Reference links

|  |  |
| --- | --- |
| Connectors | <https://docs.confluent.io/cloud/current/connectors/index.html>  oracle sink connector <https://docs.confluent.io/cloud/current/connectors/cc-oracle-db-sink.html#cc-oracle-db-sink> |
|  | <https://www.confluent.io/blog/fully-managed-connectors-make-apache-kafka-easier> |
| Sink connectors | <https://docs.confluent.io/cloud/current/connectors/cc-mongo-db-sink/cc-mongo-db-sink.html#mongodb-atlas-sink-connector-for-ccloud> |
| All examples | <https://github.com/confluentinc/demo-scene> |
| Download JDBC connector from here | <https://www.confluent.io/hub/confluentinc/kafka-connect-jdbc> |
| Ref code for all connectors | <https://github.com/confluentinc/demo-scene/tree/master/connect-jdbc> |
| Monitor kafka connector | <https://docs.confluent.io/platform/current/connect/monitoring.html> |
| To add , delete, update a connector | <https://docs.confluent.io/platform/current/connect/references/restapi.html> |
|  |  |

Kafka Connect

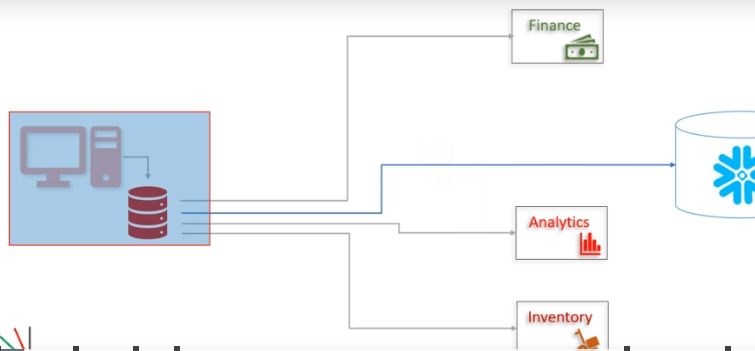
Lets say if u want to pump the data between 2 databases then use kafka connect

Or if u want to pump the data from 1 source app to many other appns then use kafka so that they will subscribed to same data stream

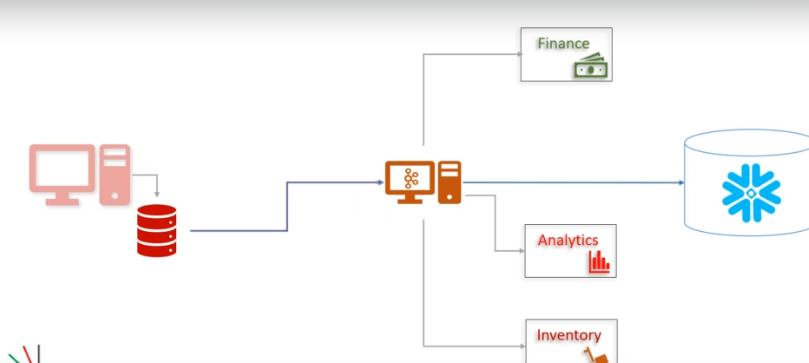
Most of the connectors instead of just dumping the data, u can even transform the data as well

Without kafka

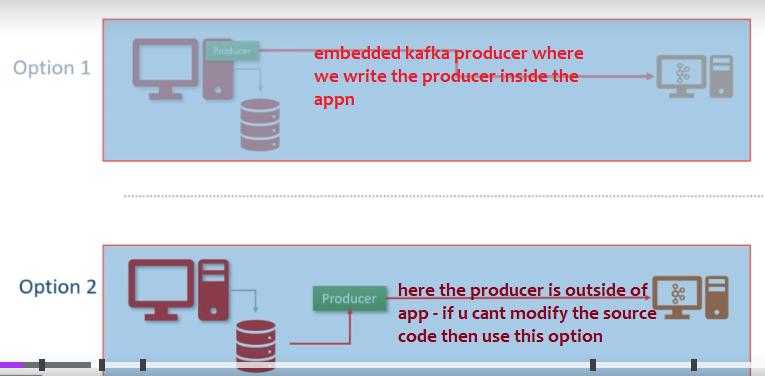
Then load will be there on the source appn to send data to 4 consumers



With kafka

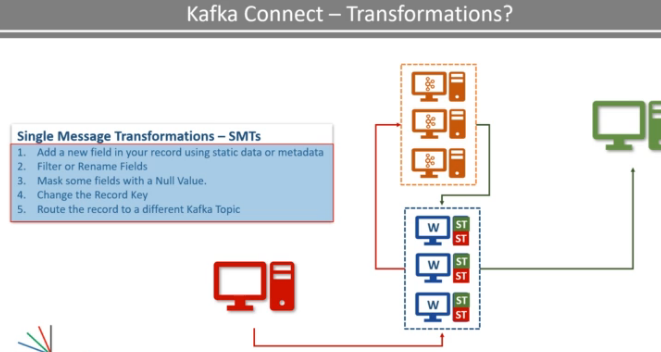


If u have source code then open the appn then write embedded kafka producer API in that appn same like yom-Event processor appln where we wrote the kafka producer which will send the data to kafka cluster, if u don’t have the source code? Then use kafka source connector api which resides outside of appn and it’s a separate application called source connector



Message transformation in Kafka connect api

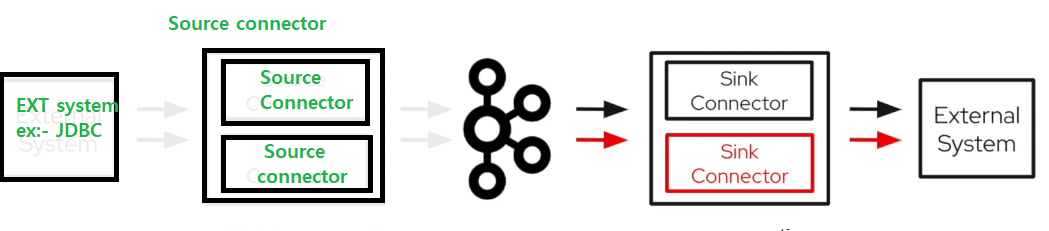
Not just reading from tables and putting into Kafka cluster we can do some message transformations



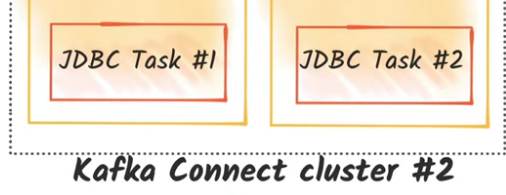
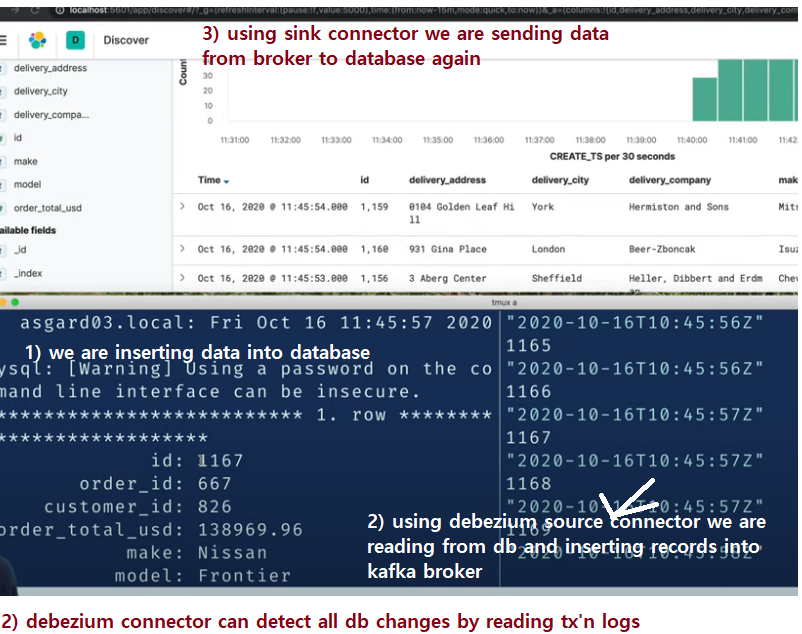
Kafka connect

Kafka connect is a cluster uses one or more workers these workers are fault tolerant and self managed

It is a component of kafka for connecting and moving data between kafka and external system



|  |  |
| --- | --- |
| Push data to cluster (From **database 🡪 kafka broker**) | Data From kafka **broker -> database** |
| Use **source connector** (to read data from database and insert data into broker)  Source connector will internally use Kafka producer api, because it will read from db and put to cluster | Use **sink connector** to take data from kafka broker & insert into database  Sink connector will internally use consumer api to consume data from broker |



We can create a Kafka connect cluster with the workers

We should run kafka connect as a jvm process which should run continosuly (same like how kafka broker runs continuouslythis also should run continuously)

LETS Assume kafka cluster runs with 3 workers, same like all consumers in a consumer group shares the work, similarly all workers in a worker group shares the work

since cluster is self-managed, If a worker crashes the other workers will automatically shares the load by taking that died workers work, similarly, when a new worker joins the group then automatically the existing workers re assigns the tasks same like when new team member join the team , other team members will automatically gives some work to the newly joined team member.

Kafka source connector

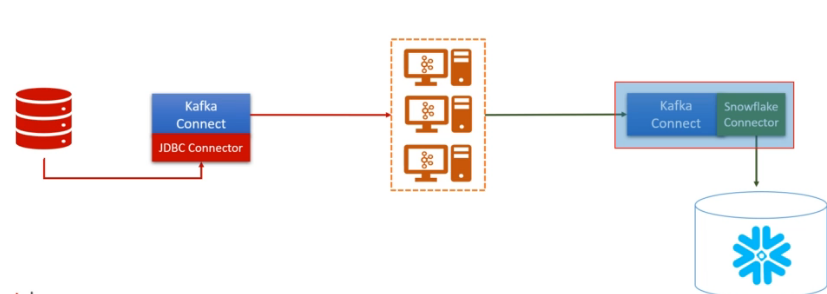
Source connector means – this connector acts as a source to Kafka broker (database to cluster), these are to get the data from common data sources like data base and write data to cluster

Top of Form

To import data from external databases, I should use-kafka source connector

Ex:- oracle JDBC source connector, this connector can read data from oracle db and pushes all the data to kafka cluster

Here jdbc connector is the source connector which will pull data from jdbc database and push the data to Kafka cluster



Kafka sink connector

Top of Form

To continuously export data from Kafka into a target database, I should use

**Kafka Sink Connector**

Sink connector is where it can read from cluster and put the data to database, it will internally use the consumer api

Bottom of Form

These are to publish the data into common data stores

It is a part of ETL pipeline- extract transform load

Ex:- read the data from kafka and put the data into a sink, the actual person who does that work is called worker thread

We should run the Kafka connector then only it will start, once we started Kafka connect, worker will take data from broker and he will insert into database

Features of connectors

Confluent’s fully managed connectors also come with built-in productivity features like

[single message transforms (SMTs)](https://docs.confluent.io/cloud/current/connectors/single-message-transforms.html?_ga=2.124480650.1247261532.1721975960-667539689.1719662201&_gac=1.81978852.1721977404.CjwKCAjw74e1BhBnEiwAbqOAjNNZX730ILXFu21FH3JiNm_XTcfg3ip8tPZBJvNIW397nLl43JXMjBoCXT8QAvD_BwE&_gl=1*ofcq2c*_gcl_aw*R0NMLjE3MjE5Nzc0MDQuQ2p3S0NBanc3NGUxQmhCbkVpd0FicU9Bak5OWlg3MzBJTFhGdTIxRkgzSmlObV9YVGNmZzNpcDh0UFpCSnZOSVczOTduTGw0M0pYTWpCb0NYVDhRQXZEX0J3RQ..*_gcl_au*OTEzMjAxOTU1LjE3MTk2NjIyMDE.*_ga*NjY3NTM5Njg5LjE3MTk2NjIyMDE.*_ga_D2D3EGKSGD*MTcyMTk3NTk1OS4xMi4xLjE3MjE5Nzc1MjIuMzcuMC4w), (we can even write our own custom transformations)

[exposed connect logs](https://docs.confluent.io/cloud/current/connectors/logging-cloud-connectors.html#view-events-in-the-ccloud-console), and

[data preview](https://docs.confluent.io/cloud/current/connectors/data-preview.html).

SMTs enable you to perform lightweight data transformations like **masking** and **filtering in flight** within the connector itself and log events provide contextual information to simplify debugging and troubleshooting. Data preview, uniquely available with Confluent Cloud connectors, lets you test a source connector’s output prior to launching the connector. This helps with iterative testing so that you can confidently launch connectors into production.

Scaling

You can happily scale kafka connect by adding more workers

Like how we can scale producers to have more producers to put data into single topic

And scaling consumers like we have more consumers in same consumer group to read data from multiple partitions similarly

You can happily scale kafka connect, you can have 4 source connector workers pulling data from multiple tables and you can have 4 sink tasks where each can push to diff tables

As per above image we can have multiple kafka connect workers in

we can have all of them in same cluster too

If u are using jdbc connector , we have to analyse the parallelism means number of parallel tasks. Lets say we have to read data from 5 tables , then parallelism should be 5, then connector will start 5 parallel tasks it may internally use 5 workers

Connector will not copy the data, it will start 5 parallel tasks worker is the one who is actually do the work, here tasks list will be given to the workers

### Task vs worker

Now the task is responsible for connecting to the source system, polling the data at a regular interval,

collecting the records, and handing over it to the worker. Yes, that's correct.

They do not send the record to the Kafka Cluster.

That task is only responsible for interacting with the external system.

This source task will handover the data to the worker, and the worker is responsible for sending it to

the Kafka.

In the case of the Sink task, they get the Kafka record from the worker, and the task is only responsible for inserting the record into the target system.

Why is it designed like this?

Reading and writing data to a Kafka Cluster is a standard activity.

So it is taken care of by the framework.

We have two things that are changing for different source and target systems.

How to split the input for parallel processing.

This is taken care of by the Connector class. How to interact with the external system.

This is also taken care of by the Task class. And these are the things that are connector

developer needs to take care of.

Most of the other stuff like interacting with Kafka, handling configurations,

errors, monitoring connectors, and tasks, scaling up and down, and handling failures are standard things

and are taken care of by the Kafka Connect Framework. Makes sense?

Configure it using below properties

Source table list= T1, T2,T3,T4,T5

Maximum parallelism=5

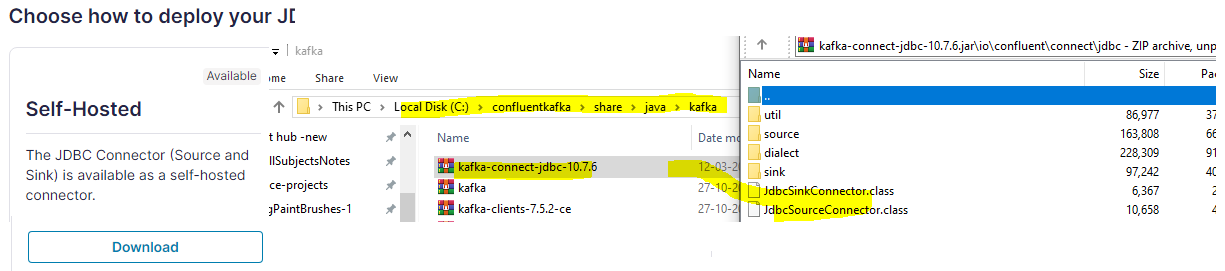
Polling frequency= 5Min

Sample Sink connector config

1. Download JDBC connector from here <https://www.confluent.io/hub/confluentinc/kafka-connect-jdbc>

This will download a jar with full list of connectors for all databases like oracle, MySQL, mongo, we should only keep relative jar file, so here relative jar file is “kafka-connect-jdbc”

We should only keep this particular jar in the below folder, then only the Kafka will recognize



U can read the document from here <https://github.com/confluentinc/demo-scene/tree/master/connect-jdbc>

1. Place that jar in this path (plugin.path=/usr/share/java) This path is mentioned in “C:/confluentkafka/etc/kafka/connect-standalone.properties”
2. After placing that jar then start **kafka-connect server** with following command

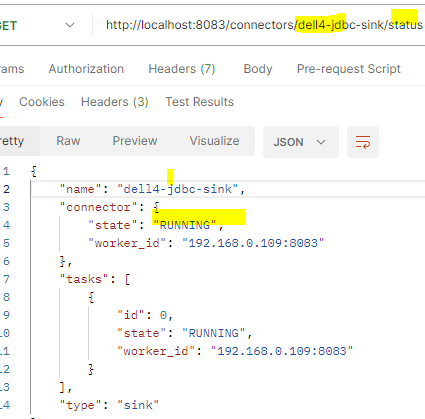
.\bin\windows\connect-standalone.bat .\etc\kafka\connect-standalone.properties

1. Then we have to register our JDBCSource/Sink connector with kafka-connect server

Hit endpoint 8083 is where kafka connect server is running

|  |  |
| --- | --- |
| 1. To see list of installed/available connectors   GET request🡪 <http://localhost:8083/connector-plugins>   1. List Currently available connectors on a kafka connect server:   GET request <http://localhost:8083/connectors>   1. To Stop a connector (PUT request)   <http://localhost:8083/connectors/my-jdbc-sink/stop>  Note:- after stopping/ after any action u should see the status of the connector to check whether it is in running or which state   1. To see the status of a connector (GET)   <http://localhost:8083/connectors/my-jdbc-sink/status>  http://localhost:8083/connectors/<your connector name>/status   1. To pause a connector (PUT request)   http://localhost:8083/connectors/my-jdbc-sink/pause | To register a connector with below details  From postman fire a POST request with below json data [**http://localhost:8083/connectors**](http://localhost:8083/connectors)  {      "name": "my-jdbc-sink",      "config": {          "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",          "connection.url": "jdbc:mysql://localhost:3306/sprbatch",          "connection.user": "mani",          "connection.password": "mani",          "table.name": "student",          "topics": "accenture",          "key.converter": "org.apache.kafka.connect.json.JsonConverter",          "value.converter": "org.apache.kafka.connect.json.JsonConverter"      }  } |
|  |  |
| To get all tasks of a connector  localhost:8083/connectors/<your connector name>/tasks  http://localhost:8083/connectors/my-jdbc-sink/tasks | To restart a connector  localhost:8083/connectors/<your connector name>/restart |

To see the status of a connector



Here Once sink connector is started it should consume data from broker and insert into database, currently I am facing some issues here even though connector thread is running

Data from topic is not being pushed to database, better refer book “kafka connect”