**How to add MySQL Connector to Kafka:**

In this example, I will explain both ways to add MySQL connector to docker compose as well as for local db. Follow these steps to achieve it:

1. First, we reproduce the issue. When will it occur?

Run these docker-compose. It will have all the docker images provided for the Kafka:

<https://raw.githubusercontent.com/confluentinc/cp-all-in-one/7.3.0-post/cp-all-in-one/docker-compose.yml>

1. Now, create DockerFile for the project and up the docker-compose:



Make sure mention the final name under pom.xml as we are defining here as “spring-kafka”

Graphical user interface, text, application, email

Description automatically generated

1. Once the docker-compose file up, you can access the control center on port 9021. Inside the connect, try to upload connector config file that we have created:

Graphical user interface, application, website

Description automatically generated

You will get this error, because there is no JDBC connector here and we are trying to upload MySQL config here so that’s why we are getting this issue.

This is the JSON file that we are trying to upload:  
<https://raw.githubusercontent.com/Maninder416/apache-kafka/feature/silicon-valley-project/source/docker.json>

In this file, we are mentioned the type of data we are sending through this connector and DB details from where we get the data and mentioning the topic where we are sending the data.

Text, letter

Description automatically generated

Follow these steps to fix this issue:

1. We need **MySQL jar file as well as JDBC source connector plugin**. JDBC source connector plugin is common for all the DBs and in this example we are creating MySQL connector, so we need MySQL Jar file too.

How to install JDBC source connector plugin:

We need to run this command to add JDBC source plugin. On your local machine, you can directly run this command and install the plugin but if you are using the docker container in this side you must add the volumes otherwise when the container will down it will install the plugin because when you run the container again it will run on different IP:

2 ways to do it: One is already mentioned like using adding volume and other way is to add the step while running the docker-compose.

**confluent-hub install confluentinc/kafka-connect-jdbc:10.6.3**

we create one docker file and in this we mention that we need this image and, on this image, we also need to install the this plugin as well as also need jar file for MySQL:

Make sure copy the MySQL jar file into some location as you must provide the path for it. In this example, we have copied the Jar file under jars folder:

Inside the docker container, we must copy this jar file under this location:

**/usr/share/confluent-hub-components/confluentinc-kafka-connect-jdbc/lib/**

Graphical user interface, text, application, email

Description automatically generated

Link for this file:

<https://github.com/Maninder416/apache-kafka/blob/feature/silicon-valley-project/Dockerfile-mysql>

After creating this, Inside the docker-compose file:

You must use this image for Kafka connect:

Graphical user interface, application

Description automatically generated with medium confidence

In this we mentioned the image name. Below is the link for working docker compose file:

<https://github.com/Maninder416/apache-kafka/blob/feature/silicon-valley-project/docker-compose.yml>

Now, we are done with the changes. Do below mentioned steps:

1. mvn clean package
2. docker-compose build
3. docker-compose up

now you can see the “**JdbcSource and Sink connector**” here. It comes after adding the plugin that we copied above:

upload the connector config again to check whether it works or not:

After adding the config file, sometimes issues come like this:

Graphical user interface, text, application, email

Description automatically generated

Steps to see the logs:

In new terminal, run the connect like this:

Docker-compose up connect

Now restart the connector you have added and observe the logs:

Text

Description automatically generated

It is clearly seen that we don’t have tables on which we are trying to run this query.

After creating the table, it worked fine:

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text

Description automatically generated