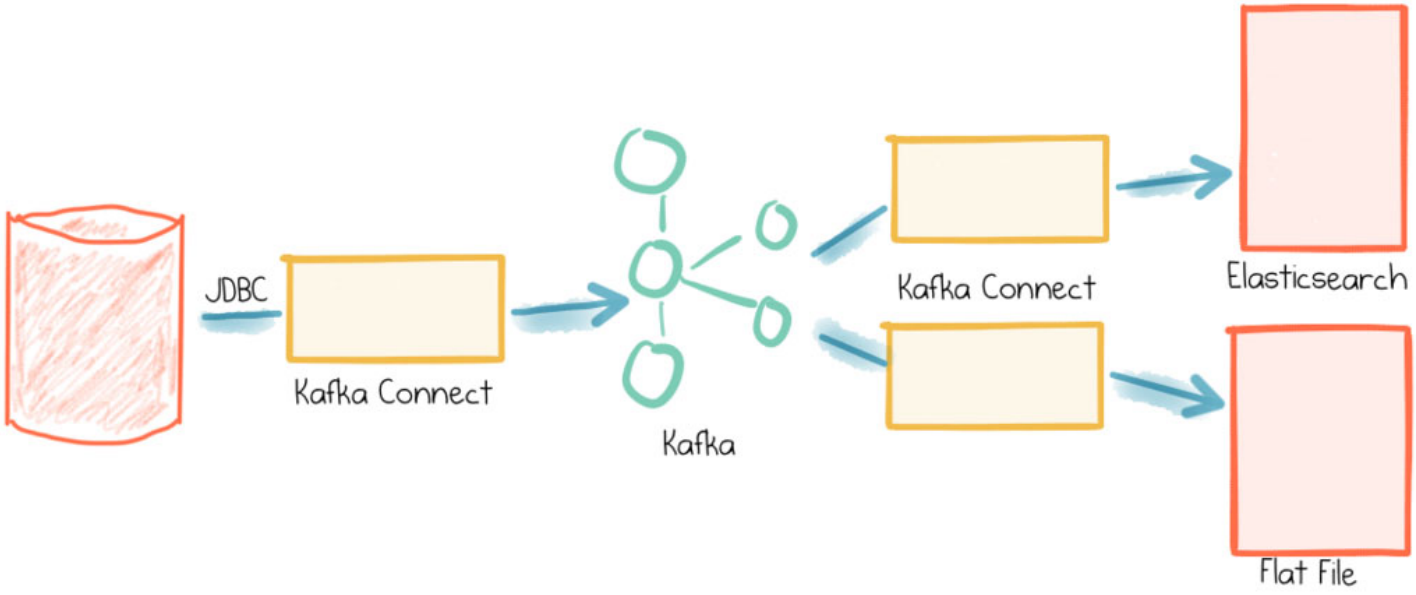
Kafka Connect Data Pipeline from Mysql to Kafka Topic

This short series of articles is going to show you how to stream data from a database (MySQL) into Apache Kafka® and from Kafka into both a text file and Elasticsearch—all with the Kafka Connect API.

Why? To demonstrate how easy it is to integrate data from sources into targets, with no coding needed!



1. Pull data using JDBC Kafka Connect connector, based on a timestamp column in the source table to identify new and modified rows
2. Stream data to an Elasticsearch index
3. Also stream data to a flat file—just because we can!

Getting Started

Prerequisites

```
sudo apt update

sudo apt install wget unzip jq -y
```

```
sudo apt install openjdk-8-jdk-headless -y

sudo update-alternatives --config java

// choose the option number that points to jdk 1.8

java -version
```

```
openjdk version "1.8.0_265"
OpenJDK Runtime Environment (build 1.8.0_265-8u265-b01-0ubuntu2~18.04-b0
1)
OpenJDK 64-Bit Server VM (build 25.265-b01, mixed mode)
```

MySQL

```
sudo apt install mysql-server -y
```

Run the security script

```
sudo mysql_secure_installation
```

Check MySQL status

```
sudo systemctl status mysql.service
```

```
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor pr
eset: enabled)
   Active: active (running) since Sat 2018-11-03 16:19:30 UTC; 38s ago
 Main PID: 11729 (mysqld)
   CGroup: /system.slice/mysql.service
           └─11729 /usr/sbin/mysqld

Nov 03 16:19:29 mysql-kafka-00 systemd[1]: Starting MySQL Community Serv
er...
Nov 03 16:19:30 mysql-kafka-00 systemd[1]: Started MySQL Community Serve
r.
```

```
sudo mysqladmin -p -u root version
```

```
Enter password:
mysqladmin  Ver 8.42 Distrib 5.7.24, for Linux on x86_64
```

```
Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reser
ved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version          5.7.24-0ubuntu0.16.04.1
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 52 sec

Threads: 1  Questions: 5  Slow queries: 0  Opens: 115  Flush tables: 1
Open tables: 34  Queries per second avg: 0.096
```

Confluent Platform

```
wget https://www.dropbox.com/s/4i5kxmv74cnx11s/confluent-oss-5.0.0-2.11.
zip
unzip confluent-oss-5.0.0-2.11.zip
cd confluent-5.0.0/
```

To use the JDBC connector, you'll need to make available the relevant JDBC driver for your source database. The connector ships with drivers for PostgreSQL and sqlite. For all others download the appropriate JAR and place it in `share/java/kafka-connect-jdbc`.

```
wget https://www.dropbox.com/s/x0k7ghvyh00q2cc/mysql-connector-java-8.0.
13.jar \
  -P ~/confluent-5.0.0/share/java/kafka-connect-jdbc
```

Start Confluent Platform

```
./bin/confluent start
```

```
This CLI is intended for development only, not for production
https://docs.confluent.io/current/cli/index.html

Using CONFLUENT_CURRENT: /tmp/confluent.2HR5iWZ3
Starting zookeeper
zookeeper is [UP]
```

```
Starting kafka
kafka is [UP]
Starting schema-registry
schema-registry is [UP]
Starting kafka-rest
kafka-rest is [UP]
Starting connect
connect is [UP]
Starting ksql-server
ksql-server is [UP]
```

Create a Database Table and Some Data

```
mysql -u root -p
```

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.24-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reser
ved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stat
ement.

mysql>
```

```
GRANT ALL PRIVILEGES ON *.* TO 'admatic'@'%' IDENTIFIED BY 'admatic123';
FLUSH PRIVILEGES;

CREATE DATABASE admatic_demo;
USE admatic_demo;

CREATE TABLE foobar (c1 int, c2 varchar(255),
  create_ts timestamp DEFAULT CURRENT_TIMESTAMP,
  update_ts timestamp DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMEST
```

```
AMP);

INSERT INTO foobar (c1, c2) values(1, 'foo');

SELECT * FROM foobar;
```

```
+------+------+---------------------+---------------------+
| c1   | c2   | create_ts           | update_ts           |
+------+------+---------------------+---------------------+
|    1 | foo  | 2018-11-03 16:25:58 | 2018-11-03 16:25:58 |
+------+------+---------------------+---------------------+
1 row in set (0.00 sec)
```

Create Kafka Connect Source JDBC Connector

Confluent Open Source ships with a JDBC source (and sink) connector for Kafka Connect.

To configure the connector, first write the config to a file(for example, `~/kafka-connect-jdbc-source.json`). Here I've added some verbose comments to it, explaining what each item does. These comments are purely for annotation, and are ignored by Kafka Connect:

```
cd ~
cat << EOF > kafka-connect-jdbc-source.json
{
  "name": "jdbc_source_mysql_foobar_01",
  "config": {
    "_comment":
      "The JDBC connector class. Don't change this if you want to use th
e JDBC Source.",
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",

    "_comment":
      "How to serialise the value of keys - here use the Confluent Avro
serialiser. Note that the JDBC Source Connector always returns null for
the key ",
    "key.converter": "io.confluent.connect.avro.AvroConverter",

    "_comment":
      "Since we're using Avro serialisation, we need to specify the Conf
```

```
luent schema registry at which the created schema is to be stored. NB Sc
hema Registry and Avro serialiser are both part of Confluent Open Source
.",
    "key.converter.schema.registry.url": "http://localhost:8081",

    "_comment":
      "As above, but for the value of the message. Note that these key/v
alue serialisation settings can be set globally for Connect and thus omi
tted for individual connector configs to make them shorter and clearer",
    "value.converter": "io.confluent.connect.avro.AvroConverter",
    "value.converter.schema.registry.url": "http://localhost:8081",

    "_comment": " --- JDBC-specific configuration below here  --- ",
    "_comment":
      "JDBC connection URL. This will vary by RDBMS. Consult your manufa
cturer's handbook for more information",
    "connection.url":
      "jdbc:mysql://localhost:3306/admatic_demo?user=admatic&password=ad
matic123",

    "_comment": "Which table(s) to include",
    "table.whitelist": "foobar",

    "_comment":
      "Pull all rows based on an timestamp column. You can also do bulk
or incrementing column-based extracts. For more information, see http://
docs.confluent.io/current/connect/connect-jdbc/docs/source_config_option
s.html#mode",
    "mode": "timestamp",

    "_comment": "Which column has the timestamp value to use?  ",
    "timestamp.column.name": "update_ts",

    "_comment":
      "If the column is not defined as NOT NULL, tell the connector to i
gnore this  ",
    "validate.non.null": "false",

    "_comment":
      "The Kafka topic will be made up of this prefix, plus the table na
me  ",
    "topic.prefix": "mysql-"
  }
}
EOF
```

Load the connector config into Connect

```
cd ~/confluent-5.0.0/
./bin/confluent load jdbc_source_mysql_foobar_01 -d ~/kafka-connect-jdbc
-source.json
```

```
This CLI is intended for development only, not for production
https://docs.confluent.io/current/cli/index.html

{
  "name": "jdbc_source_mysql_foobar_01",
  "config": {
    "_comment": "The Kafka topic will be made up of this prefix, plus th
e table name  ",
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "key.converter": "io.confluent.connect.avro.AvroConverter",
    "key.converter.schema.registry.url": "http://localhost:8081",
    "value.converter": "io.confluent.connect.avro.AvroConverter",
    "value.converter.schema.registry.url": "http://localhost:8081",
    "connection.url": "jdbc:mysql://localhost:3306/admatic_demo?user=adm
atic&password=admatic123",
    "table.whitelist": "foobar",
    "mode": "timestamp",
    "timestamp.column.name": "update_ts",
    "validate.non.null": "false",
    "topic.prefix": "mysql-",
    "name": "jdbc_source_mysql_foobar_01"
  },
  "tasks": [],
  "type": null
}
```

Check its status:

```
./bin/confluent status jdbc_source_mysql_foobar_01
```

```
This CLI is intended for development only, not for production
https://docs.confluent.io/current/cli/index.html

{
  "name": "jdbc_source_mysql_foobar_01",
  "connector": {
    "state": "RUNNING",
    "worker_id": "10.142.0.2:8083"
```

```
  },
  "tasks": [
    {
      "state": "RUNNING",
      "id": 0,
      "worker_id": "10.142.0.2:8083"
    }
  ],
  "type": "source"
}
```

Test that the Source Connector is Working

We'll use the avro console consumer to check the topic:

```
./bin/kafka-avro-console-consumer \
--bootstrap-server localhost:9092 \
--property schema.registry.url=http://localhost:8081 \
--property print.key=true \
--from-beginning \
--topic mysql-foobar
```

You should see almost instantly the row of data that we inserted above, now on the Kafka topic:

```
null    {"c1":{"int":1},"c2":{"string":"foo"},"create_ts":1541262358000,
"update_ts":1541262358000}
```

Return to your MySQL session and insert/update some data:

```
mysql --user=admatic --password=admatic123 admatic_demo
```

```
mysql: [Warning] Using a password on the command line interface can be i
nsecure.
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
```

```
Server version: 5.7.24-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reser
ved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stat
ement.

mysql>
```

```
insert into foobar (c1,c2) values(2,'foo');
insert into foobar (c1,c2) values(3,'foo');

update foobar set c2='bar' where c1=1;

select * from foobar;
```

```
+------+------+---------------------+---------------------+
| c1   | c2   | create_ts           | update_ts           |
+------+------+---------------------+---------------------+
|    1 | bar  | 2018-11-03 16:25:58 | 2018-11-03 16:43:55 |
|    2 | foo  | 2018-11-03 16:43:45 | 2018-11-03 16:43:45 |
|    3 | foo  | 2018-11-03 16:43:51 | 2018-11-03 16:43:51 |
+------+------+---------------------+---------------------+
3 rows in set (0.00 sec)
```

In your console consumer session you should see original first row, plus the additional data appearing in the topic – both the two new rows (c1=2, c1=3), as well as the updated row (c1=1).

```
null    {"c1":{"int":2},"c2":{"string":"foo"},"create_ts":1541263425000,
"update_ts":1541263425000}
null    {"c1":{"int":3},"c2":{"string":"foo"},"create_ts":1541263431000,
"update_ts":1541263431000}
null    {"c1":{"int":1},"c2":{"string":"bar"},"create_ts":1541262358000,
"update_ts":1541263435000}
```

The `update_ts` column is managed automagically by MySQL (other RDBMS have similar functionality), and Kafka Connect's JDBC connector is using this to pick out new and updated rows from the database.

As a side note here, Kafka Connect tracks the offset of the data that its read using the `connect-offsets` topic. Even if you delete and recreate the connector, if the connector has the same name it will retain the same offsets previously stored. So if you want to start from scratch, you'll want to change the connector name – for example, use an incrementing suffix for each test version you work with. You can actually check the content of the `connect-offsets` topic easily:

```
cd ~/confluent-5.0.0/
./bin/kafka-console-consumer \
--bootstrap-server localhost:9092 \
--from-beginning \
--property print.key=true \
--topic connect-offsets
```

```
["jdbc_source_mysql_foobar_01",{"protocol":"1","table":"admatic_demo.foo
bar"}]  {"timestamp_nanos":0,"timestamp":1541263435000}
```