Understanding Kafka Connect

Intern Names	:	Parth.V.Sharma & Nikhil Mehta
Mentor Name	:	Manas Joshi
Manager Name	:	Aniket Khandelwal
Team	:	Platform

Project Overview :

Area of Study:

- What is Kafka Connect
- Provide Customization Features and explore flexibility
- Advantages

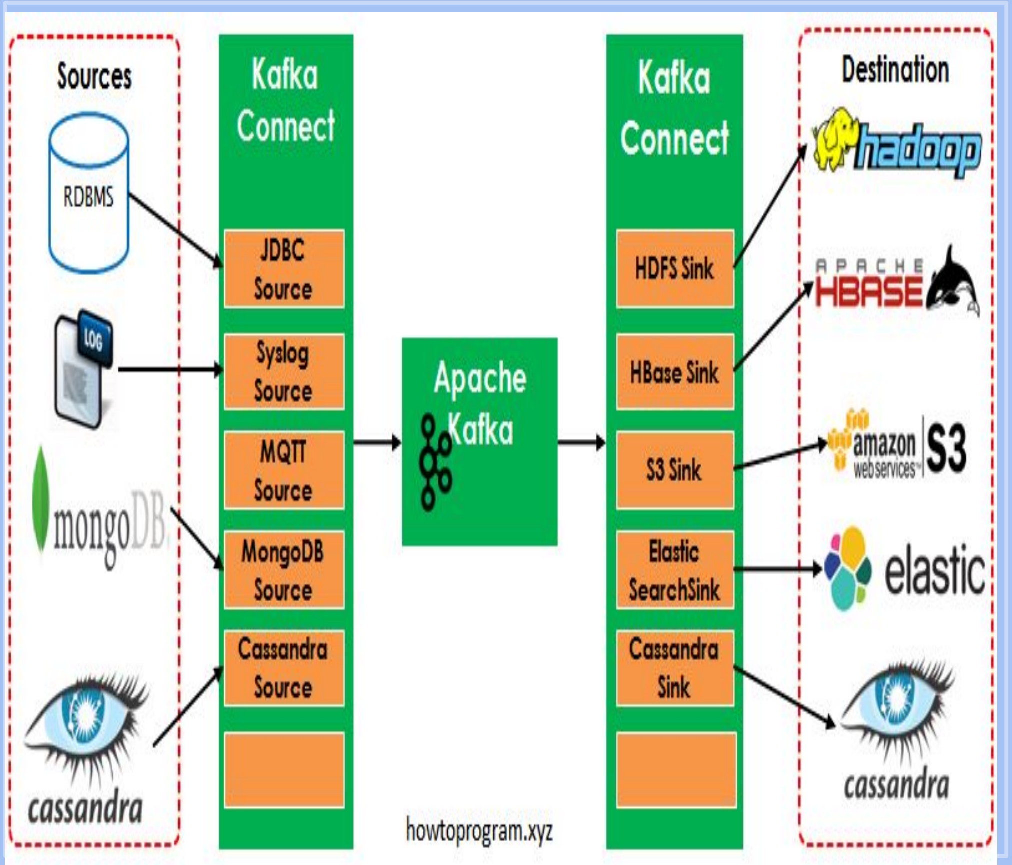
Major Achievements:

➔ Data pipeline to move Data from **MongoDb** to **Redis** and/or **ElasticSearch**

➔ Created mechanisms for Error handling

➔ Provided Capability to modify data flowing in the pipeline

What is Kafka Connect?



- ❏ A tool for scalably and reliably streaming data between Apache Kafka and other data systems
- ❏ Made up of a set of connectors which act as the medium of interaction between kafka and external systems

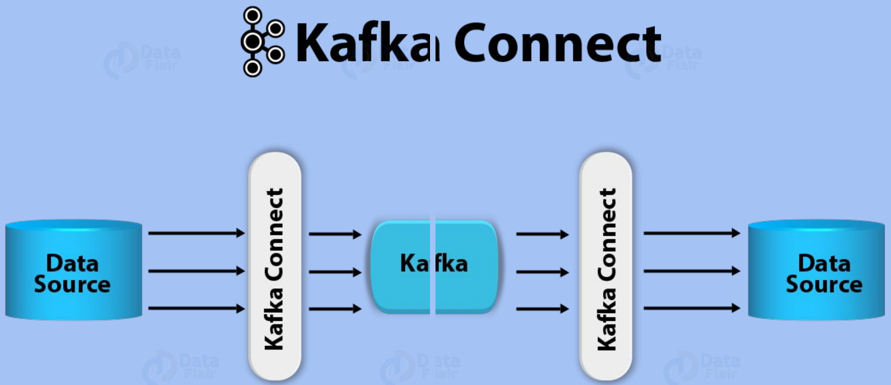
Types of Kafka Connectors:

Source Connector

1. Imports data from any external system into an Apache Kafka topic

Sink Connector

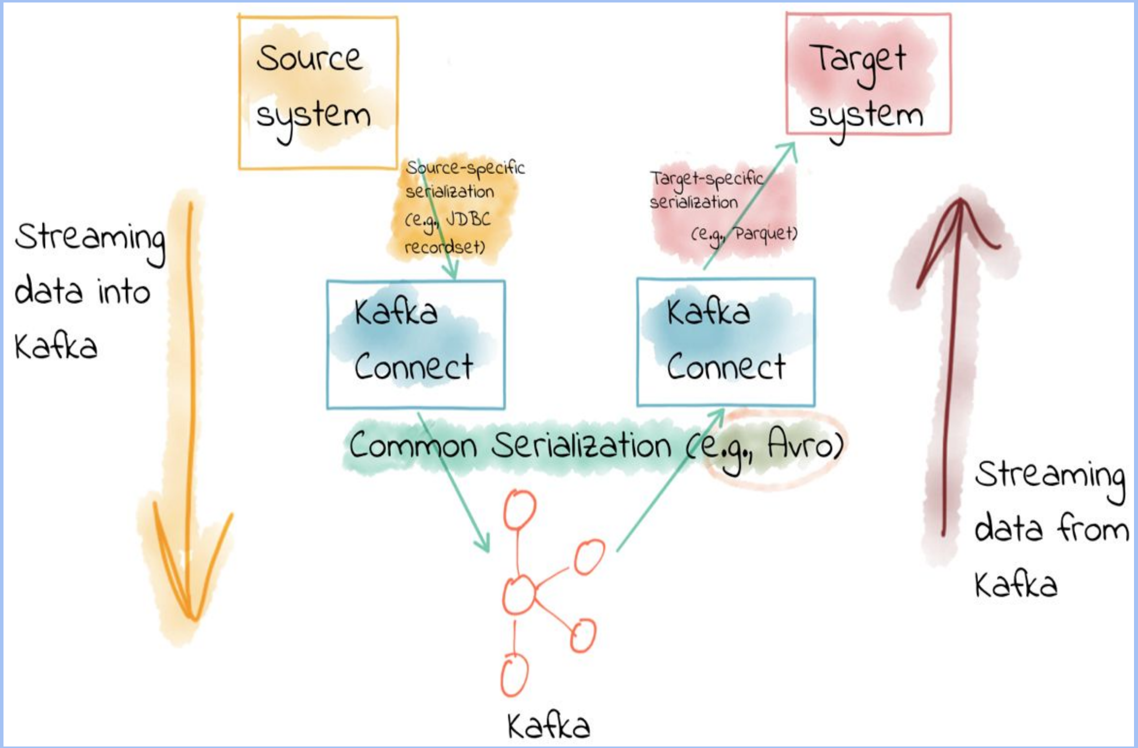
1. Sink connector allows you to export data from Apache Kafka topics to any other system



Important Kafka Connect Features & Concepts

Convertors

Block Diagram

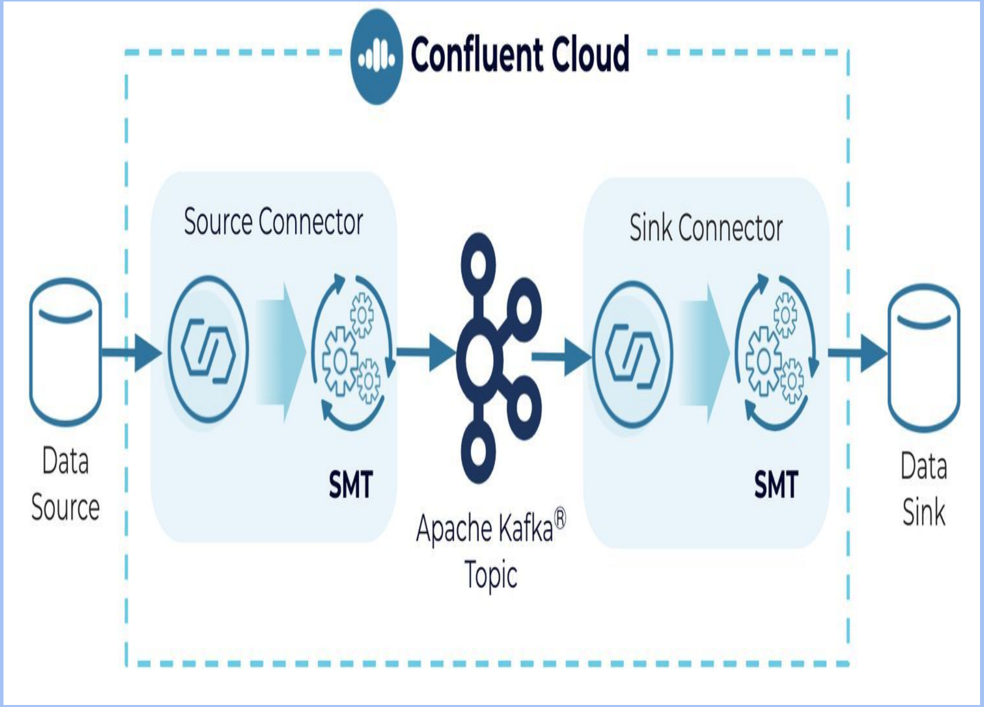

Source system
Target system
Source-specific serialization (e.g., JDBC recordset)
Target-specific serialization (e.g., Parquet)
Streaming data into Kafka
Kafka Connect
Kafka Connect
Common Serialization (e.g., Avro)
Streaming data from Kafka
Kafka

Description:

- Handle serialization and deserialization of data
- Data is in the form of bytes when stored in Kafka
- Standardized serialization format
  - Json
  - Avro
  - String

Single Message Transforms

Block Diagram



Description:

- Provides a way to Manipulate data when in flows in the pipeline
- Transform inbound messages after a source connector has produced them
- SMTs transform outbound messages before they are sent to a sink connector

Single Message Transforms

Some Useful SMTs

- ValuetoKey
    - Replace the record key with a new key formed from a subset of fields in the record value.
- ReplaceField
    - Filter or rename fields.
- Add - Drop Fields
- AlterSchema
    - Change the schema of the key or value of the record
- Custom SMT

Custom Single Message Transforms

Manipulate all parts of the `Record`: the `Key`, the `Value`, the `Key` and `Value schemas`, `destination topic`, `destination partition` , and `timestamp`

```
public class CustomSMT implements Transformation
```

Configure method

Determines the input parameters to be defined in the connector configuration and properties regarding them

- ❑ Default Value
- ❑ Importance
- ❑ Validators

Apply method

Takes a kafka record as input , performs certain operations and returns a modified kafka record to be inserted

```
@Override
public R apply(R record) {}
```

Custom SMT Examples

Customized Dead Letter Queue

Store various exceptions that occur in the pipeline without stopping the connector

- Include more exceptions than the already existing ones
- Different DLQs for different exceptions

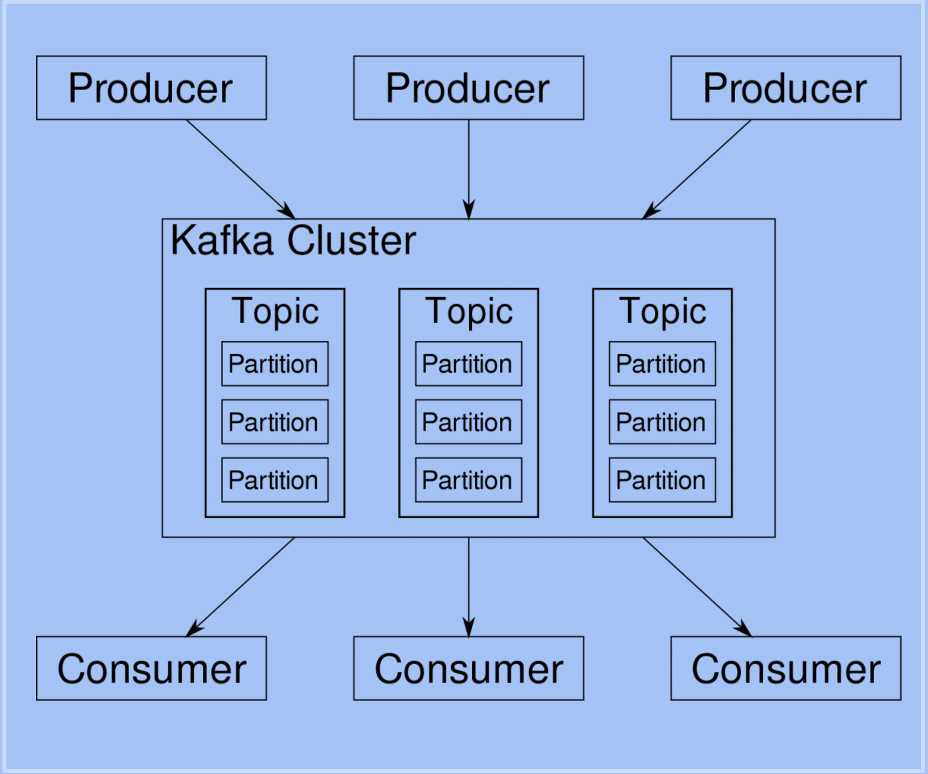
Tracking records

Keep a count of :

I. messages flowing out from source connector
II. Messages flowing into sink connector

Ensure they are equal

Kafka Partitioner & Partition Key



Usage and Explanation

- ❏ Kafka partitioner is responsible for deciding partition number for each message.

- ❏ The default partitioner follows these rules.
  1. If a partition number is explicitly defined straightaway use it
  2. Else if it uses the partition key to choose a partition based on a hash value of this key
     a. eg) hash(key)%num_partition.
  3. If no partition number or key is present, pick a partition in a round-robin fashion.

Creating Custom Partitioner

What?

- ❑ Create our own partitioning logic for the messages flowing from the source connector into a kafka topic

Why?

- ➔ Default partitioner does not cater to some scenarios
- ➔ Data from same producer to go to the same partition
  - ◆ Using Composite key to partition data
  - ◆ The way Hashing works
  - ◆ Number of partitions increase
  - ◆ Reserving some partitions

How?

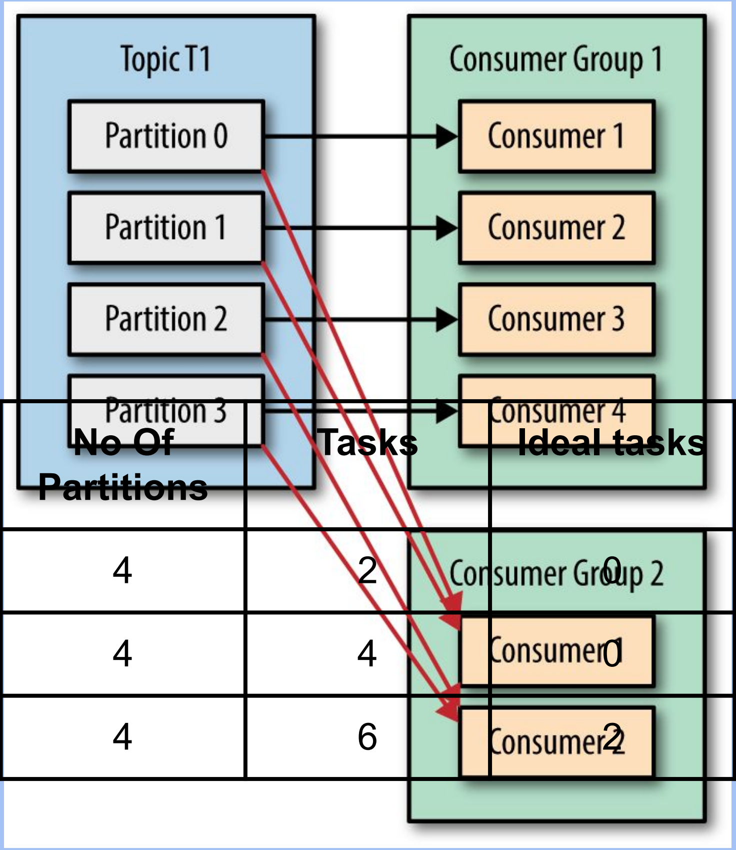
- ➔ Kafka provides an interface called Partitioner
- ➔ custom partitioner class must implement three methods from the interface
  - ◆ `public void configure(Map<String, ?> configs)`
  - ◆ `public int partition(String topic, Object key, byte[] keyBytes, Object value, byte[] valueBytes, Cluster cluster)`
  - ◆ `public void close()`

Any Questions ?

Thank You

Message Ordering

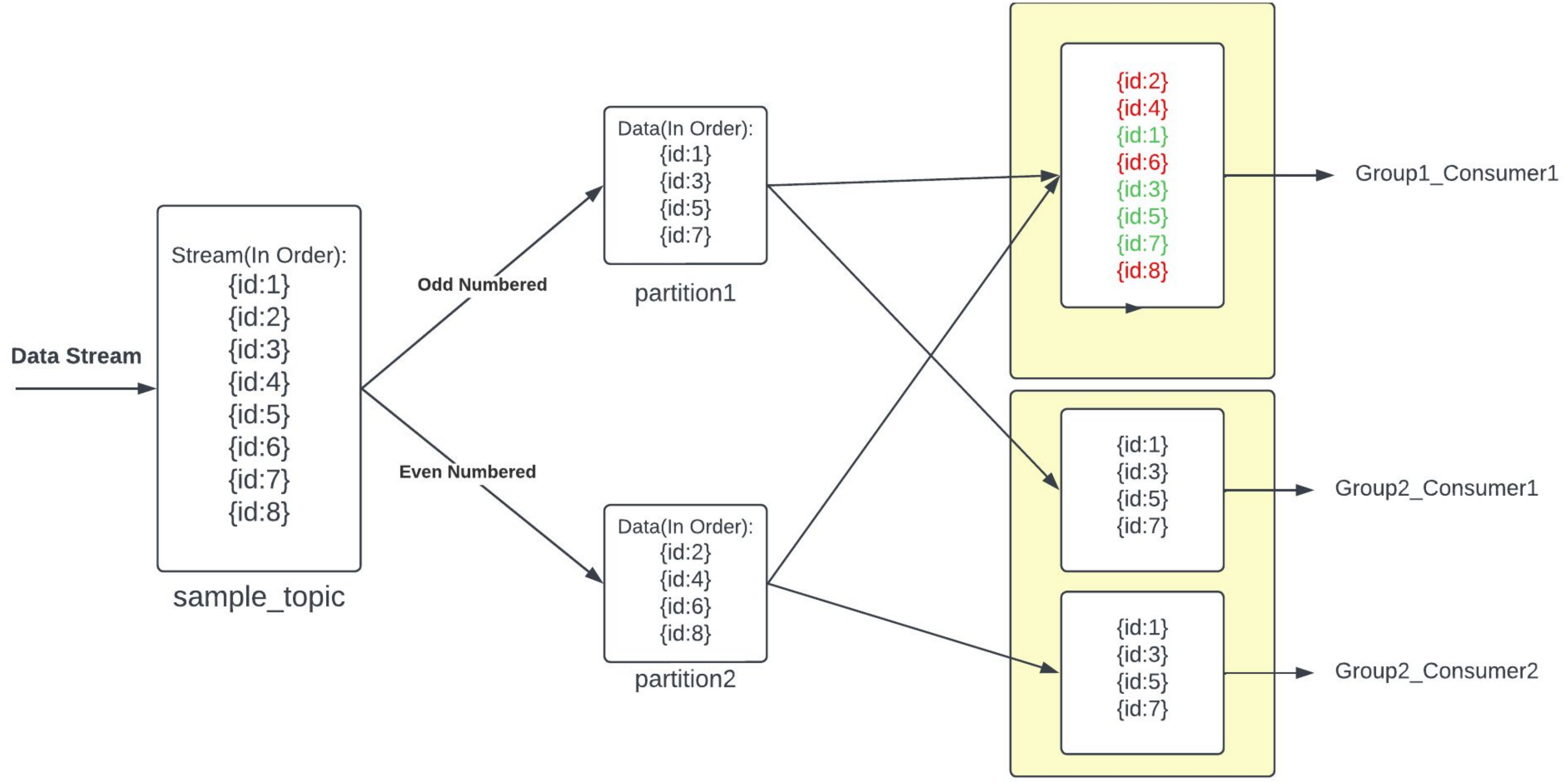
- Messages coming to a particular topic can be divided into partitions through partition key.
- Only one consumer from a group can subscribe to a partition.
- Consumers in different groups can subscribe to same partition.
- A consumer can subscribe to multiple partitions.



No Of Partitions	Tasks	Ideal tasks
4	2	0
4	4	0
4	6	2

Source: Oreilly

Unit of Parallelism : Partition


Data Stream
Stream(In Order):
{id:1}
{id:2}
{id:3}
{id:4}
{id:5}
{id:6}
{id:7}
{id:8}
sample_topic
Odd Numbered
Even Numbered
Data(In Order):
{id:1}
{id:3}
{id:5}
{id:7}
partition1
Data(In Order):
{id:2}
{id:4}
{id:6}
{id:8}
partition2
{id:2}
{id:4}
{id:1}
{id:6}
{id:3}
{id:5}
{id:7}
{id:8}
Group1_Consumer1
{id:1}
{id:3}
{id:5}
{id:7}
Group2_Consumer1
{id:1}
{id:3}
{id:5}
{id:7}
Group2_Consumer2

Error Handling

Error Handling By Kafka Connect

Stage	Handled?
Start	NO
poll(source connector)	NO
convert	YES
transform	YES
put(sink connector)	NO

Error Handling By Kafka Connect

Fail Fast	Ignore	Dead Letter Queue(DLQ)
`tolerance = none (default)`	`tolerance = all`	`tolerance = all`
	`log.enable = true`	`deadletterqueue.topic.name = <name>`
	`log.include.messages = true`	`deadletterqueue.context.headers.enable = true`

DLQ topic can be subscribed by other consumers or if the error can be rectified, can be sent to same producer.

No DLQ In Put Stage?

Sink Database may be down.

- We can set number of retries to a high number with high backoff interval.
- Use custom logic such as retries for particular error and DLQ for others.
- ES Sink Connector uses '_bulk' api and error could be due to one erroneous record.

Error Handling in Put Stage

We can analyse connectors source code and apply custom error handling strategies.

Redis Connector

```
if (null == record.key()) {
      throw new DataException("The key for the
            record cannot be null. " +
            formatLocation(record)
 );
```

ES Connector

```
if (shouldSkipRecord(record)) {
      logTrace("Ignoring {} with null value.",
                  record);
      offsetState.markProcessed();
      reportBadRecord(record, new
      ConnectException("Cannot write null valued
                  record."));
      continue;
}
```

Advantages of Kafka connect

The benefits of Kafka Connect include:

- **Data Centric Pipeline** – Uses meaningful data abstractions
- **Flexibility and Scalability** – Standalone or Distributed
- **Reusability and Extensibility** – Connect leverages existing connectors or extends them to tailor to your needs and provides lower time to production.
- **Schema Registry** – Confluent provides a layer to manage schemas and schema evolution.

Any Questions ?

Thank You

Learnings :

1. Java as a language for development .
2. Java Concurrency .
3. Distributed Systems .
4. Kafka Apache .
5. Implementation of Kafka building basic group chat api .
6. Unstructured Databases Redis , Scylla .
7. Understanding and reading of source codes .
8. Debugging and Fixing Bugs .
9. Jedis Java Api .
10. Methods of benchmarking .
11. Writing clean and extendible codes .
12. Using various tools on Git.
13. Getting familiar with jar files , Config files etc.
14. Using various helpful tools , docker , homebrew , throttle etc .