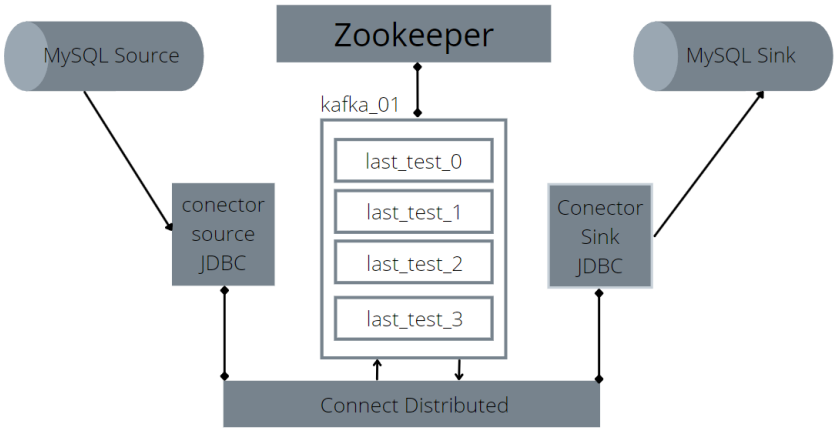
PAD

Data Pipeline con Kafka Connect - PADSA

Este ejemplo hace uso de conectores para obtener datos de nuevos registros de una base de datos en MySQL y escribirlos en otra BD en MySQL haciendo uso de kafka Connect con el conector preconstruido JDBC

Diagrama del Pipeline de Datos



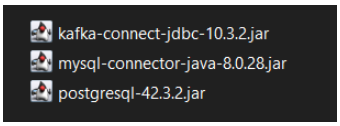
Entorno de trabajo utilizado:

- Windows 10 home
- Kafka 3.1.0
- Postman
- MySQL Workbench 8.0

Desarrollo del Ejemplo

1. Descargar y extraer los Conectores preconstruidos JDBC
   - https://www.confluent.io/hub/confluentinc/kafka-connect-jdbc (Kafka Connect)
   - https://jdbc.postgresql.org/download.html (PostgreSQL)
   - https://dev.mysql.com/downloads/connector/j/ (MySQL)

   1.1 Una vez descargados los conectores de deben colocar en el directorio libs del directorio de kafka.



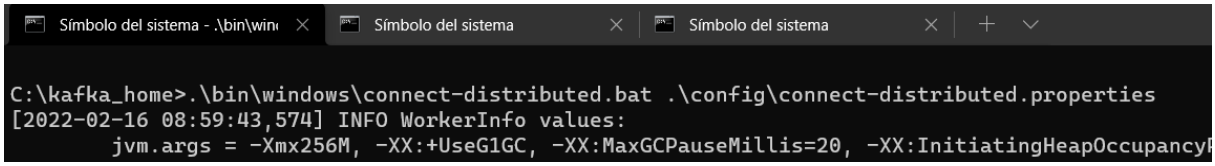
2. Levantar **Zookeeper**, **Kafka server** y **Kafka connect,** con los siguientes comandos correspondientes cada uno desde una ventana de terminal independiente:
   - `.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties`

```
C:\kafka_home>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

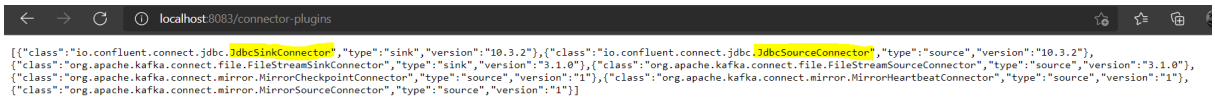
- `.\bin\windows\kafka-server-start.bat .\config\server.properties`

```
C:\kafka_home>.\bin\windows\kafka-server-start.bat .\config\server.properties
```

- `.\bin\windows\connect-distributed.bat .\config\connect-distributed.properties`



3. Comprobar que se agregaron correctamente los conectores Source Connector y Sink Connector usando es siguiente endpoint.
   - http://localhost:8083/connector-plugins



4. Crear el Topic [de nombre *last_test* para este ejemplo] desde terminal con el siguiente comando.
   - `.\bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --create --topic last_test --partitions 4 --replication-factor 1`
5. Listat los Topics para comprobar que se creo correctamente o consultar los detalles del topic
   - `.\bin\windows\kafka-topics --list --bootstrap-server localhost:9092`

```
C:\kafka_home>.\bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092
__consumer_offsets
clientstable_clients
connect-configs
connect-offsets
connect-status
last_test
```

   - `.\bin\windows\kafka-topics.bat --describe --bootstrap-server localhost:9092 --topic last_test`

```
C:\kafka_home>.\bin\windows\kafka-topics.bat --describe --bootstrap-server localhost:9092 --topic last_test
Topic: last_test        PartitionCount: 4       ReplicationFactor: 1    Configs: segment.bytes=1073741824
        Topic: last_test        Partition: 0    Leader: 1       Replicas: 1     Isr: 1
        Topic: last_test        Partition: 1    Leader: 1       Replicas: 1     Isr: 1
        Topic: last_test        Partition: 2    Leader: 1       Replicas: 1     Isr: 1
        Topic: last_test        Partition: 3    Leader: 1       Replicas: 1     Isr: 1
```

6. (Opcional) Crear un consumidor de datos del Topic en consola para monitorear el funcionamiento del Source connector.
   - `.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic last_test --from-beginning`
7. Elegir o crear la base de datos y la tabla de donde tomara los datos el Source Connector

En este ejemplo de utiliza la siguiente configuración:

```
// Crear la base de datos
CREATE DATABASE `kafkaconnect`;

// Crear tabla de prueba
CREATE TABLE `timestamp_test` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
)
```

id	name	created_at	updated_at
1	edgar3	2022-02-15 22:53:43	2022-02-15 23:02:07
2	lucero	2022-02-15 22:54:30	2022-02-15 22:54:30
3	anahi2	2022-02-15 22:59:21	2022-02-15 23:01:45
4	lofe	2022-02-15 23:03:50	2022-02-15 23:03:50
NULL	NULL	NULL	NULL

8. Crear la base de datos y la tabla en donde escribirá los datos el Sink Connector

   En este ejemplo de utiliza la siguiente configuración:

```
// Crear la base de datos
CREATE DATABASE `sinkconnect`;

// Crear tabla de prueba
CREATE TABLE `last_test` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
)
```

8. Crear el conector Source con la API REST de Kakfa Connect con ayuda de un tester de API´s en este ejemplo se realiza con “*Postman”.*

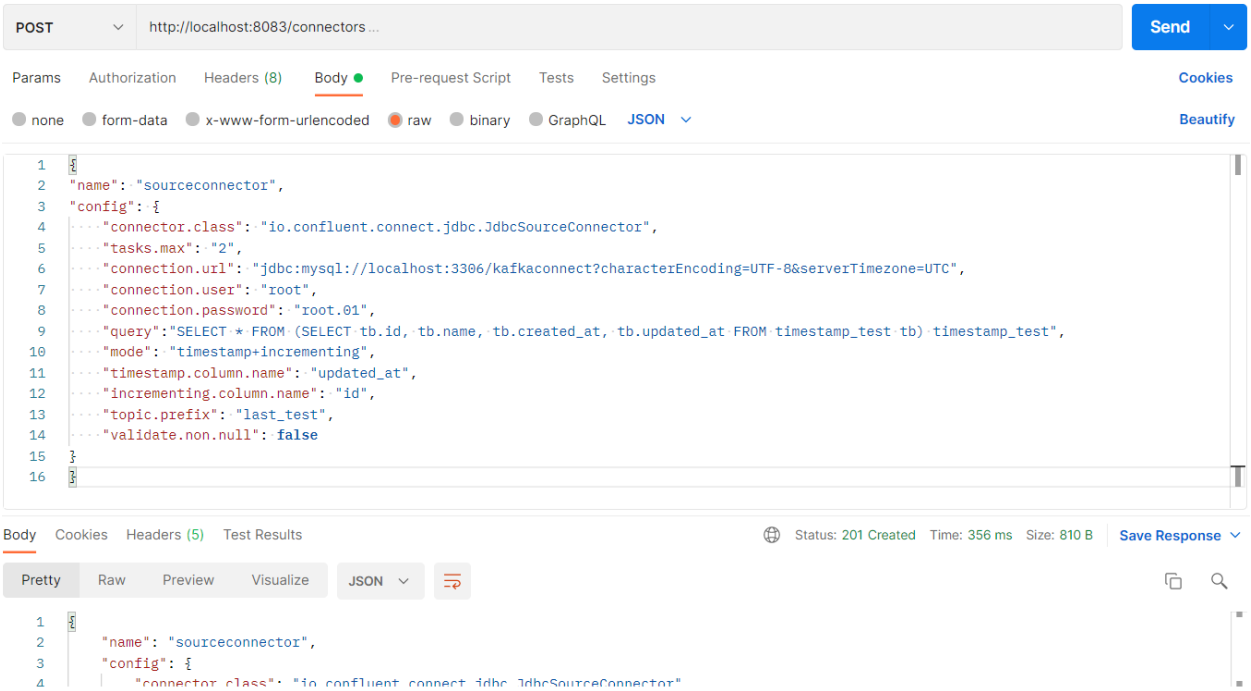
   Para este ejemplo se utiliza el siguiente archivo JSON de configuración:

   - Se conecta a la base de datos en MySQL con la url, user y password
   - La forma de obtención de los datos es mediante un Query
   - El modo `“mode”` utilizado es `"timestamp+incrementing"` que hara registros al detectar cambios en una columna de tipo “tiemstamp” definida en este caso "updated_at" y la columna del valor incremental “id”
   - La configuración "topic.prefix": "last_test" para conectar el topic creado anteriormente.
   - Para mas detalles de configuración ver la documentación oficial del Conector JDBC

```
{
"name": "sourceconnector",
"config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "tasks.max": "2",
    "connection.url": "jdbc:mysql://localhost:3306/kafkaconnect?characterEncoding=UTF-8&serverTimezone=UTC",
    "connection.user": "root",
    "connection.password": "root.01",
    "query":"SELECT * FROM (SELECT tb.id, tb.name, tb.created_at, tb.updated_at FROM timestamp_test tb) timestamp_test",
    "mode": "timestamp+incrementing",
    "timestamp.column.name": "updated_at",
    "incrementing.column.name": "id",
    "topic.prefix": "last_test",
    "validate.non.null": false
```

```
  }
}
```

Visto desde postman al realizar el POST un Status 201 de respuesta indica que el conector se creo correctamente.

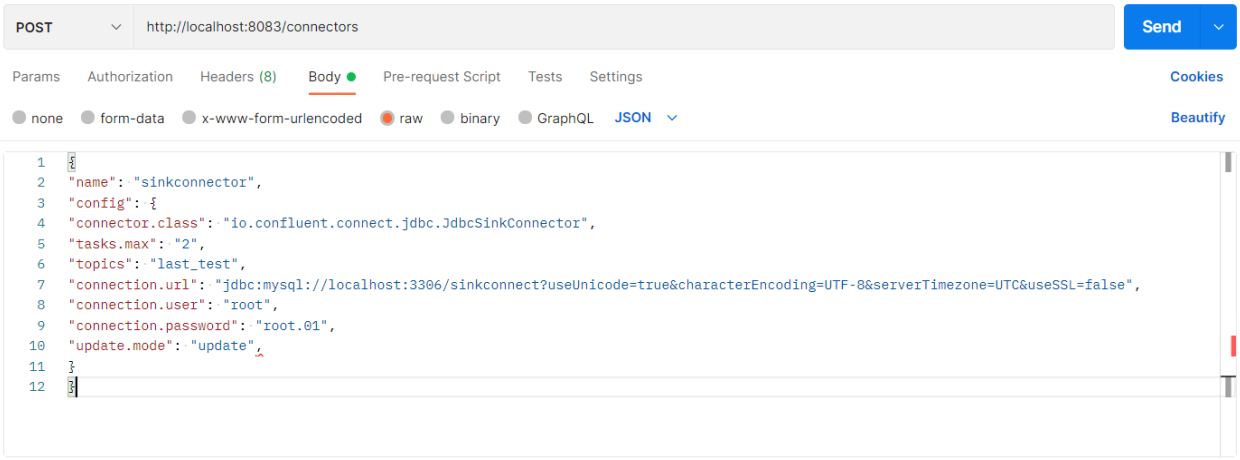


10. Crear el conector Sink con la API REST de Kakfa Connect con ayuda de un tester de API´s en este ejemplo se realiza con *“Postman”.*

    Para este ejemplo se utiliza el siguiente archivo JSON de configuración:

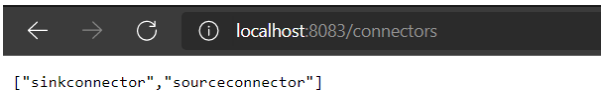
    - Se conecta al topic “last_test” que además será el nombre la tabla en la que se escribíran los datos.
    - Se conecta a la base de datos de destino en MySQL con la url, user y password
    - El modo de update es update.
    - Para mas detalles de configuración ver la documentación oficial del Conector JDBC

```
{
"name": "sinkconnector",
"config": {
  "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
  "tasks.max": "2",
  "topics": "last_test",
  "connection.url": "jdbc:mysql://localhost:3306/sinkconnect?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC&useSSL=false",
  "connection.user": "root",
  "connection.password": "root.01",
  "update.mode": "update",
  }
}
```

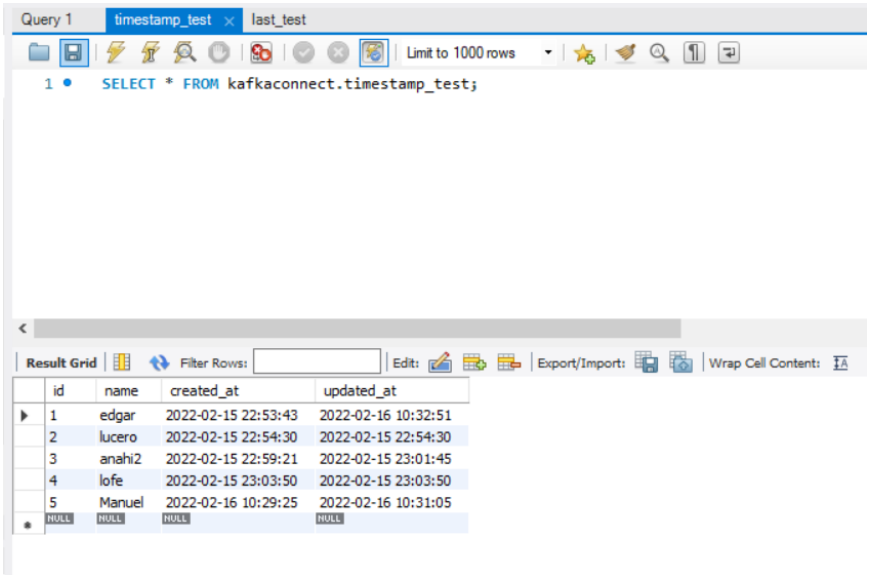


11. Consultar los connectores creados con el siguiente endpoint

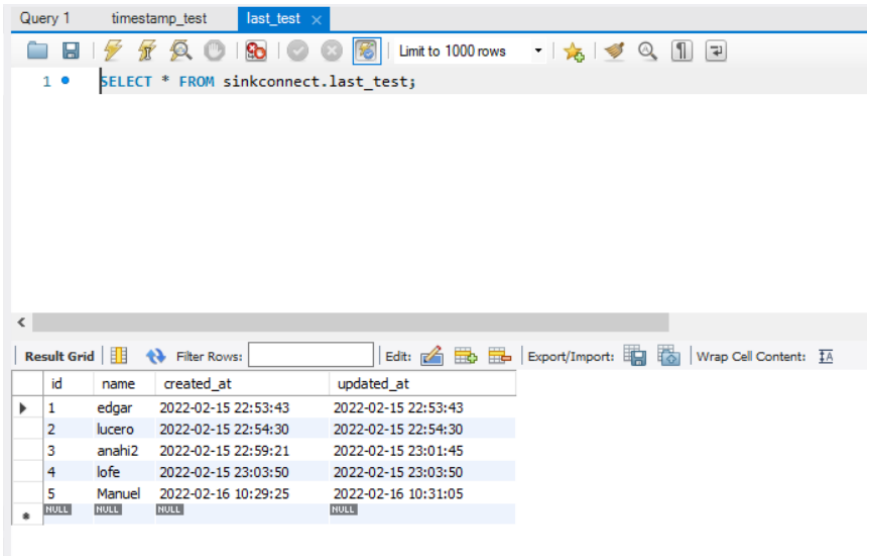
    - http://localhost:8083/connectors



12. A este punto el pipeline de datos debe estar corriendo correctamente y cualquier nuevo registro hecho en la tabla productora “tiemstamp_test” se escribirá automáticamente en la tabla consumidora “last_test”.



> Recuerda dar click en el botón de actualizar para ver los cambios.


Query 1
timestamp_test
last_test
Limit to 1000 rows
1 SELECT * FROM sinkconnect.last_test;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
id name created_at updated_at
1 edgar 2022-02-15 22:53:43 2022-02-15 22:53:43
2 lucero 2022-02-15 22:54:30 2022-02-15 22:54:30
3 anahi2 2022-02-15 22:59:21 2022-02-15 23:01:45
4 lofe 2022-02-15 23:03:50 2022-02-15 23:03:50
5 Manuel 2022-02-16 10:29:25 2022-02-16 10:31:05
NULL NULL NULL NULL