Kafka Connect

Raghavi, Janaswamy
Sr.Principal Engineer (TLCP)
raghavi.janaswamy@optum.com

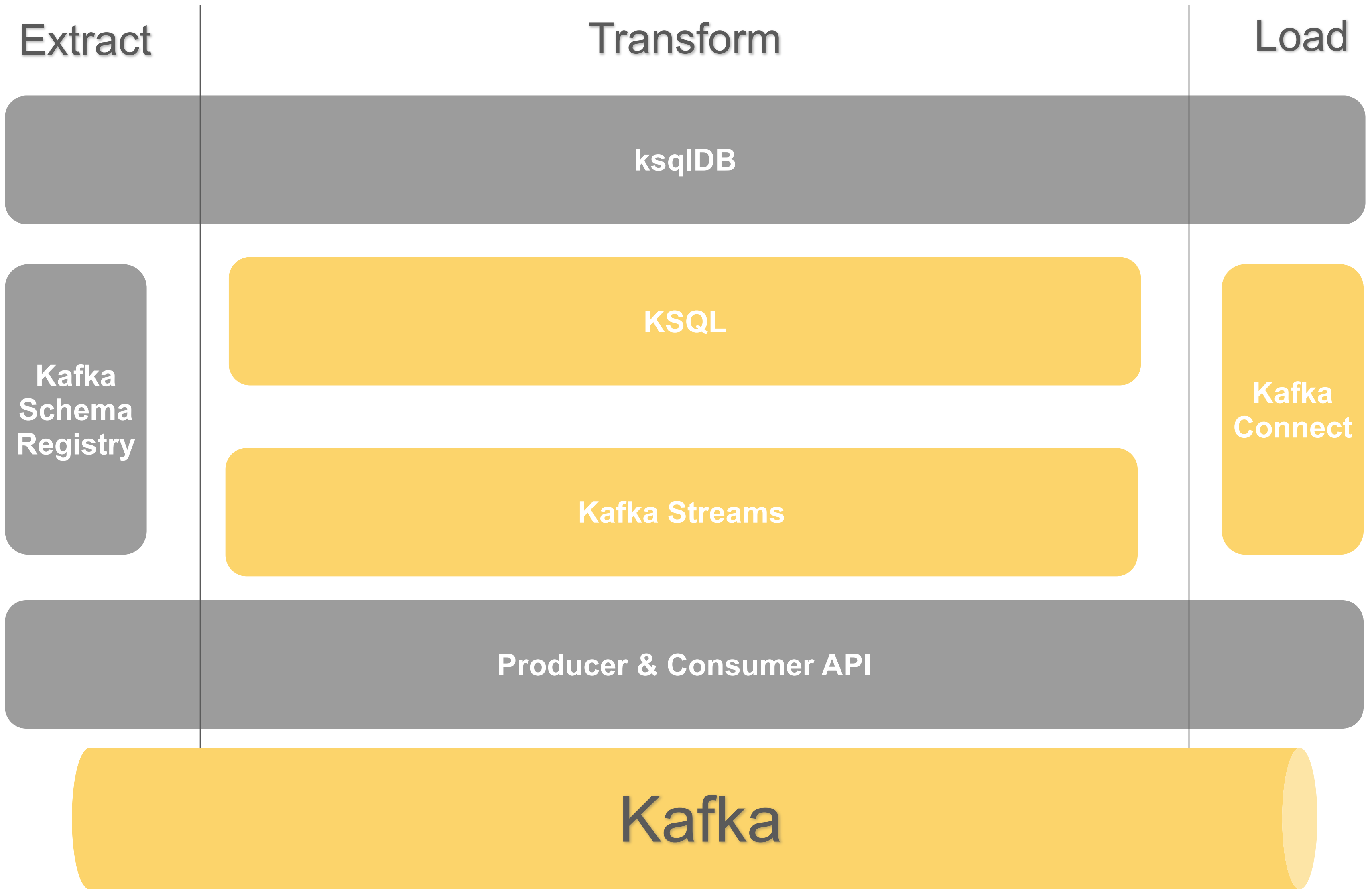
OPTUM®

Agenda

- Kafka & Kafka connect Concepts
- Lab 1: Kafka platform Setup - Local Docker Environment
- Lab 2: Kafka Connect – Build Source Connector
- Lab 3: Kafka Connect – Build Sink Connector
- Kafka Connect implementations in OptumCare & P360
- Pointers to the Advanced Concepts
- Q&A

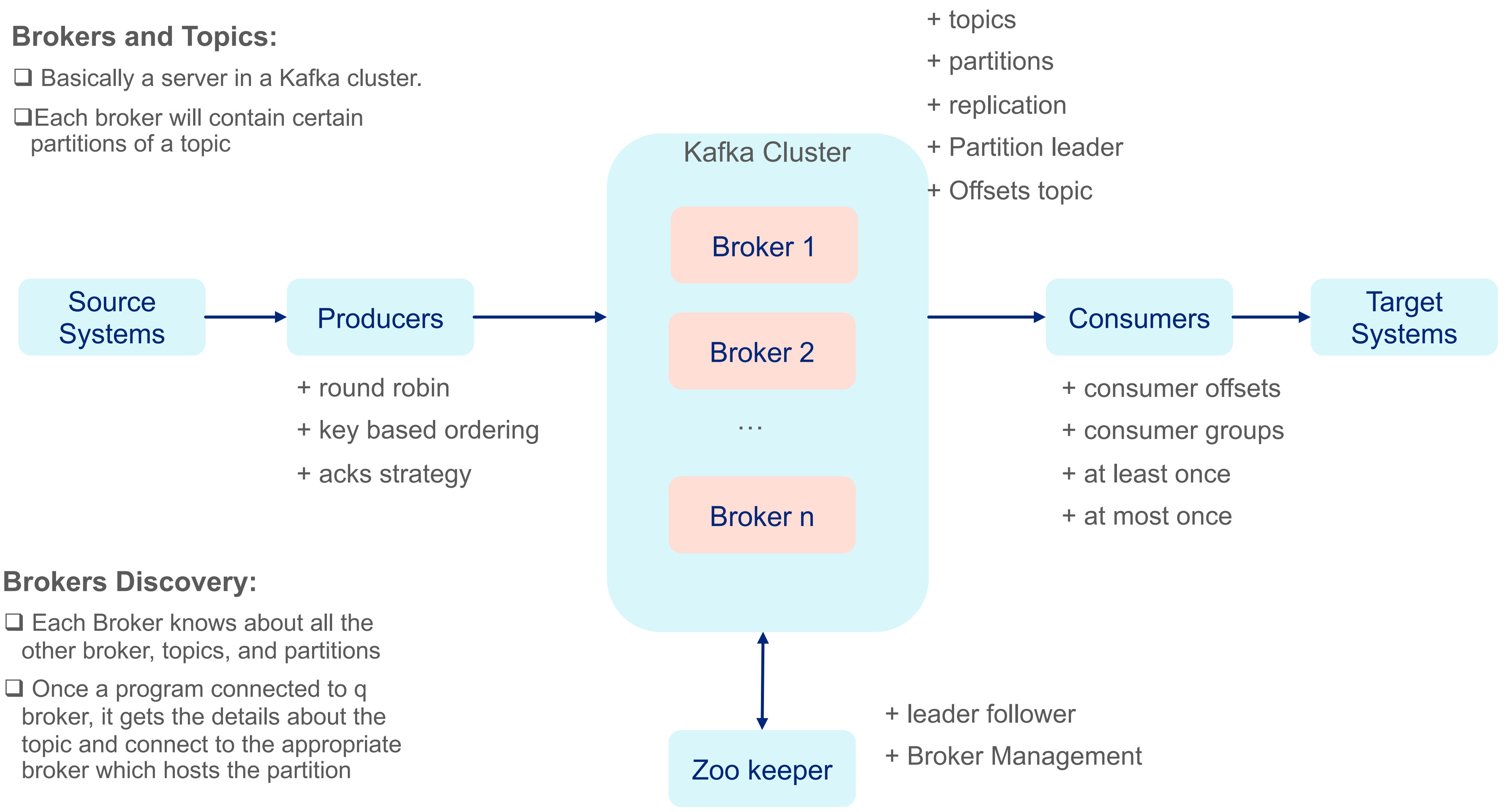
Optum

Kafka Ecosystem



Kafka Core - Terminology

- ❑ **Brokers and Topics:**
  - ❑ Basically a server in a Kafka cluster.
  - ❑Each broker will contain certain partitions of a topic

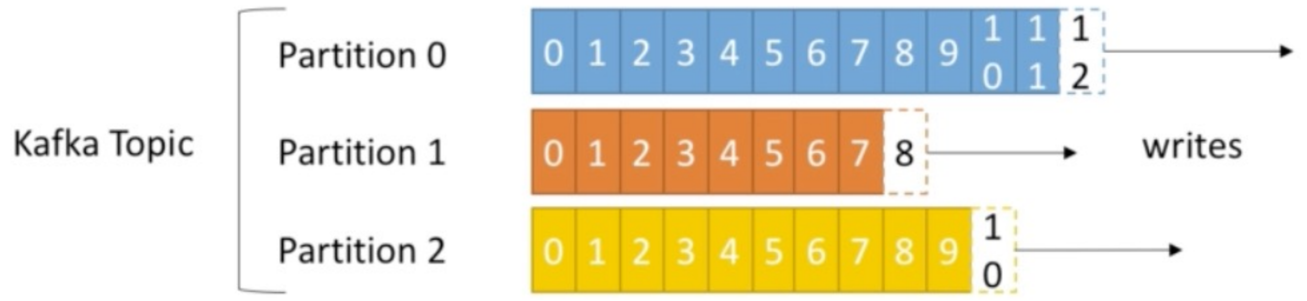


- ❑ **Brokers Discovery:**
  - ❑ Each Broker knows about all the other broker, topics, and partitions
  - ❑ Once a program connected to q broker, it gets the details about the topic and connect to the appropriate broker which hosts the partition

Kafka - Internals

- **Topic Replication:**
  - Distributed world needs replication.
  - At any point in time, there will be one broker for a partition which will serve as a leader.
  - Leader sends and receives data. Other brokers just stay in sync.
- **Producer - Data Acknowledgement**
  - Producer can choose to receive acknowledgment from the Broker
    - Producer won't wait for acknowledgment
    - Producer will wait for acknowledgment
    - Producer will wait for acknowledgment for leader and replicas
- **Message keys**
  - Data is sent in round robin across partitions
  - Data for a particular key is always sent to a particular partition. Key hashing.
- **Consumers and Consumer Groups:**
  - Consumers can read from multiple partitions for a single topic in parallel.
  - Consumer read data in groups, called consumer groups
  - Two consumers of the same group will read from mutually exclusive partitions
- **Consumer Offsets/ Acknowledgement**
  - Stored to a Kafka topic "__consumer_offsets"
  - Consumers can continue from where they left
- **Delivery Semantics**
  - Consumer choose when to commit offsets
    - Commit once received
    - Commit once received and processed

Terminology - Kafka Topics and Messages

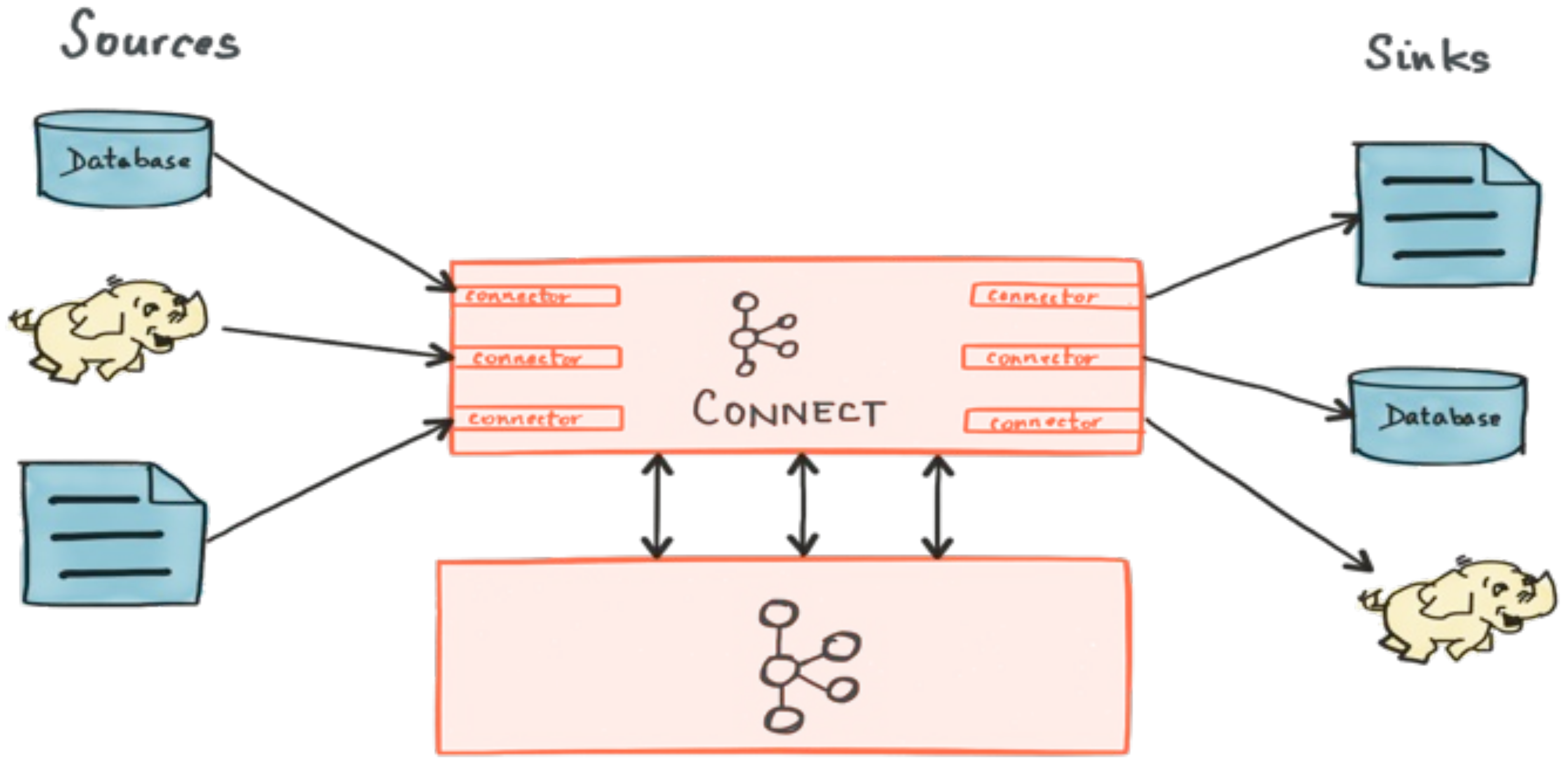


- ❑ **Topic:**
  - ❑ A particular stream of data.
  - ❑ Every record is called a message.
  - ❑ A topic will have a name
  - ❑ Data is kept for a limited duration
- ❑ **Partitions:**
  - ❑ Topics are split in partitions.
  - ❑ Partitions are ordered
  - ❑ Data once written to the partition, it CANNOT be changed. It's immutable.
- ❑ **Offset:**
  - ❑ Unique id given to a message with in a partition
  - ❑ First message will have an offset of "0"
  - ❑ Offset will NEVER go back to zero, even after the earlier messages are deleted.
- ❑ **Message**
  - ❑ To identify a message - use the partition id & offset together
  - ❑ Message in a particular partition is chronologically ordered.
  - ❑Messages across partitions in a topic are NOT chronologically ordered

Optum

What is Kafka Connect

Free, open-source component of Kafka ecosystem that provides extensible interface to connect to different data sources & load to or from Kafka in scalable and reliable way.



Advantages

- Standalone & Distributed Compute: Kafka connect can run as a cluster providing multiple workers on which a job can be run.
- Extensible: Provides extensible interface that can be leveraged to create different source and sink connectors. Many free/open-source sink connectors can be plugged into a connect cluster for connecting to different sources.
- Configurable: All jobs can be configured at the start of the cluster or can be provided via REST API to a running cluster instance.
- Reliable: Provides at least once processing & retry capabilities with logging and metrics exposed for monitoring & alerting.
- Data Transformations: Provides capability to perform light weight transformations on the data.

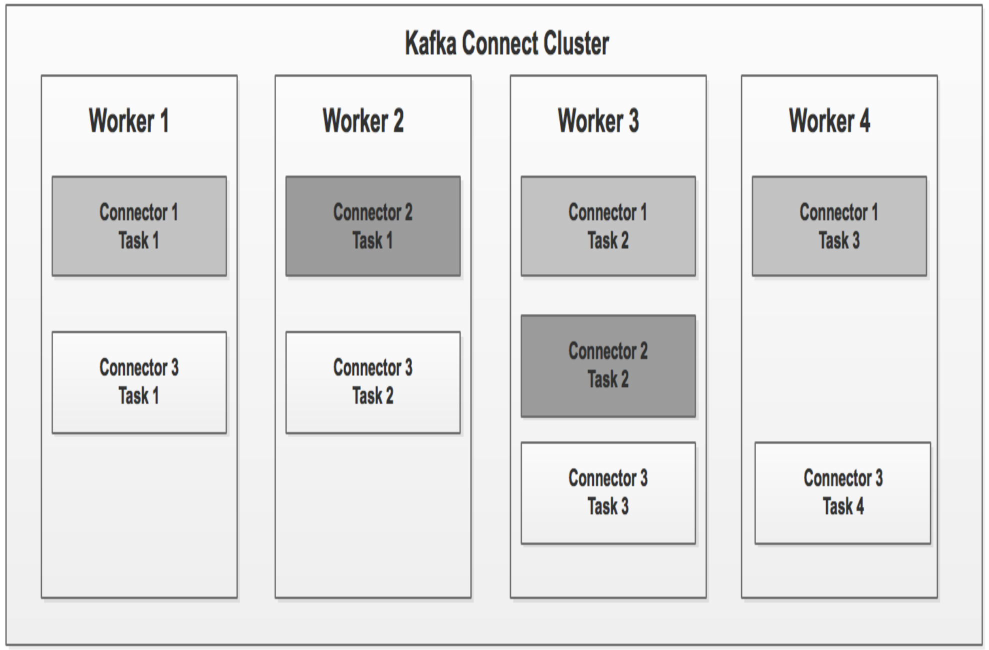
Optum

How Kafka Connect works

Terminologies

Worker: Processes running on instances that make a Kafka cluster.

- Types:
  - Standalone
    - Single process running on an instance.
    - Scalability/Fault-tolerance not achieved.
    - Very useful in testing/small extractions.
  - Distributed
    - Scalable solution to have parallel processing.
    - Embedded fault tolerance based on inherent Kafka Infrastructure utilization.



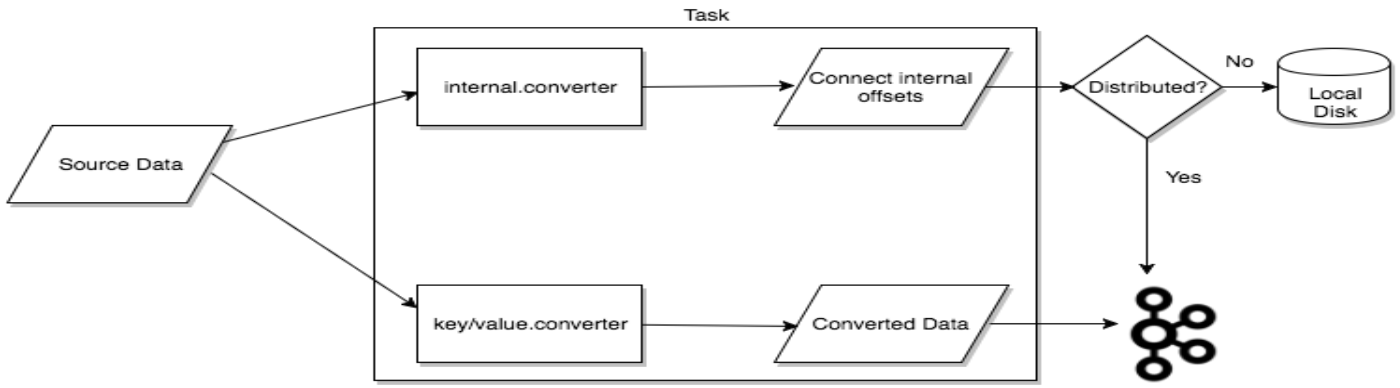
Optum

Connect Internals: Tasks

- Tasks : Implementation that actually copy the data

  Connector Jobs -> Task 1, Task 2, ……. Task n

- Internal Topics (Compacted) : Task State storage
  - Config Topic: Stores all connector configuration.
  - Offset Topic: Stores all source connector offsets to support incremental data extaction. Please note that Sink connector works on default consumer_offsets topic for consumer offset storage.
  - Status Topic: Stores all connector statuses.



Optum

Type of Kafka Connectors

Connectors: Job configuration that connects Kafka to different sources/destinations

- Source Connectors : Imports data into Kafka Topic

Ex: JDBC Source

JMS Source

- Sink Connectors : Exports data from Kafka Topic

JDBC Sink

Elasticsearch Sink

Amazon S3 Sink

Sample Source Connector

```
{
  "name": "mysql-curl-claimdop-volumes",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "incrementing.column.name": "Id",
    "transforms.createKey.type": "org.apache.kafka.connect.transforms.ValueToKey",
    "connection.password": "optum",
    "tasks.max": "2",
    "transforms": "createKey,extractInt",
    "transforms.extractInt.type": "org.apache.kafka.connect.transforms.ExtractField$Key",
    "batch.max.rows": "10",
    "table.types": "TABLE",
    "table.poll.interval.ms": "2000",
    "table.whitelist": "claims_dop_metrics_volumes_sink_test",
    "mode": "incrementing",
    "topic.prefix": "kaas.pdp.dev.p360-",
    "transforms.extractInt.field": "tin",
    "connection.user": "root",
    "transforms.createKey.fields": "tin",
    "poll.interval.ms": "2000",
    "name": "mysql-curl-claimdop-volumes",
    "connection.url": "jdbc:mysql://apsrt4811.uhc.com:30898/optum",
    "client.id": "kaas.pdp.dev"
  }
}
```

Sample C* Sink Connector

```
{
    "name": "cassandra-sink-claims-dop-volumes",
    "config": {
    "connector.class": "com.datamountaineer.streamreactor.connect.cassandra.sink.CassandraSinkConnector",
    "tasks.max": "1",
    "topics": "kaas.pdp.p360-claims-dop-metrics-volumes-volumes-output",
    "connect.cassandra.contact.points": "apvrd22444.uhc.com,apvrd22443.uhc.com,apvrd22445.uhc.com",
    "connect.cassandra.port": "9042",
    "connect.cassandra.username": "pdp",
    "connect.cassandra.password": ”<password>",
    "connect.cassandra.key.space": "claims_streams",
    "connect.cassandra.kcql": "INSERT INTO claim_tin_dop_metrics SELECT tin AS provider_tin,ownedOrAffiliated AS owned_affiliated,claimTypeCode AS
claim_type_code,lineOfBusiness AS line_of_business,submissionType AS submission_type,claimNetworkType AS claim_in_out_network_type,claimSystemId AS
claim_system_id,fundingTypeCode AS funding_type_code,dateOfProcess AS date_of_process,hcoName AS hco_name,claimsApproved AS
approved_count,claimsDenied AS denied_count FROM kaas.pdp.p360-claims-dop-metrics-volumes-volumes-output;",
    "value.converter.schema.registry.url": "http://kaas-test-schema-registry-a.optum.com",
    "value.converter": "io.confluent.connect.avro.AvroConverter",
    "key.converter.schema.registry.url": "http://kaas-test-schema-registry-a.optum.com",
    "key.converter": "io.confluent.connect.avro.AvroConverter",
    "connect.cassandra.default.value": "UNSET",
    "connect.cassandra.ssl.enabled": "true",
    "connect.cassandra.trust.store.password": ”<truststore_password>",
    "connect.cassandra.trust.store.path": "/mnt/pdp-cassandra-certs/cassandra.truststore.jks",
    "connect.cassandra.key.store.password": ”<keystore_password>",
    "connect.cassandra.key.store.path": "/mnt/pdp-cassandra-certs/cassandra.keystore.jks",
    "connect.cassandra.ssl.client.cert.auth": true
    }
}
```

Optum

Kafka Connect: Converters

Purpose: Translate the data between source/sink systems and Kafka connector

- Converter: The format conversion while writing to/reading from Kafka.
  - AVROConverter, JSONConverter, StringConverter
- Transform: Simple/lightweight conversions to be applied on each individual message.
  - Cast, Drop, Flatten, InsertField, ValueToKey
- Plugin: Implementation classes which define the execution behavior of Connectors.

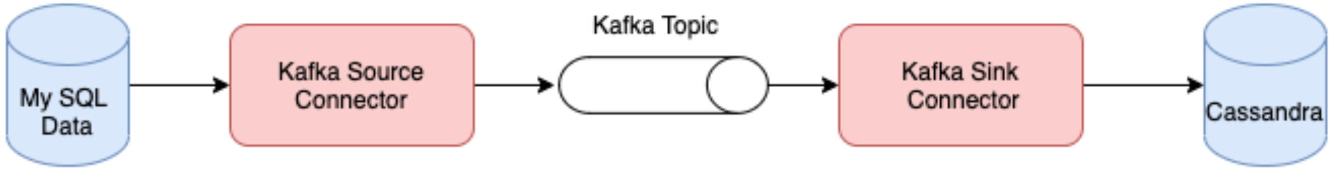
Kafka connect - REST APIs

- GET http://localhost:8085/connectors
- GET http://localhost:8085/connector-plugins
- POST http://localhost:8085/connectors
- GET http://localhost:8085/connectors/<connector_name>/status
- PUT http://localhost:/connectors/<connector_name>/<pause_resume_restart>
- DELETE http://localhost:8085/connectors/<connector_name>

Optum

Lab – Scenarios

- Lab 1 – Setup
  Internal topics setup
  docker-compose to setup kafka environment/database for source and sink
- Lab 2 – Kafka Connect Source
  Kafka Source Connector setup
  Verify data in the kafka topic
- Lab 3 – Kafka Connect Sink
  Kafka Sink Connector Setup
  Verify data in the Cassandra Database



Optum

Kafka Connect @ P360 / OptumCare

- OptumCare
- https://github.optum.com/OCNP/OCNP-caredata-kconnect-source-facets
- https://github.optum.com/OCNP/OCNP-caredata-kconnect-sink-facets

- Open source Kafka connector:
  - https://github.optum.com/EnterpriseProviderPlatform/stream-reactor/tree/master/kafka-connect-cassandra
- Deployment Image Creation:
  - https://github.optum.com/EnterpriseProviderPlatform/pdp-kconnect-rdbms-cassandra
- Example:
  - https://github.optum.com/EnterpriseProviderPlatform/network-measures-kconnect-s3-sink-lab/tree/master/network-measures-kconnect-s3-sink-lab-chart

Transformations (SMTs)

Purpose: Single Message transformations on individual messages.

Cast: cast to a specific type

- "transforms": "Cast", "transforms.Cast.type": "org.apache.kafka.connect.transforms.Cast$Value", "transforms.Cast.spec": "ID:string,score:float64"

ExtractField: Extract a specified field when not null

- "transforms": "extractField", "transforms.ExtractField.type":"org.apache.kafka.connect.transforms.ExtractField$Key", "transforms.ExtractField.field":"id"

ExtractTopic: Replace topic with a new topic derived from its key/value

- "transforms": "keyFieldExample", transforms.KeyFieldExample.type=io.confluent.connect.transforms.ExtractTopic$Value transforms.KeyFieldExample.field=f3 transforms.KeyFieldExample.skip.missing.or.null=tru

Flatten: Flatten a nested data structure

- "transforms": "flatten", "transforms.flatten.type": "org.apache.kafka.connect.transforms.Flatten$Value", "transforms.flatten.delimiter": "."

ReplaceField: Filter/replace fields

- "transforms": "ReplaceField", "transforms.ReplaceField.type": "org.apache.kafka.connect.transforms.ReplaceField$Value", "transforms.ReplaceField.blacklist": "c2"

TimestampConverter:

- "transforms": "keyFieldExample", transforms.KeyFieldExample.type=io.confluent.connect.transforms.ExtractTopic$Value transforms.KeyFieldExample.field=f3 transforms.KeyFieldExample.skip.missing.or.null=true

ValueToKey:

- "transforms": "ValueToKey", "transforms.ValueToKey.type":"org.apache.kafka.connect.transforms.ValueToKey", "transforms.ValueToKey.fields":"userId,city,state"

Optum

Error Handling:

How: Create Dead letter queue with below parameters

```
"errors.tolerance": "all",
  "errors.deadletterqueue.topic.name":"dlq_topic_name",
  "errors.deadletterqueue.topic.replication.factor": 1
```

```
errors.deadletterqueue.context.headers.enable = true
```

Q/A

Thank you

Feedback Link

https://www.metricsthatmatter.com/url/u.aspx?74E258F91202202062