# Building Data Pipeline using the Kafka Connect.

Table of Contents

* Introduction.
* Apache Kafka High level View
* What and Why Kafka Connector?
* Lets Understand Kafka Connector Architecture
* Data Pipelines Management using Connectors!
* Advantages of Kafka Connectors
* Connector Examples and Community
* Building Custom Connectors using Connect API
* Conclusion
* References

## Introduction

From our experiences we all know that its always hard to get the old system retired and often we need to deal with situations where we would need the old and new systems to coexists (if not forever , but at least for quite a considerable time) .

This is one of the scenarios where we would need a data pipeline and by definition, a data pipeline represents the flow of data between two or more systems. In other words we can say data pipeline is a set of instruction that would define how and when the data should be transferred between the systems.

There are many ways one can choose to create the data pipelines and I this article we will look at one way of using the pub-sub (publish/subscribe) queues like Kafka for the transfer of data between multiple systems. Most of them must be aware of the Kafka its producer-consumer / streaming capabilities, but here we will concentrate more on how to connected the data pipes by one of the flexible/scalable/fault-tolerant component of this Kafka Platform called the Kafka Connect.

## Apache Kafka High level View

Before we get into the details of Kafka Connect, Let’s have a quick high level overview of Apache Kafka. Apache Kafka is a distributed streaming platform, with three key capabilities listed below :

* Publish and Subscribe streams of records.
* Store steams of records in a fault tolerant way.
* Process Streams of records as they occur.

The Kafka runs as a cluster and this cluster stores the streams of records in categories called topics. A topic is a category or the feed name to which the records are published. The topics in Kafka is always a Multi-Subscriber and is partitioned where a partition is a ordered, immutable sequence of records that is continually appended to a structured commit log.



As show in the Figure – 1 above the Apache Kafka has the following Core Components.

* **Producers** – Producers Publish data to the topics of their choice. The producer is responsible for choosing which record to assign to which partition within the topic. This can be done in a round-robin fashion simply to balance load or it can be done according to some semantic partition function (say based on some key in the record).
* **Consumers** – Consumers label themselves with a consumer group name, and each record published to a topic is delivered to one consumer instance within each subscribing consumer group. Consumer instances can be in separate processes or on separate machines.
* **Stream Processors** – In Kafka a stream processor is anything that takes continual streams of data from input topics, performs some processing on this input, and produces continual streams of data to output topics.
* Connectors – Kafka connectors give you toolsets that can interconnect data between Apache Kafka ad other data systems scalably and reliably. More on the connectors in the next section.

## What and Why Kafka Connector?