* **Generic steps for Kafka connect:** <https://docs.google.com/document/d/1vfUKEg-uQxmMcOTRRp9bm2288AEBXXkxIOFP9Plkt7A/edit>
* **Kafka - JDBC source connector useful links:**
* <https://www.tutorialkart.com/apache-kafka/kafka-connector-mysql-jdbc/>
* <https://www.confluent.io/blog/simplest-useful-kafka-connect-data-pipeline-world-thereabouts-part-1/>
* <https://turkogluc.com/kafka-connect-jdbc-source-connector/>

**Note: please ignore steps 0 - 2 if Connect has the classes and it’s already started. The steps below are described in order to document all necessary steps.**

**Step 0: SSH to the Kafka Connect machine and stop Kafka Connect if running**

* ps aux | grep connect-distributed.sh
* Kill the process
* curl localhost:8083 // to confirm its not running

**Step 1: get the MySQL class in the JDBC connector directory (this needs to be done only once)**

* **The JDBC connector (to be downloaded from here :** [**https://d1i4a15mxbxib1.cloudfront.net/api/plugins/confluentinc/kafka-connect-jdbc/versions/10.2.0/confluentinc-kafka-connect-jdbc-10.2.0.zip**](https://d1i4a15mxbxib1.cloudfront.net/api/plugins/confluentinc/kafka-connect-jdbc/versions/10.2.0/confluentinc-kafka-connect-jdbc-10.2.0.zip) **)** comes with some **classes, including ones for Postgres and sqllite options - but by default not with the MySQL one**. Other extensions of the connector for MySQL, Oracle, SQL Server, DB2, and Teradata - need to have the jars downloaded in the JDBC directory, in our case located in /opt/connectors/confluentinc-kafka-connect-jdbc-10.2.0
* **So, download the MySQL specific connector from here:** https://dev.mysql.com/downloads/connector/j/5.1.html and placed the jar file in the /opt/onnectors/confluentinc-kafka-connect-jdbc-10.2.0 (the specific jar has to be placed in the same directory as the default jdbc jar).
* **In order for the Kafka Connect to take this path into consideration when loading the class - meaning when starting Connect - we need to place this specific path in the /opt/kafka/config/connect-distributed.properties file.** Please check this file and you will see on the last line the /opt/connect besides the standard path /opt/kafka/libs where the others connectors jars are located.
* **Note**: it works to just mention /opt/connectors in the connect configuration file - it will recursively see all sub-directories.

**Step 2: Start Kafka Connect Distributed to take into consideration the new added jars**

* sudo /opt/kafka/bin/connect-distributed.sh /opt/kafka/config/connect-distributed.properties &
* Check that is works: curl localhost:8083

**Check that the classes for JDBC are visible: curl localhost:8083/connector-plugins |jq**

**Step 3: create in MySQL a database and a table for your testing**. For example I created database vali and table nume. The table has entries of id int and nume varchar.

* Connect to MySQL - you will find the MySQL details here: <https://docs.google.com/document/d/1Xaucls2_TIODTLW4gRYwagHFfYRvmOjz5dS2BC2oLgM/edit#>
* Connection: From any machine in the cluster:

mysql -h 10.156.0.14 -u student -p

* Passw: OroJulMy21
* Once database and table created we will need to declare the id column as a primary key and an incremental one. We will use the option to see only the new added records in the connector and this will be based on an incremental column - in our case id
* ALTER TABLE nume ADD PRIMARY KEY (id);
* ALTER TABLE nume MODIFY COLUMN id INT auto\_increment;

OR we can create the table with the primary key and autoincrement from beginning:

create table nume(id int AUTO\_INCREMENT PRIMARY KEY, name varchar(255));

* See why here: <https://www.tutorialkart.com/apache-kafka/kafka-connector-mysql-jdbc/>

**Step 4: configure the connector**

* You will find in /tmp/**vali-source-mysql-jdbc-autoincrement.json** an example of the configuration file
* Content below - please copy this in a different file and modify the specifics (see them in red below)

{

"name": "**source-mysql-jdbc-autoincrement**",

"config": {

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

"connection.url": "jdbc:mysql://10.156.0.14:3306/curs?user=student&password=OroJulMy21",

"table.whitelist": "demo",

"mode": "incrementing",

"incrementing.column.name": "id",

"topic.prefix": "mysql-jdbc-"

}

}

**Please note:** I used the incrementing option of the connector - there are also other options like: bulk (every time connects to mysql reads again the whole table) and timestamp (Pull all rows based on a timestamp column. You can also do bulk or incrementing column-based extracts. For more information, see **mode** in <https://docs.confluent.io/kafka-connect-jdbc/current/source-connector/source_config_options.html> )

**Step 5: We have the Kafka Connect started properly and the JDBC classes are seen. Now we need to load the connector based on our configuration file:**

curl -d @"/tmp/source-mysql-jdbc-autoincrement.json" -H "Content-Type: application/json" -X POST http://localhost:8083/connectors

Check that the connector has started:

curl localhost:8083/connectors

**Now, in our table let’s insert some data: make sure to use incrementing id’s (use a select and see which is the last id)**

* **Example: insert into demo(id,name) values(12,"test");**

**Check the created topic and the contents:**

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

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic mysql-jdbc-demo

You can go back to mysql and add some more rows with incrementing ID and you will see the new records in the topic.

**Note: with this connector setting we can only monitor the new inserts in the demo table -based on the incrementing id column. If we update a record (update demo set name="update" where id=2;) this will not be visible in the Kafka topic.**

**To stop a connector:**

curl -X DELETE localhost:8083/connectors/name-connector

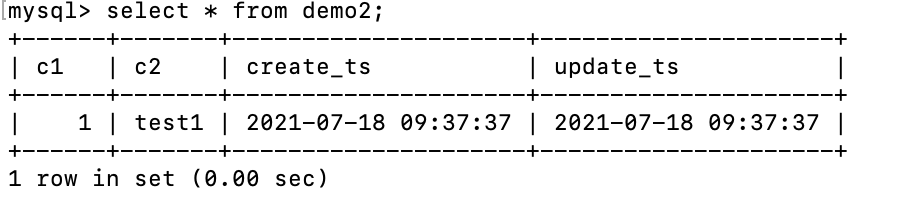
**-------stop here--------------------**

**Test - MySQL connector with timestamp update (on update\_ts column - everytime this column changes it means it’s a new record)**

create table time(c1 int, c2 varchar(255),create\_ts timestamp DEFAULT CURRENT\_TIMESTAMP , update\_ts timestamp DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP );

Insert into demo2(c1,c2) values (1,’test1’);

Output will be:



**Config file (/tmp/source-mysql-jdbc-timestamp.json -- sursa)**

{

"name": "source-mysql-jdbc-timestamp",

"config": {

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

"connection.url": "jdbc:mysql://10.156.0.14:3306/curs?user=student&password=OroJulMy21",

"table.whitelist": "foobar",

"mode": "timestamp",

"timestamp.column.name": "update\_ts",

"validate.non.null":"false",

"topic.prefix": "mysql-jdbc-"

}

}

curl -d @"/tmp/source-mysql-jdbc-timestamp.json" -H "Content-Type: application/json" -X POST http://localhost:8083/connectors

curl localhost:8083/connectors

curl localhost:8083/connectors/vali-source-mysql-jdbc-timestamp |jq

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

/opt/kafka/bin/kafka-console-consumer.sh --topic mysql-jdbc-demo2 --bootstrap-server localhost:9092 --from-beginning

**Class exercise: Create a connector that reads from our mysql topic and saves the data in a file**