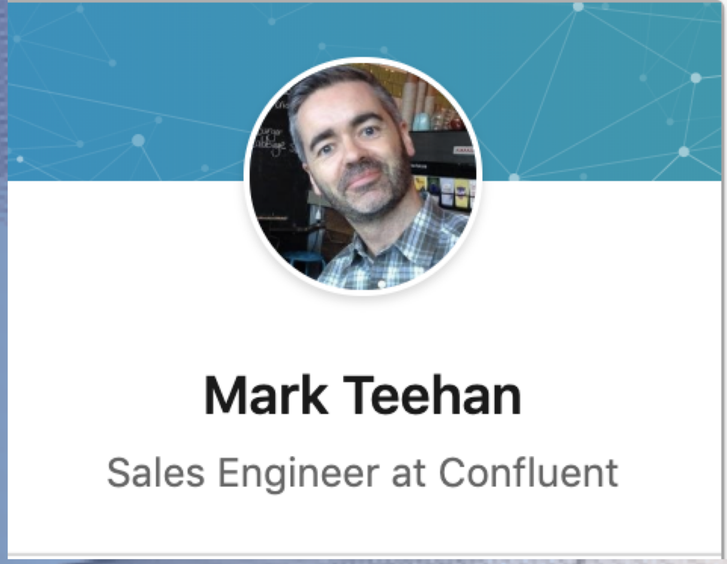
kafka KRU



SEOUL - 18. OCT. 2019

From Zero to Hero With Kafka Connect

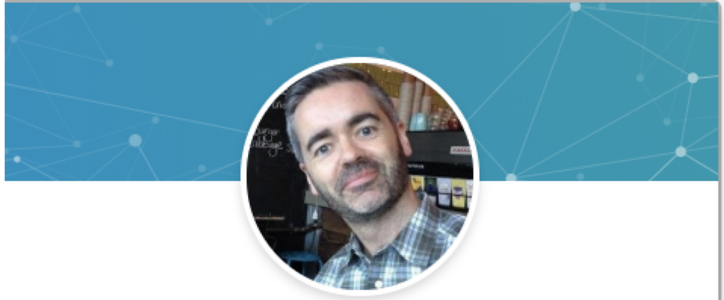
Mark Teehan
Sales Engineer
Confluent

Agenda

- What is Kafka Connect
- Configuring Kafka Connect
- Deployment
- JDBC Connectors

Duration: ~40m

Author Credit: Robin Moffatt



Mark Teehan

Sales Engineer at Confluent

Confluent: 18 months
Singapore: 20 years

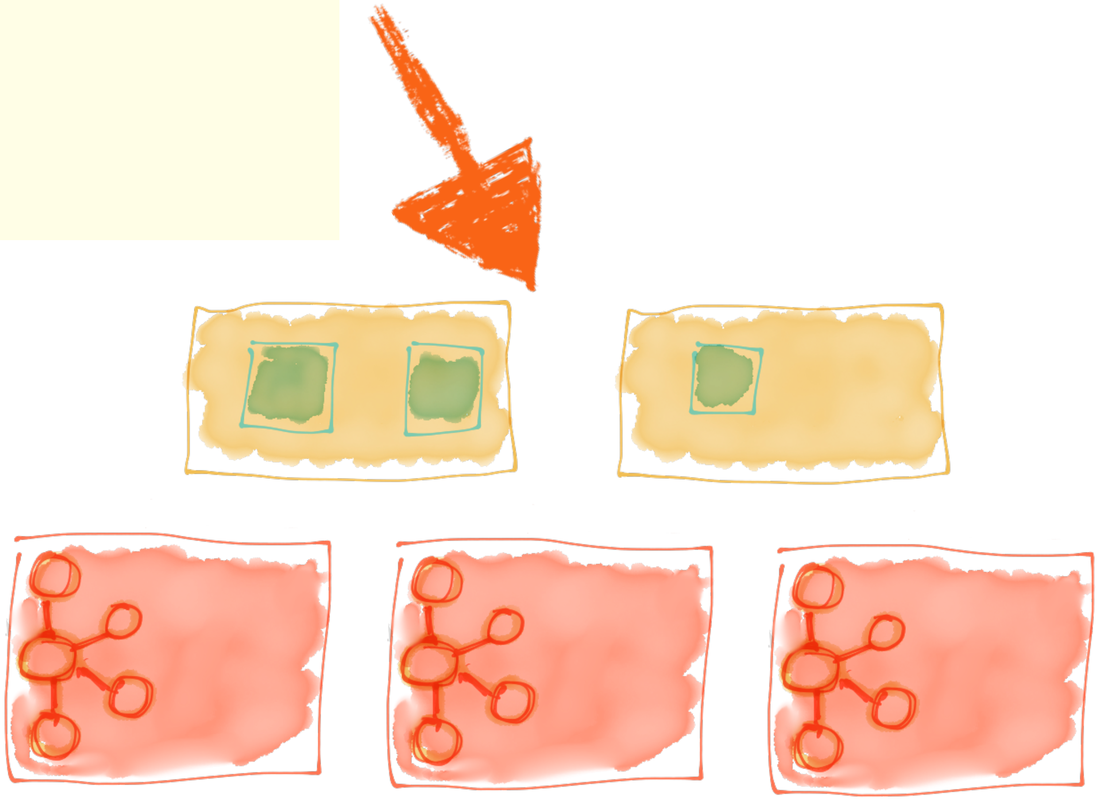
Before that:
SAP HANA
Oracle DBA

confluent

What is Kafka Connect?

Streaming Integration with Kafka Connect

Sources

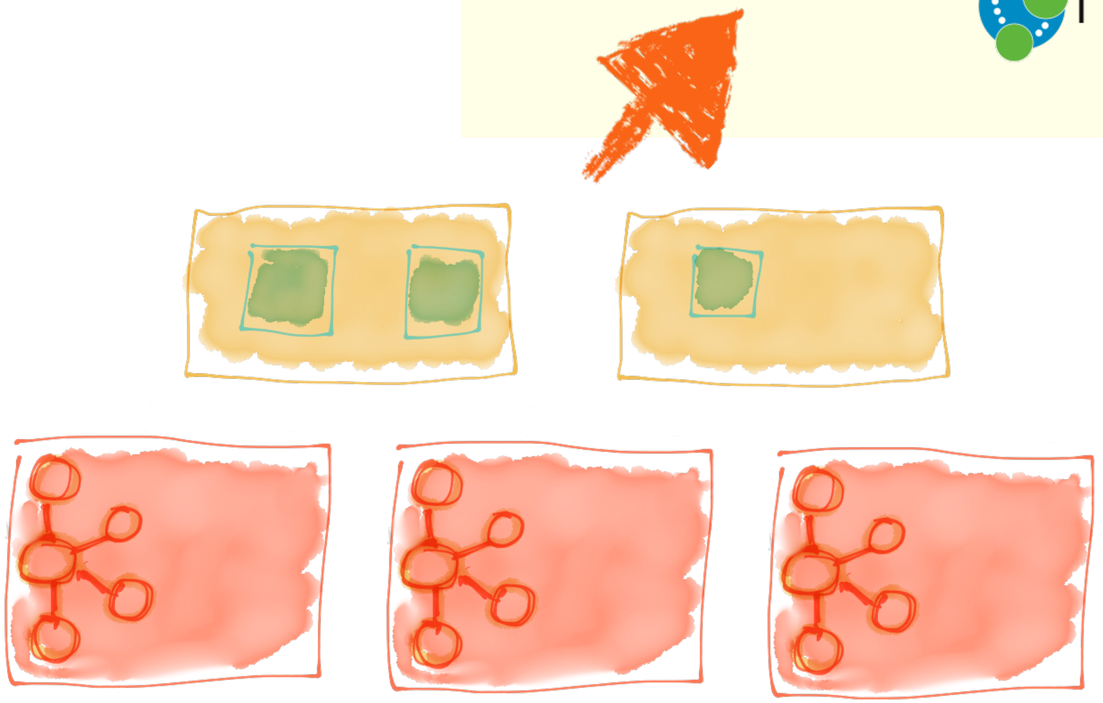


Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect

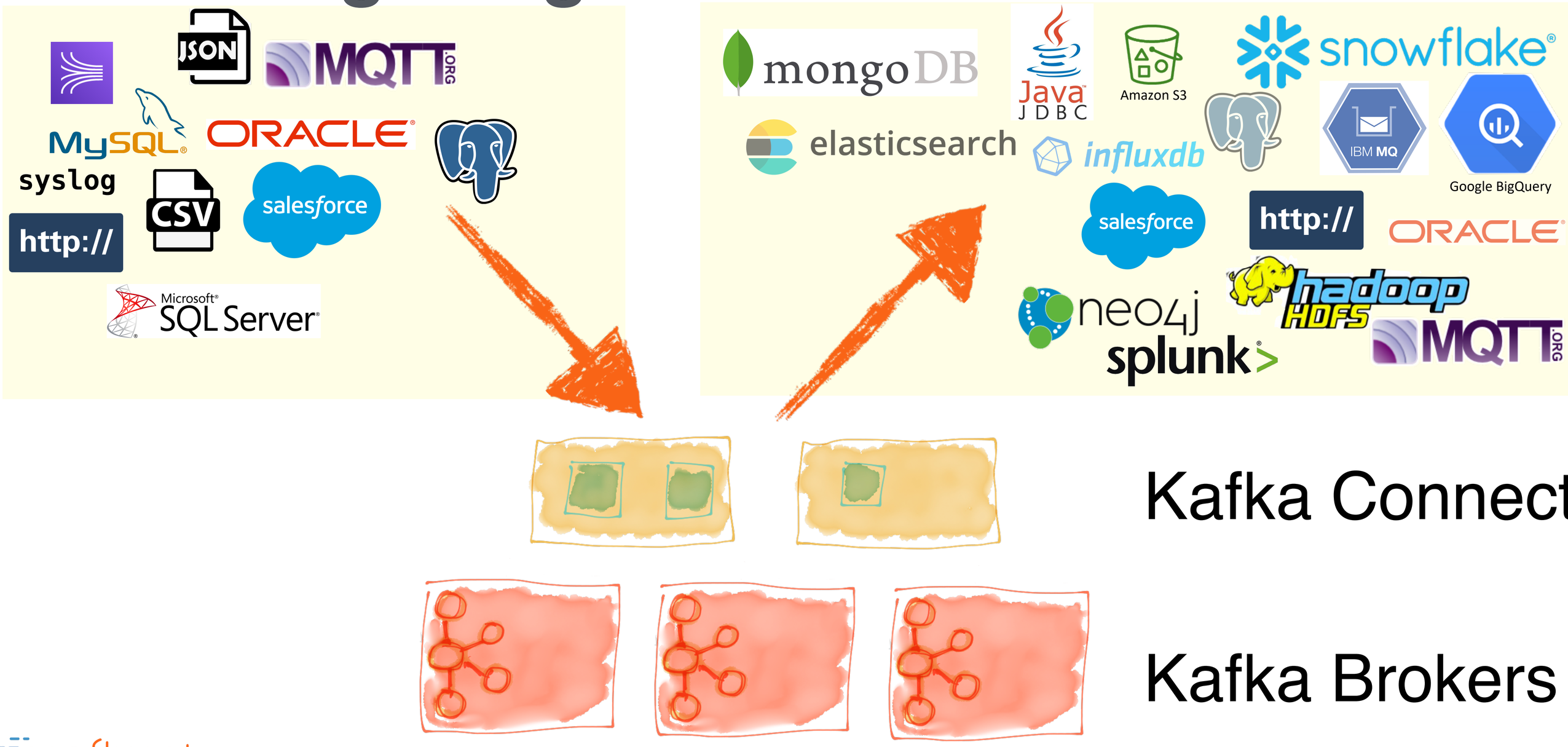
Sinks



Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect

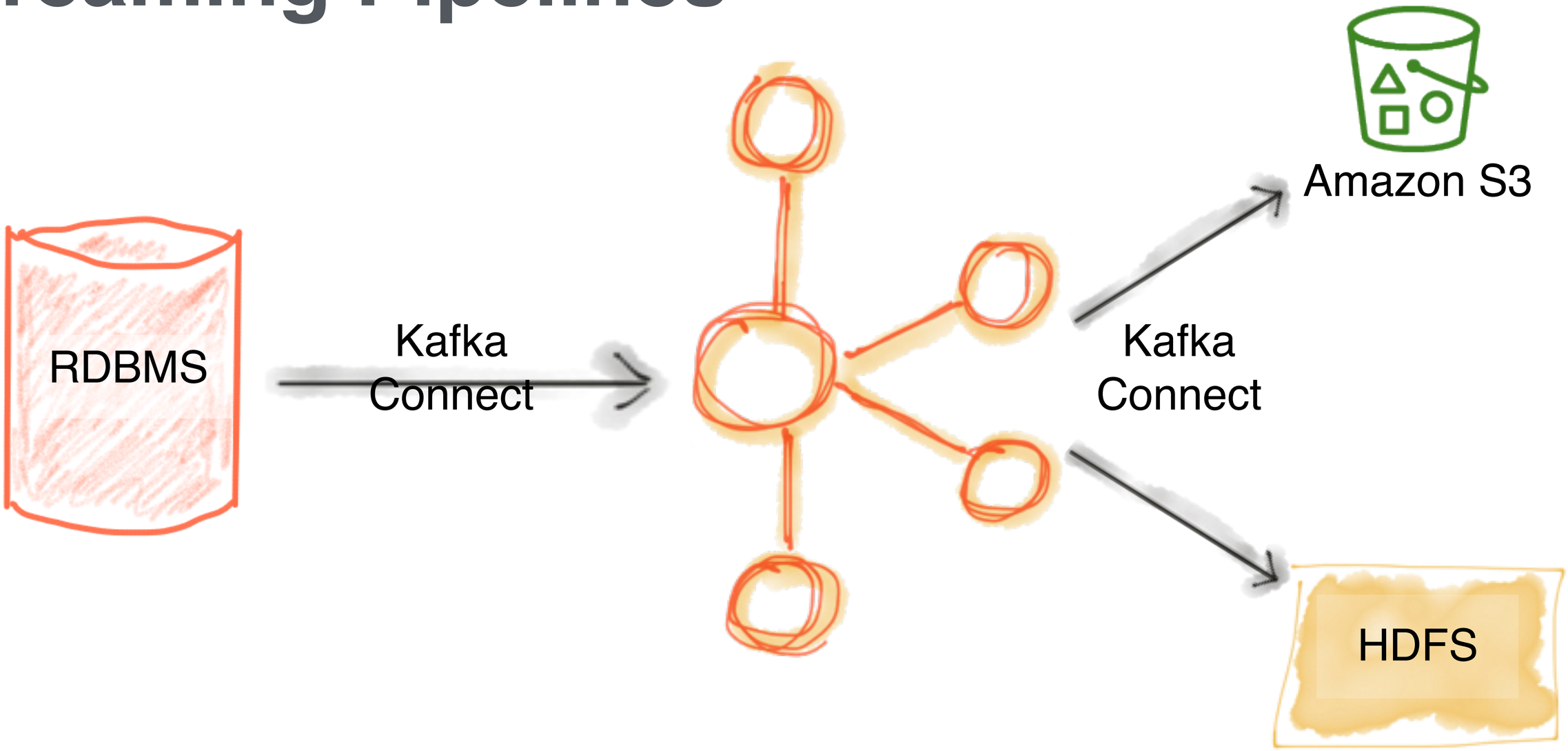


Look Ma, No Code!

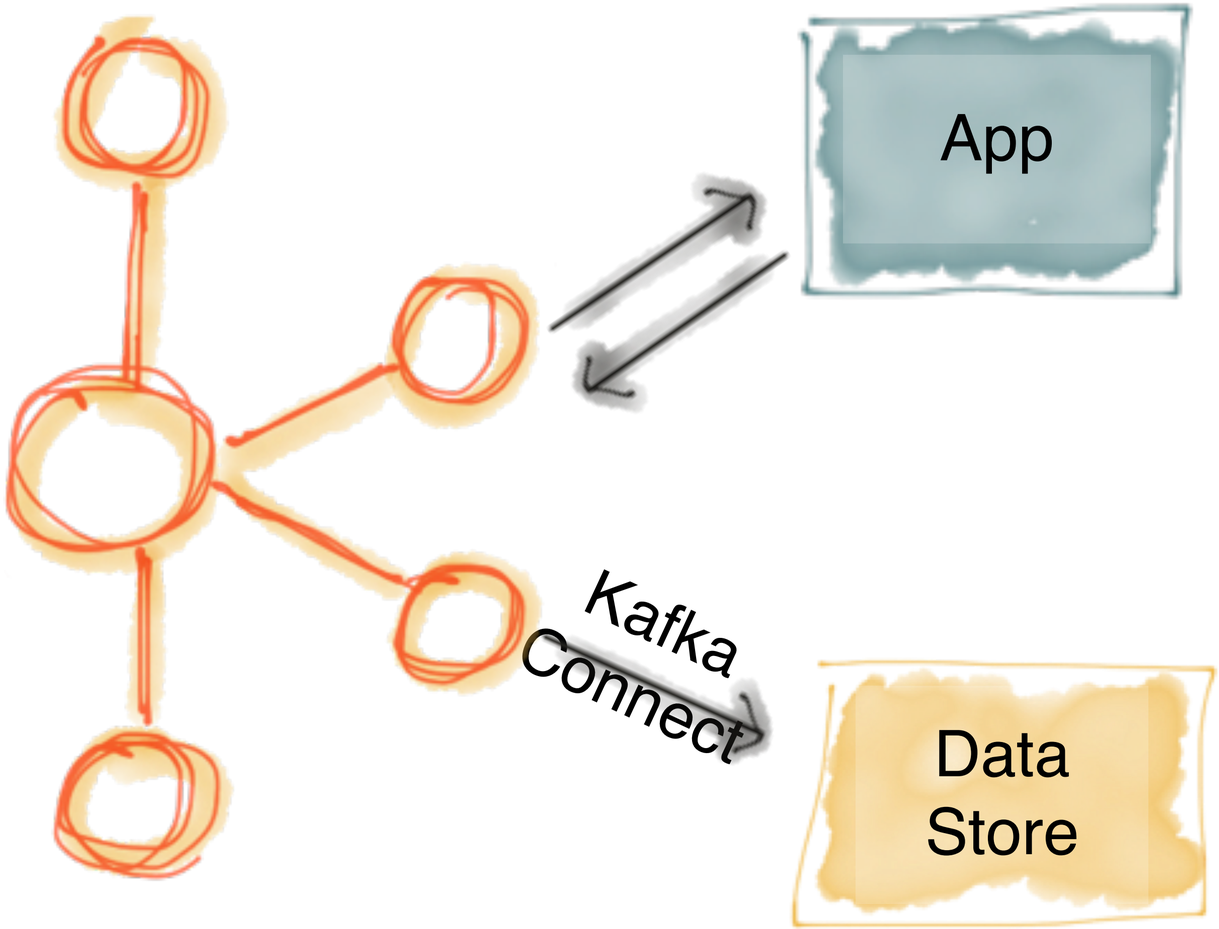
```
{
  "connector.class":
    "io.confluent.connect.jdbc.JdbcSourceConnector",
  "connection.url":
    "jdbc:mysql://asgard:3306/demo",
  "table.whitelist":
    "sales,orders,customers"
}
```

https://docs.confluent.io/current/connect/

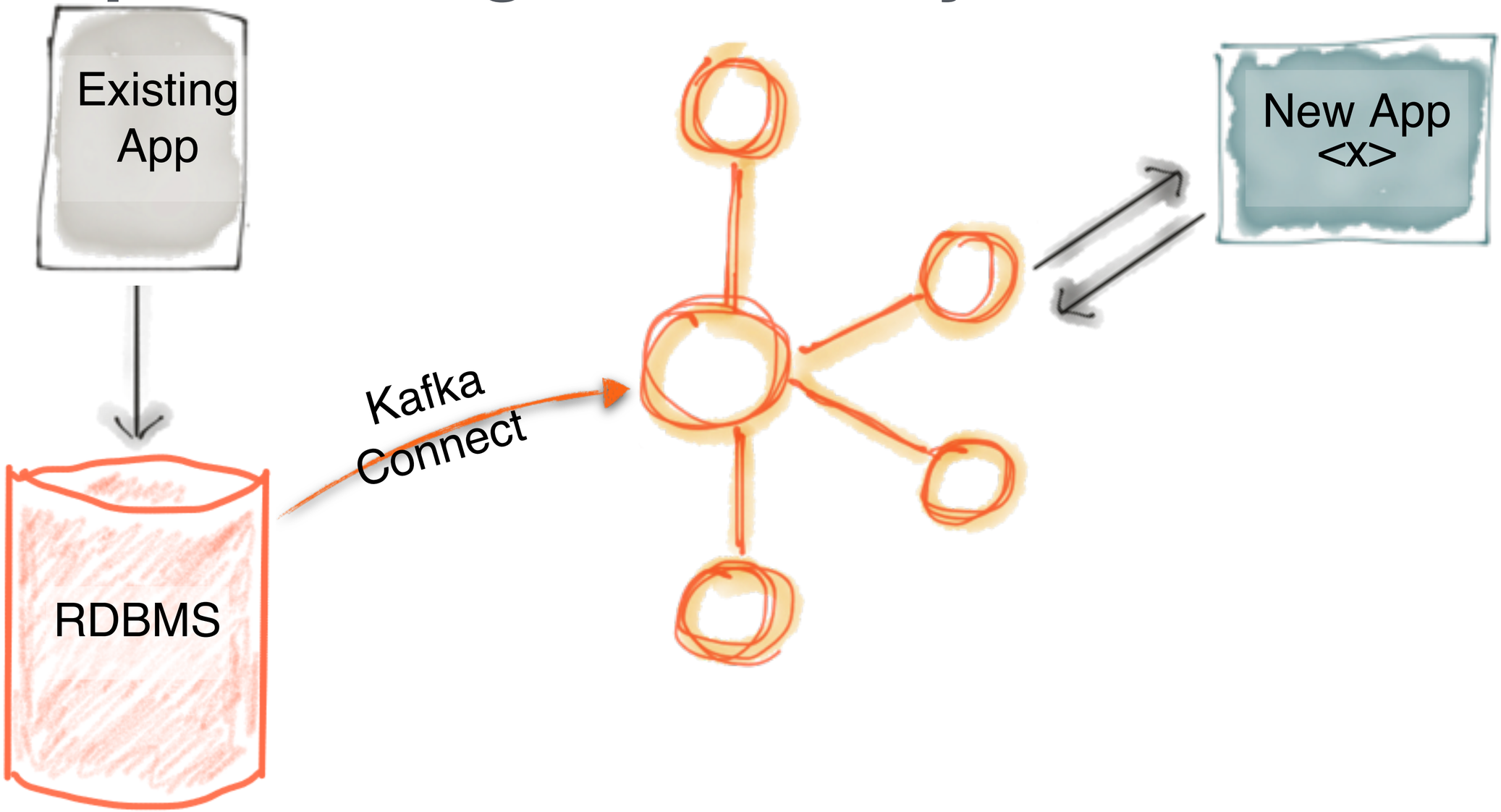
Streaming Pipelines



Writing to data stores from Kafka



Evolve processing from old systems to new

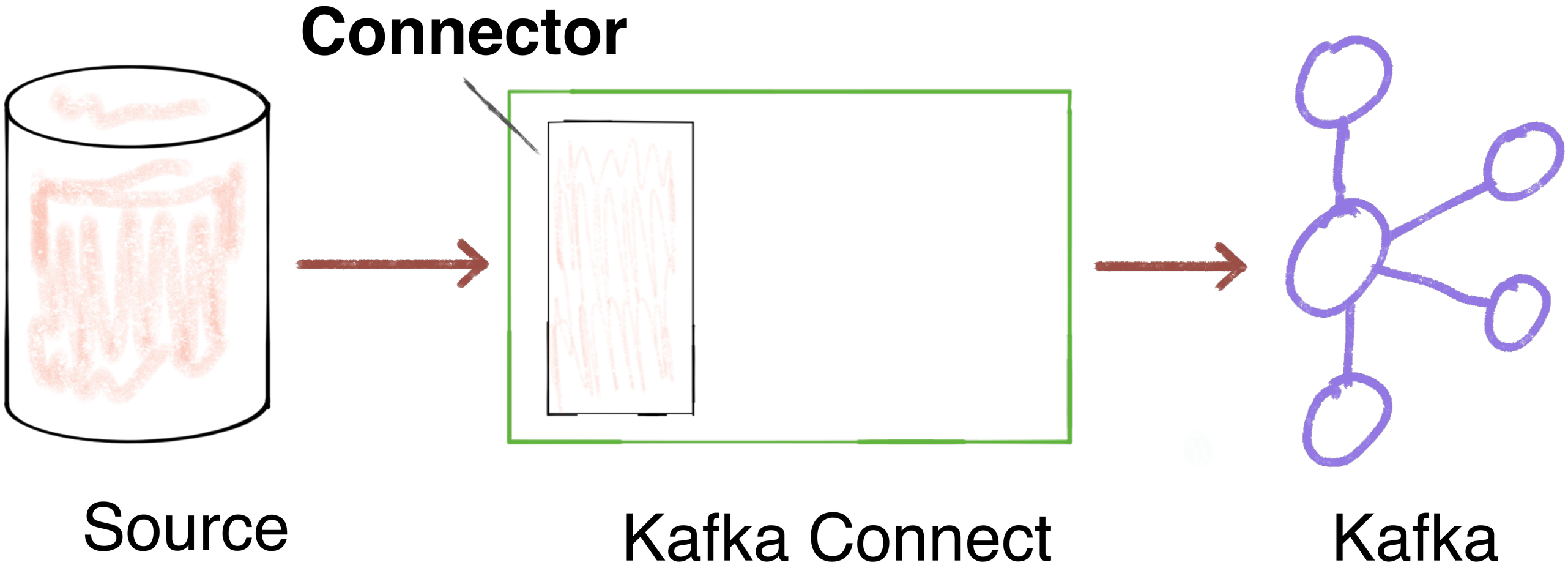


Configuring Kafka Connect

Inside the API - connectors, transforms, converters

Kafka Connect basics

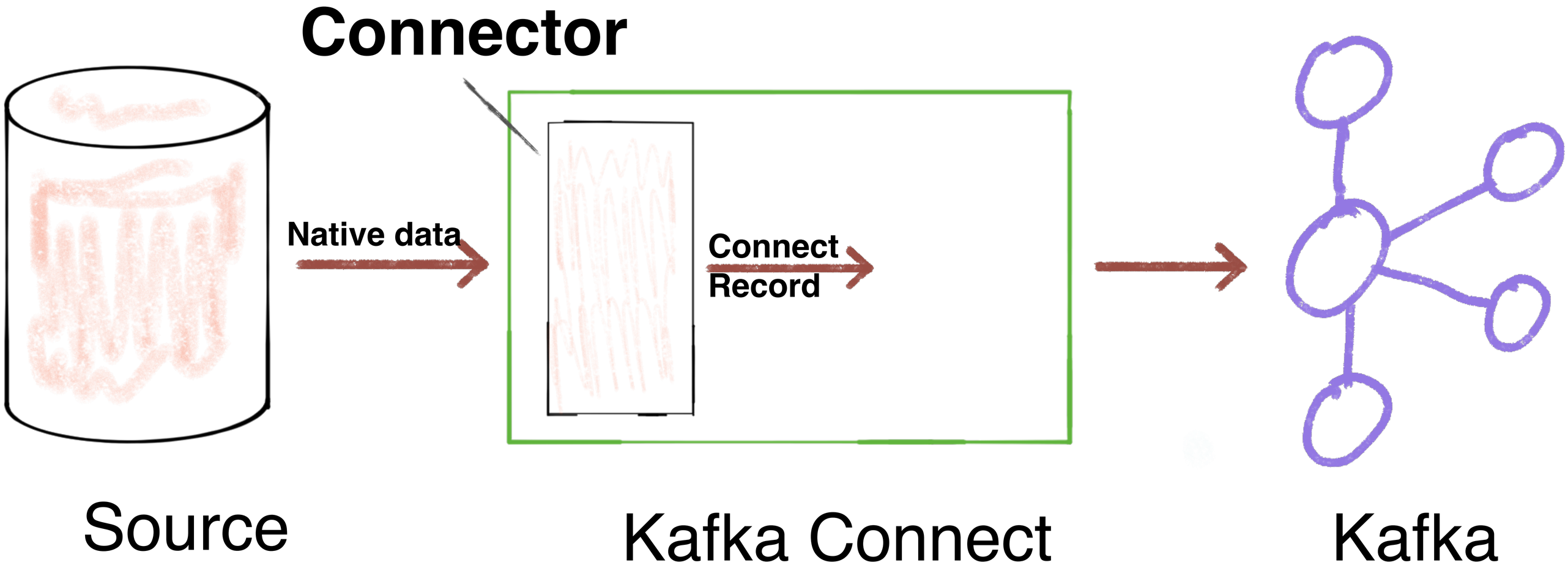
Connectors



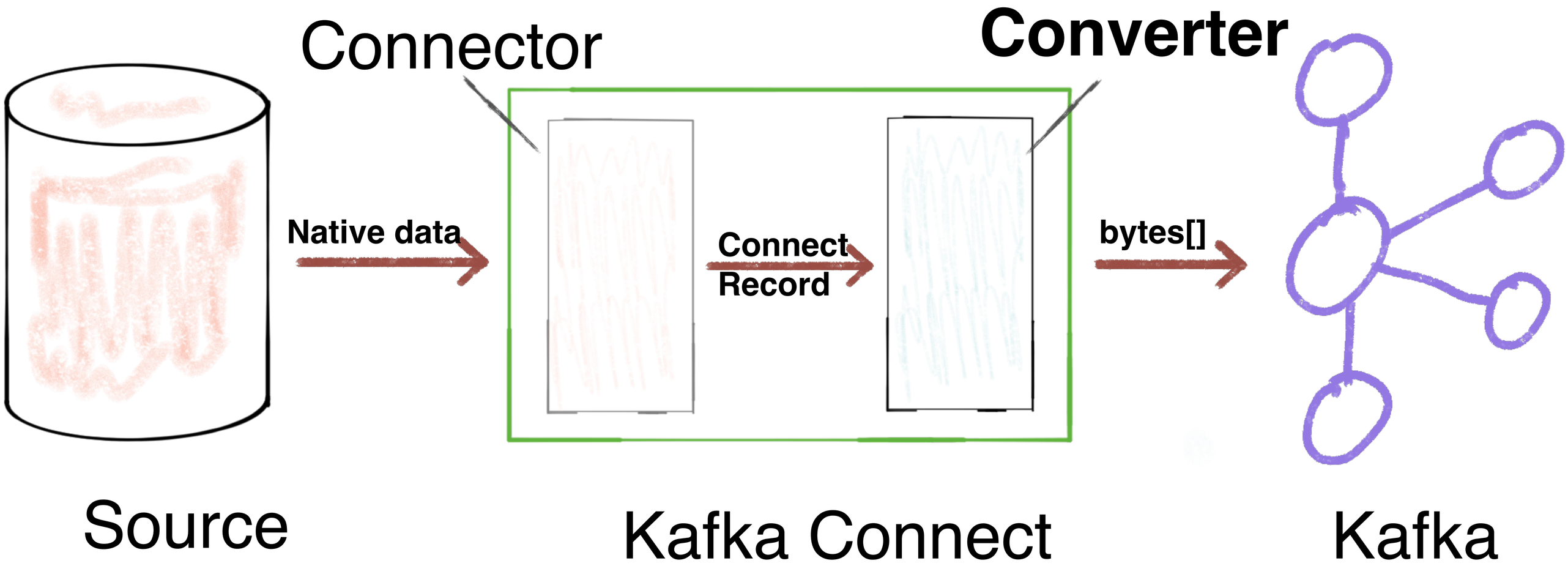
Connectors

```
"config": {
  [...]
  "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
  "connection.url": "jdbc:postgresql://postgres:5432/",
  "topics": "asgard.demo.orders",
}
```

Connectors


Connector
Native data
Connect
Record
Source
Kafka Connect
Kafka

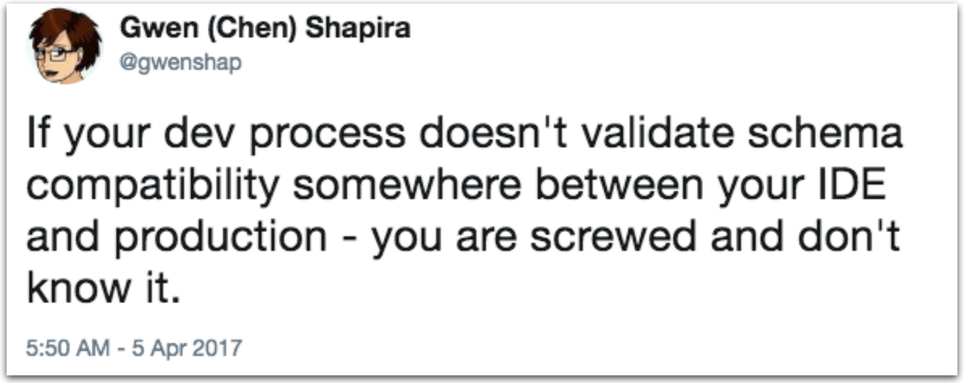
Converters


Connector
Converter
Native data
Connect
Record
bytes[]
Source
Kafka Connect
Kafka

Serialisation & Schemas

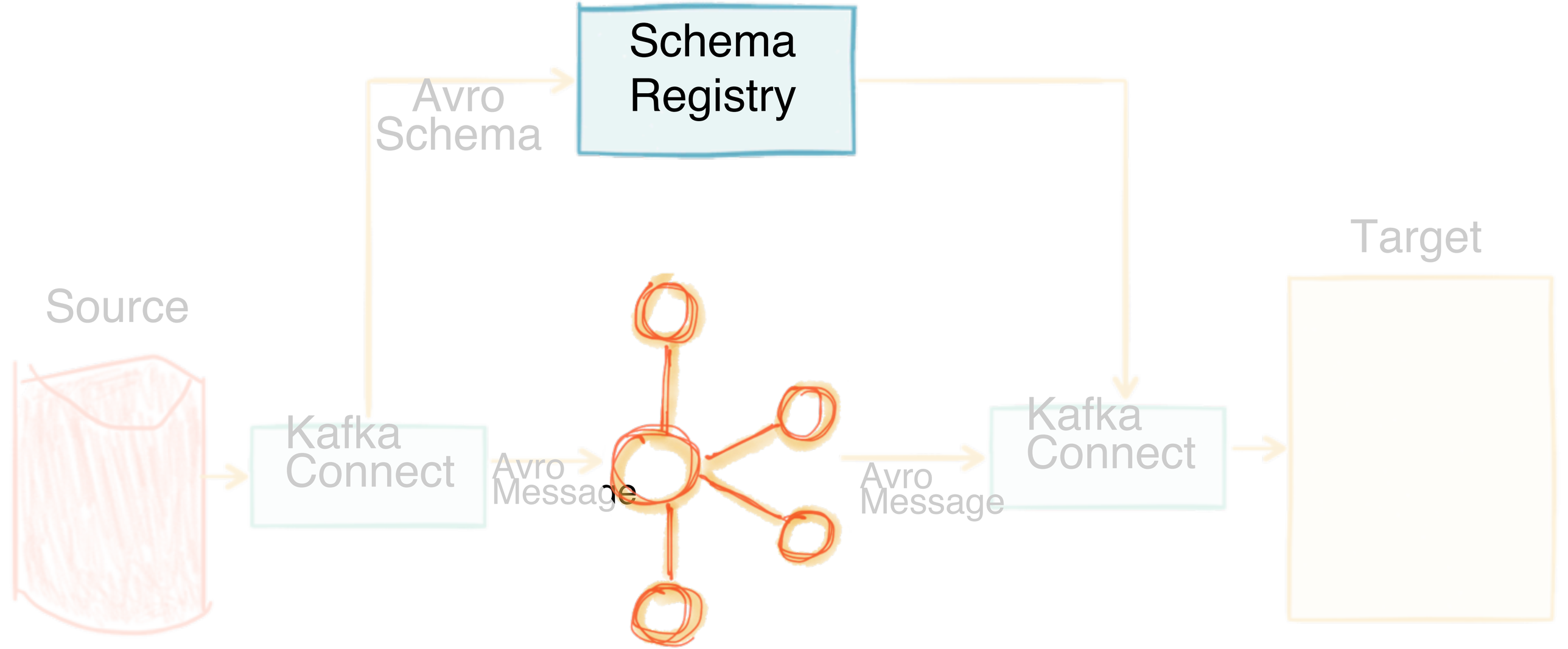
Avro Protobuf JSON CSV

-> Confluent Schema Registry

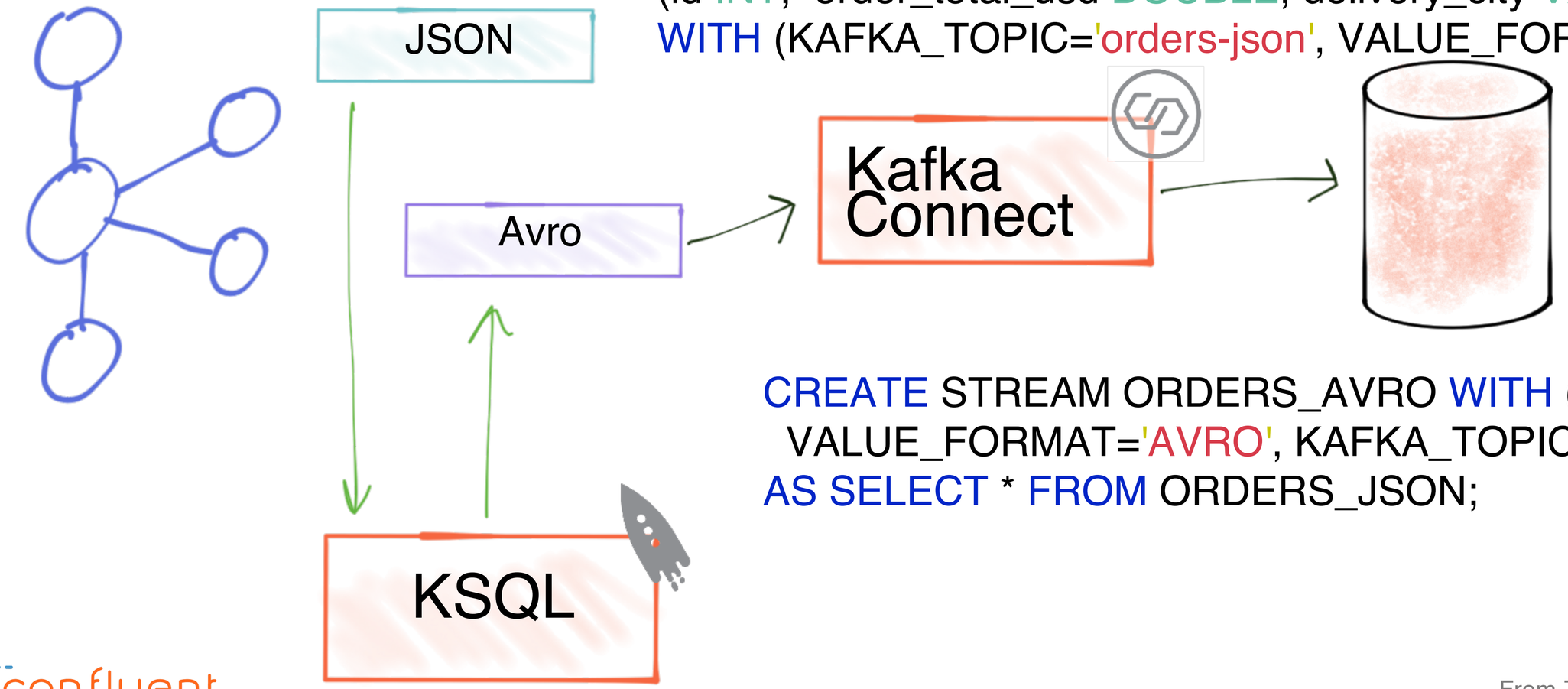


https://qconnewyork.com/system/files/presentation-slides/qcon_17_-_schemas_and_apis.pdf

The Confluent Schema Registry



Using KSQL to apply schema to your data




```
CREATE STREAM ORDERS_JSON
  (id INT,  order_total_usd DOUBLE, delivery_city VARCHAR)
  WITH (KAFKA_TOPIC='orders-json', VALUE_FORMAT='JSON');
```

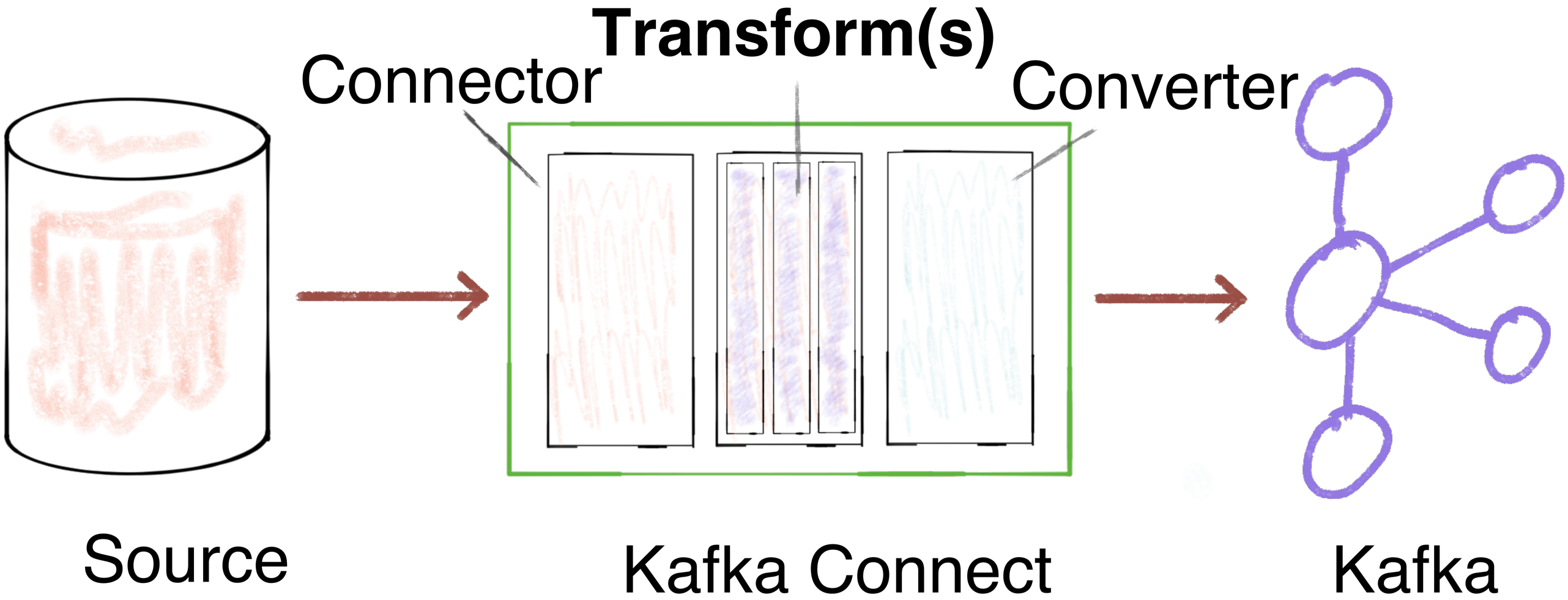
```
CREATE STREAM ORDERS_AVRO WITH (
  VALUE_FORMAT='AVRO', KAFKA_TOPIC='orders-avro')
AS SELECT * FROM ORDERS_JSON;
```

Converters

```
key.converter=io.confluent.connect.avro.AvroConverter
key.converter.schema.registry.url=http://localhost:8081
value.converter=io.confluent.connect.avro.AvroConverter
value.converter.schema.registry.url=http://localhost:8081
```

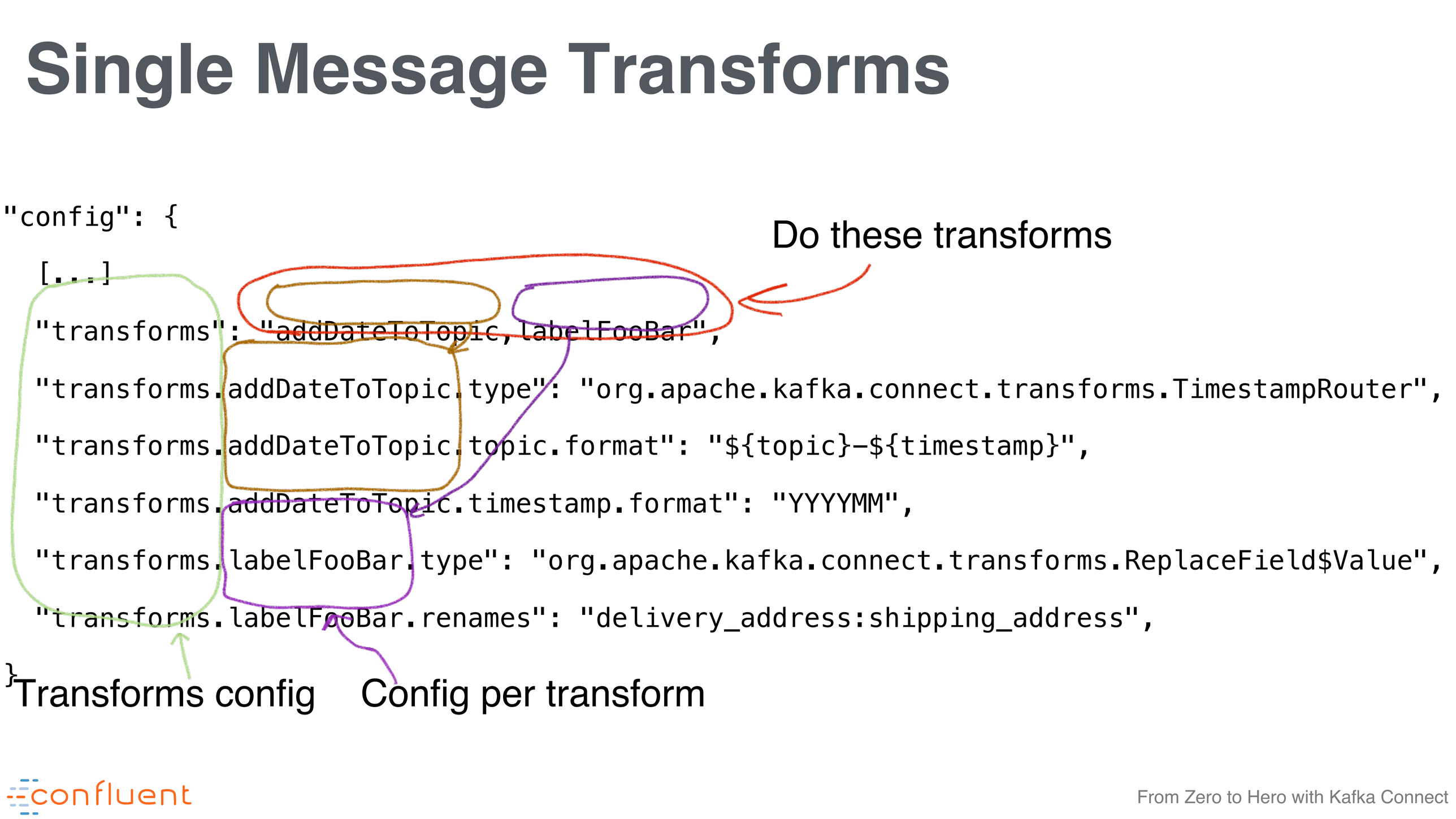
Set as a global default per-worker; optionally can be overriden per-connector

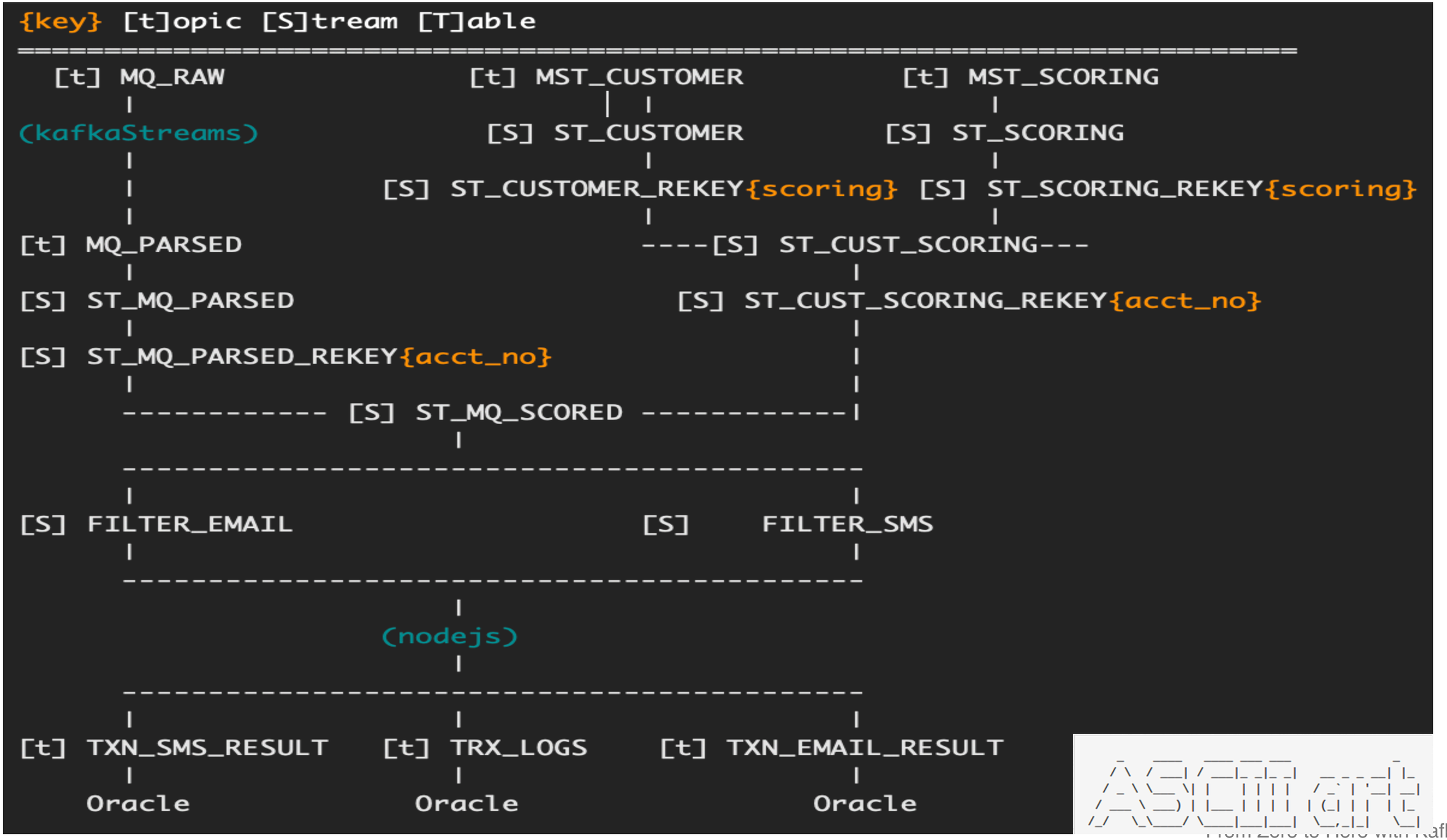
Single Message Transforms



Single Message Transforms

```
"config": {
  [...]
  "transforms": "addDateToTopic,labelFooBar",
  "transforms.addDateToTopic.type": "org.apache.kafka.connect.transforms.TimestampRouter",
  "transforms.addDateToTopic.topic.format": "${topic}-${timestamp}",
  "transforms.addDateToTopic.timestamp.format": "YYYYMM",
  "transforms.labelFooBar.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
  "transforms.labelFooBar.renames": "delivery_address:shipping_address",
}
```

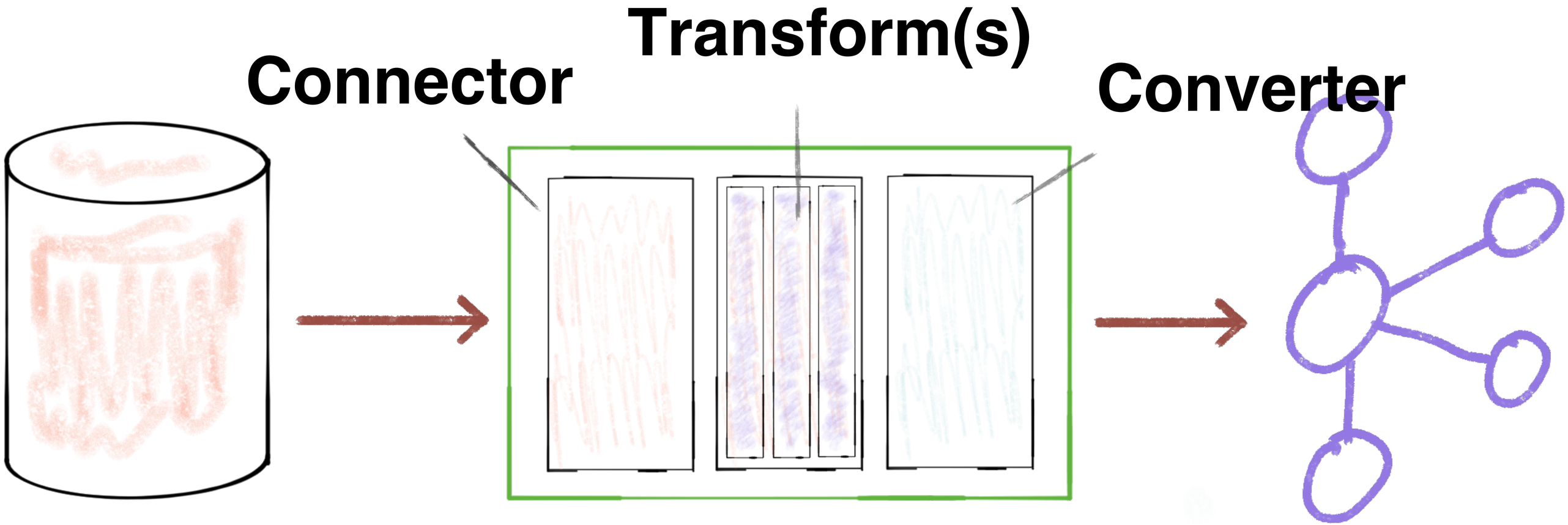



{key} [t]opic [S]tream [T]able
[t] MQ_RAW
[t] MST_CUSTOMER
[t] MST_SCORING
(kafkaStreams)
[S] ST_CUSTOMER
[S] ST_SCORING
[S] ST_CUSTOMER_REKEY{scoring}
[S] ST_SCORING_REKEY{scoring}
[t] MQ_PARSED
[S] ST_CUST_SCORING
[S] ST_MQ_PARSED
[S] ST_CUST_SCORING_REKEY{acct_no}
[S] ST_MQ_PARSED_REKEY{acct_no}
[S] ST_MQ_SCORED
[S] FILTER_EMAIL
[S] FILTER_SMS
(nodejs)
[t] TXN_SMS_RESULT
[t] TRX_LOGS
[t] TXN_EMAIL_RESULT
Oracle
Oracle
Oracle

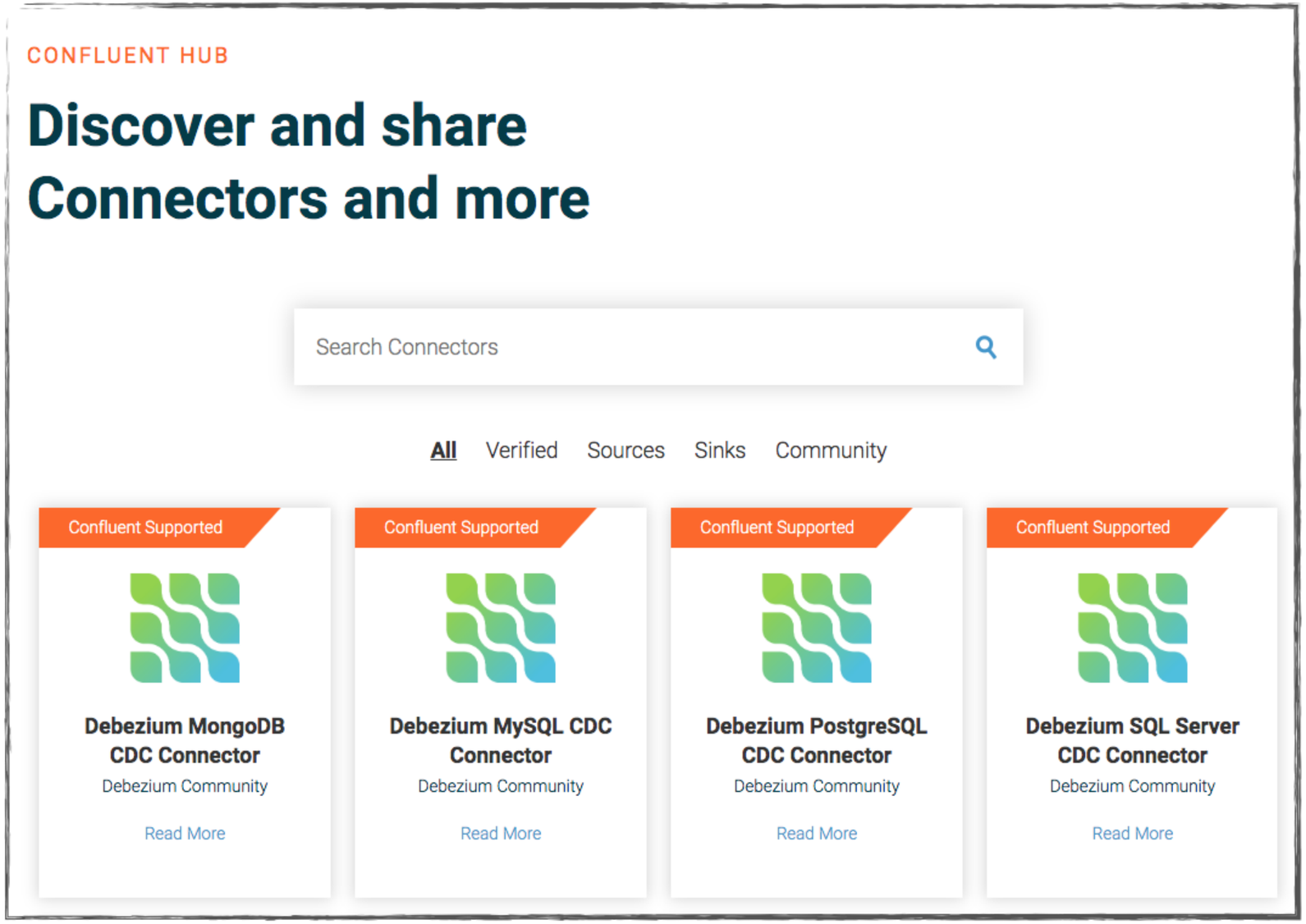
```
                                    "name": "${P}_mq_source_${DT3}",
"config": {
                          "connector.class": "io.confluent.connect.ibm.mq.IbmMQSourceConnector"
                                    "name": "${P}_mq_source_${DT3}"
        ,                         "tasks.max": "1"
        ,                       "kafka.topic": "${P}_MQ_RAW"
        , "confluent.topic.bootstrap.servers": "10.20.215.212:9092"
        ,"confluent.topic.replication.factor": "3"
        ,                       "transforms": "unpack"
        ,          "transforms.unpack.type": "org.apache.kafka.connect.transforms.ExtractField\$Value"
        ,         "transforms.unpack.field": "text"
        ,                    "mq.hostname": "10.20.215.209"
        ,                        "mq.port": "2311"
        ,                "mq.transport.type": "client"
        ,                 "mq.queue.manager": "QMDD9.CFL.01"
        ,                       "mq.channel": "CHLDD.CFL.KFK.CONN"
        ,                    "mq.username": ""
        ,                    "mq.password": ""
        ,              "jms.destination.name": "POC.TRX_LOG.CFL"
        }
```

IBM MQ: *Unpack JMS field “Value” as the Kafka message payload, ignore other JMS fields*

Extensible



Confluent Hub

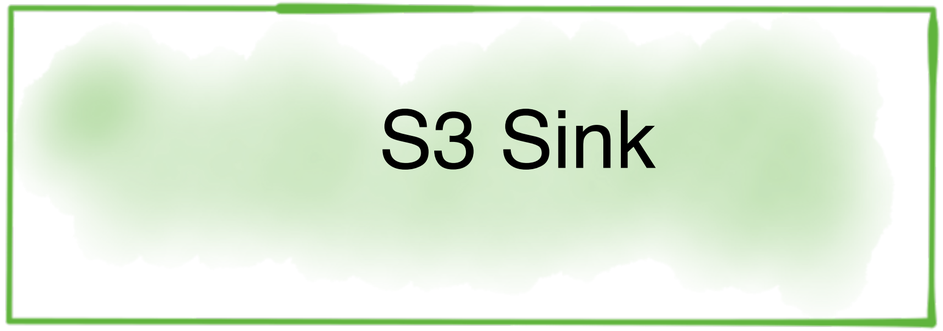


hub.confluent.io

Deploying Kafka Connect

Connectors, Tasks, and Workers

Connectors and Tasks



Connectors and Tasks

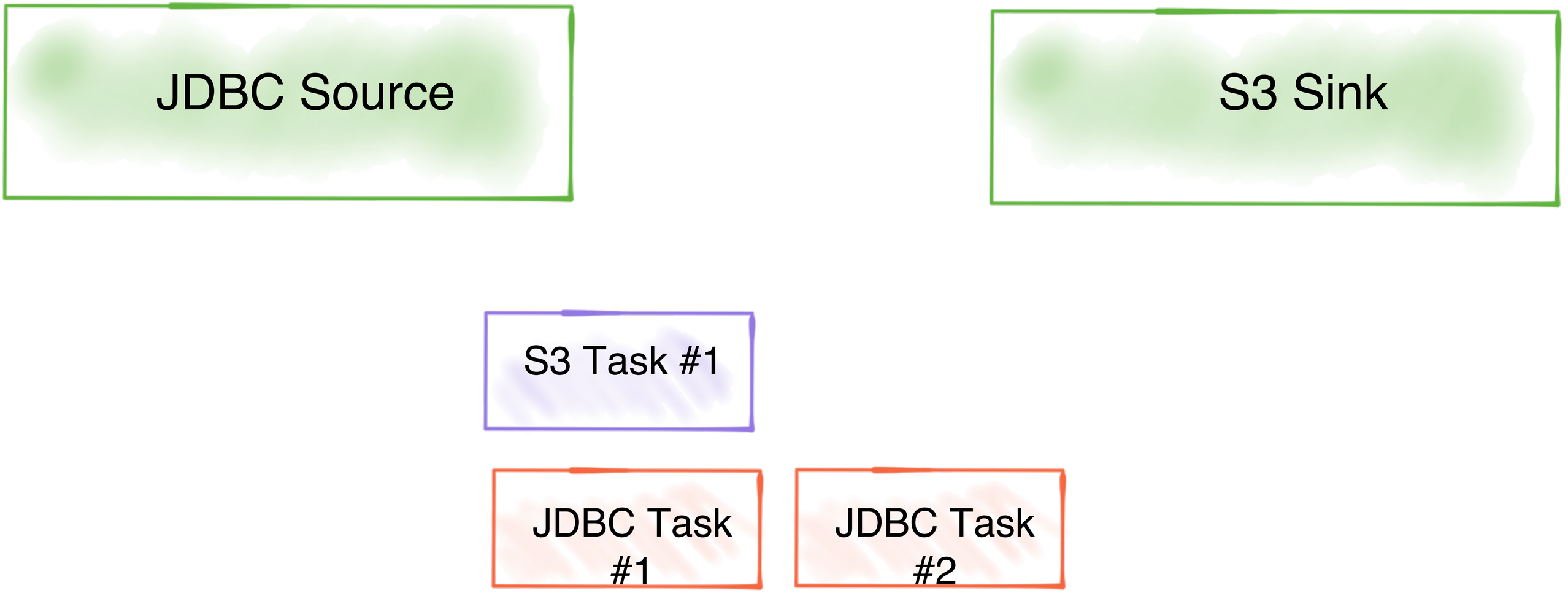
JDBC Source

S3 Sink

S3 Task #1

JDBC Task #1

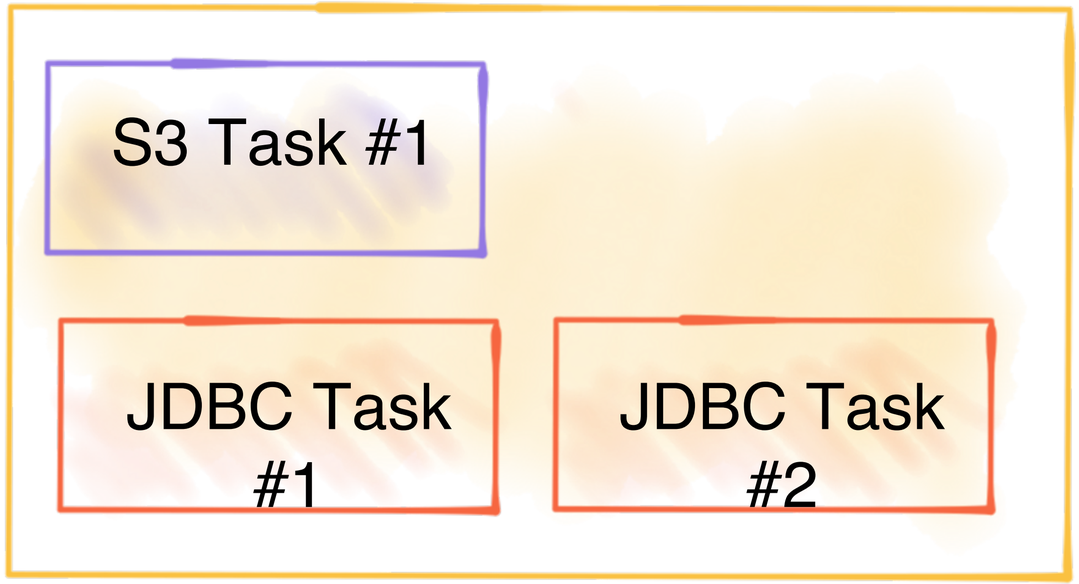
Connectors and Tasks



Tasks and Workers

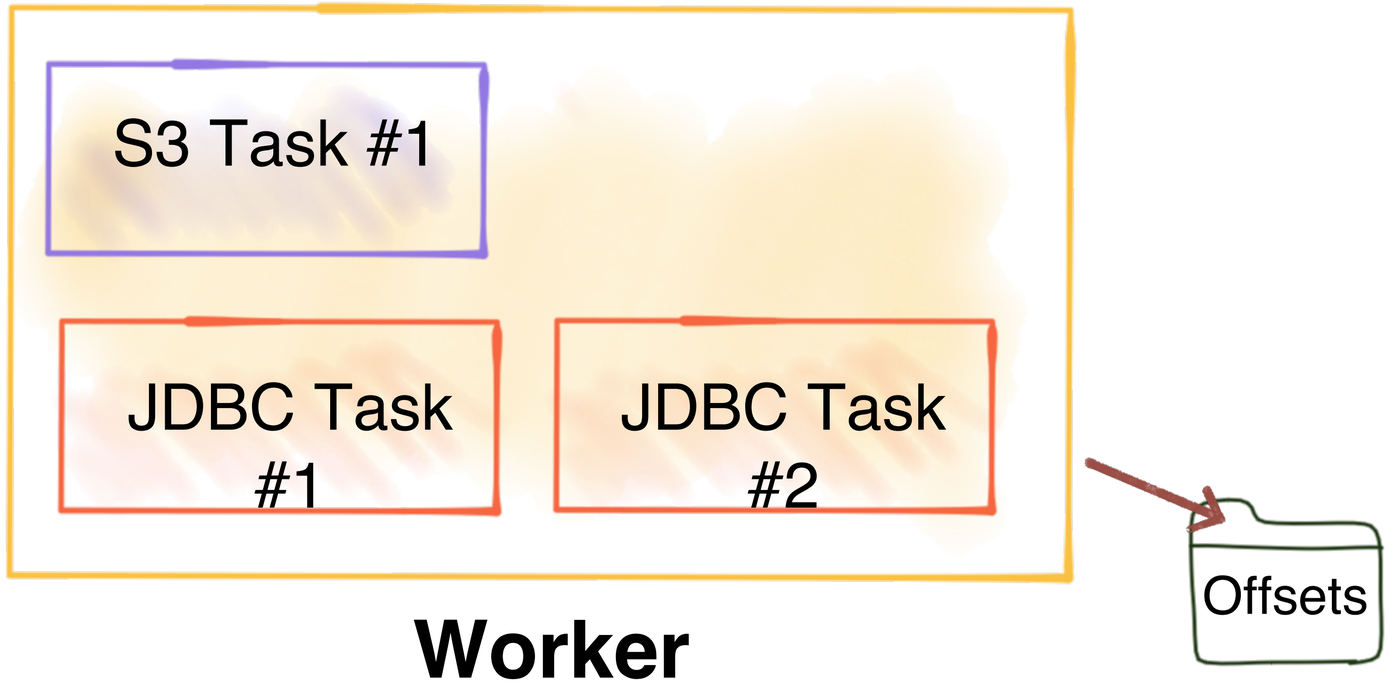
JDBC Source

S3 Sink

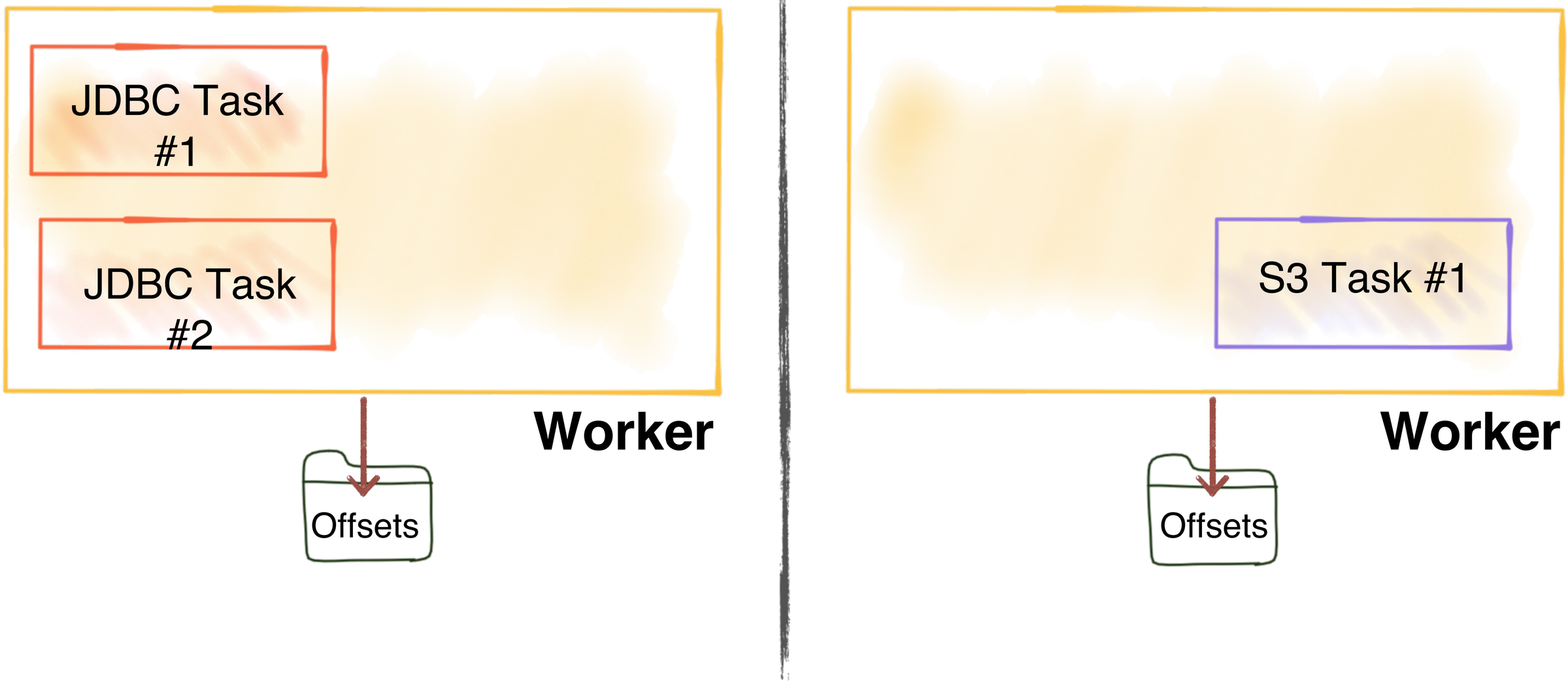


Worker

Kafka Connect **Standalone** Worker

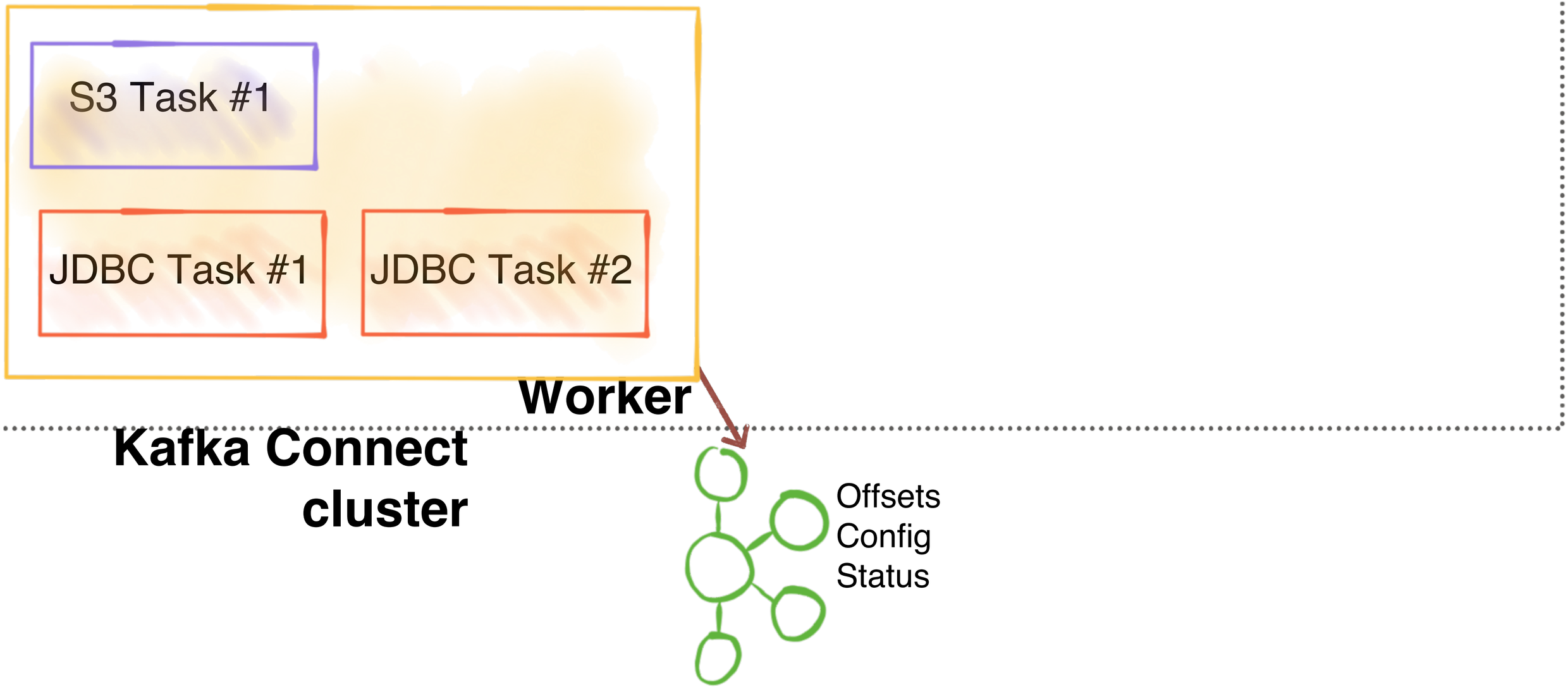


"Scaling" the Standalone Worker



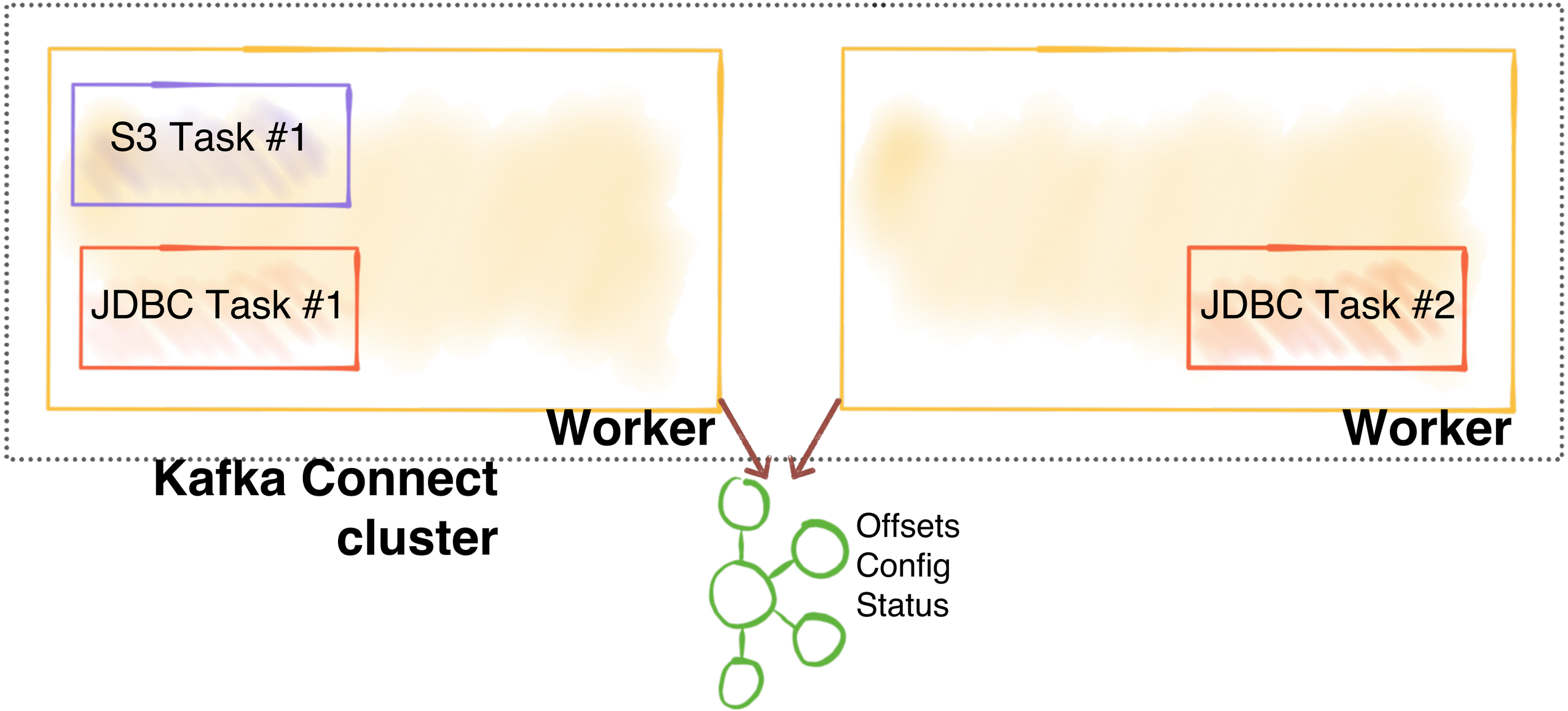
Fault-tolerant? **Nope.**

Kafka Connect **Distributed** Worker



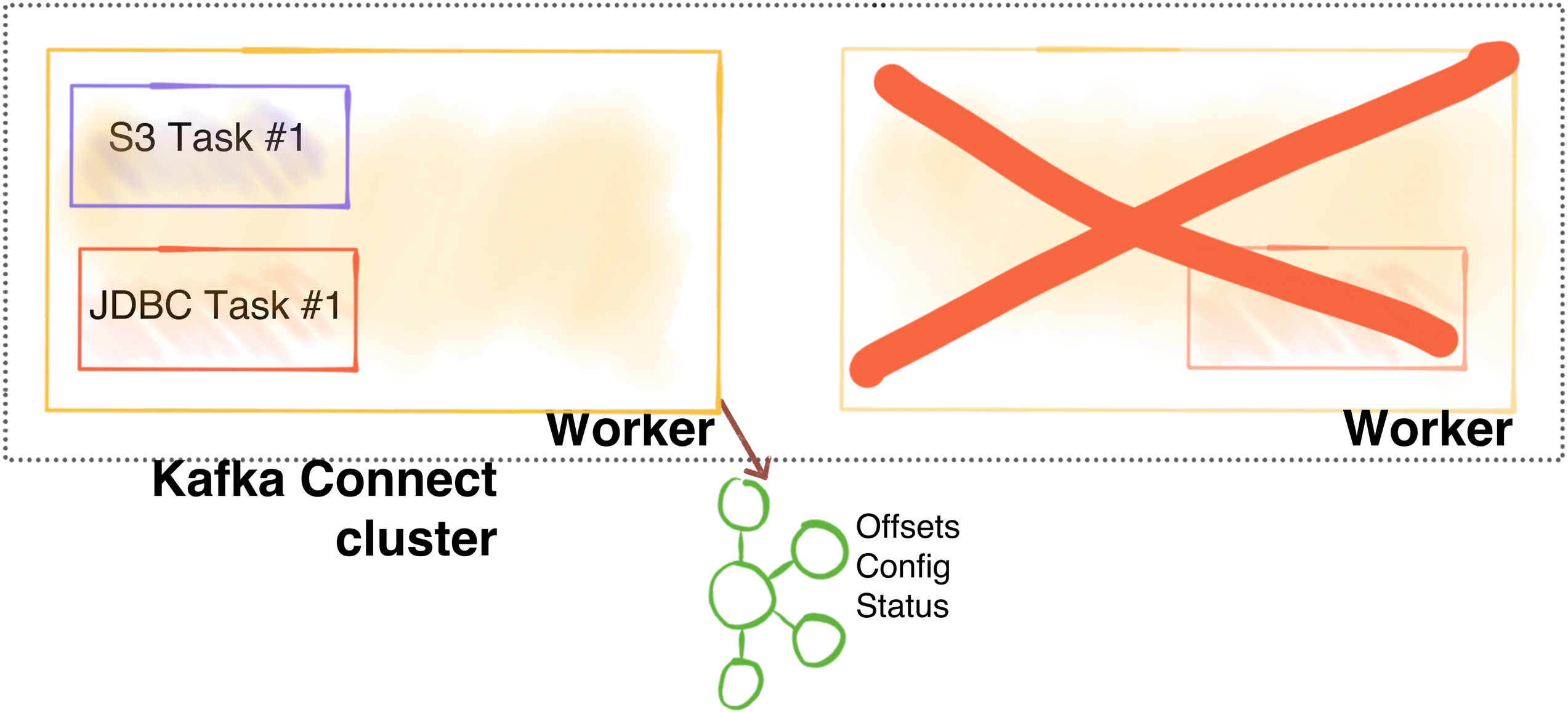
Fault-tolerant? **Yeah!**

Scaling the Distributed Worker

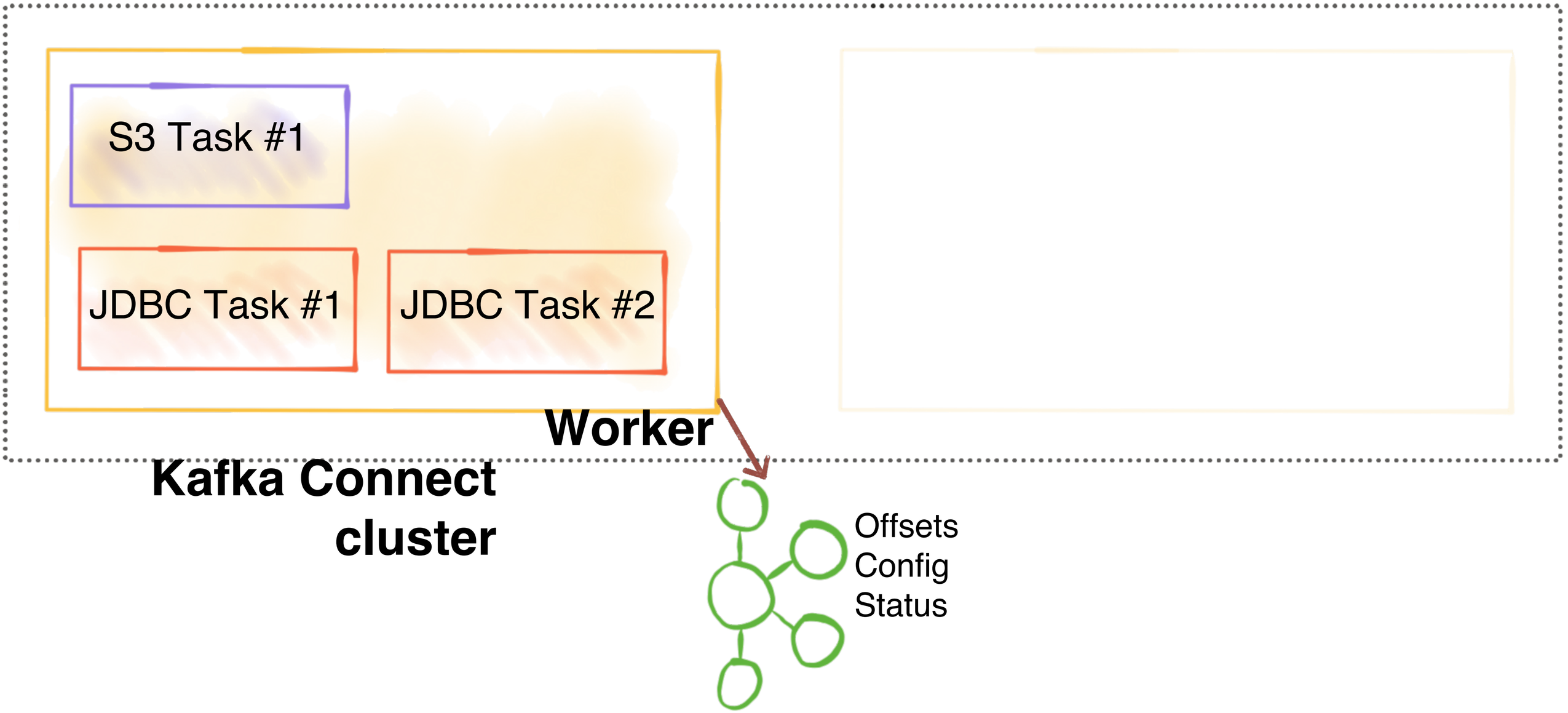


Fault-tolerant? **Yeah!**

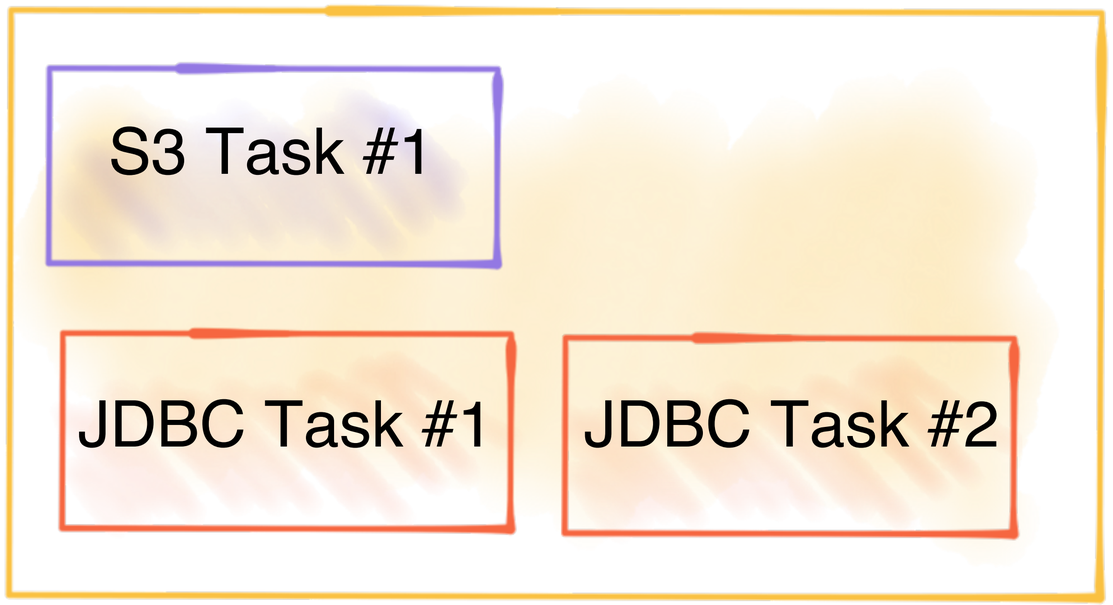
Distributed Worker - fault tolerance



Distributed Worker - fault tolerance



....Kafka 2.3: no more "stop the world"

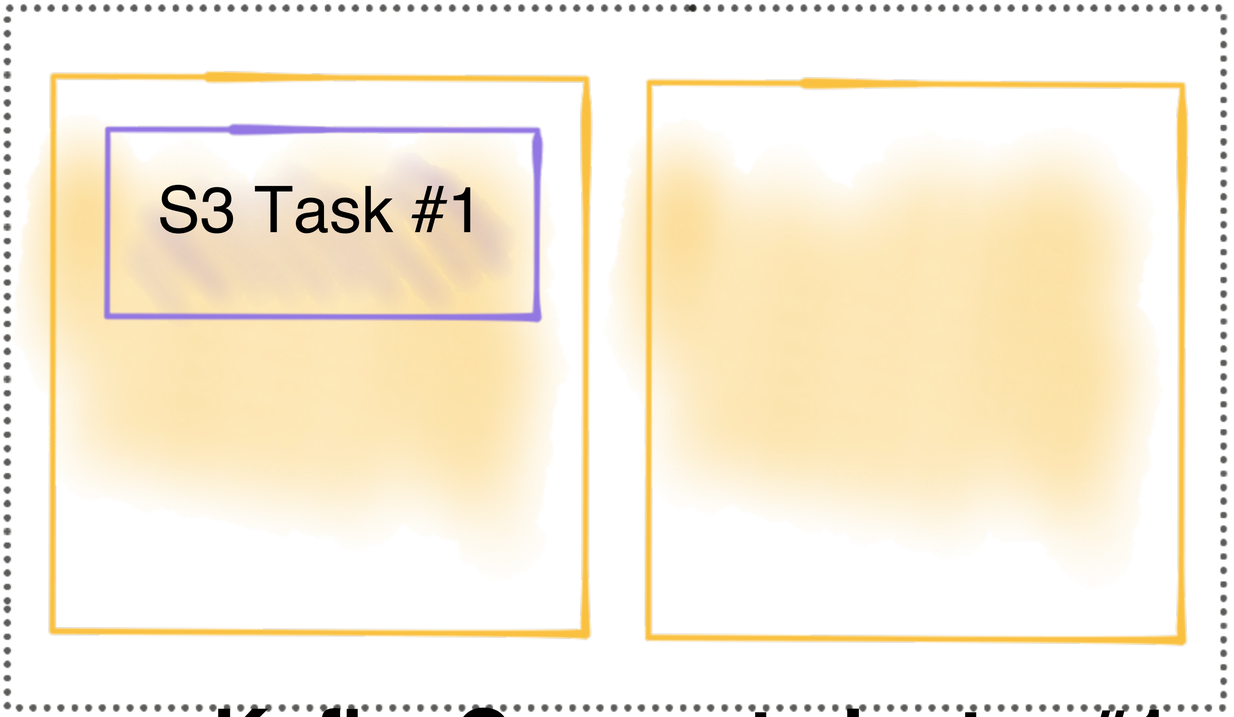


New in 2.3: KIP-415: Incremental Cooperative Rebalancing in Kafka Connect

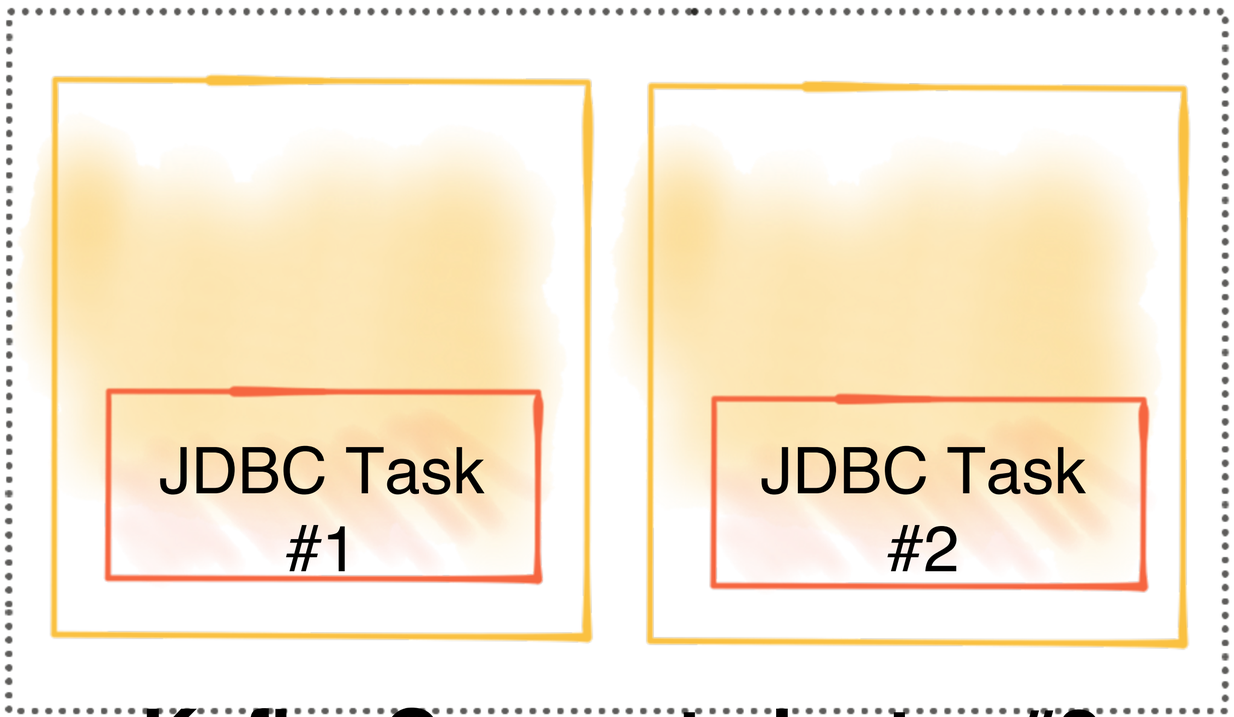
A new connector won't stop the existing tasks in a Kafka Connect cluster

With KIP-415, a rebalance happens more gracefully. It stops only the tasks that need to move between workers (if any), leaving the rest running on their assigned worker

Multiple Distributed Clusters



Kafka Connect cluster #1



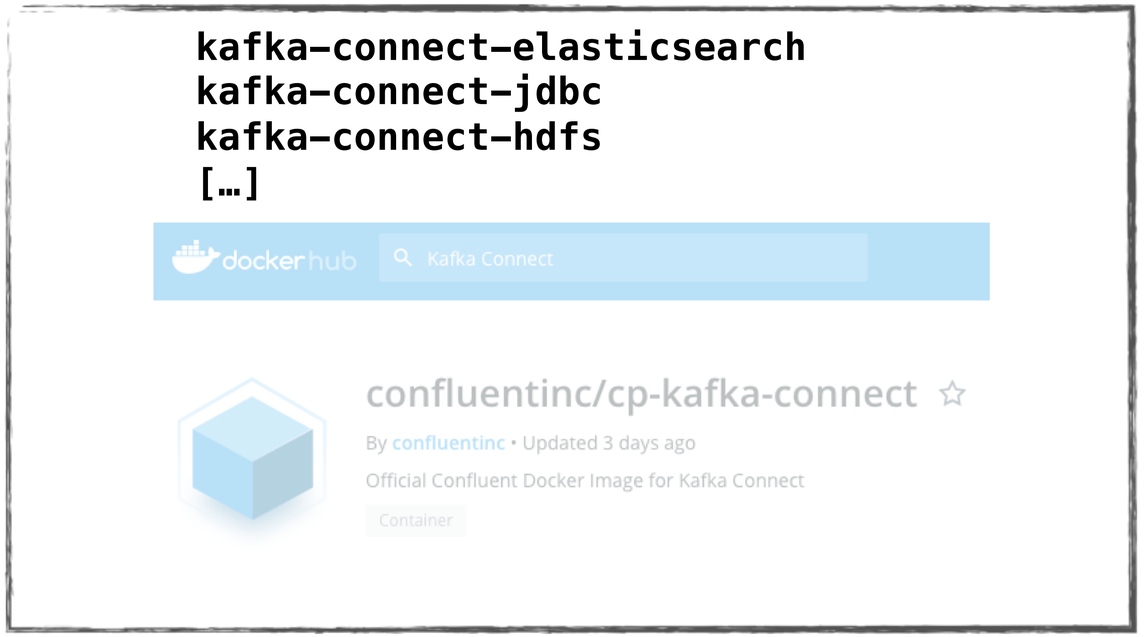
Kafka Connect cluster #2

Containers

Kafka Connect images on Docker Hub

confluentinc/cp-kafka-connect-base

kafka-connect-elasticsearch
kafka-connect-jdbc
kafka-connect-hdfs
[…]



confluentinc/cp-kafka-connect

Adding connectors to a container

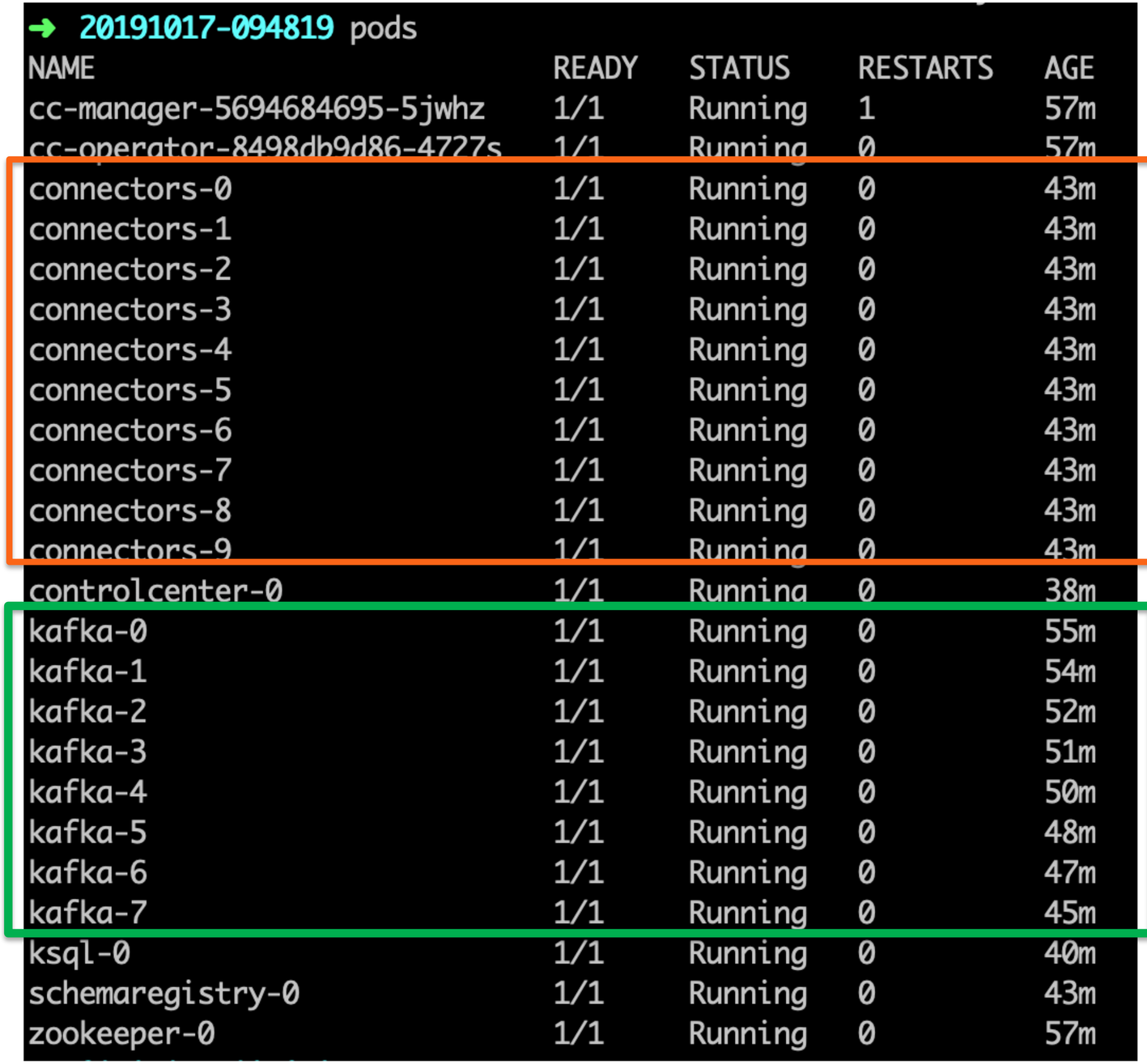
Confluent Hub

confluentinc/cp-kafka-connect-base

At runtime

```
kafka-connect:
  image: confluentinc/cp-kafka-connect:5.2.1
  environment:
    CONNECT_PLUGIN_PATH: '/usr/share/java,/usr/share/confluent-hub-components'
command:
    - bash - -c - |
      confluent-hub install --no-prompt neo4j/kafka-connect-neo4j:1.0.0
      /etc/confluent/docker/run
```

http://rmoff.dev/ksln19-connect-docker



```
➜  20191017-094819 pods
NAME                                  READY   STATUS    RESTARTS   AGE
cc-manager-5694684695-5jwhz           1/1     Running   1          57m
cc-operator-8498db9d86-4727s          1/1     Running   0          57m
connectors-0                          1/1     Running   0          43m
connectors-1                          1/1     Running   0          43m
connectors-2                          1/1     Running   0          43m
connectors-3                          1/1     Running   0          43m
connectors-4                          1/1     Running   0          43m
connectors-5                          1/1     Running   0          43m
connectors-6                          1/1     Running   0          43m
connectors-7                          1/1     Running   0          43m
connectors-8                          1/1     Running   0          43m
connectors-9                          1/1     Running   0          43m
controlcenter-0                       1/1     Running   0          38m
kafka-0                               1/1     Running   0          55m
kafka-1                               1/1     Running   0          54m
kafka-2                               1/1     Running   0          52m
kafka-3                               1/1     Running   0          51m
kafka-4                               1/1     Running   0          50m
kafka-5                               1/1     Running   0          48m
kafka-6                               1/1     Running   0          47m
kafka-7                               1/1     Running   0          45m
ksql-0                                1/1     Running   0          40m
schemaregistry-0                      1/1     Running   0          43m
zookeeper-0                           1/1     Running   0          57m
```

Kafka Connect Demo system: stream Telco *Change Data Records (CDR)* into a topic. Each CDR has 160 attributes.

All Services run on Kubernetes pods, on Google Cloud. Deployed to Kubernetes using *Confluent Operator*

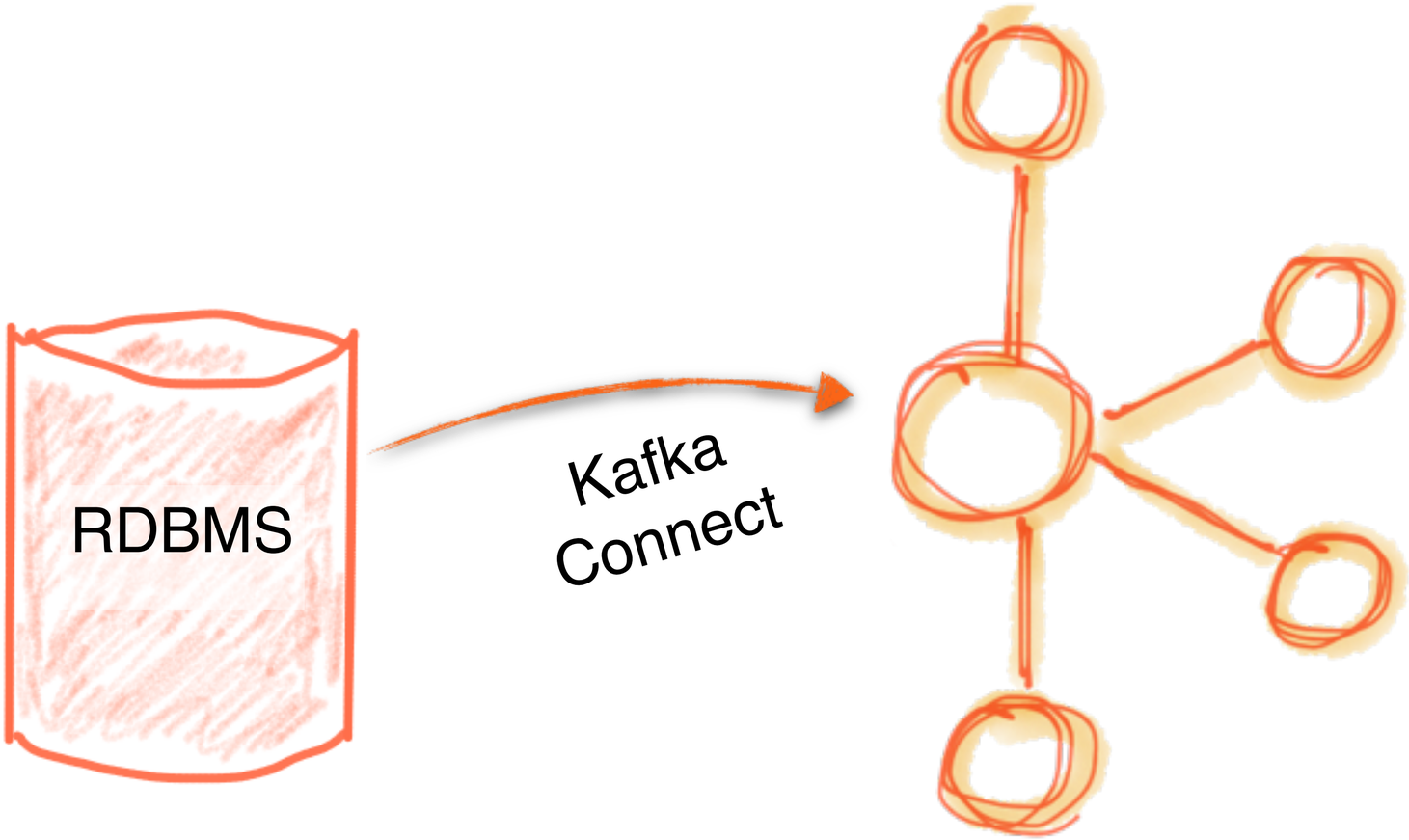
Use 10 Kafka Connect Workers, each running SPOOLDIR (source), Elastic (Sink)
8 Kafka Brokers

Use kSQL for aggregation and filtering

CDR's flow to another Google Cloud Kafka System for Disaster Recovery using Confluent Replicator.

Demo system should process 100,000 CDR's per second, average.

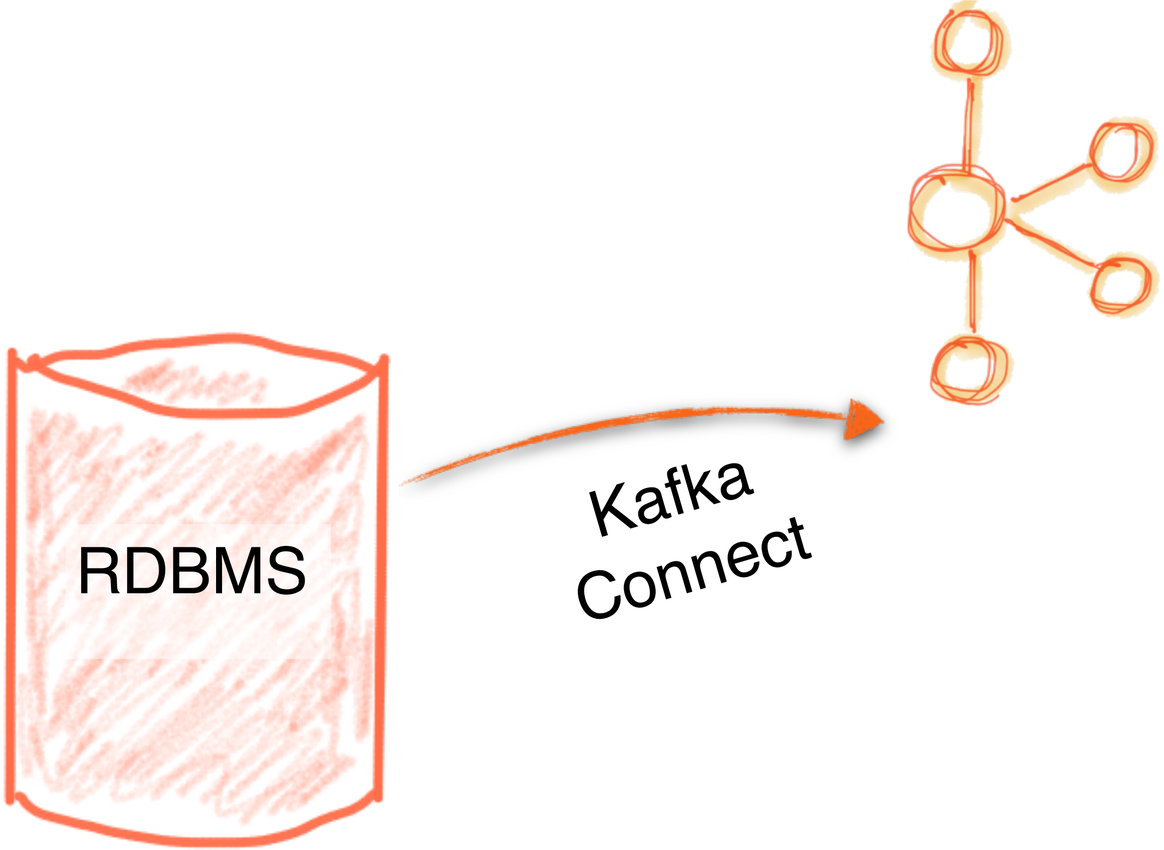
JDBC Source



Dialect Support for

- Microsoft SQL Server
- PostgreSQL
- Oracle
- IBM DB2
- SAP HANA
- SQLite
- Generic JDBC 4.0 Support

JDBC Source



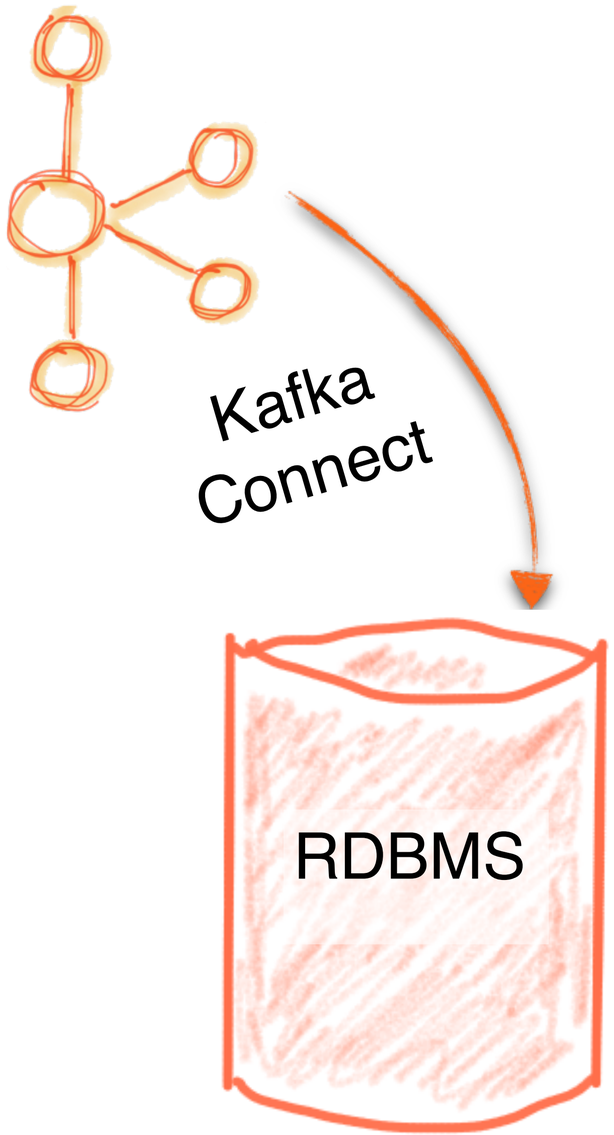
SELECT every x seconds

Incremental Query Modes:

1, Incrementing Column
2, Timestamp Column
3, Timestamp + Incrementing Column
4, Custom Query
5, Bulk

Create/Evolve the table schema in the Schema Registry

JDBC Sink



INSERT whenever a new message is sent to the topic

Idempotent writes - *insert.mode* to do INSERT, UPDATE or MERGE/UPSERT

Schema auto.create and auto.evolve

#EOF

teehan@confluent.io

AMA – I'll try....

http://talks.rmoff.net/ - Author Credit – Robin Moffatt

Confluent Community - What next?

Join the Confluent Community Slack Channel

About 10,000 Kafkateers are collaborating every single day on the Confluent Community Slack channel!

cnfl.io/community-slack

The Confluent Community Catalyst Program

Subscribe to the Confluent blog

Get frequent updates from key names in Apache Kafka® on best practices, product updates & more!

cnfl.io/read

confluent