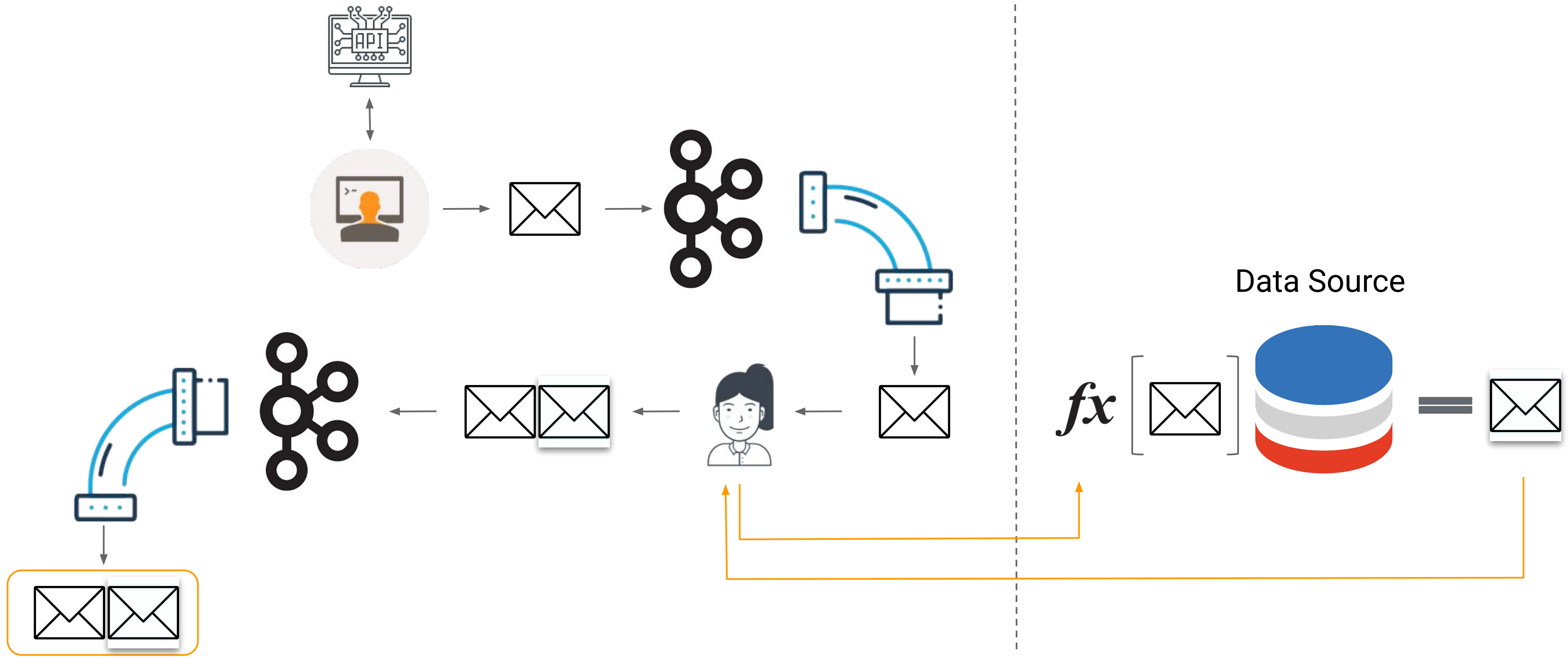
Apache Kafka Data Enrichment

Using kafka-connect and kafka-streams

Introduction



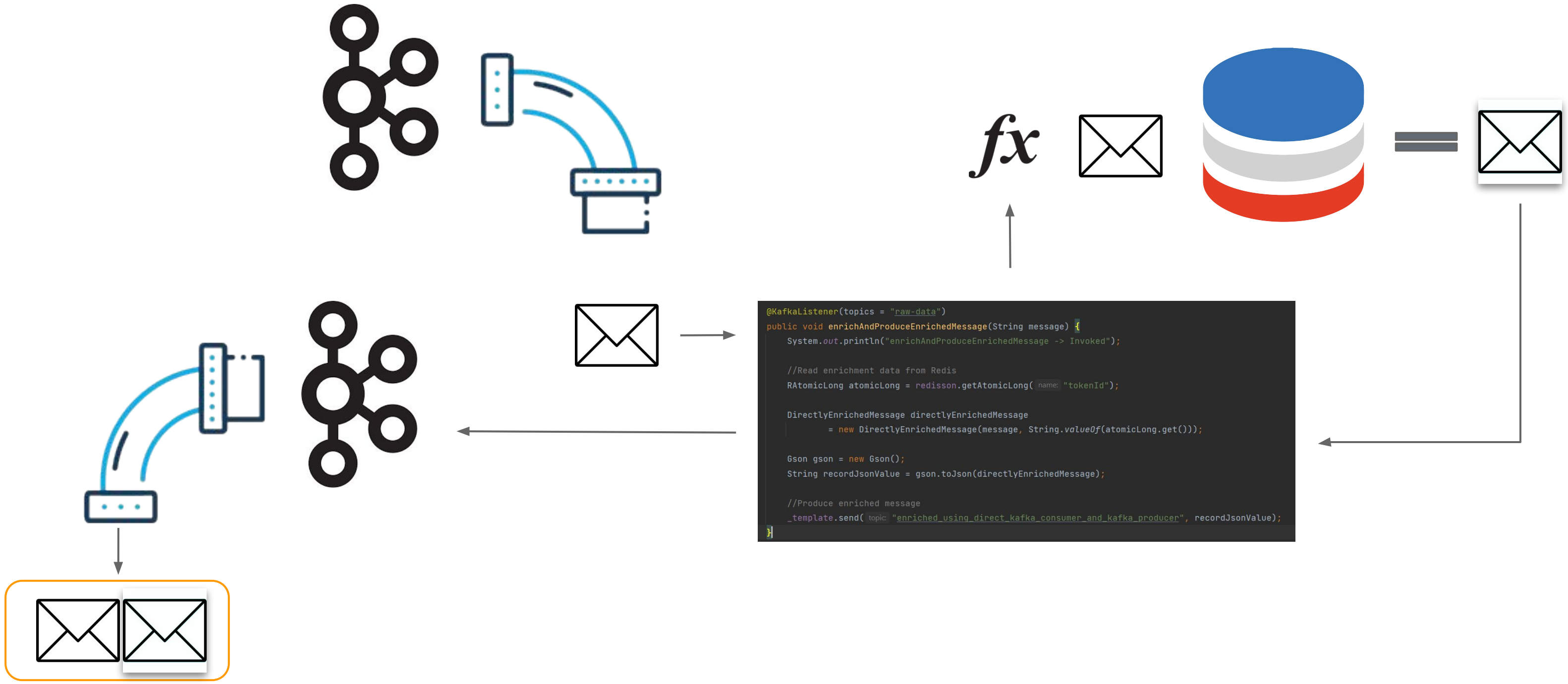
The Three Scales for Data Enrichment

- Scales for Data Enrichment
  - Data Enrichment **Coding Flexibility**
  - Data Enrichment Function **Overhead**
    - How long does it take to receive a response?
    - What is the byte size of the response?
    - How much data do I have to scan in order to receive a response?
  - Data Enrichment **Hosting** and **Management**
- Related Concerns
  - Can We Pre Calculate Enrichment Data?
  - Is it Possible to Clone the Data Source Into a kafka Topic?
    - Can we “pay” the network bandwidth overhead?
    - Can we “pay” the storage overhead?
  - Should The Data Enrichment Process Preserve Message Ordering?

Let’s Try

Attempt #1

"On The Fly" Data Enrichment
Using a kafka Consumer and a kafka Producer



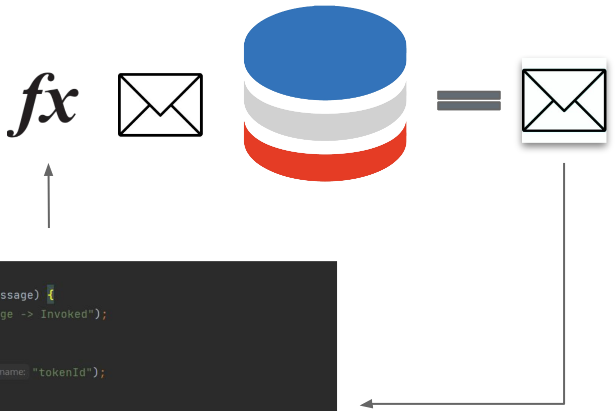
```java
@KafkaListener(topics = "raw-data")
public void enrichAndProduceEnrichedMessage(String message) {
    System.out.println("enrichAndProduceEnrichedMessage -> Invoked");

    //Read enrichment data from Redis
    RAtomicLong atomicLong = redisson.getAtomicLong( name: "tokenId");

    DirectlyEnrichedMessage directlyEnrichedMessage
            = new DirectlyEnrichedMessage(message, String.valueOf(atomicLong.get()));

    Gson gson = new Gson();
    String recordJsonValue = gson.toJson(directlyEnrichedMessage);

    //Produce enriched message
    _template.send( topic: "enriched_using_direct_kafka_consumer_and_kafka_producer", recordJsonValue);
}
```

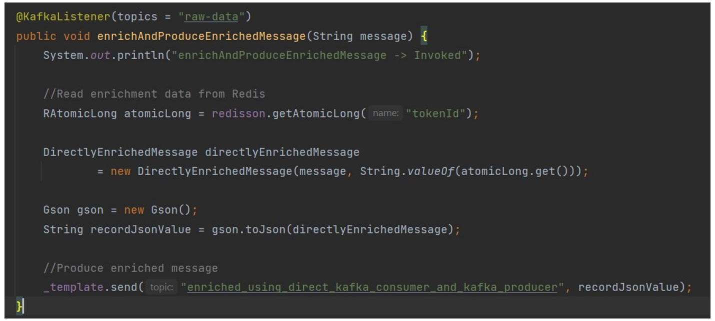
Demo Time

Kobi Hikri

“On The Fly” Data Enrichment Considerations

Enrichment Function is Not Limited. Anything We Can Code - Can Become An Enrichment Function

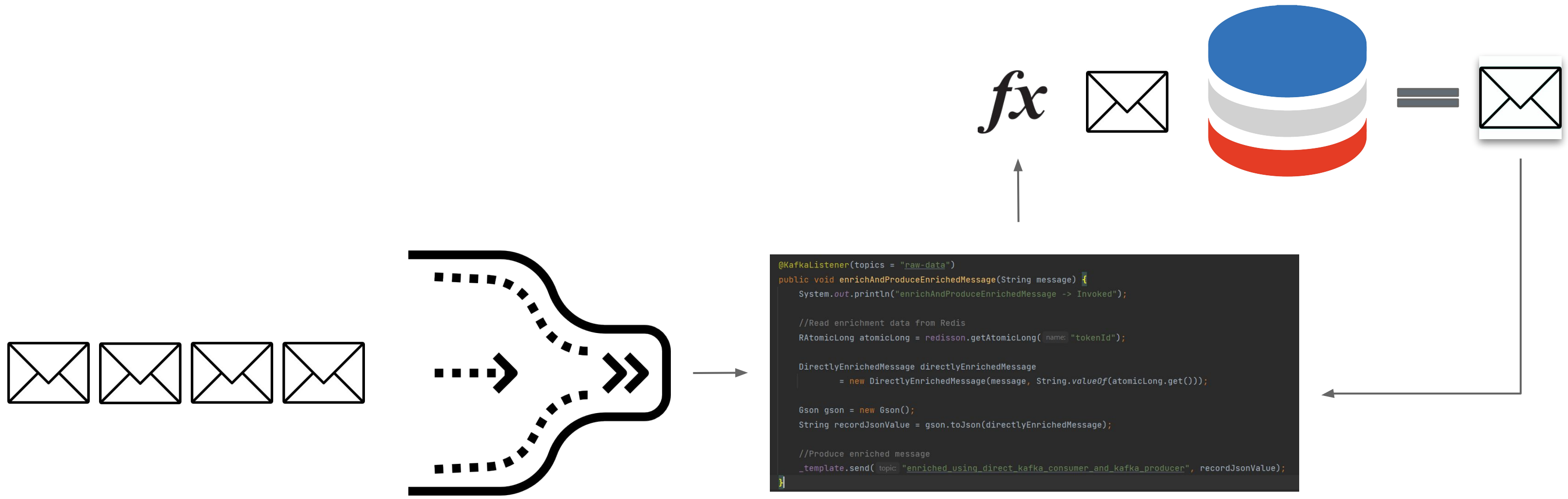




```
@KafkaListener(topics = "raw-data")
public void enrichAndProduceEnrichedMessage(String message) {
    System.out.println("enrichAndProduceEnrichedMessage -> Invoked");

    //Read enrichment data from Redis
    RAtomicLong atomicLong = redisson.getAtomicLong( name: "tokenId");

    DirectlyEnrichedMessage directlyEnrichedMessage
            = new DirectlyEnrichedMessage(message, String.valueOf(atomicLong.get()));

    Gson gson = new Gson();
    String recordJsonValue = gson.toJson(directlyEnrichedMessage);

    //Produce enriched message
    _template.send( topic: "enriched_using_direct_kafka_consumer_and_kafka_producer", recordJsonValue);
}
```

“On The Fly” Data Enrichment Considerations

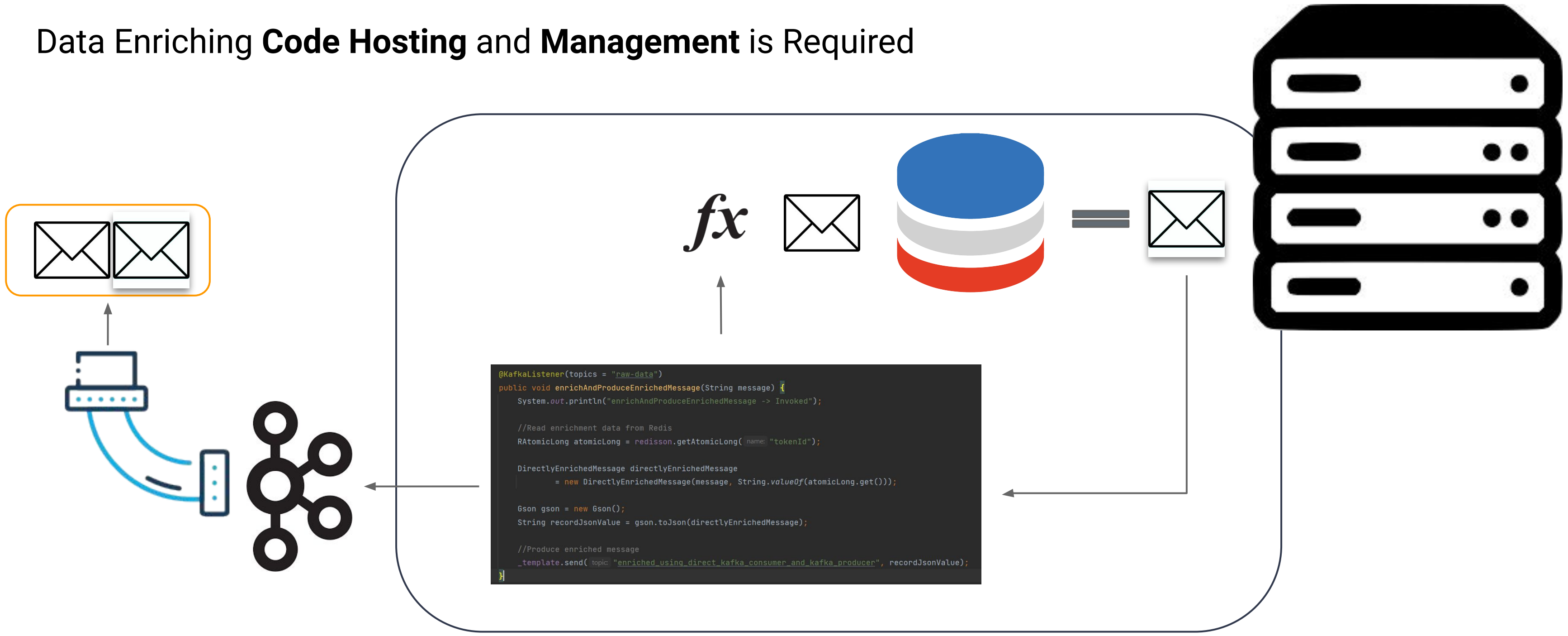
Our Enrichment Function Becomes a **Bottleneck**



```java
@KafkaListener(topics = "raw-data")
public void enrichAndProduceEnrichedMessage(String message) {
    System.out.println("enrichAndProduceEnrichedMessage -> Invoked");

    //Read enrichment data from Redis
    RAtomicLong atomicLong = redisson.getAtomicLong( name: "tokenId");

    DirectlyEnrichedMessage directlyEnrichedMessage
            = new DirectlyEnrichedMessage(message, String.valueOf(atomicLong.get()));

    Gson gson = new Gson();
    String recordJsonValue = gson.toJson(directlyEnrichedMessage);

    //Produce enriched message
    _template.send( topic: "enriched_using_direct_kafka_consumer_and_kafka_producer", recordJsonValue);
}
```

“On The Fly” Data Enrichment Considerations

Data Enriching **Code Hosting** and **Management** is Required

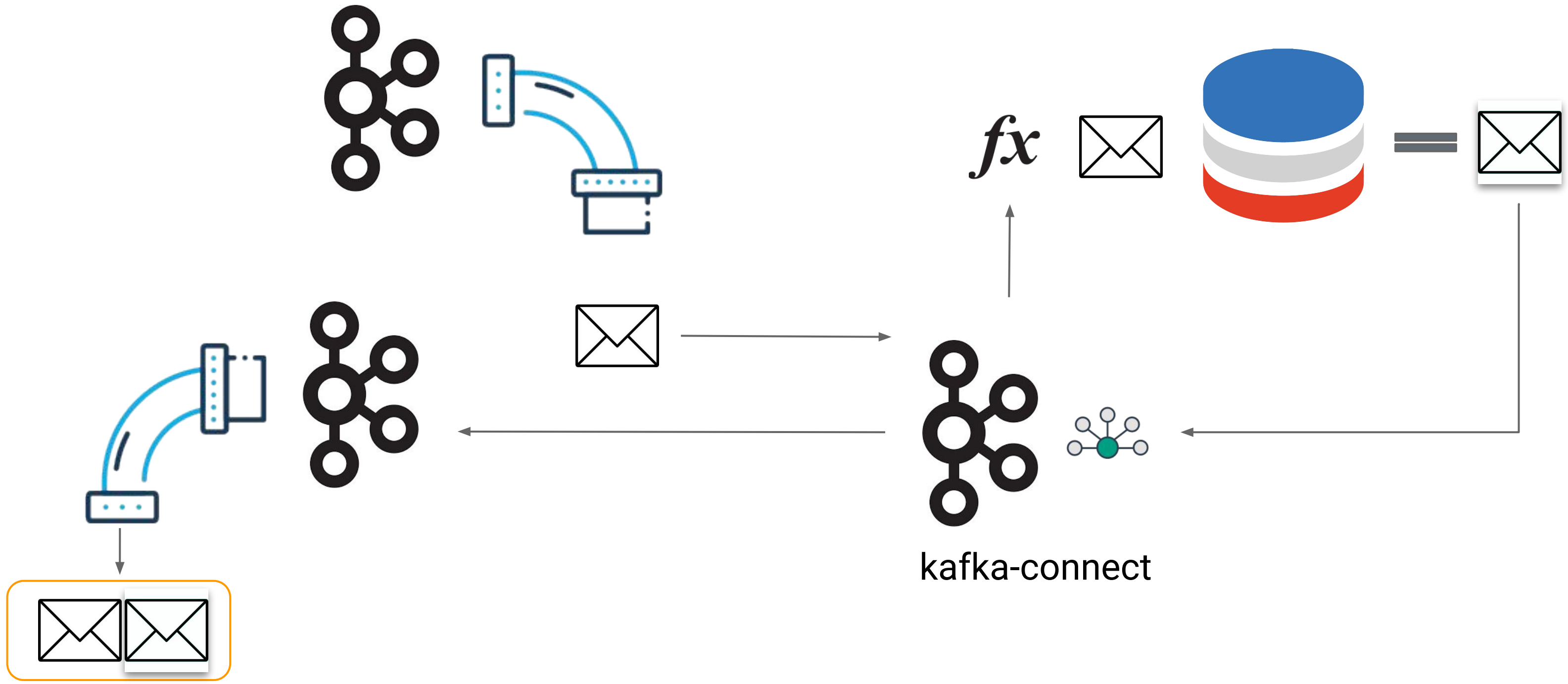


```java
@KafkaListener(topics = "raw-data")
public void enrichAndProduceEnrichedMessage(String message) {
    System.out.println("enrichAndProduceEnrichedMessage -> Invoked");

    //Read enrichment data from Redis
    RAtomicLong atomicLong = redisson.getAtomicLong( name: "tokenId");

    DirectlyEnrichedMessage directlyEnrichedMessage
            = new DirectlyEnrichedMessage(message, String.valueOf(atomicLong.get()));

    Gson gson = new Gson();
    String recordJsonValue = gson.toJson(directlyEnrichedMessage);

    //Produce enriched message
    _template.send( topic: "enriched_using_direct_kafka_consumer_and_kafka_producer", recordJsonValue);
}
```

Let’s Try Again

Attempt #2

"On The Fly" Data Enrichment Using kafka-connect



kafka-connect

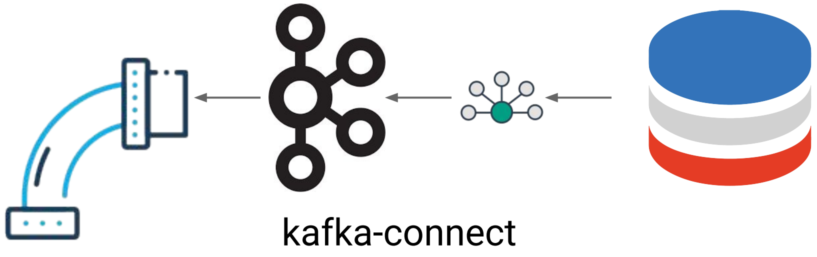
Brief Description

- The Integration Mechanism for Apache kafka
  - Allows exporting data from a kafka cluster into 3rd-party
  - Allows importing data from 3rd-party into a kafka Cluster
- Deployed as a Distributed Cluster of Workers
- Exposes a Rest API for Management and Configuration
- Work is Done Inside Connector Tasks Which Are Managed by the Connect Cluster

kafka-connect Connector Types

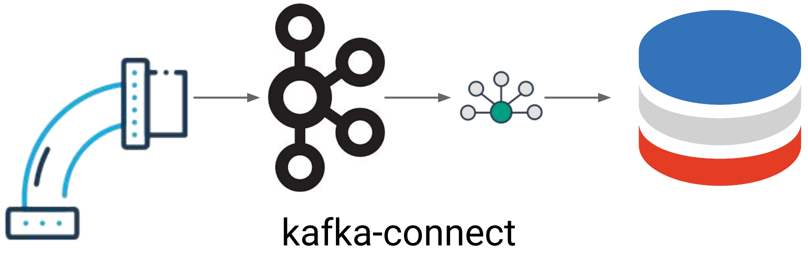
Source Connectors

Responsible for importing data from a 3rd-party data source - into kafka topics



Sink Connectors

Responsible for exporting data from kafka topics - into a 3rd-party

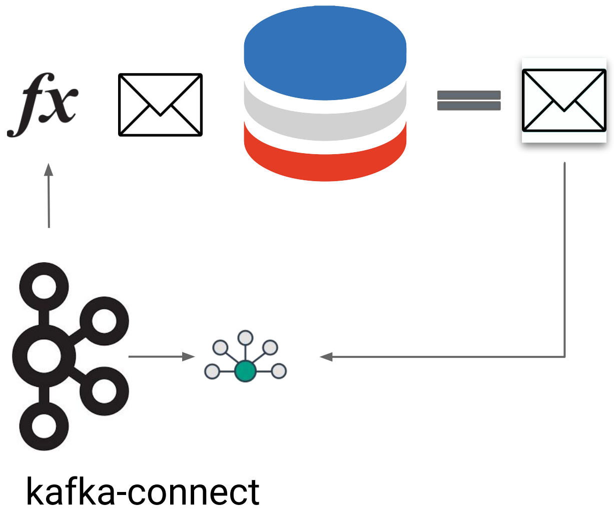


Demo Time - a kafka Sink Connector for “On The Fly” Data Enrichment

Kobi Hikri

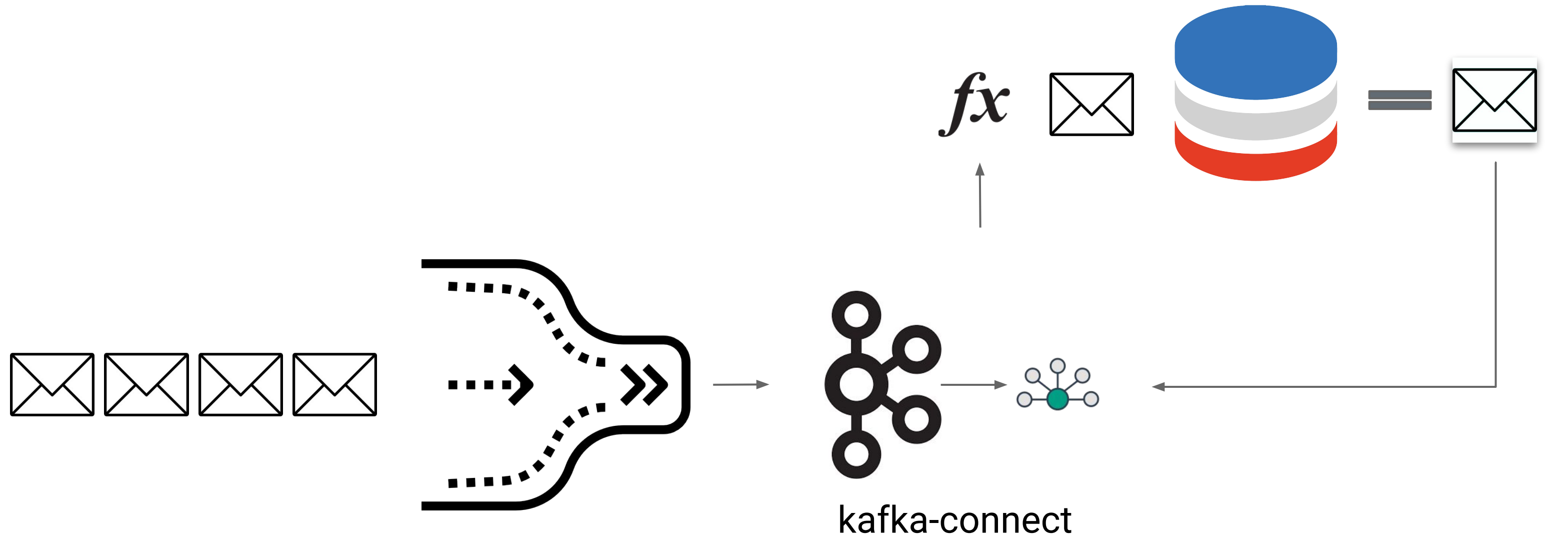
“On The Fly” Data Enrichment Using kafka-connect Considerations

Enrichment Function is Not Limited. Anything We Can Code - Can Become An Enrichment Function



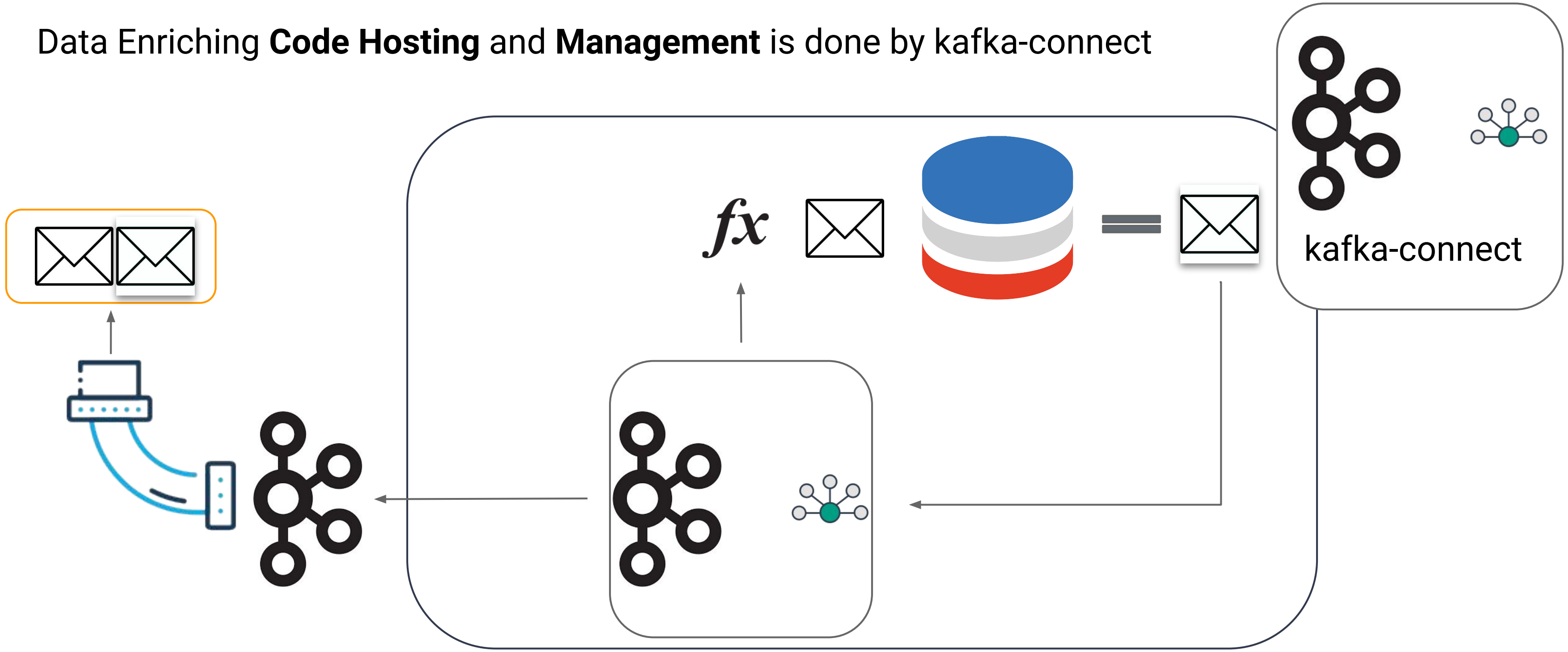
“On The Fly” Data Enrichment Using kafka-connect Considerations

Our Enrichment Function is Still a **Bottleneck**



“On The Fly” Data Enrichment Using kafka-connect Considerations

Data Enriching **Code Hosting** and **Management** is done by kafka-connect



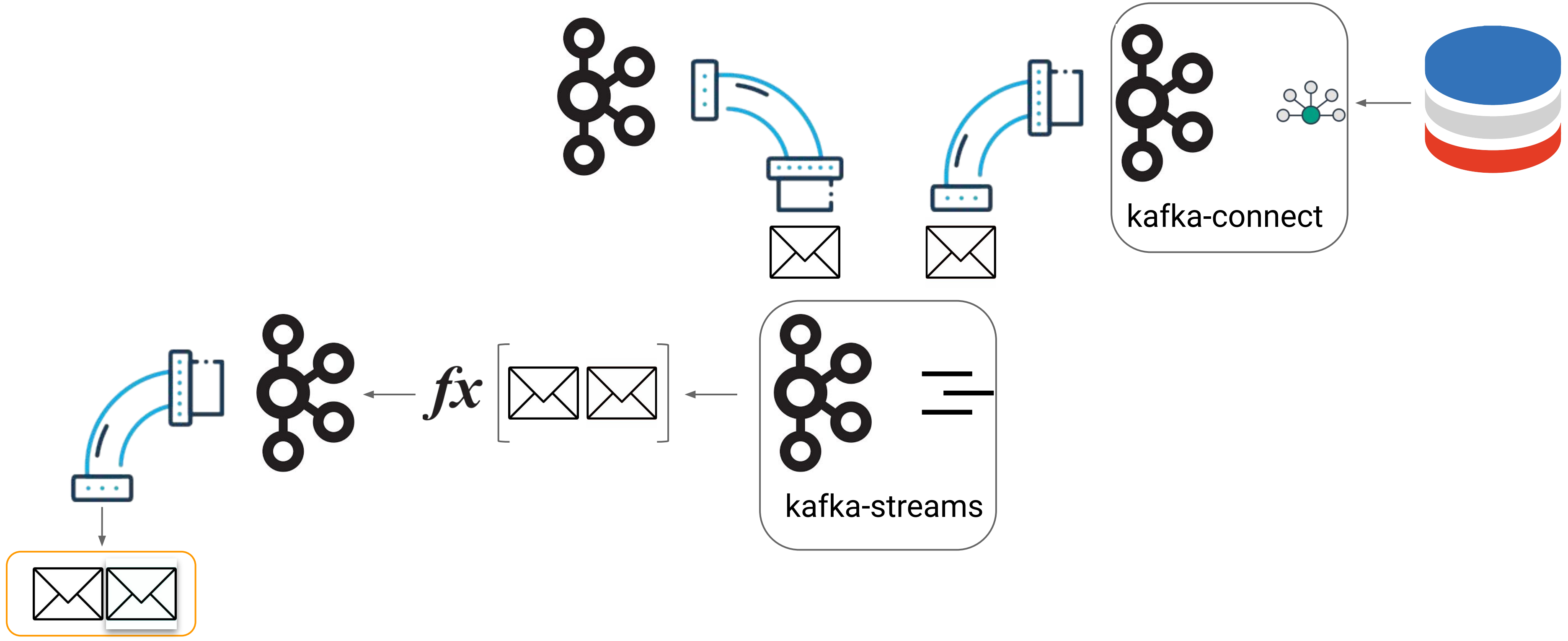
Can Anything Be Done About Our Enrichment Function Posing a Bottleneck?

- **It Depends**
  - Can we pre calculate enrichment data (i.e. Is our raw data predictable to some degree)?
  - If so - is the overhead of writing the enrichment data to a a kafka topic (e.g. storage overhead, bandwidth overhead) acceptable?
- **if** you answered **yes** to both aforementioned questions - **then the answer is yes**.
- **else** - **consider scaling out** your enrichment code. Run more instances of your enrichment code in parallel. Keep in mind that original message ordering is not guaranteed among multiple enrichment function instances.

Next Attempt

Attempt #3

“On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams



kafka-streams

Brief Description

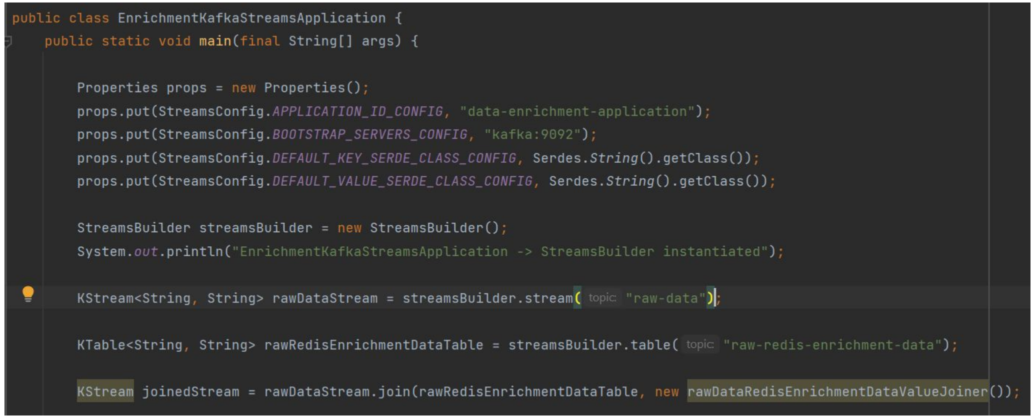
- A Client Library Allowing Data Manipulation for kafka topics Based Data
  - Data manipulation includes consuming, transforming, filtering and producing data
- kafka-streams Applications Combine Processing Nodes into Topologies
  - Topologies Are Made of Nodes
    - Source Nodes
    - Stream Processor Nodes
    - Sink Nodes
- kafka-streams Applications Requires **Hosting** and Rely on a Java Virtual Machine
- Exposes a **D**omain **S**pecific **L**anguage, Allowing for The Abstractions of Data Streams And Data Tables
  - KStream
  - KTable

Demo Time - “On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams

Kobi Hikri

“On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams - Considerations

Enrichment Function is Limited By Compliance To kafka-streams DSL (But Flexible Through Development of kafka-streams DSL Supported Facilities (e.g. Value Joiners)

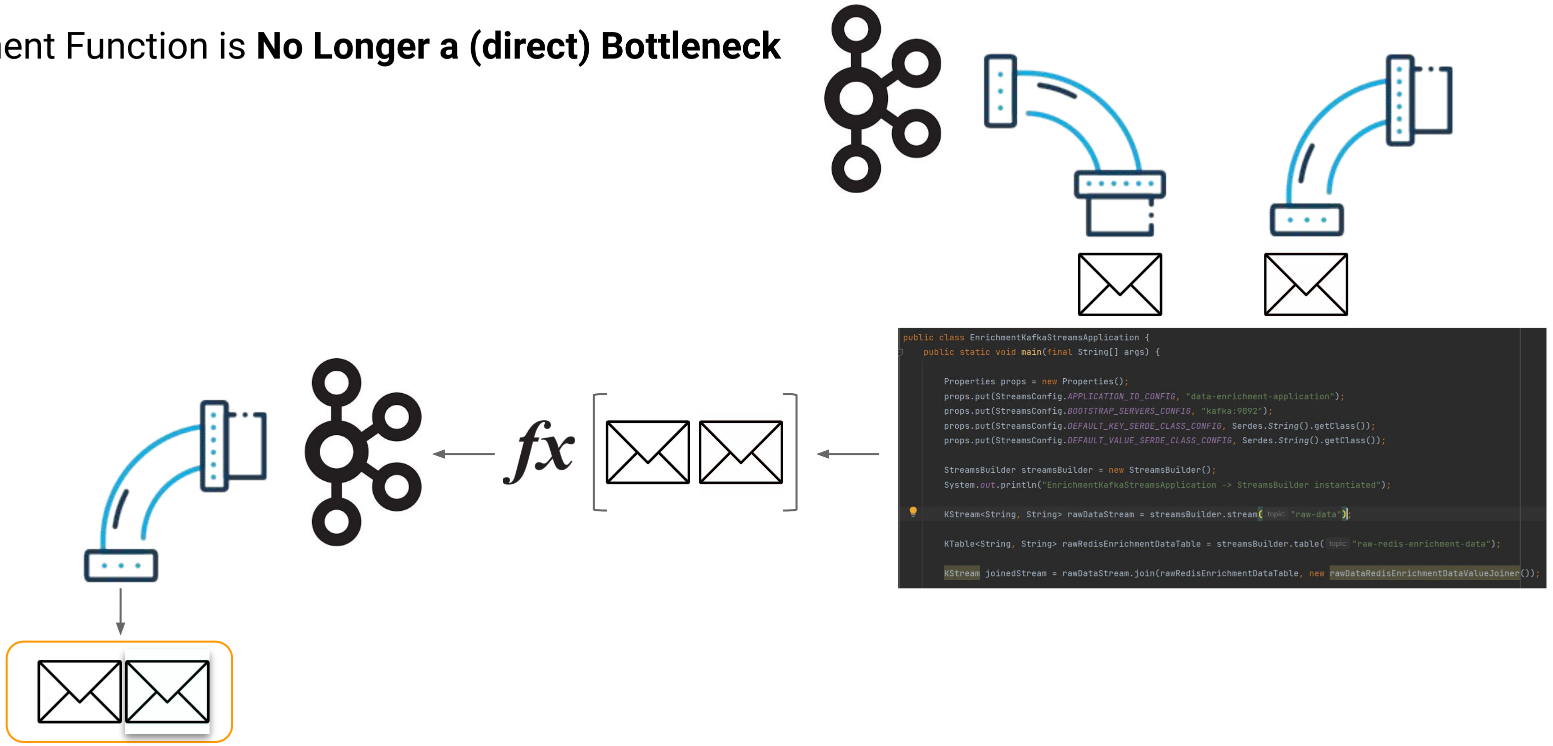


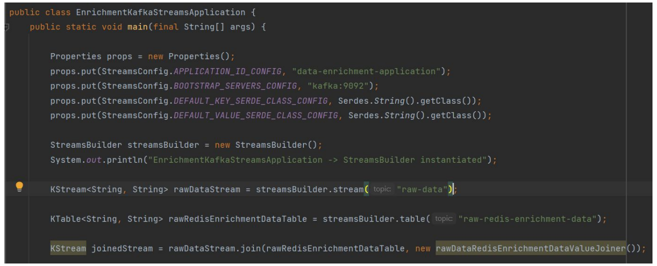
```java
public class EnrichmentKafkaStreamsApplication {
    public static void main(final String[] args) {

        Properties props = new Properties();
        props.put(StreamsConfig.APPLICATION_ID_CONFIG, "data-enrichment-application");
        props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka:9092");
        props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
        props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());

        StreamsBuilder streamsBuilder = new StreamsBuilder();
        System.out.println("EnrichmentKafkaStreamsApplication -> StreamsBuilder instantiated");

        KStream<String, String> rawDataStream = streamsBuilder.stream( topic: "raw-data");

        KTable<String, String> rawRedisEnrichmentDataTable = streamsBuilder.table( topic: "raw-redis-enrichment-data");

        KStream joinedStream = rawDataStream.join(rawRedisEnrichmentDataTable, new rawDataRedisEnrichmentDataValueJoiner());
```

kafka-streams

“On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams - Considerations

Our Enrichment Function is **No Longer a (direct) Bottleneck**

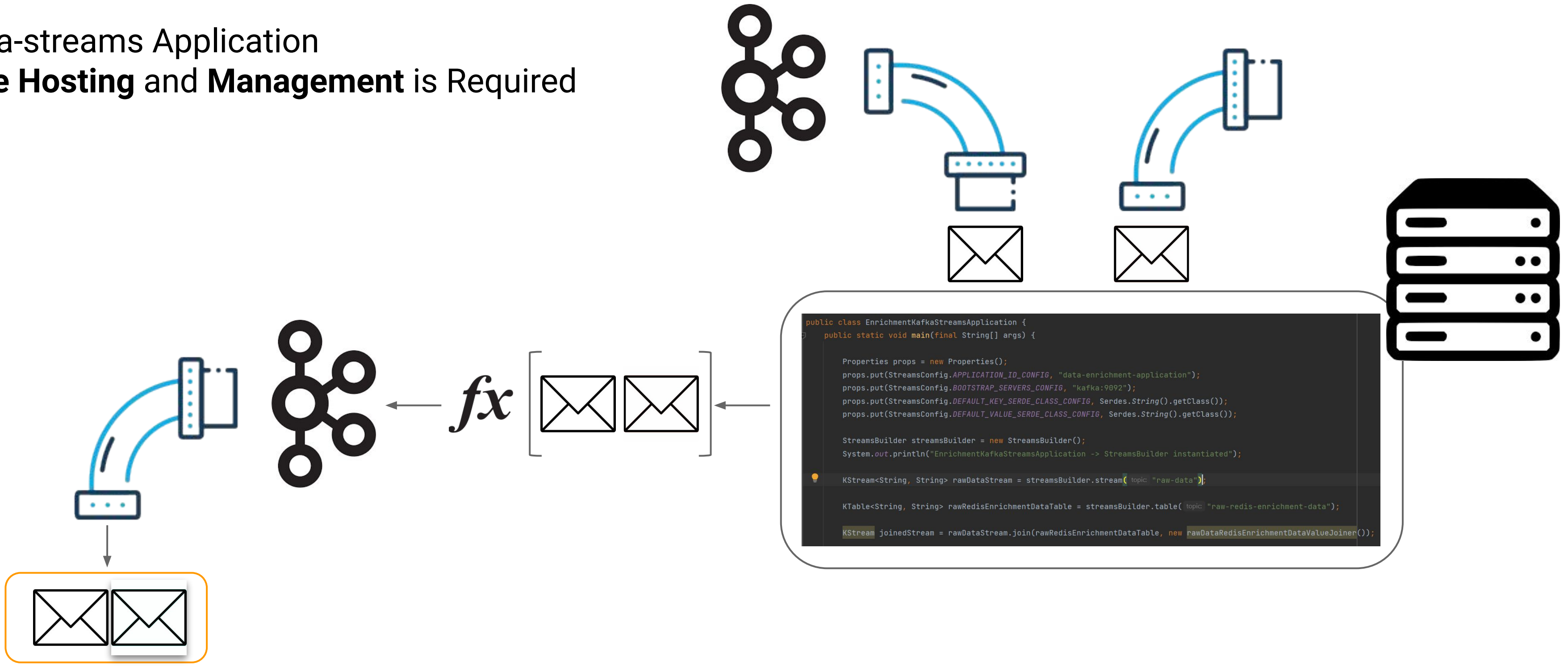




```
public class EnrichmentKafkaStreamsApplication {
    public static void main(final String[] args) {

        Properties props = new Properties();
        props.put(StreamsConfig.APPLICATION_ID_CONFIG, "data-enrichment-application");
        props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka:9092");
        props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
        props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());

        StreamsBuilder streamsBuilder = new StreamsBuilder();
        System.out.println("EnrichmentKafkaStreamsApplication -> StreamsBuilder instantiated");

        KStream<String, String> rawDataStream = streamsBuilder.stream( topic: "raw-data");

        KTable<String, String> rawRedisEnrichmentDataTable = streamsBuilder.table( topic: "raw-redis-enrichment-data");

        KStream joinedStream = rawDataStream.join(rawRedisEnrichmentDataTable, new rawDataRedisEnrichmentDataValueJoiner());
```

“On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams - Considerations

kafka-streams Application
Code Hosting and **Management** is Required



```java
public class EnrichmentKafkaStreamsApplication {
    public static void main(final String[] args) {

        Properties props = new Properties();
        props.put(StreamsConfig.APPLICATION_ID_CONFIG, "data-enrichment-application");
        props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka:9092");
        props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
        props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());

        StreamsBuilder streamsBuilder = new StreamsBuilder();
        System.out.println("EnrichmentKafkaStreamsApplication -> StreamsBuilder instantiated");

        KStream<String, String> rawDataStream = streamsBuilder.stream( topic: "raw-data");

        KTable<String, String> rawRedisEnrichmentDataTable = streamsBuilder.table( topic: "raw-redis-enrichment-data");

        KStream joinedStream = rawDataStream.join(rawRedisEnrichmentDataTable, new rawDataRedisEnrichmentDataValueJoiner());
```

So How Can We Avoid Having to **Write** and Host kafka-streams Applications?

- **Someone** Should Write and Host The kafka-streams Code
- Several Options Available
  - Lenses Sql (aka LSQL)
  - Confluent ksqlDB
- Both Are Good Tools!
- Both Provide an “SQL Like” DSL
  - Both Are Non ANSI SQL Compliant
- Lenses Sql Has No Free License Plan
- Confluent ksqlDB Offers a “Standalone” Free Plan

ksqlDb

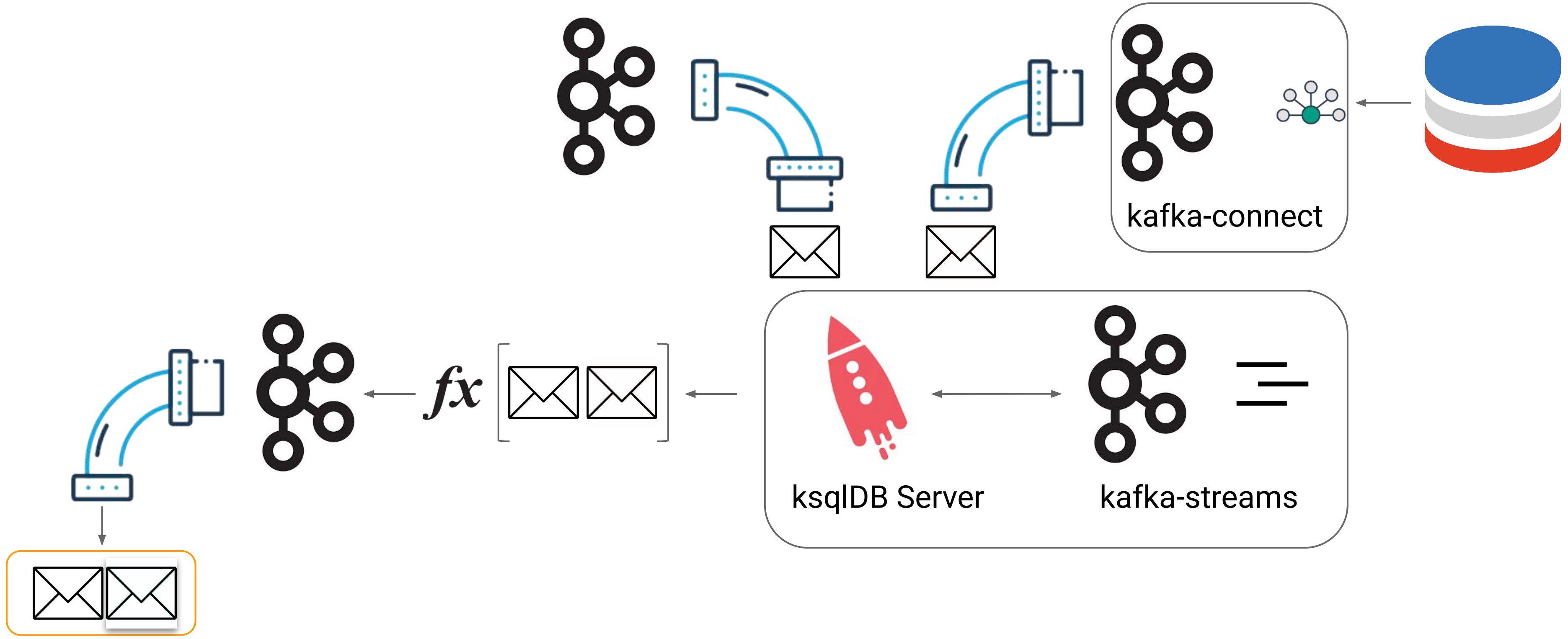
Brief Description

- Creates a Database Abstraction On Top of Your Kafka Topics
- Allows For Referencing Your Topics as Streams or As Tables
- Can Integrate With Other Data Sources Via kafka-connect Connectors

One More Time

Attempt #4

“On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams **via ksqlDB**

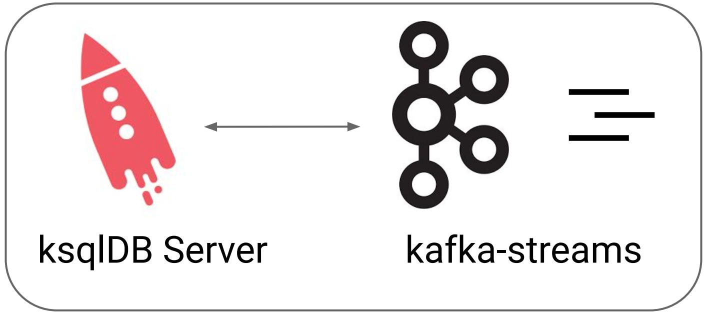


Demo Time - “On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams **via ksqlDB**

Kobi Hikri

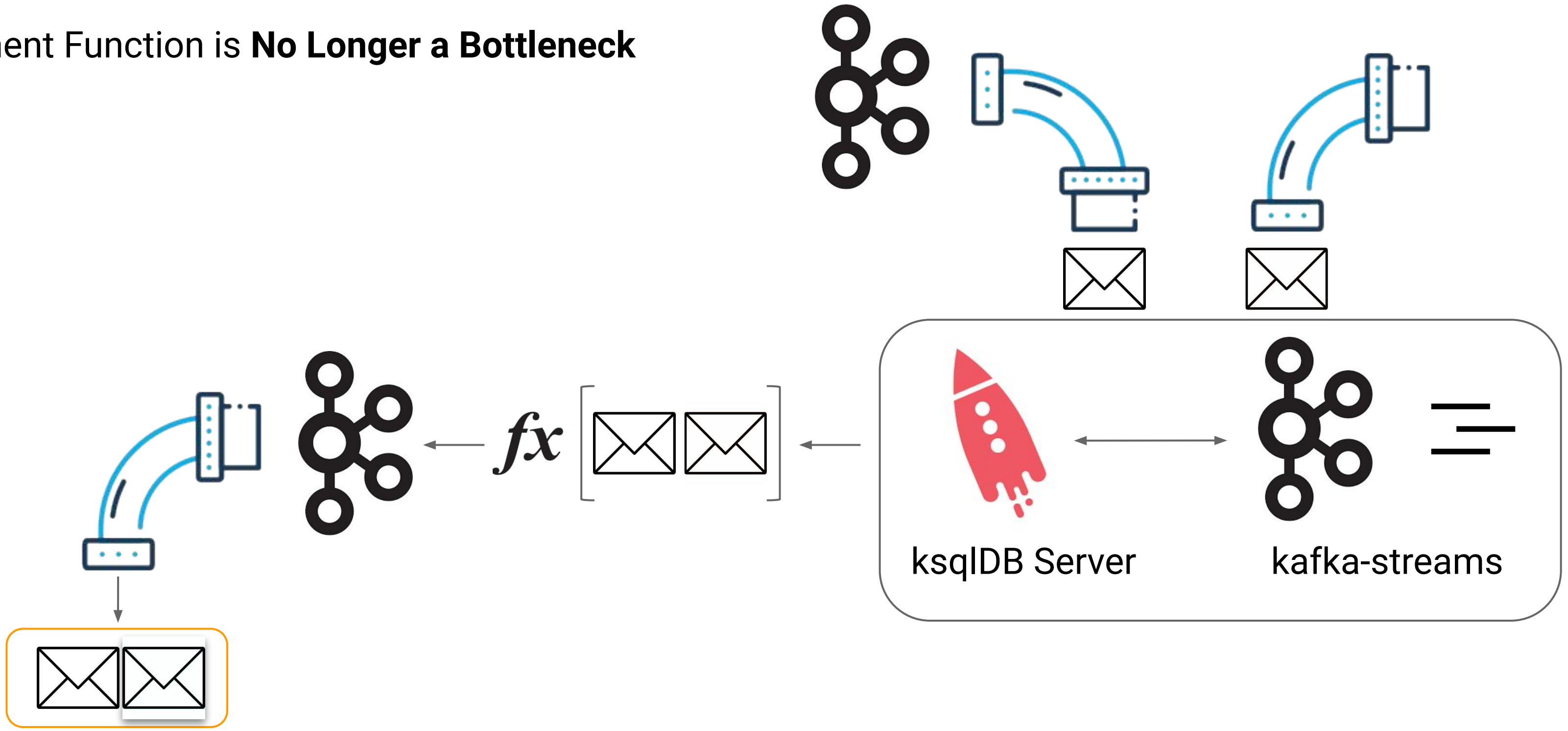
“On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams **via ksqlDB** - Considerations

Enrichment Function is Limited By Compliance To kafka-streams DSL **And** ksqlDB DSL



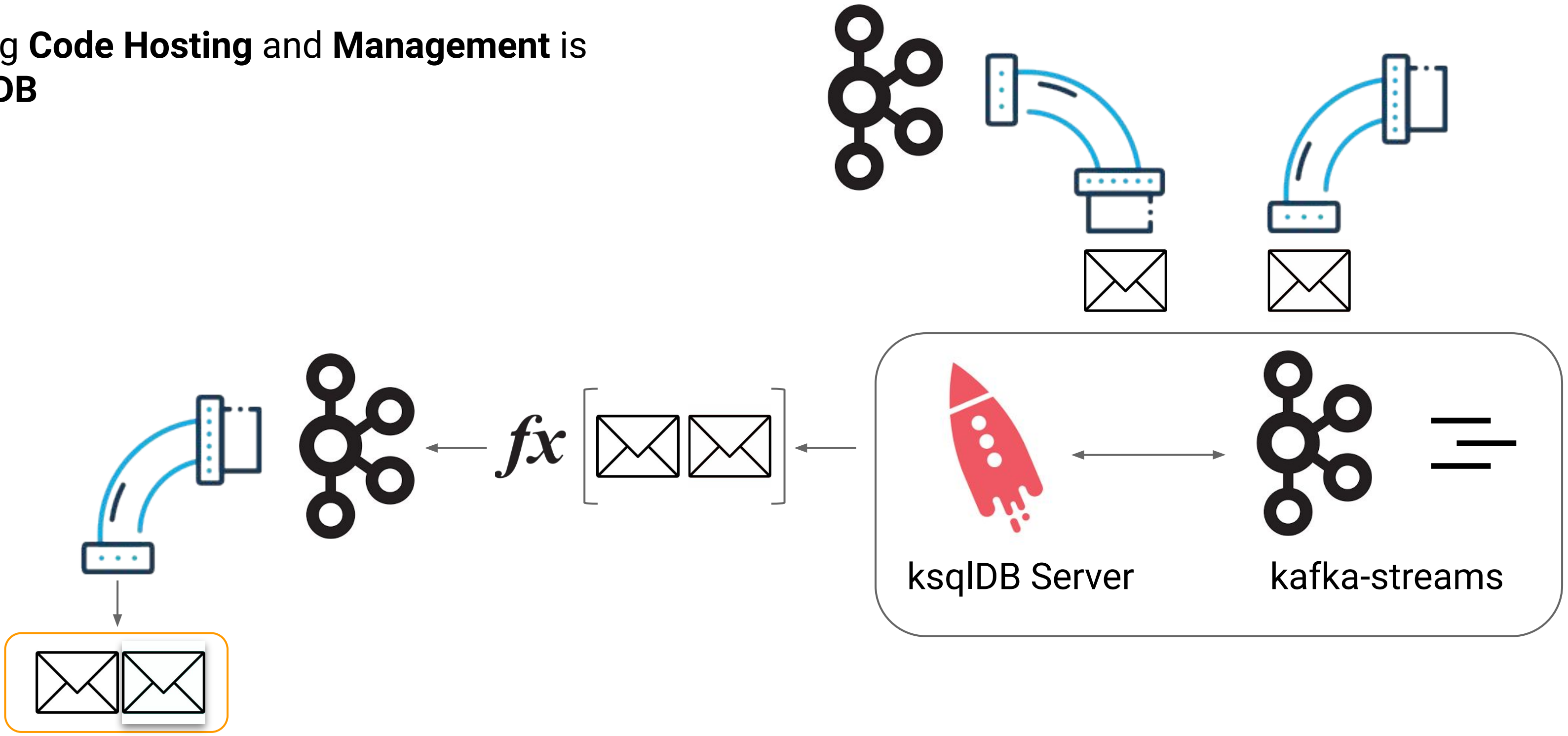
"On The Fly" Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams via ksqlDB - Considerations

Our Enrichment Function is **No Longer a Bottleneck**



“On The Fly” Data Enrichment Using Pre Calculated Enrichment Data - Utilizing kafka-connect and kafka-streams via ksqlDB - Considerations

Data Enriching **Code Hosting** and **Management** is **done by ksqlDB**



Is This All We Need to Know About Kafka Data Enrichment?

No, but It’s a Start!

Resources To Check Out:

https://kafka.apache.org/documentation/streams/

Things To Consider:

- Enrichment Data Caching
  - Using an External Distributed Cache
- Data Ordering and Reordering
- Exactly Once
  - **Very** Difficult to Achieve In a Full Data Pipeline
    - Plan for Idempotent Data Processing Code!
    - Be Aware That Data Could Be Missing From Your Pipeline!
- “Real-time” + Real Life = “Online” At Best

In Conclusion

What we have covered

- Apache kafka Data Enrichment - The Process of Adding External Data To Messages We Consume From a kafka Topic
- The Attempts We've Made
  - Writing Our Own kafka Consumers and kafka Producers
  - Using kafka-connect as an Enrichment Code Hosting Mechanism
  - When Our Enrichment Data Can Be Pre - Fetched
    - Using kafka-connect as an Enrichment Data Fetcher and kafka-streams as an Enrichment Engine
    - Abstracting The Usage of kafka-streams (using ksqlDB as an example)

More Attempts

Attempt #next

Kobi Hikri - Software Simplifier

Q&A

Kobi Hikri

Thanks!

Your Time is Appreciated