POST -H "Content-Type: application/json" --data @pg-source-connector.json http://localhost:8083/connectors
```

확인 및 삭제 명령어

```
#확인 # todo-connector 는 pg-source-connector.json 에서 임의로 정한 connector 이름
curl -s http://localhost:8083/connectors/pgs-connector/status

#원가 오류가 있다면 하기의 명령어로 conetor 삭제
curl -X DELETE http://localhost:8083/connectors/pgs-connector
```

확인시 정상

```
{ "name": "pgs-connector", "connector": { "state": "RUNNING", "worker_id": "connect:8083" }, "tasks": [ { "id": 0, "state": "RUNNING", "worker_id": "connect:8083" }, { "type": "source" } ] }
```

CDC 테스트

💡 테스트를 위한 방법은 여러가지가 있지만 이번 시나리오에서는 Kcat tool을 이용합니다.

kafkacat 설치를 위한 준비 (On CentOS 8)

~/config/kcat.conf 생성 (/root/.config/kcat.conf)

`sasl.password` 항목에 "" 를 사용하면 안됨.

```
metadata.broker.list=ehants.servicebus.windows.net:9093
security.protocol=SASL_SSL
sasl.mechanisms=PLAIN
sasl.username=$ConnectionString
sasl.password=Endpoint=sb://ehants.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=ILZSbjNszY6isX3D7LX0j0b9fM48z7m48Dwaley1jzM=
```

kcat 을 위한 패키지 설치 및 빌드

```
cd ~
dnf group install -y "Development Tools"
dnf install -y git cyrus-sasl-devel cmake libcurl-devel

git clone https://github.com/edenhill/kafkacat.git
cd kafkacat
./bootstrap.sh
```

환경 변수 설정 및 실행

```
export KCAT_CONFIG=/root/.config/kcat.conf
export BROKER=antevh.servicebus.windows.net:9093
export TOPIC=antns.public.stables
./kcat -b $BROKER -t $TOPIC -o beginning
```

결과

```
^ Reached end of topic ant-server.public.todos [0] at offset 4
{"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}, {"type":"string","optional":true,"field":"description"}],("type":"string","optional":true,"field":"todo_status"}],("type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}, {"type":"string","optional":true,"field":"description"}, {"type":"string","optional":true,"field":"todo_status"}],("type":"int32","optional":false,"field":"id"}, {"type":"string","optional":true,"field":"description"}, {"type":"string","optional":false,"field":"version"}],("type":"string","optional":false,"field":"connector"}, {"type":"string","optional":false,"field":"name"}, {"type":"int64","optional":false,"field":"ts_ms"}, {"type":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters":{"allowed":["true","last","false"],"default":"false","field":"snapshot"}, {"type":"string","optional":false,"field":"db"}, {"type":"string","optional":false,"field":"schema"}, {"type":"string","optional":false,"field":"table"}, {"type":"int64","optional":true,"field":"txid"}, {"type":"int64","optional":true,"field":"lsn"}, {"type":"int64","optional":true,"field":"xmin"}],("type":"string","optional":false,"name":"io.debezium.connector.postgresql.Source","field":"source"}, {"type":"string","optional":false,"field":"op"}, {"type":"int64","optional":true,"field":"ts_ms"}, {"type":"struct","fields":[{"type":"string","optional":false,"field":"id"}, {"type":"int64","optional":false,"field":"total_order"}, {"type":"int64","optional":false,"field":"data_collection_order"}],("type":"string","optional":true,"field":"transaction"}],("type":"string","optional":false,"name":"ant-server.public.todos.Envelope"},"payload":{"before":null,"after":{"id":5,"description":"my test","todo_status":"complete"},"source":{"version":"1.2.0.Final","connector":"postgresql","name":"ant-server","ts_ms":1652423228873,"snapshot":"false","db":"postgres","schema":"public","table":"todos","txid":19096,"lsn":3724545736,"xmin":null,"op":"c","ts_ms":1652423228255,"transaction":null}}
```

오류



경우에 따라 다를 수 있음

1. `Creation of replication slot failed`

`ALTER ROLE ant WITH REPLICATION;` 을 실행했는지 확인.

```
{ "name": "todo-connector", "connector": { "state": "RUNNING", "worker_id": "connect:8083" }, "tasks": [
  { "id": 0, "state": "FAILED", "worker_id": "connect:8083", "trace": "io.debezium.DebeziumException:
  Creation of replication slot failed....." }
```

2. `DB나 Table 오류`

이벤트 허브에 다음 사진과 같이 허브들이 생겨있는지 확인.

이후 `connect-distributed.properties` 와 `pg-source-connector.json`를 확인.

이름

[antns.public.stables](#)

[connect-cluster-configs](#)

[connect-cluster-offsets](#)

[connect-cluster-status](#)

3. SASL authentication error

sasl.password 값 확인

```
%3|1652703560.288|FAIL|rdkafka#producer-1|
[thrd:sasl_ssl://ehants.servicebus.windows.net:9093/bootstrap]:
sasl_ssl://ehants.servicebus.windows.net:9093/bootstrap: SASL authentication error: The
Service Bus connection string is not of the expected format. Either there are unexpected
properties within the string or the format is incorrect. Please check the string before trying
again. (after 1ms in state AUTH_REQ)
```

4. NoSuchMethodError: java.nio.ByteBuffer.position

creation of replication slot failed 가 나와도 먼저 관련 설정 말고 위의 오류가 있는지 확인 할 것.
있다면 자바 버전을 11 로 업그레이드.

5. 하나의 커넥터에 대한 레플리카 슬롯... 어찌구저찌구...

이전 커넥터 설정을 동일하게 이용하거나 다른 이름으로 재생성.

만약 안되면 Postgre 에 접속하여 아래 명령어로 레플리카 슬롯 확인 후 drop으로 제거 후 재작업

```
SELECT * FROM pg_replication_slots;
select pg_drop_replication_slot('debezium');
```

6. LOGICAL....

wal_level 이 LOGICAL 이 아니거나 관련 권한이 없을 때

Oracle DB Install [Oracle Linux on CentOS 8]



다음의 명령어는 root 로 실행합니다.

자동 Setup

```
dnf install -y oracle-database-preinstall-21c
```

LINUX.X64_213000_db_home.zip

Oracle 사이트에서 Oracledb 21c zip 파일을 구한다.

Oracle 사용자 패스워드 설정

```
passwd oracle
```

방화벽 끄기

```
systemctl stop firewalld  
systemctl disable firewalld
```

DB가 설치될 디렉토리 생성

```
mkdir -p /u01/app/oracle/product/21.0.0/dbhome_1  
mkdir -p /u02/oradata  
chown -R oracle:oinstall /u01 /u02  
chmod -R 775 /u01 /u02
```

스크립트용 디렉토리 생성

```
mkdir /home/oracle/scripts
```

환경 파일 생성

```
cat > /home/oracle/scripts/setEnv.sh <<EOF  
# Oracle Settings  
export TMP=/tmp  
export TMPDIR=$TMP  
  
export ORACLE_HOSTNAME=ol8-21.localdomain  
export ORACLE_UNQNAME=cdb1  
export ORACLE_BASE=/u01/app/oracle  
export ORACLE_HOME=$ORACLE_BASE/product/21.0.0/dbhome_1  
export ORA_INVENTORY=/u01/app/oraInventory  
export ORACLE_SID=cdb1  
export PDB_NAME=pdb1  
export DATA_DIR=/u02/oradata  
  
export PATH=/usr/sbin:/usr/local/bin:$PATH  
export PATH=$ORACLE_HOME/bin:$PATH  
  
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib  
export CLASSPATH=$ORACLE_HOME/jlib:$ORACLE_HOME/rdbms/jlib  
EOF
```

환경 파일 적용 - 만약 적용이안된다면 직접 입력

```
echo ". /home/oracle/scripts/setEnv.sh" >> /home/oracle/.bash_profile
```

Start & stop 생성


```

cat > /home/oracle/scripts/start_all.sh <<EOF
#!/bin/bash
. /home/oracle/scripts/setEnv.sh

export ORAENV_ASK=NO
. oraenv
export ORAENV_ASK=YES

dbstart \$ORACLE_HOME
EOF

cat > /home/oracle/scripts/stop_all.sh <<EOF
#!/bin/bash
. /home/oracle/scripts/setEnv.sh

export ORAENV_ASK=NO
. oraenv
export ORAENV_ASK=YES

dbshut \$ORACLE_HOME
EOF

chown -R oracle:oinstall /home/oracle/scripts
chmod u+x /home/oracle/scripts/*.sh

```

사용법 (지금은 사용 안함)

```

~/scripts/start_all.sh
~/scripts/stop_all.sh

```

zip 파일을 압축해제 후 설치 (Silent mode 사용)



다음 스크립트는 Oracle 유저로 실행하세요.

```

# Unzip software.
cd $ORACLE_HOME
unzip -oq /ocpack/LINUX.X64_213000_db_home.zip

# Interactive mode.
#./runInstaller

# Silent mode.
./runInstaller -ignorePrereq -waitforcompletion -silent \
  -responseFile ${ORACLE_HOME}/install/response/db_install.rsp \
  oracle.install.option=INSTALL_DB_SWONLY \
  ORACLE_HOSTNAME=${ORACLE_HOSTNAME} \
  UNIX_GROUP_NAME=oinstall \
  INVENTORY_LOCATION=${ORA_INVENTORY} \
  SELECTED_LANGUAGES=en,en_GB \
  ORACLE_HOME=${ORACLE_HOME} \
  ORACLE_BASE=${ORACLE_BASE} \
  oracle.install.db.InstallEdition=EE \
  oracle.install.db.OSDBA_GROUP=dba \
  oracle.install.db.OSBACKUPDBA_GROUP=dba \
  oracle.install.db.OSGDGDBA_GROUP=dba \
  oracle.install.db.OSKMDBA_GROUP=dba \
  oracle.install.db.OSRACDBA_GROUP=dba \
  SECURITY_UPDATES_VIA_MYORACLESUPPORT=false \
  DECLINE_SECURITY_UPDATES=true

```

waring 등 에러가 나오지만 무시하고 다음 단계 진행



다음은 root로 실행

```
/u01/app/oraInventory/oraInstRoot.sh
/u01/app/oracle/product/21.0.0/dbhome_1/root.sh
```



다음은 다시 Oracle 유저로

Oracle 리스너 시작

```
lsnrctl start
```

DB 생성

```
# 경우 따라 bin 폴더를 찾아가서 실행
./dbca -silent -createDatabase
        -templateName General_Purpose.dbc
        -gdbname ${ORACLE_SID} -sid ${ORACLE_SID} -responseFile NO_VALUE
        -characterSet AL32UTF8
        -sysPassword SysPassword1
        -systemPassword SysPassword1
        -createAsContainerDatabase true
        -numberOfPDBs 1
        -pdbName ${PDB_NAME}
        -pdbAdminPassword PdbPassword1
        -databaseType MULTIPURPOSE
        -memoryMgmtType auto_sga
        -totalMemory 2000
        -storageType FS
        -datafileDestination "${DATA_DIR}"
        -redoLogFileSize 50
        -emConfiguration NONE
        -ignorePreReqs
```

서비스 다시시작 설정

/etc/oratab

```
cdb1:/u01/app/oracle/product/21.0.0/dbhome_1:Y
```

```
sqlplus / as sysdba <<EOF
alter system set db_create_file_dest='${DATA_DIR}';
alter pluggable database ${PDB_NAME} save state;
exit;
EOF
```

SQL cmd 접속 (관리자 계정 접속 명령어)

```
sqlplus '/as sysdba'
```

Oracle server 계정 생성 (C## 뒤가 유저이름)

- 기본 설정으로는 C##을 붙이지 않고 유저생성 x

```
create user C##oant identified by "QWERasdf123!!";
```

유저에 오라클 관련 권한 부여

```
grant connect, DBA, RESOURCE to C##oant;
```

설정 변경 적용

```
commit;
```

exit로 나간 후 생성한 유저로 로그인

```
sqlplus
```

테이블 생성

Postgre 에서 처럼 ID 컬럼의 데이터 형식에서 SERIAL 은 지원하지 않기에,
값이 들어갈 수 있도록 바꿔만 주었습니다.
마찬가지로 지원 안되는 데이터 형식이 있다면 알맞는 형식으로 맞추어 생성합니다.
Oracle 은 모두 소문자로 표기해도 ""로 감싸지 않으면 대문자로 생성됩니다.

```
CREATE TABLE stables (id Number(20), description VARCHAR(50), todo_status VARCHAR(12), PRIMARY KEY(id));
```

Debezium Setting

Oracle JDBC Sink Connector 설정



Kafka Connector 가 설치, 설정된 서버에서 실행합니다.



명령어는 root 로 실행합니다.

Oracle 용 JDBC 드라이버 jar 파일 다운로드 및 위치 이동

```
wget https://download.oracle.com/otn-pub/otn_software/jdbc/215/ojdbc11.jar  
cp ./ojdbc11.jar /root/kafka_2.12-3.2.0/libs/
```

confluent 용 JDBC 커넥터 플러그인 확장 다운로드 및 위치 이동

(jar 파일이 중복되면 덮어쓰지 않아도 됨)

```
wget https://d1i4a15mxbxib1.cloudfront.net/api/plugins/confluentinc/kafka-connect-jdbc/versions/10.4.1/confluentinc-kafka-connect-jdbc-10.4.1.zip  
  
unzip ./confluentinc-kafka-connect-jdbc-10.4.1.zip  
  
cp ./confluentinc-kafka-connect-jdbc-10.4.1/lib/*jar /root/kafka_2.12-3.2.0/libs/
```

Kafka connector 서버 에서 oracle-sink-connector.json 생성

```
{  
  "name": "oracle-sink-connector",  
  "config": {  
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",  
    "tasks.max": "1",  
    "connection.url": "jdbc:oracle:thin:@//10.0.0.7:1521/cdb1",  
    "auto.create": "false",  
    "auto.evolve": "false",  
    "insert.mode": "upsert",
```

```

"table.name.format": "STABLES",
"tombstones.on.delete": "true",
"connection.user": "C#oant",
"connection.password": "QwERasdf123!!",
"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",
"transforms.unwrap.type": "io.debezium.transforms.ExtractNewRecordState",
"transforms.unwrap.drop.tombstones": "false",
"quote.sql.identifier": "always",
"pk.mode": "record_value",
"pk.fields": "ID",
"topics": "antns.public.stables"
}
}

```

`connection.url` : Oracle DB JDBC 접근 URL

`auto.create` : 테이블이 없으면 자동생성 (비활성 권장)

`insert.mode` : upsert 는 update와 insert 모두

`table.name.format` : 데이터 전송 대상 테이블 (대소문자 주의)

`transforms` : 이벤트 허브에 저장된 데이터 중 after 내의 데이터만 전송하게 하는 모듈 (**중요**)

`quote.sql.identifier` : 데이터에 "" 를 씌울지 말지 선택 `always` or `never`

`pk.mode` : upsert 선택시 필수 항목

`pk.fields` : PK가 될 컬럼 선택 (대소문자 주의)

Oracle sink connector 생성

```
curl -X POST -H "Content-Type: application/json" --data @oracle-sink-connector.json http://localhost:8083/connectors
```

확인 및 삭제

```

curl -s http://localhost:8083/connectors/oracle-sink-connector/status

curl -X DELETE http://localhost:8083/connectors/oracle-sink-connector

```

데이터 확인

```
select * from STABLES;
```

오류

1. ORA-12514

"connection.url": "jdbc:oracle:thin:@//10.0.0.7:1521/cdb1"

이 부분의 경로, 형식등이 정확한지 확인

2. ORA-00904

"quote.sql.identifier": "always" 항목 추가,

혹은 이벤트 허브에 저장된 데이터 컬럼 대소문자 확인.

- kafkacat으로 확인
- Oracle 기본은 대문자여야함

3. 테이블 찾지 못함.

"table.name.format":"STABLES" 의 테이블 명 대소문자 확인

4. `UPSERT mode requires key field names to be known, check the primary key configuration`

하기 PK 관련 설정 추가

"pk.mode": "record_value"

"pk.fields": "ID"

최종 데이터 연동 확인

postgresql 데이터 입력

```
INSERT INTO "STABLES" ("DESCRIPTION", "TODO_STATUS") VALUES ('final test', 'complete');
```

Kcat - source connector 동작 확인

```
% Reached end of topic antns.public.stables [0] at offset 5
{"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"default":0,"field":"ID"}, {"type":"string","optional":true,"field":"DESCRIPTION"}, {"type":"string","optional":true,"field":"TODO_STATUS"}], "optional":true, "name":"antns.public.STABLES.Value", "field":"before"}, {"type":"struct","fields":[{"type":"int32","optional":false,"default":0,"field":"ID"}, {"type":"string","optional":true,"field":"DESCRIPTION"}, {"type":"string","optional":true,"field":"TODO_STATUS"}], "optional":true, "name":"antns.public.STABLES.Value", "field":"after"}, {"type":"struct","fields":[{"type":"string","optional":false,"field":"version"}, {"type":"string","optional":false,"field":"connector"}, {"type":"string","optional":false,"field":"name"}, {"type":"int64","optional":false,"field":"ts_ms"}], "type":"string","optional":true, "name":"io.debezium.data.Enum", "version":1, "parameters":{"allowed":["true,last,false,incremental"], "default":"false", "field":"snapshot"}, {"type":"string","optional":false,"field":"db"}, {"type":"string","optional":true,"field":"sequence"}, {"type":"string","optional":false,"field":"schema"}, {"type":"string","optional":false,"field":"table"}, {"type":"int64","optional":true,"field":"txid"}, {"type":"int64","optional":true,"field":"lsn"}, {"type":"int64","optional":true,"field":"xmin"}], "optional":false, "name":"io.debezium.connector.postgresql.Source", "field":"source"}, {"type":"string","optional":false,"field":"op"}, {"type":"int64","optional":true,"field":"ts_ms"}, {"type":"struct","fields":[{"type":"string","optional":false,"field":"id"}, {"type":"int64","optional":false,"field":"total_order"}, {"type":"int64","optional":false,"field":"data_collection_order"}], "optional":true, "field":"transaction"}, {"type":"int64","optional":false,"name":"antns.public.STABLES.Envelope"}, {"payload":{"before":null,"after":{"ID":6,"DESCRIPTION":"final test","TODO_STATUS":"complete"},"source":{"version":"1.9.2.Final","connector":"postgresql","name":"antns","ts_ms":1653288003943,"snapshot":"false","db":"pgdbant","sequence":["8690608544","8724161832"],"schema":"public","table":"STABLES","txid":44614,"lsn":8724161832,"xmin":null},"op":"c","ts_ms":1653288004337,"transaction":null}}
```

Oracle DB table 조회

```
select * from STABLES;
```

Oracle DB - Sink connector 동작 확인

```
SQL> select * from STABLES;

  ID DESCRIPTION                                TODO_STATUS
-----
  1 setup postgresql on azure                    complete
  2 setup kafka connect                         complete
  3 configure and install connector              in-progress
  4 start connector                             pending
  5 my way222                                    hahaant22222
  6 final test                                  complete

6 rows selected.

SQL>
```

Oracle DB에 별도의 작업 없이 6 번이 새로 추가 되었다.

END

Oracle db source connector 세팅



가능한 방법은 Logminer / XStream API 의 두가지 방식이 있으며,
이 시나리오 예서는 Logminer 로 진행합니다.

```
mkdir /u02/oradata/CDB1/pdb1/recovery_area
```

```
sqlplus '/as sysdba'
```

DB복제 활성화 (Oracle instance 재시작 됨)

```
alter system set db_recovery_file_dest_size = 10G;
alter system set db_recovery_file_dest = '/u02/oradata/CDB1/pdb1/recovery_area' scope=spfile;
shutdown immediate
startup mount
alter database archivelog;
alter database open;
-- Should now "Database log mode: Archive Mode"
archive log list
```

복제 활성화 확인

```
show parameters enable_goldengate_replication
```

DB or TABLE 로깅 활성화 - 둘중 하나만 합니다. 테이블의 경우는 잘 고쳐쓸것.

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

위의 설정 확인

```
select supplemental_log_data_min from v$database;
```

테이블의 경우

```
ALTER TABLE inventory.customers ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
```

sys 계정의 비밀번호를 변경한다. (기본값 없음)

```
alter user sys identified by QWERasdf123
```

Logminer 용 테이블 및 관련 사용자 생성 후 권한 부여

(메모장에 복사해서 공백 지우고 로그인 별로 따로 실행)

```
sqlplus sys/QWERasdf123@//localhost:1521/CDB1 as sysdba
CREATE TABLESPACE logminer_tbs DATAFILE '/u02/oradata/CDB1/logminer_tbs.dbf'
```

```

        SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
    exit;

sqlplus sys/QWERasdf123@//localhost:1521/pdb1 as sysdba
    CREATE TABLESPACE logminer_tbs DATAFILE '/u02/oradata/CDB1/pdb1/logminer_tbs.dbf'
        SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
    exit;

sqlplus sys/QWERasdf123@//localhost:1521/CDB1 as sysdba
    CREATE USER c##pant IDENTIFIED BY QWERasdf123
        DEFAULT TABLESPACE logminer_tbs
        QUOTA UNLIMITED ON logminer_tbs
        CONTAINER=ALL;

    GRANT CREATE SESSION TO c##pant CONTAINER=ALL;
    GRANT SET CONTAINER TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$DATABASE TO c##pant CONTAINER=ALL;
    GRANT FLASHBACK ANY TABLE TO c##pant CONTAINER=ALL;
    GRANT SELECT ANY TABLE TO c##pant CONTAINER=ALL;
    GRANT SELECT_CATALOG_ROLE TO c##pant CONTAINER=ALL;
    GRANT EXECUTE_CATALOG_ROLE TO c##pant CONTAINER=ALL;
    GRANT SELECT ANY TRANSACTION TO c##pant CONTAINER=ALL;
    GRANT LOGMINING TO c##pant CONTAINER=ALL;

    GRANT CREATE TABLE TO c##pant CONTAINER=ALL;
    GRANT LOCK ANY TABLE TO c##pant CONTAINER=ALL;
    GRANT CREATE SEQUENCE TO c##pant CONTAINER=ALL;

    GRANT EXECUTE ON DBMS_LOGMNR TO c##pant CONTAINER=ALL;
    GRANT EXECUTE ON DBMS_LOGMNR_D TO c##pant CONTAINER=ALL;

    GRANT SELECT ON V_$LOG TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$LOG_HISTORY TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$LOGMNR_LOGS TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$LOGMNR_CONTENTS TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$LOGMNR_PARAMETERS TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$LOGFILE TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$ARCHIVED_LOG TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$ARCHIVE_DEST_STATUS TO c##pant CONTAINER=ALL;
    GRANT SELECT ON V_$TRANSACTION TO c##pant CONTAINER=ALL;

    exit;

```

생성한 pant 유저로 로그인

```
sqlplus
```

Oracle db table 생성

```
CREATE TABLE OTABLES (id Number(20), description VARCHAR(50), exam_status VARCHAR(12), PRIMARY KEY(id));
```

데이터 입력

```
INSERT INTO "OTABLES" ("ID", "DESCRIPTION", "EXAM_STATUS") VALUES (1,'first test', 'start');
```



다음은 Kafka connector 서버에서 실행합니다.

Oracle-source-connector 속성 작성

```
vi oracle-source-connector.json
```

```
{
    "name": "oracle-source-connector",
```

```

"config": {
  "connector.class" : "io.debezium.connector.oracle.OracleConnector",
  "database.hostname" : "10.0.0.7",
  "database.port" : "1521",
  "database.user" : "c##pant",
  "database.password" : "QWERasdf123",
  "database.dbname" : "CDB1",
  "table.include.list" : "OTABLES",
  "database.server.name" : "server1",
  "tasks.max" : "1",
  "database.pdb.name" : "PDB1",
  "database.history.kafka.topic": "schema-changes.otables",
  "database.history.kafka.bootstrap.servers" : "ehants.servicebus.windows.net:9093",
  "database.history.producer.security.protocol" : "SASL_SSL",
  "database.history.producer.sasl.mechanism" : "PLAIN",
  "database.history.producer.sasl.jaas.config" : "org.apache.kafka.common.security.plain.PlainLoginModule required username=\\"$ConnectionString\\" password=\\"Endpoint=sb://ehants.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=ILZSbJNszyGisX3D7LX0j0b9fM48zTm48Dwaley1jzM=\\";",
  "database.history.consumer.security.protocol" : "SASL_SSL",
  "database.history.consumer.sasl.mechanism" : "PLAIN",
  "database.history.consumer.sasl.jaas.config" : "org.apache.kafka.common.security.plain.PlainLoginModule required username=\\"$ConnectionString\\" password=\\"Endpoint=sb://ehants.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=ILZSbJNszyGisX3D7LX0j0b9fM48zTm48Dwaley1jzM=\\";"
}
}

```

debezium oracle connector 플러그인 / oracle jdbc 드라이버 다운 및 이동

```

wget https://repo1.maven.org/maven2/io/debezium/debezium-connector-oracle/1.9.4.Final/debezium-connector-oracle-1.9.2.Final-plugin.tar.gz
tar -zxvf debezium-connector-oracle-1.9.4.Final-plugin.tar.gz
cp ./debezium-connector-oracle/*.jar /root/kafka_2.12-3.2.0/libs/

```

```

wget https://repo1.maven.org/maven2/com/oracle/database/jdbc/ojdbc8/21.1.0.0/ojdbc8-21.1.0.0.jar
cp ojdbc8-21.1.0.0.jar /root/kafka_2.12-3.2.0/libs/

```

```

curl -X POST -H "Content-Type: application/json" --data @oracle-source-connector.json http://localhost:8083/connectors

```

```

#확인 # todo-connector 는 pg-source-connector.json 에서 임의로 정한 connector 이름
curl -s http://localhost:8083/connectors/oracle-source-connector/status

#원가 오류가 있다면 하기의 명령어로 connector 삭제
curl -X DELETE http://localhost:8083/connectors/oracle-source-connector

```

```

export KCAT_CONFIG=/root/.config/kcat.conf
export BROKER=ehants.servicebus.windows.net:9093
export TOPIC=schema-changes.otables
./kcat -b $BROKER -t $TOPIC -o beginning

```