Streaming SQL Server data changes using Apache Kafka and Debezium installed locally

Debezium is an open-source project, which offers various plugins to fetch the data from a database. It captures row-level changes in your databases so that your applications can see and respond to those changes. Debezium records in a transaction log all row-level changes committed to each database table.

Pre-Requisites:

- Make sure the change data capture is implemented for the respective SQL Server database, please refer to the follow document on how to Enable SQL Server Change data capture
- Make sure Java is installed and JAVA_HOME environment variable is added

Download and Install Kafka (locally):

- Create a new directory called Kafka close to the home dir, for easy organization (*/Users/SubbaReddyAlla/kafka)*.
- And, download the latest Kafka tool from the Apache Kafka page into the newly created directory. In this setup, we are using kafka_2.13-3.6.0.tgz
- Untar the downloaded file: `tar -xzf kafka_2.13-3.6.0.tgz`

Download and Install Debezium plug-in:

- Download the Debezium plug-in for the SQL Server from this page - Debezium Release Series 2.4
  - `wget https://repo1.maven.org/maven2/io/debezium/debezium-connector-sqlserver/2.4.0.Final/debezium-connector-sqlserver-2.4.0.Final-plugin.tar.gz`
  - Untar the downloaded file: `tar -xzf debezium-connector-sqlserver-2.4.0.Final-plugin.tar.gz`
- Under the newly created Kafka directory, create a new directory called plugins like this: /Users/SubbaReddyAlla/kafka/plugins
- Move the untar folder (`debezium-connector-sqlserver`) into this new directory. `mv debezium-connector-sqlserver plugins/`

Add the Kafka Connect Worker and Connector Properties:

Create these files under the Kafka directory (/Users/SubbaReddyAlla/kafka)

Worker.Properties

(use the below code and save the file as worker.properties)

```
offset.storage.file.filename=/tmp/connect.offsets
bootstrap.servers=localhost:9092
offset.flush.interval.ms=10000
#rest.port=10082
#rest.host.name=localhost
#rest.advertised.port=10082
#rest.advertised.host.name=localhost
advertised.port=9092
advertised.host.name=localhost
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
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
plugin.path=/Users/SubbaReddyAlla/kafka/plugins
#If kafka is TLS authenticated, uncomment below lines.
#security.protocol=SSL
```

```
#ssl.truststore.location=/tmp/kafka.client.truststore.jks
#producer.security.protocol=SSL
#producer.ssl.truststore.location=/tmp/kafka.client.truststore.jks
```

Connector.Properties

(use the below code and save the file as connector.properties)

```
name=nbs-cdc-test
connector.class=io.debezium.connector.sqlserver.SqlServerConnector
database.hostname=<**hostname**>
database.port=1433
database.user=<**username**>
database.password=<**password**>
database.dbname=nbs_odse
database.server.name=odse
database.names=nbs_odse
database.history.kafka.topic=nbs-page-test
topic.prefix=test
database.history.kafka.bootstrap.servers=localhost:9092
schema.history.internal.kafka.topic=odse.history
schema.history.internal.kafka.bootstrap.servers=localhost:9092
#table.whitelist=dbo.Person
table.include.list=dbo.NBS_page
database.encrypt=true
database.trustServerCertificate=true
snapshot.lock.timeout.ms=120000
snapshot.mode=schema_only
#If kafka is TLS authenticated, uncomment below lines.
#database.history.producer.security.protocol=SSL
#database.history.producer.ssl.truststore.location=/tmp/kafka.client.truststore.jks
```

Please make sure that we input the appropriate values into certain fields in these properties files

Start up Kafka Server/Broker and Zookeeper

Go to the kafka directory `kafka_2.13-3.6.0` and run the following commands:

- For Zookeeper

```
./bin/zookeeper-server-start.sh config/zookeeper.properties
```

- For Kafka Broker

```
./bin/kafka-server-start.sh config/server.properties
```

Once the Broker and Zookeeper are started successfully without any errors. we can proceed to creating a topic and start to stream the data using Kafka Connect.

Creating Kafka Topic

- Use the following command to create a new topic

```
./bin/kafka-topics.sh --create --topic nbs-page-test --bootstrap-server localhost:9092
```

- Use the following command to check and list topics in your Kafka instance

```
./bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

make sure that you provide your available and non-duplicate topic into the connector properties
`database.history.kafka.topic=nbs-page-test`

Starting Kafka Streaming

```
./bin/connect-standalone.sh worker.properties connector.properties
```

Once this command is run without any errors, we should start to see the data streaming from the database and tables that are connected using the connector properties file. And, you can see the topics created for each table in the list and you would notice the JSON files for changes made to the table.

Topic validation

For topic validation, i was using a GUI tool called **Kadeck** (we shall be using the free version), once this tool is installed. You could add the connection under **Manage Connection** for `localhost:9092`

Under the **Data Browser** window, you could see all the topics that were created in your Kafka instance. once you click on the topic for your table, you would see the JSON files for all the changes that were made.

example of different types of JSON file:

new Inserted row:


```
{"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int64","optional":false,"field":"nbs_pag
```


deleted row:


```
{"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int64","optional":false,"field":"nbs_pag
```


updated row:


```
{"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int64","optional":false,"field":"nbs_pag
```


Refer to this page to understand the Change Event Values in the JSON file, look for the section: 5.3.4.2. Change Event Values