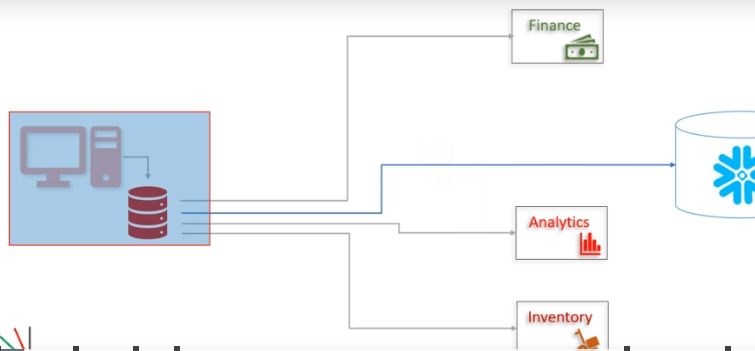
Kafka Connect

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

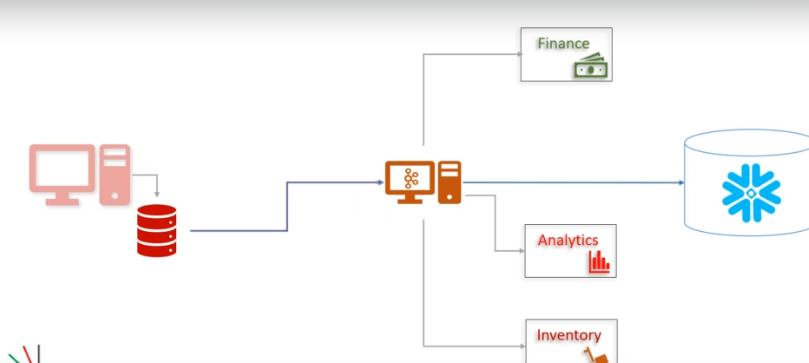
Or if u want to pump the data from 1 source app to many other appns then use kafka so that they will subscribed to same data stream

Without kafka

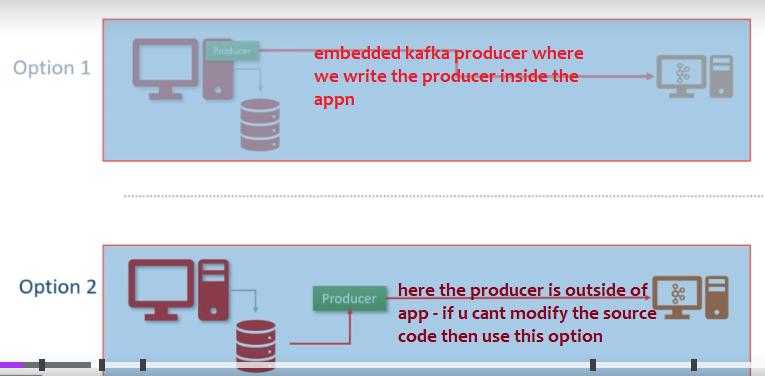
Then load will be there on the source appn to send data to 4 consumers



With kafka

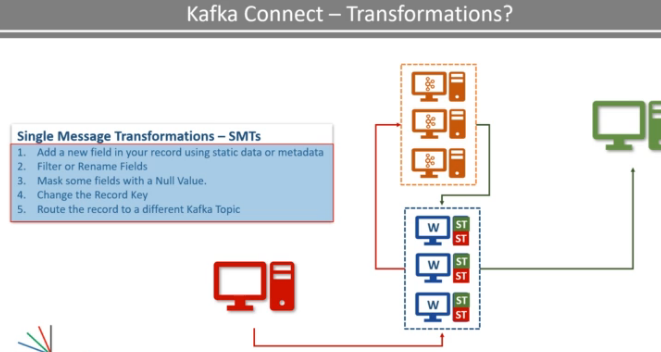


If u have source code then open the appn then write embedded kafka producer API in that appn same like yom-Event processor appln where we wrote the kafka producer which will send the data to kafka cluster, if u don’t have the source code? Then use kafka source connector api which resides outside of appn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



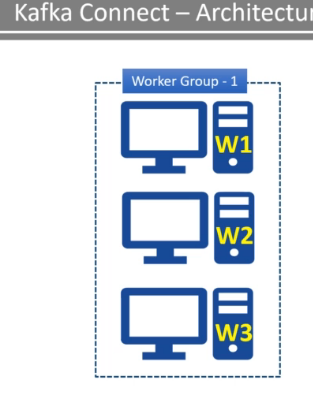
Message transformation in kafka connect api

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



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

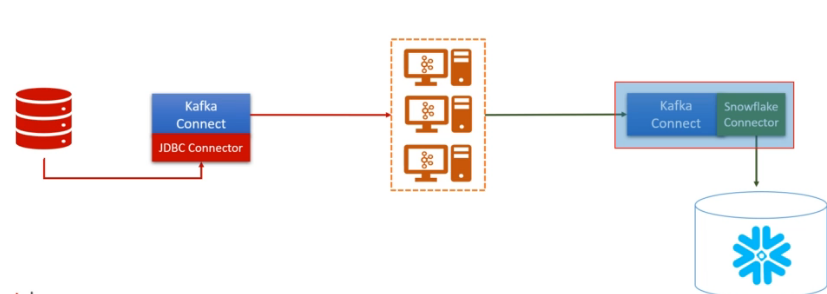
These are to get the data from common data sources

Top of Form

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

Bottom of Form

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**

Bottom of Form

These are to publish the data into common data stores

It is a part of ETL pipeline- extract transform load

It is to put the kafka data into a sink,

Ex:- read the data from kafka and put the data into a sink

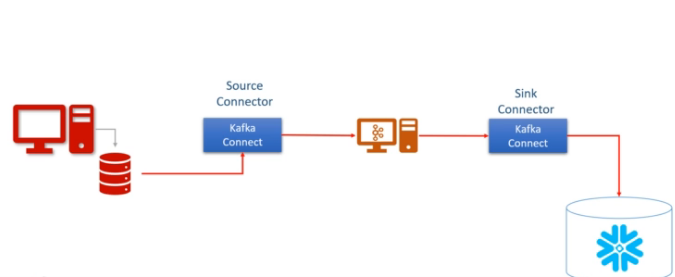
Kafka connect

It is a component of kafka for connecting and moving data between kafka and external system, we don’t need to write kafka producer as we have **source connector** and we don’t need to write kafka consumer as we have **sink connector** api

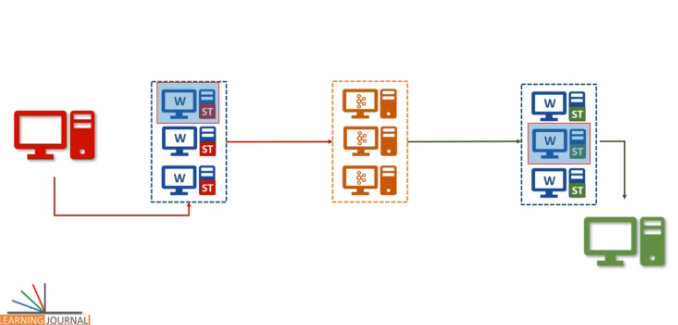
If you have source code then like yom project you can happily open the source code write kafka producer, then this kafka connect is no more required

In case if u don’t have the source code , then u can use kafka connect API which is independent and it can read from database and put the data to cluster

Here source connector is where you can read from database and writes to cluster

Source connector will internally use Kafka producer api, because it will read and put to cluster

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



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