@rmoff #KafkaMeetup

From Zero to Hero with Kafka Connect

a.k.a.

A practical guide to becoming l33t with Kafka Connect

@rmoff #KafkaMeetup

From Zero to Hero with Kafka Connect

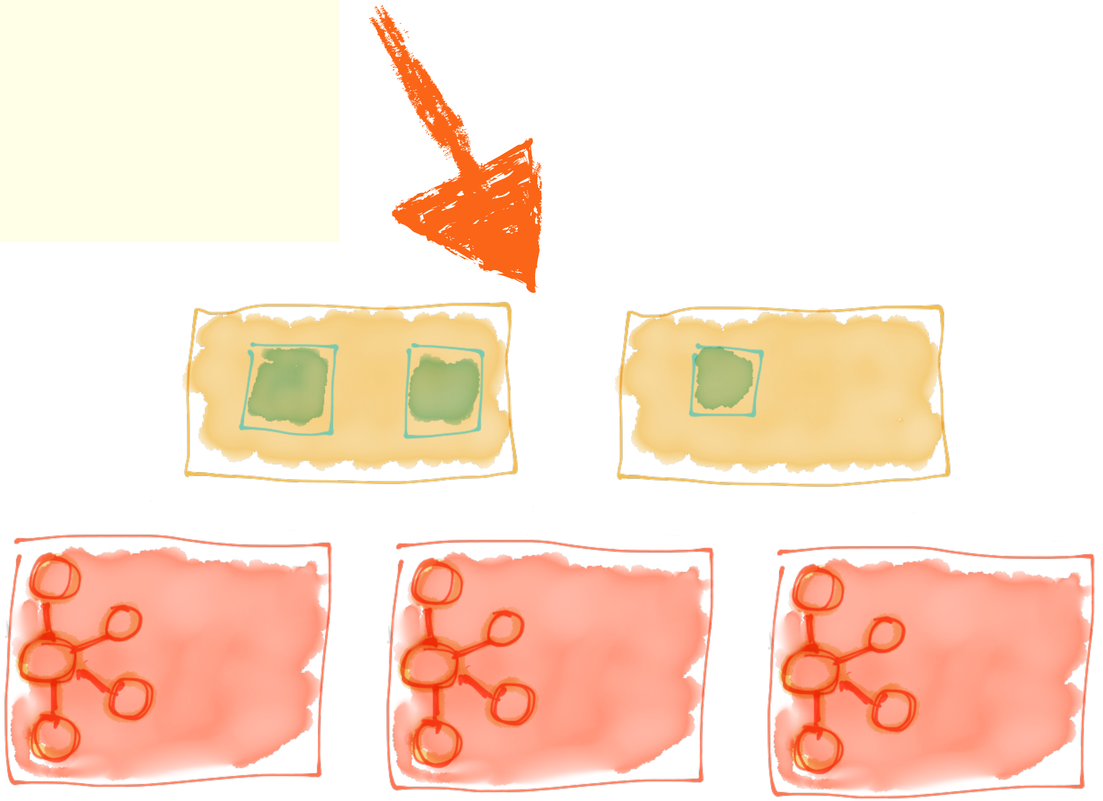
a.k.a.

A practical guide to becoming l33t with Kafka Connect

What is Kafka Connect?

Streaming Integration with Kafka Connect

Sources

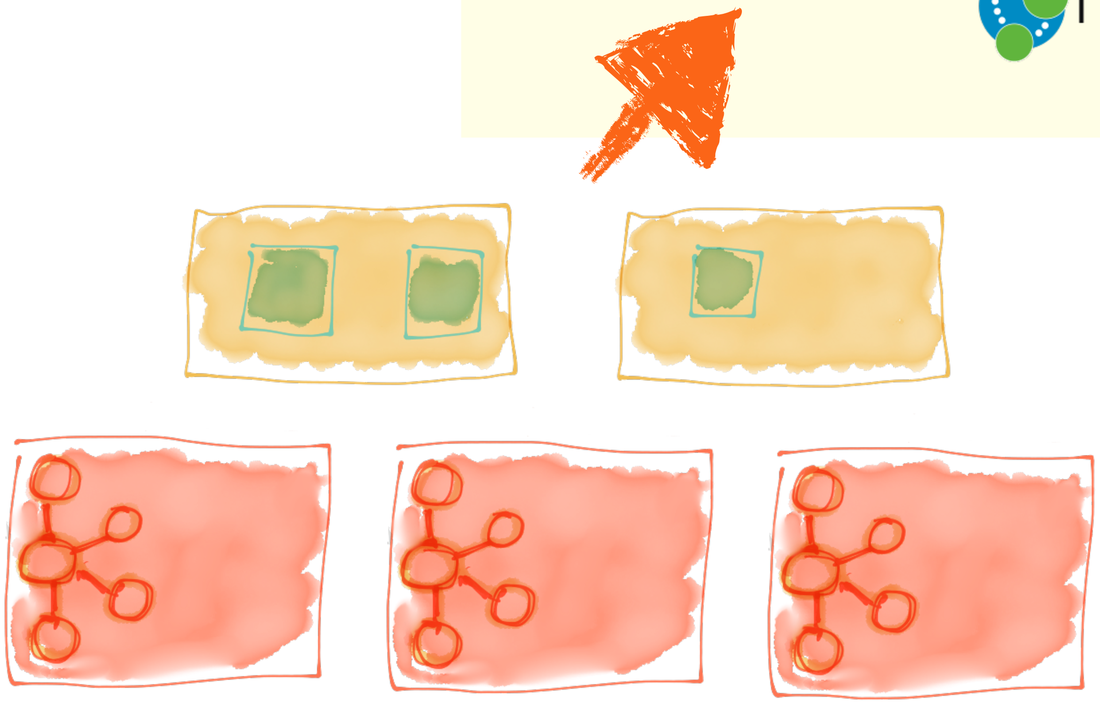


Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect

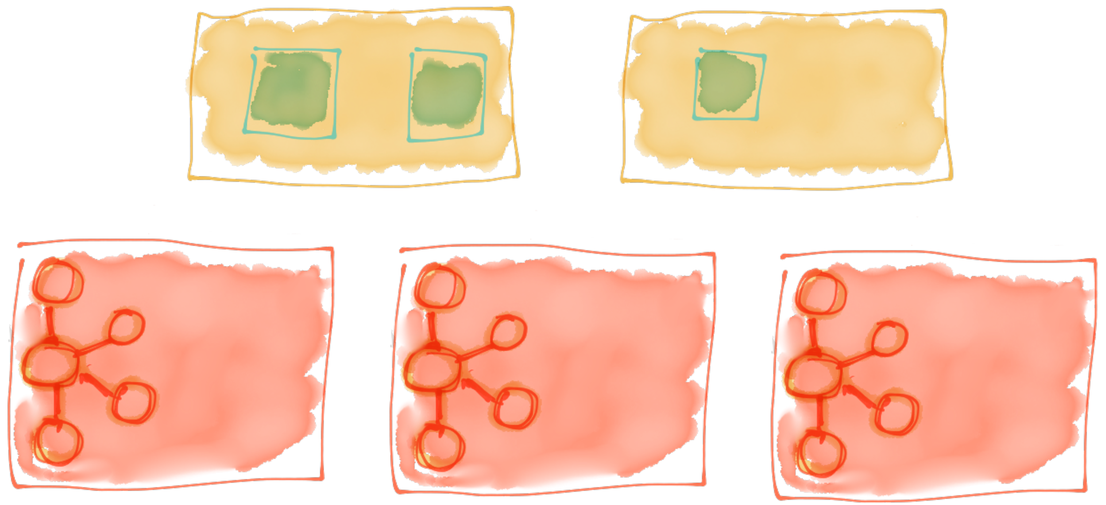
Sinks



Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect



Kafka Connect

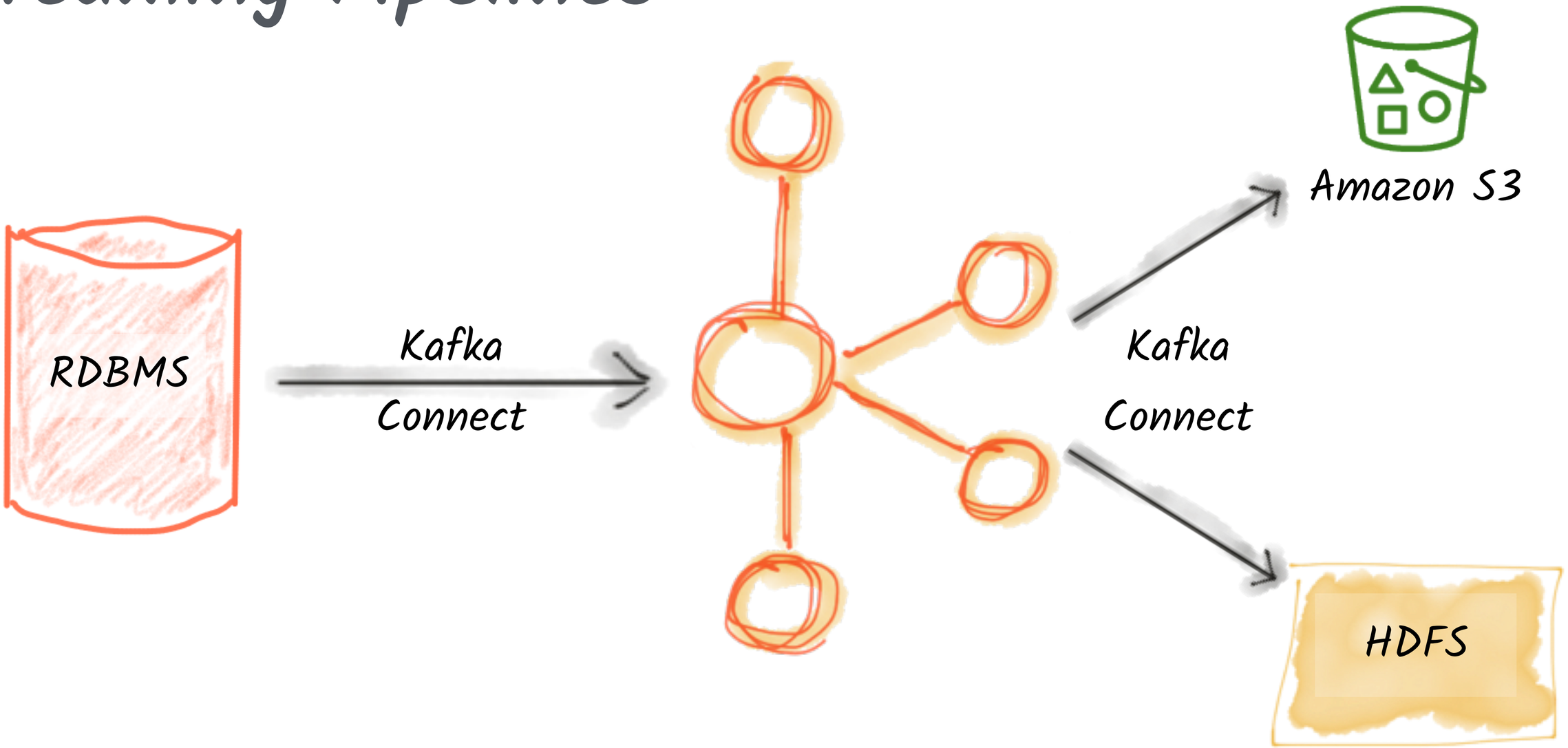
Kafka Brokers

Look Ma, No Code!

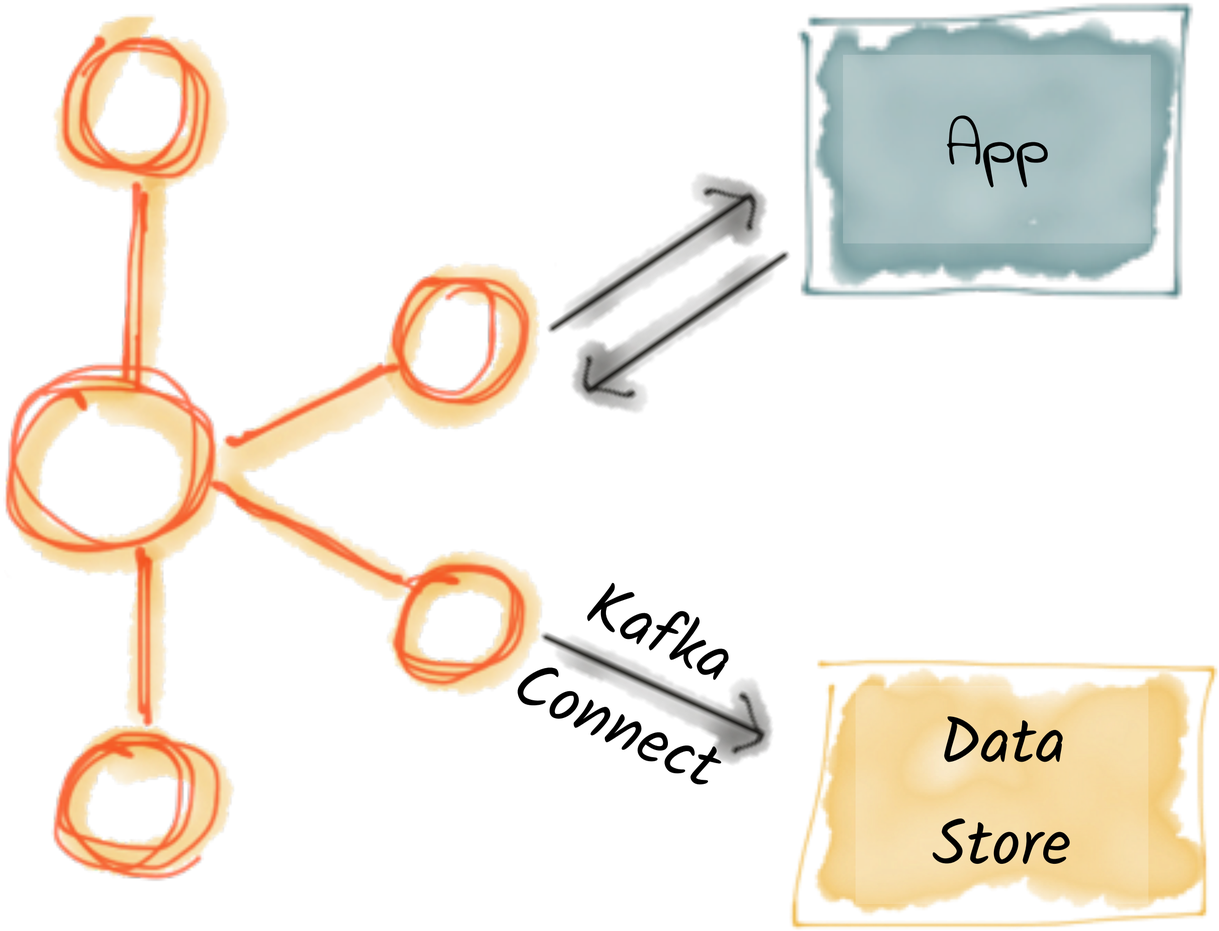
```
{
  "connector.class":
    "io.confluent.connect.jdbc.JdbcSourceConnector",
  "connection.url":
    "jdbc:mysql://asgard:3306/demo",
  "table.whitelist":
    "sales,orders,customers"
}
```

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

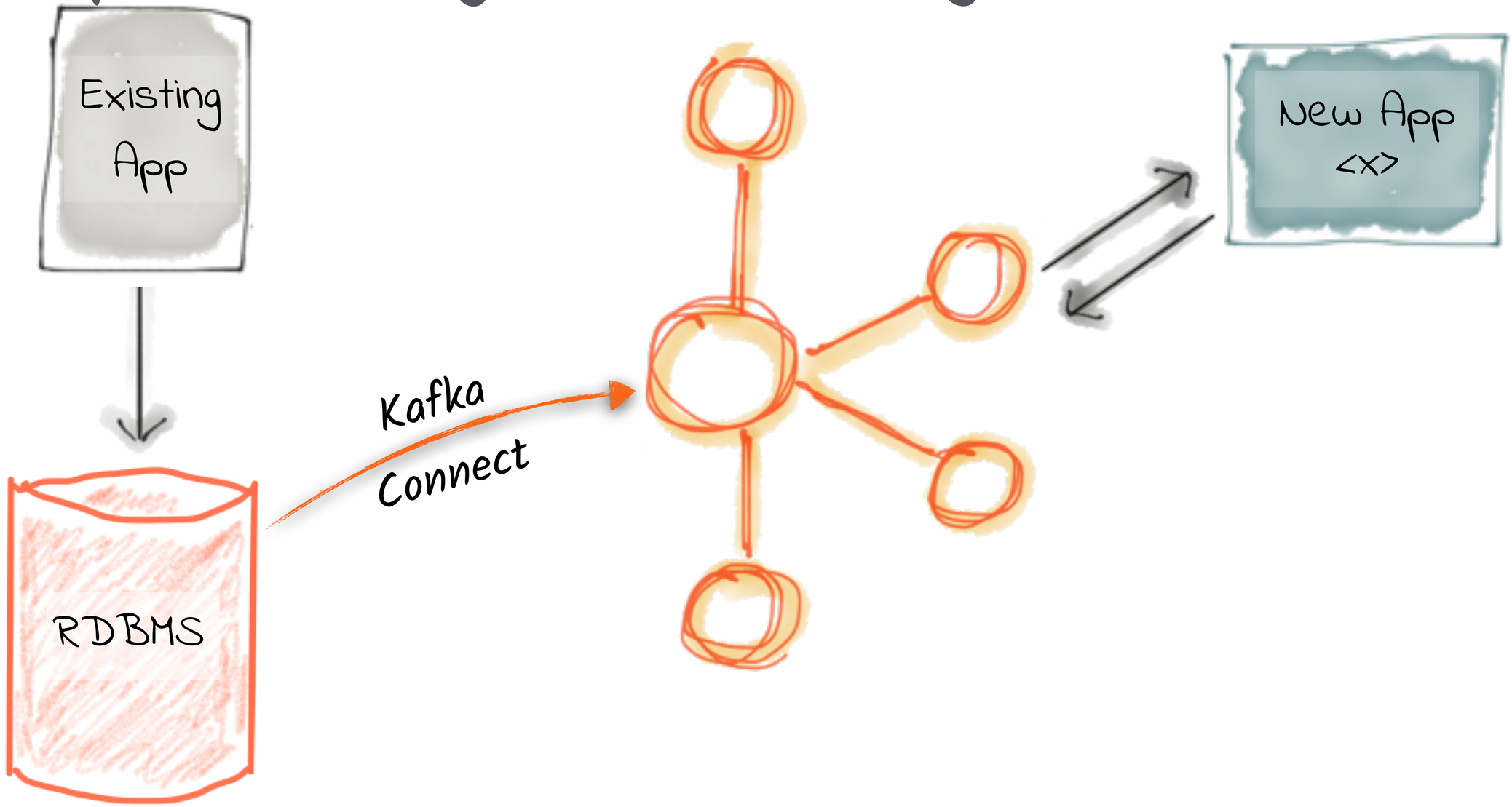
Streaming Pipelines



Writing to data stores from Kafka



Evolve processing from old systems to new



Demo

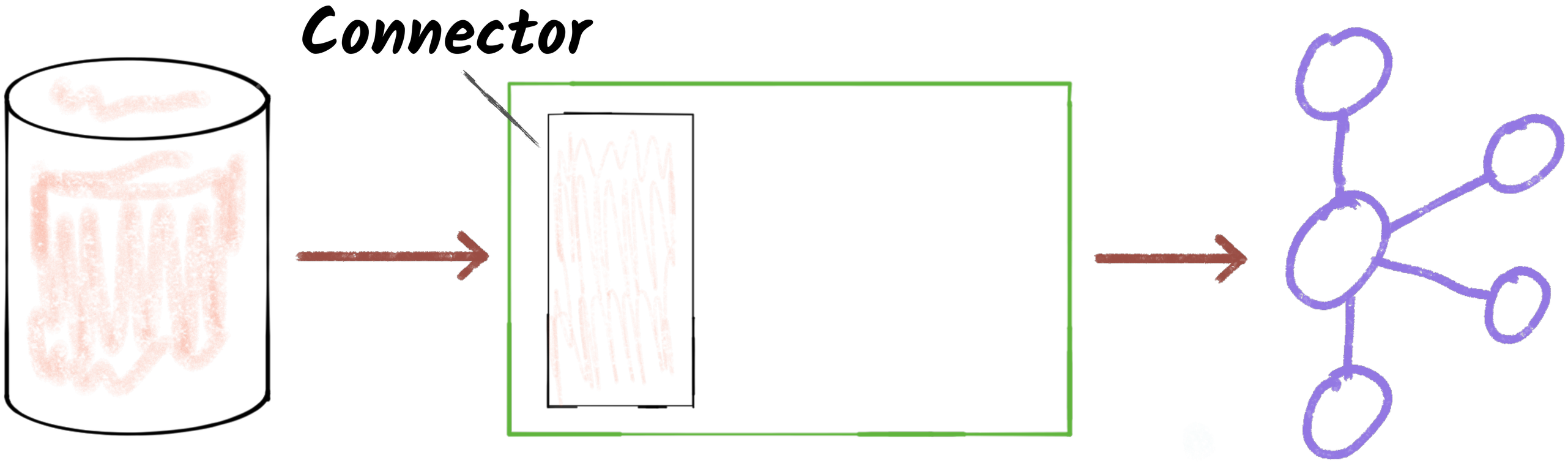
http://rmoff.dev/kafka-connect-code

Configuring Kafka Connect

Inside the API - connectors, transforms, converters

Kafka Connect basics

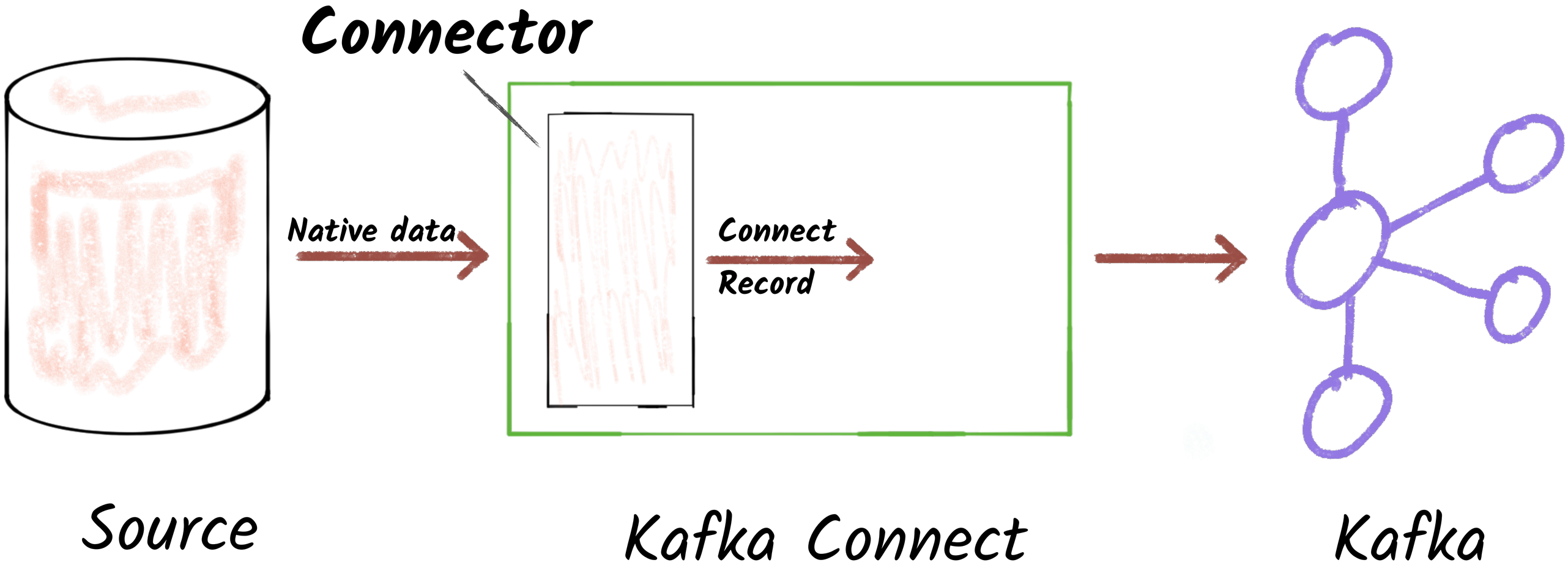
Connectors



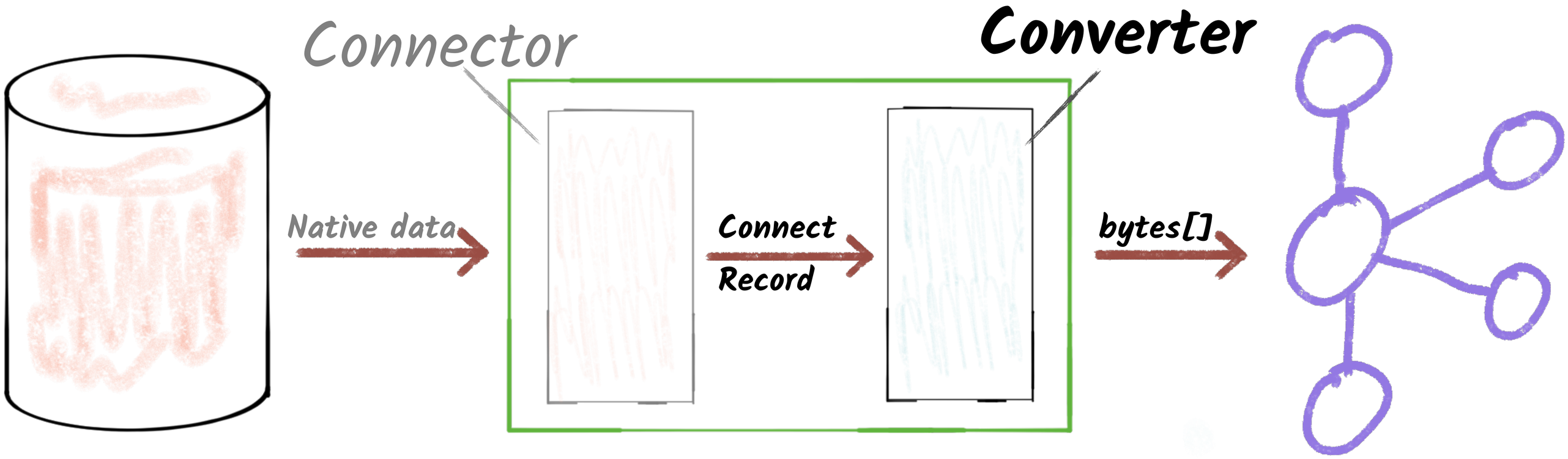
Connectors

```
"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

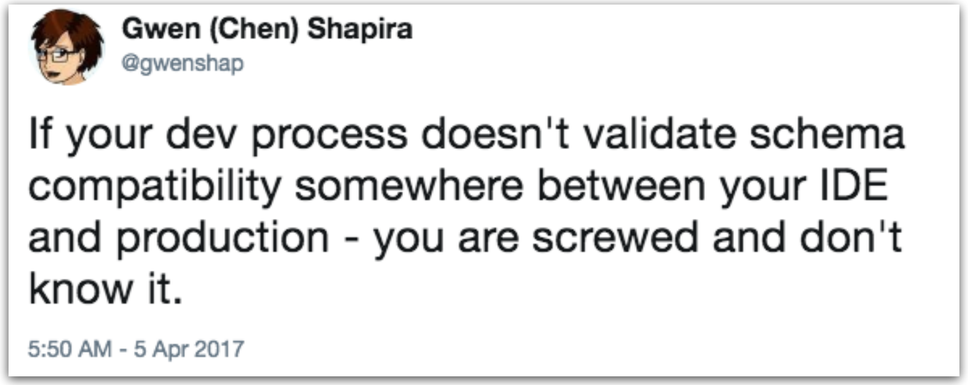
Converters


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

Serialisation & Schemas

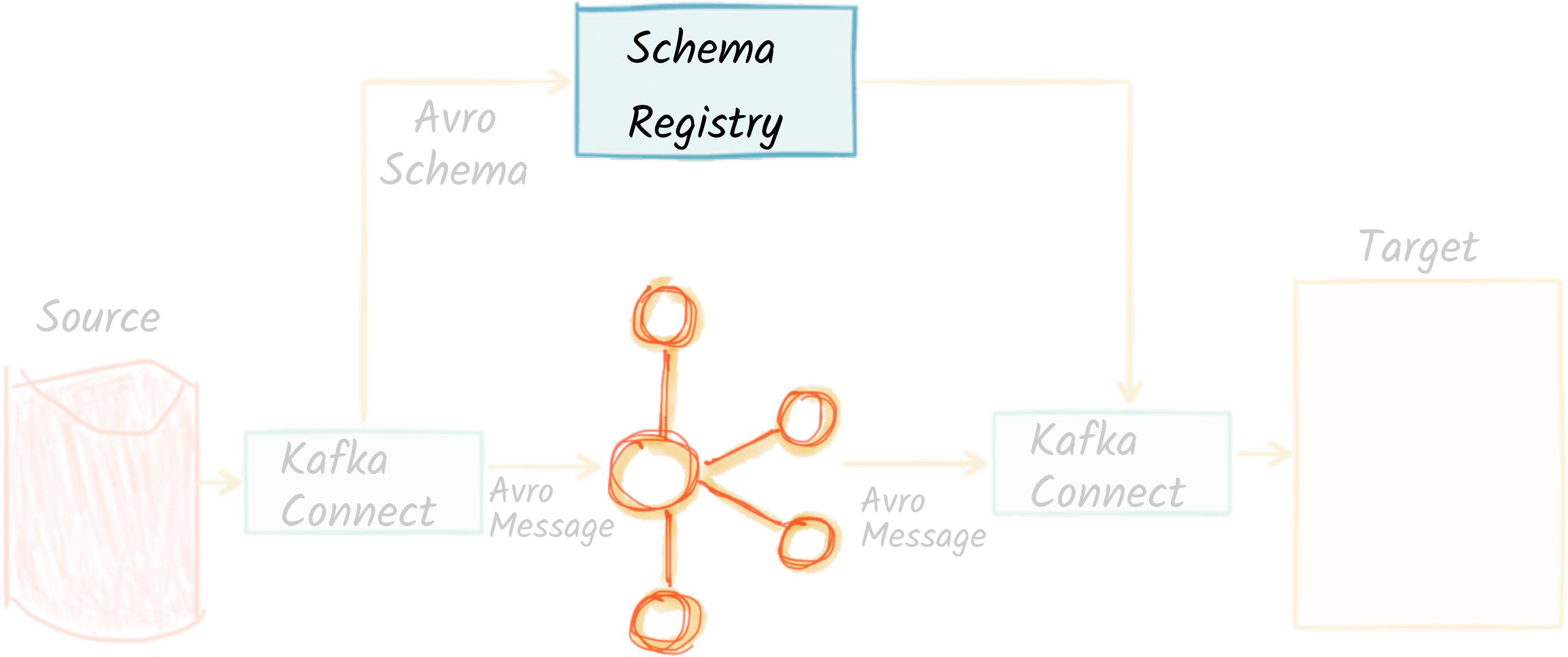
Avro Protobuf JSON CSV

-> Confluent Schema Registry



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

The Confluent Schema Registry



Converters

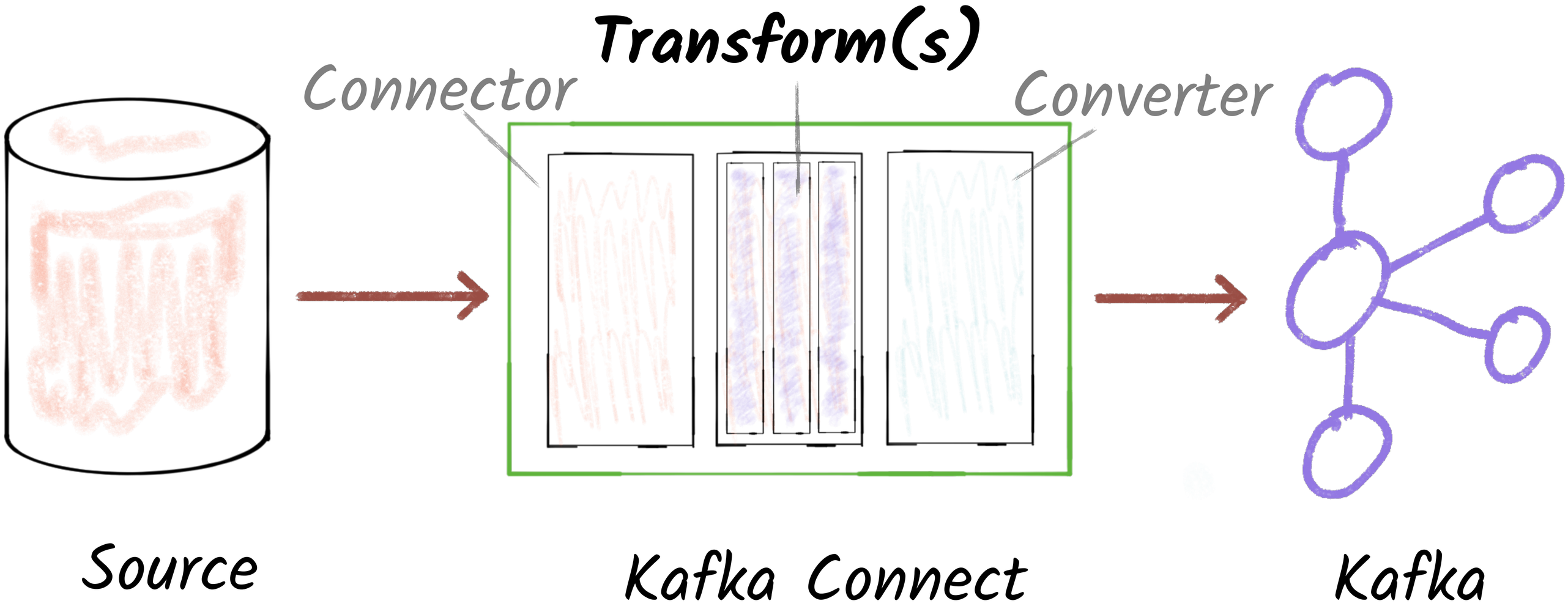
```
key.converter=io.confluent.connect.avro.AvroConverter
key.converter.schema.registry.url=http://localhost:8081
value.converter=io.confluent.connect.avro.AvroConverter
value.converter.schema.registry.url=http://localhost:8081
```

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

What about internal converters?

```
value.converter=org.apache.kafka.connect.json.JsonConverter
internal.value.converter=org.apache.kafka.connect.json.JsonConverter
key.internal.value.converter=org.apache.kafka.connect.json.JsonConverter
value.internal.value.converter=org.apache.kafka.connect.json.JsonConverter
key.internal.value.converter.bork.bork.bork=org.apache.kafka.connect.json.JsonConverter
key.internal.value.please.just.work.converter=org.apache.kafka.connect.json.JsonConverter
```

Single Message Transforms



Single Message Transforms

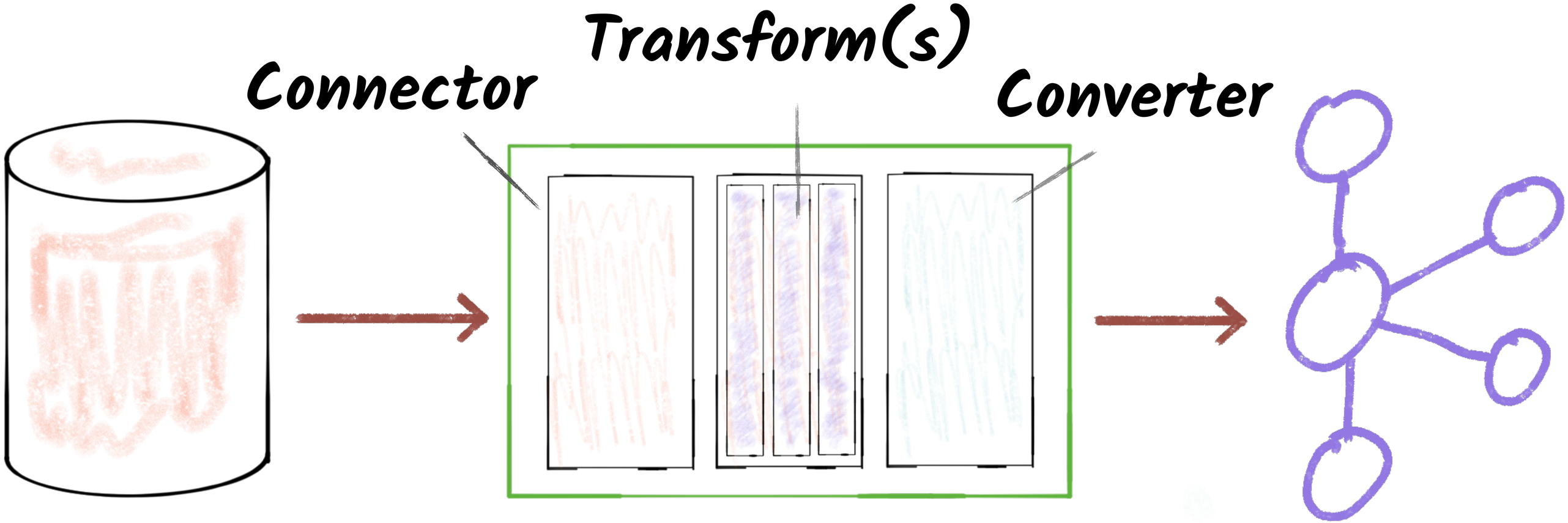
```
"config": {
  [...]
  "transforms": "addDateToTopic,labelFooBar",
  "transforms.addDateToTopic.type": "org.apache.kafka.connect.transforms.TimestampRouter",
  "transforms.addDateToTopic.topic.format": "${topic}-${timestamp}",
  "transforms.addDateToTopic.timestamp.format": "YYYYMM",
  "transforms.labelFooBar.type":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

Do these transforms

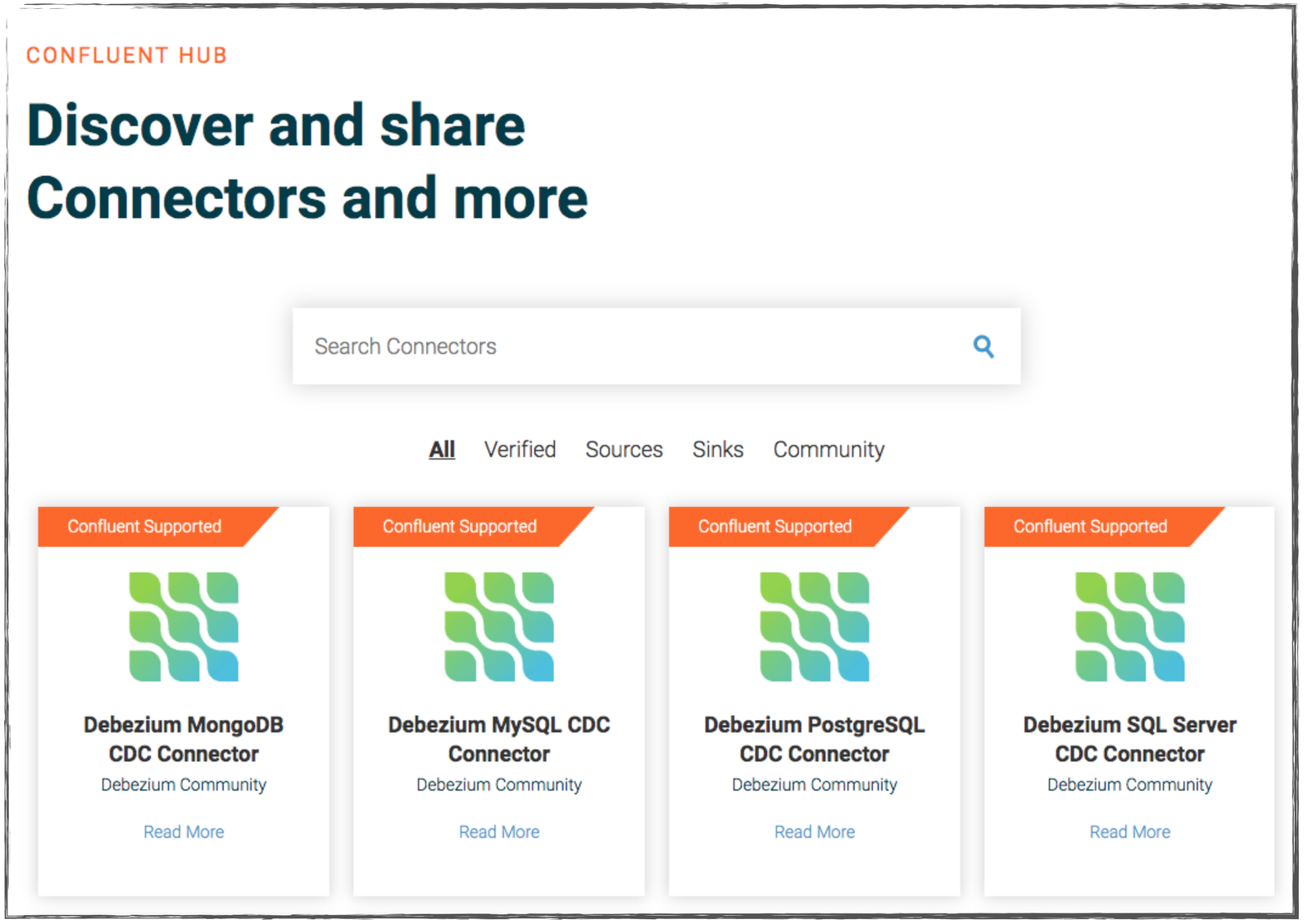
Transforms config

Config per transform

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

JDBC Source

S3 Sink

S3 Task #1

JDBC Task #1

JDBC Task #2

Tasks and Workers

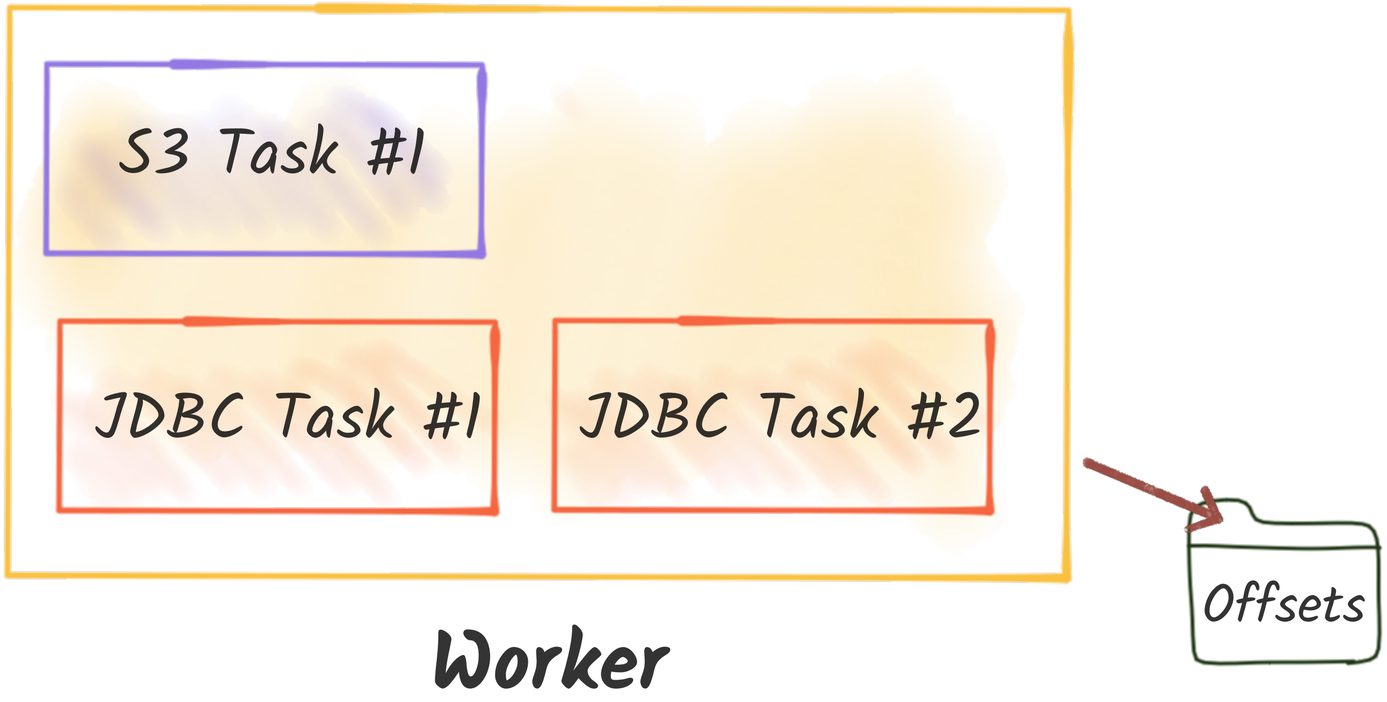
JDBC Source

S3 Sink

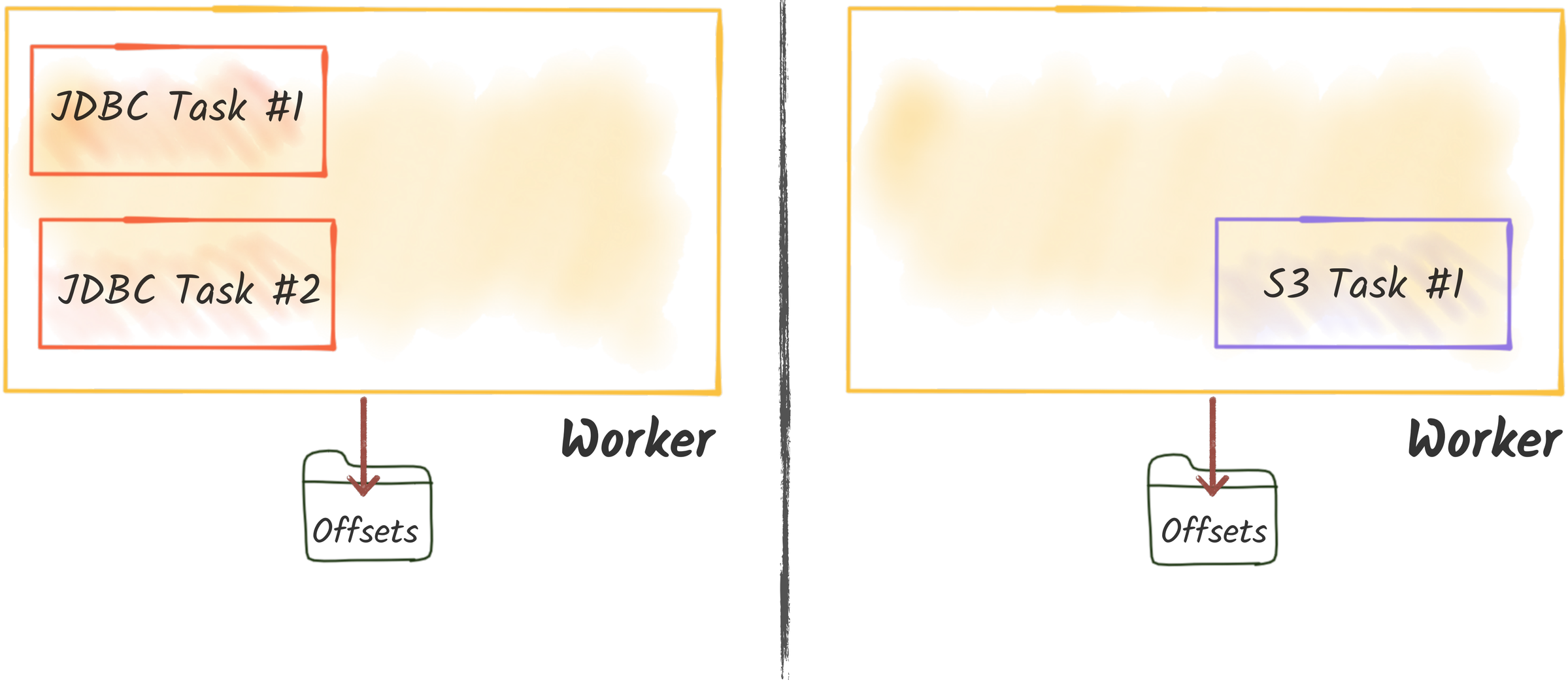
Worker


Standalone
Distributed
JAKE-CLARK.TUMBLR
imgflip.com

Kafka Connect Standalone Worker

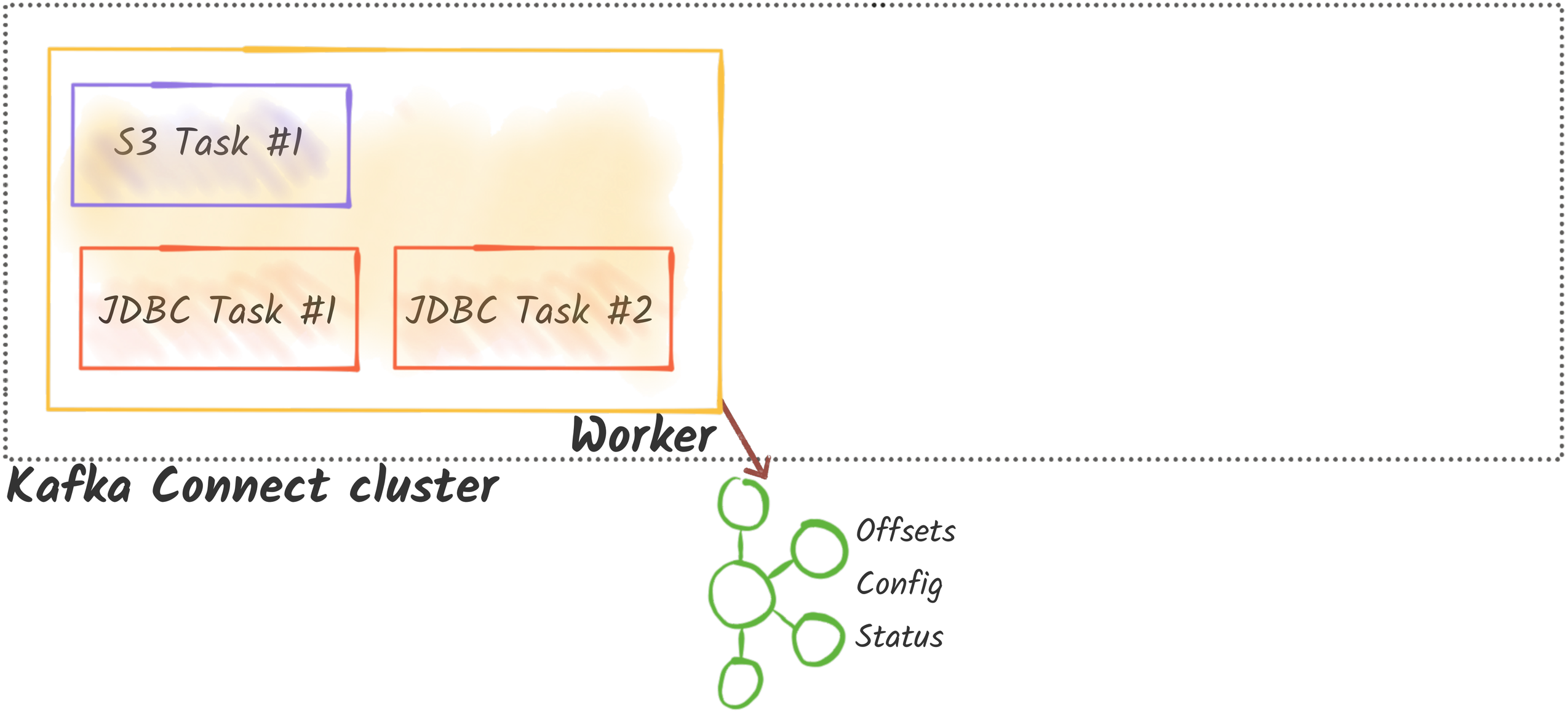


"Scaling" the Standalone Worker



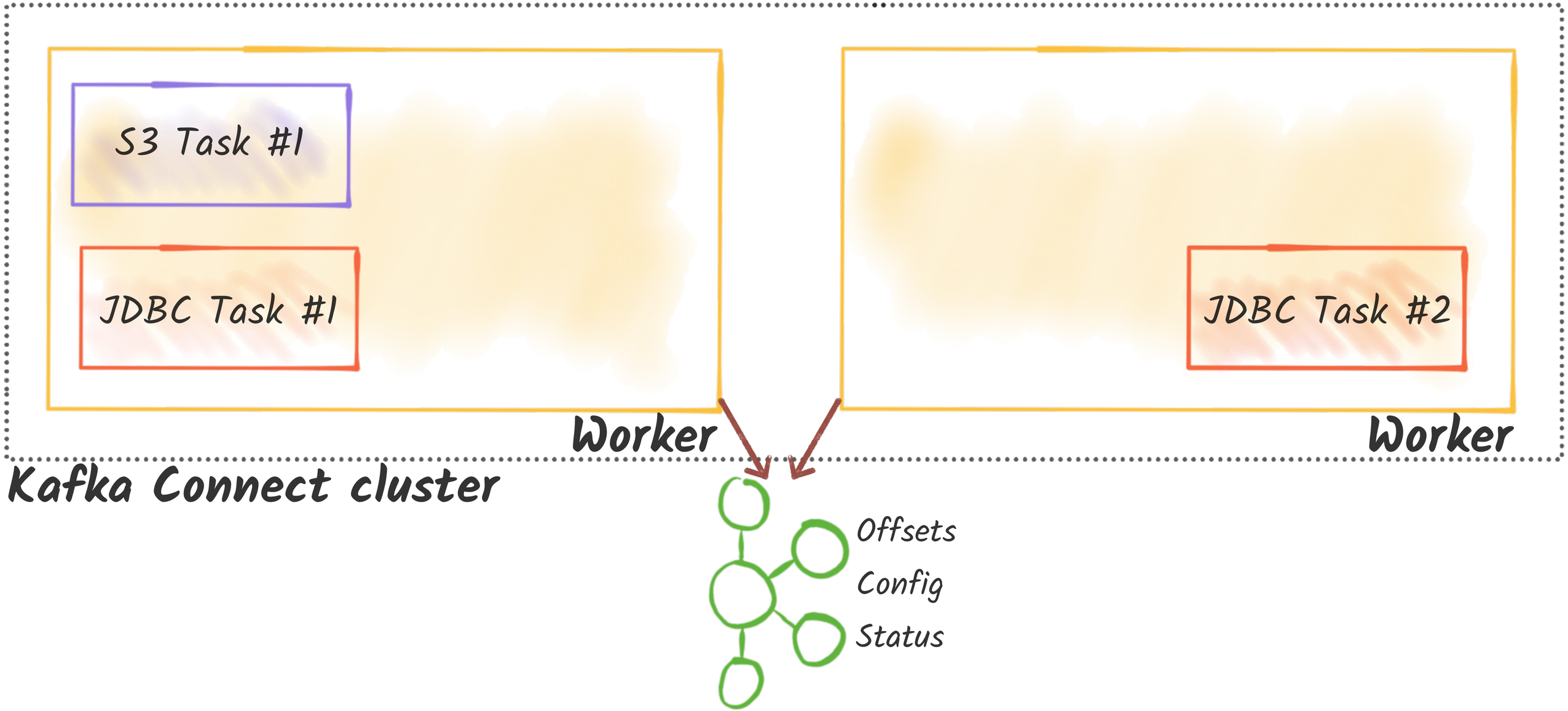
Fault-tolerant? **Nope.**

Kafka Connect Distributed Worker



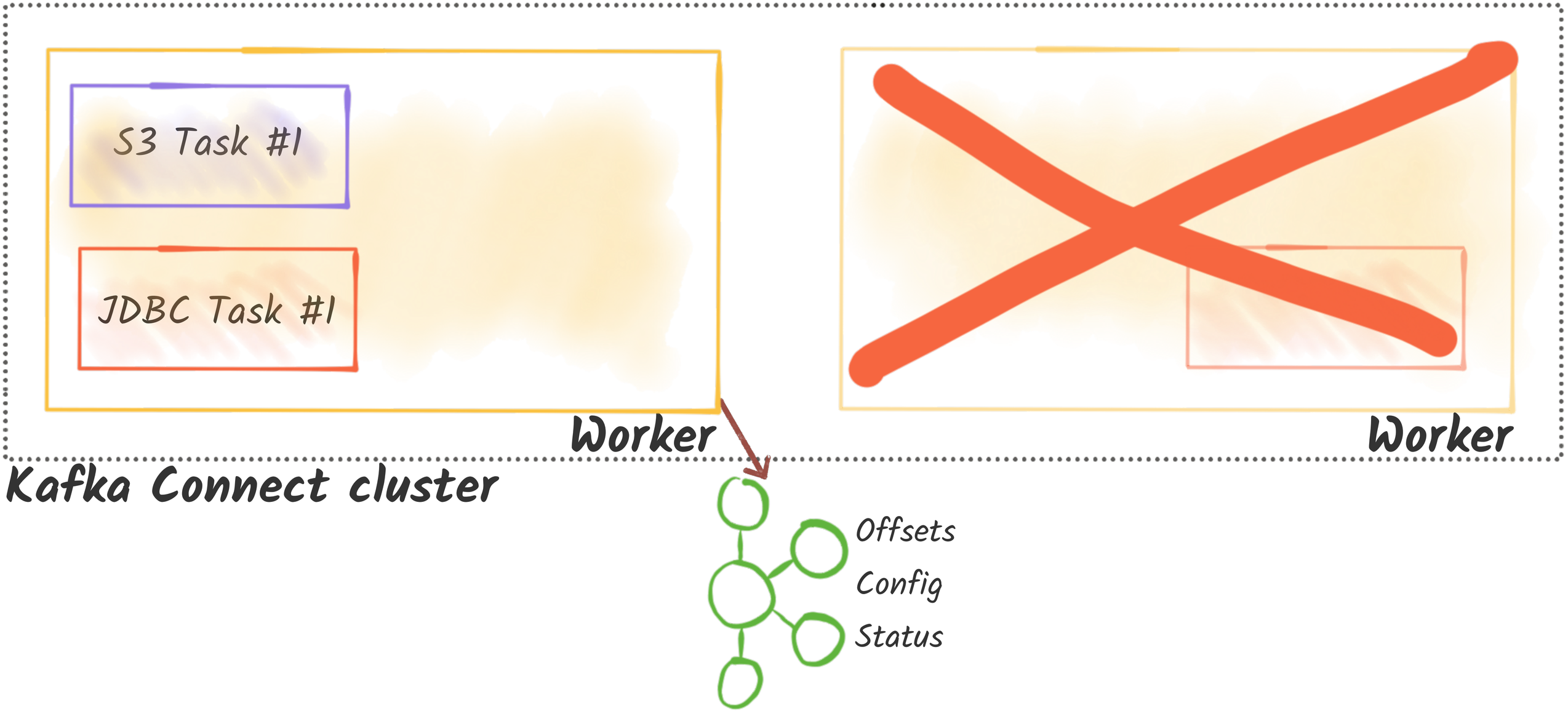
Fault-tolerant? **Yeah!**

Scaling the Distributed Worker

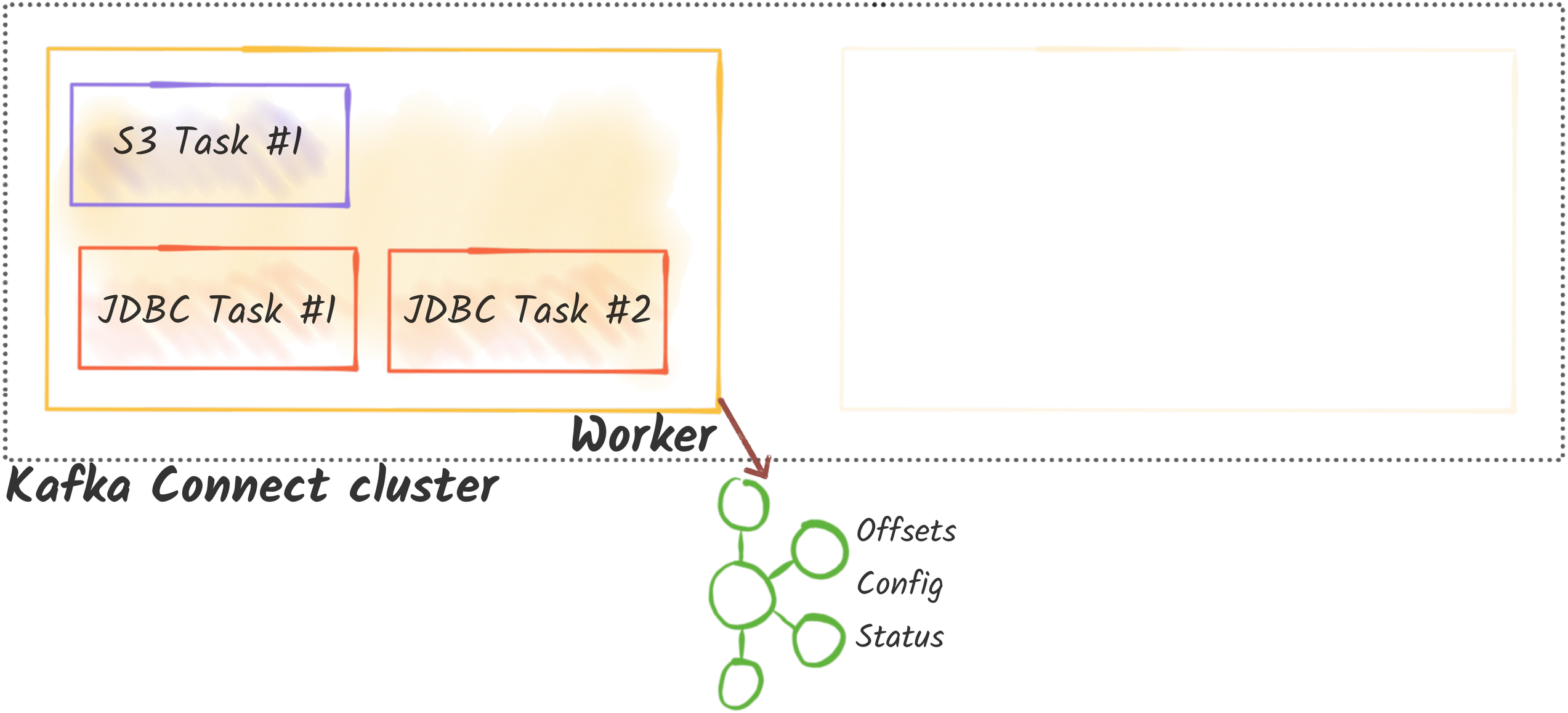


Fault-tolerant? **Yeah!**

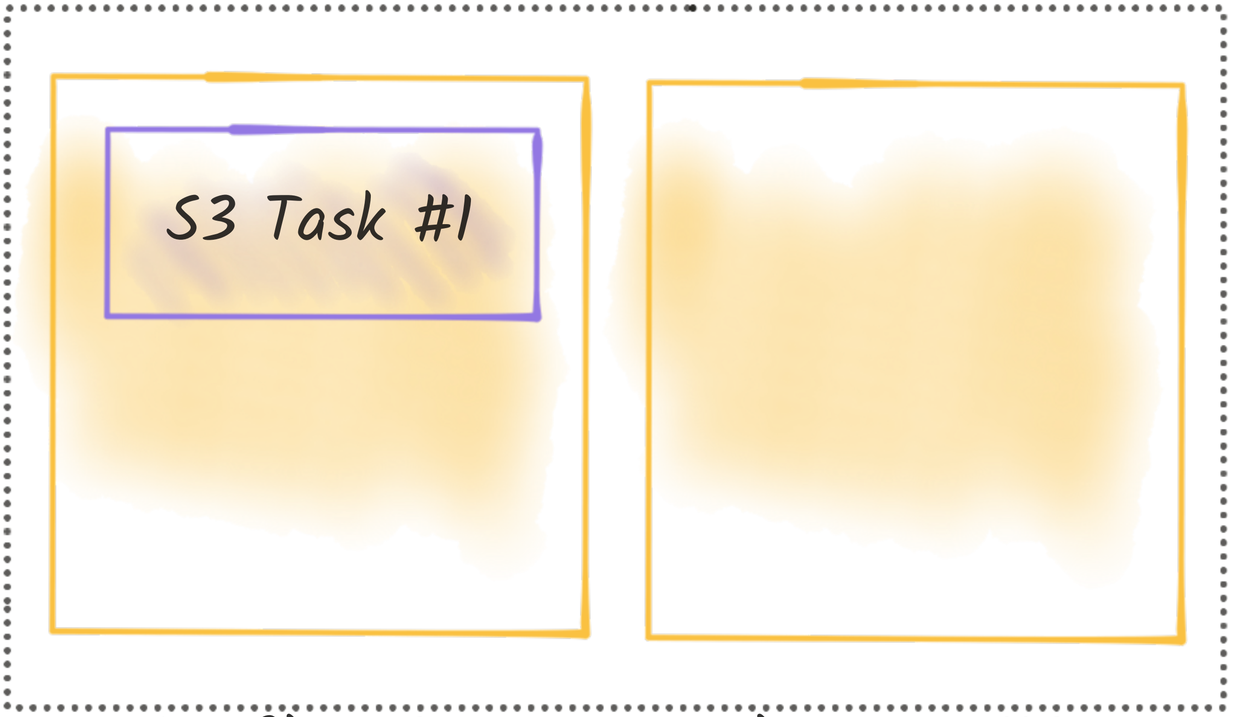
Distributed Worker - fault tolerance



Distributed Worker - fault tolerance



Multiple Distributed Clusters



Kafka Connect cluster #1

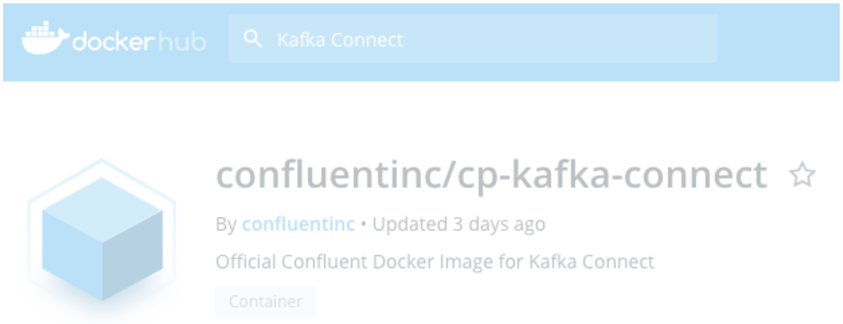
Kafka Connect cluster #2

Containers

Kafka Connect images on Docker Hub

confluentinc/cp-kafka-connect-base

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



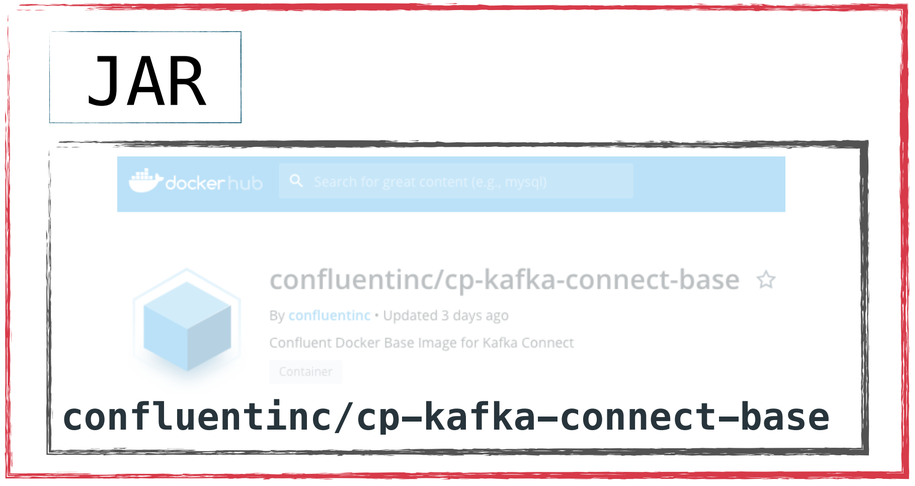
confluentinc/cp-kafka-connect

Adding connectors to a container

Confluent Hub

At runtime

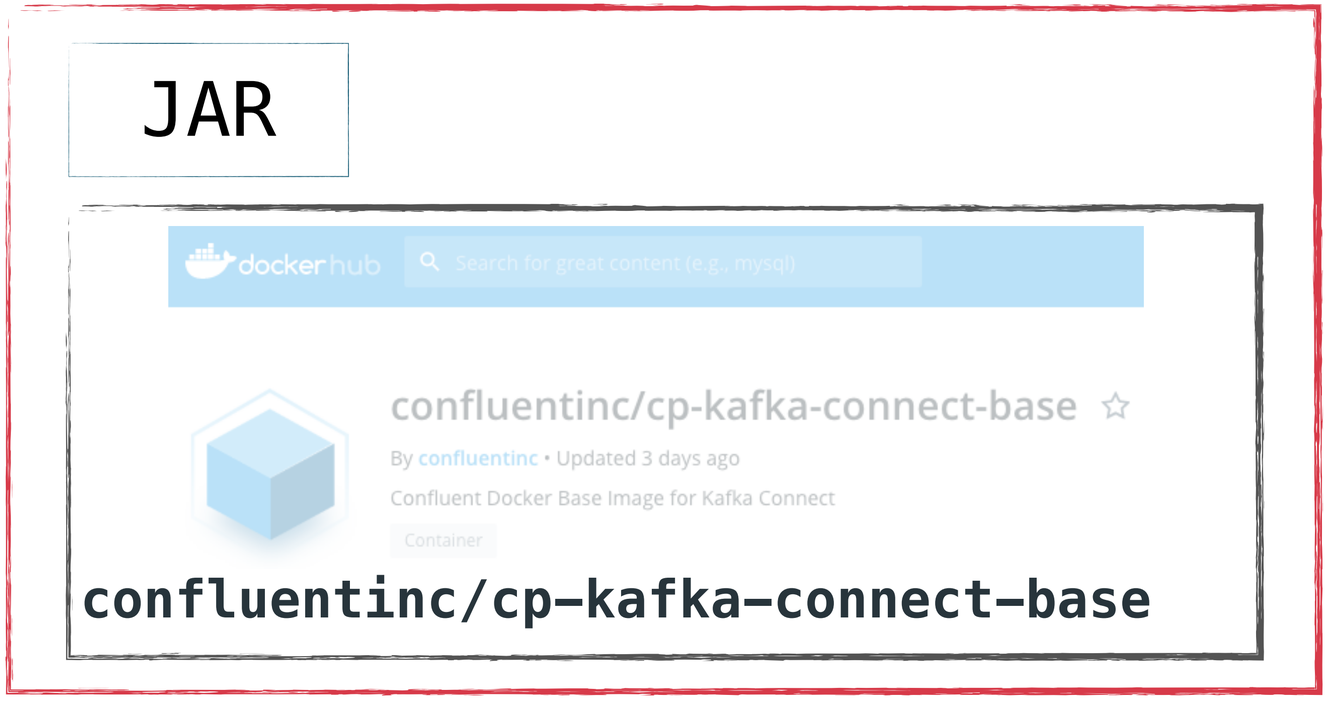
```
kafka-connect:
  image: confluentinc/cp-kafka-connect:5.2.1
  environment:
    CONNECT_PLUGIN_PATH: '/usr/share/java,/usr/share/confluent-hub-components'
command:
    - bash
    - -c
    - |
      confluent-hub install --no-prompt neo4j/kafka-connect-neo4j:1.0.0
      /etc/confluent/docker/run
```



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

Build a new image

```
FROM confluentinc/cp-kafka-connect:5.2.1
ENV CONNECT_PLUGIN_PATH="/usr/share/java,/usr/share/confluent-hub-components"
RUN confluent-hub install --no-prompt neo4j/kafka-connect-neo4j:1.0.0
```



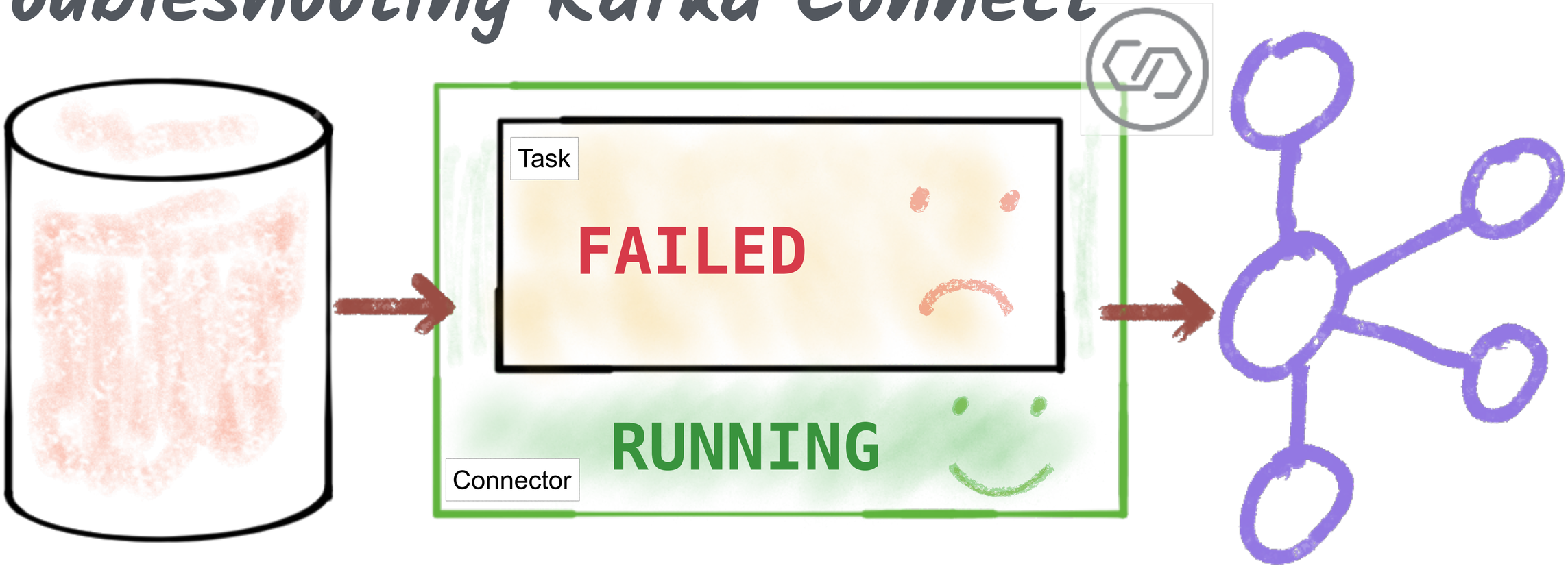
Automating connector creation

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

```
# Launch Kafka Connect
/etc/confluent/docker/run &
#
# Wait for Kafka Connect listener
while [ $$(curl -s -o /dev/null -w %{http_code} http://$$CONNEC
  echo -e $$(date) " Kafka Connect listener HTTP state: " $$(cu
  sleep 5
done
#
# Create JDBC Source connector
curl -X POST http://localhost:8083/connectors -H "Content-Type:
          "name": "jdbc_source_mysql_00",
          "config": {
                  "connector.class": "io.confluent.connect.jdbc
                  "connection.url": "jdbc:mysql://mysql:3306/de
```

Troubleshooting Kafka Connect

Troubleshooting Kafka Connect




```
$ curl -s "http://localhost:8083/connectors/source-debezium-orders/status" | \
  jq '.connector.state'
"RUNNING"
$ curl -s "http://localhost:8083/connectors/source-debezium-orders/status" | \
  jq '.tasks[0].state'
"FAILED"
```

http://go.rmoff.net/connector-status

Troubleshooting Kafka Connect

```
curl -s "http://localhost:8083/connectors/source-debezium-orders-00/status"
| jq '.tasks[0].trace'
```

```
"org.apache.kafka.connect.errors.ConnectException\n\tat
io.debezium.connector.mysql.AbstractReader.wrap(AbstractReader.java:230)\n\tat
io.debezium.connector.mysql.AbstractReader.failed(AbstractReader.java:197)\n\tat
io.debezium.connector.mysql.BinlogReader$ReaderThreadLifecycleListener.onCommunicationFailure(BinlogReader.java:
1018)\n\tat com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:950)\n\tat
com.github.shyiko.mysql.binlog.BinaryLogClient.connect(BinaryLogClient.java:580)\n\tat
com.github.shyiko.mysql.binlog.BinaryLogClient$7.run(BinaryLogClient.java:825)\n\tat java.lang.Thread.run(Thread.java:
748)\nCaused by: java.io.EOFException\n\tat
com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.read(ByteArrayInputStream.java:190)\n\tat
com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.readInteger(ByteArrayInputStream.java:46)\n\tat
com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java
:35)\n\tat
com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java
:27)\n\tat com.github.shyiko.mysql.binlog.event.deserialization.EventDeserializer.nextEvent(EventDeserializer.java:
212)\n\tat io.debezium.connector.mysql.BinlogReader$1.nextEvent(BinlogReader.java:224)\n\tat
com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:922)\n\t ... 3 more\n"
```

The log is the source of truth

```
$ confluent log connect

$ docker-compose logs kafka-connect

$ cat /var/log/kafka/connect.log
```

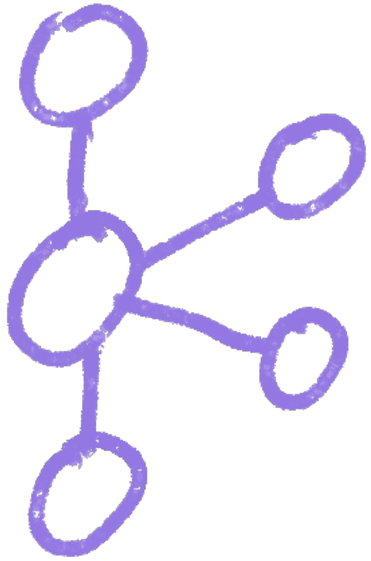
Error Handling and Dead Letter Queues

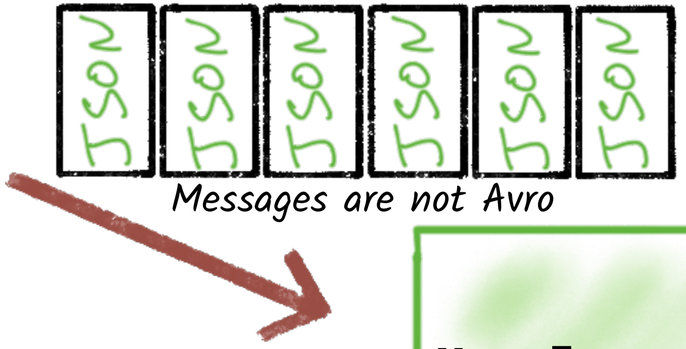
org.apache.kafka.common.errors.SerializationException: Unknown magic byte!

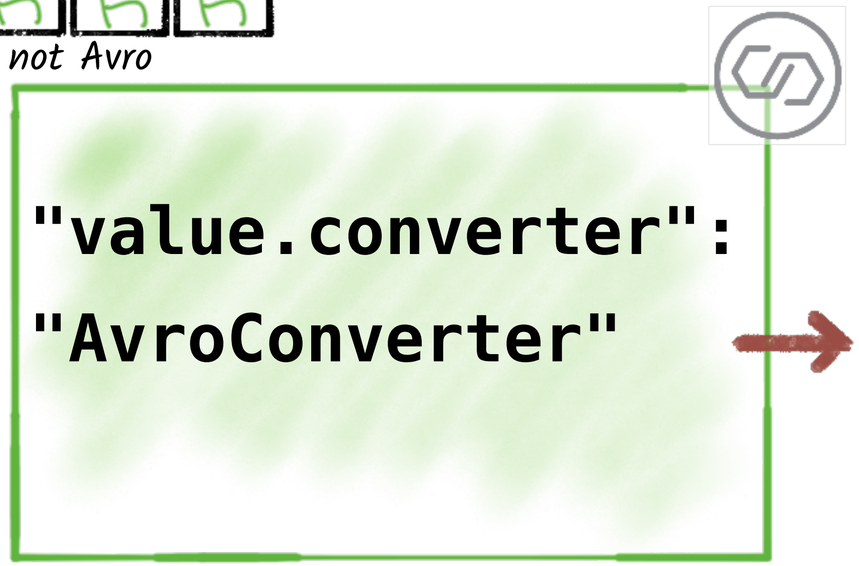
Mismatched converters

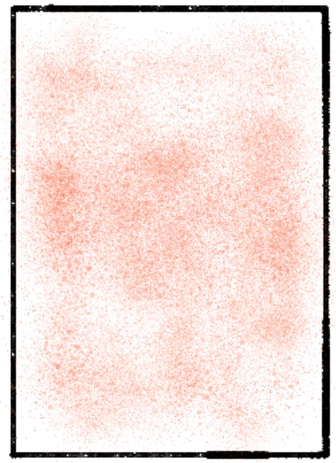
org.apache.kafka.common.errors.SerializationException:

Unknown magic byte!







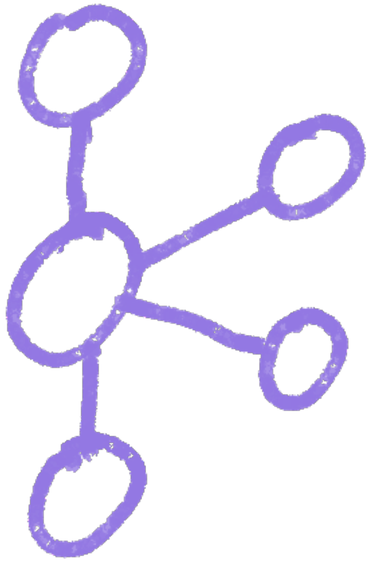


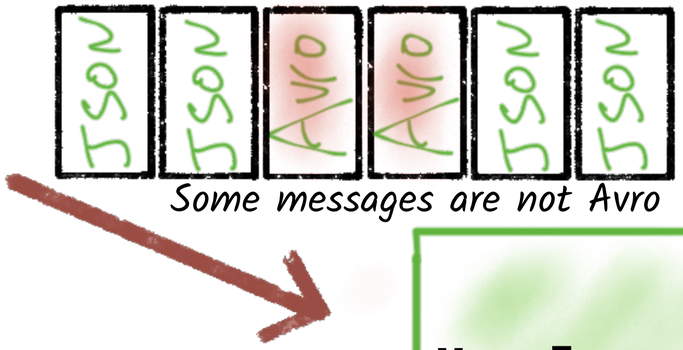
ⓘ Use the correct Converter for the source data

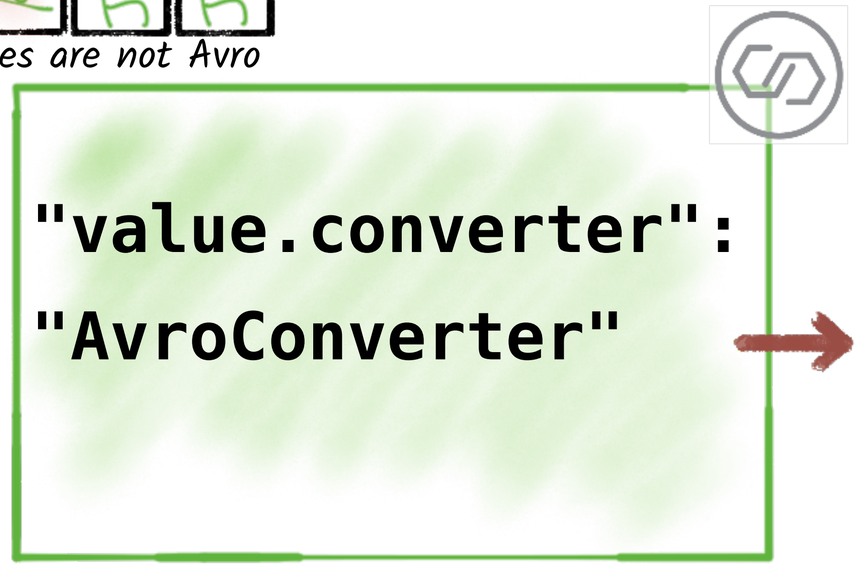
Mixed serialisation methods

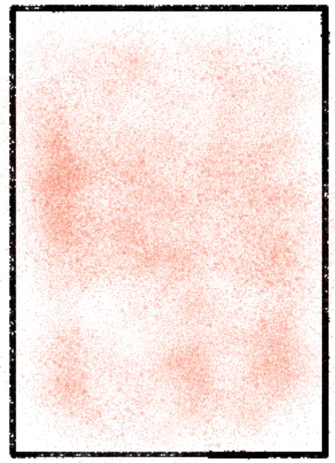
org.apache.kafka.common.errors.SerializationException:

Unknown magic byte!









ⓘ Use error handling to deal with bad messages

Error Handling and DLQ

Handled

Convert

-> read/write from Kafka

-> [de]-serialisation

Transform

Not Handled

Start

-> Connections to a data store

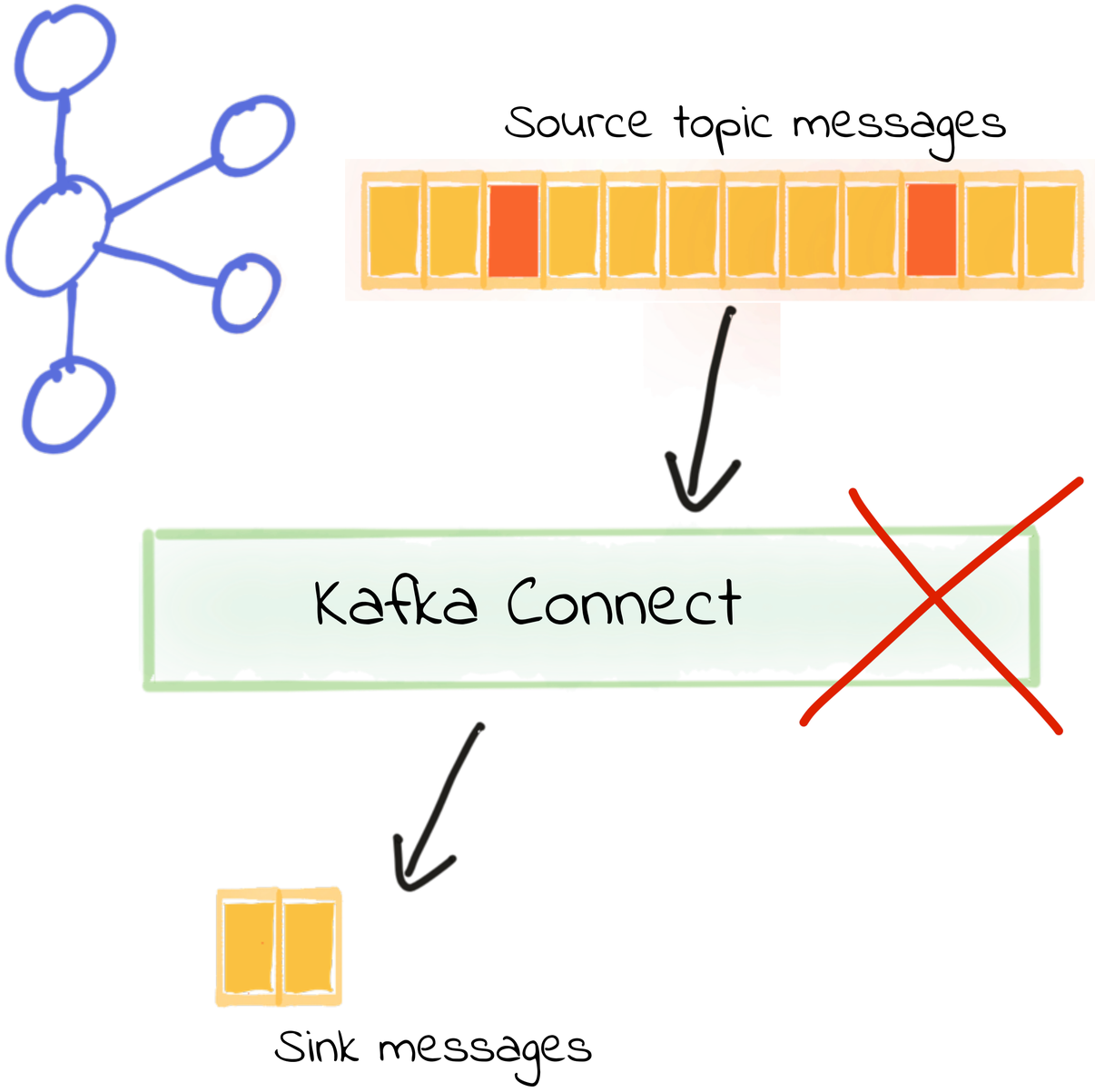
Poll / Put

-> Read/Write from/to data store*

* can be retried by Connect

https://cnfl.io/connect-dlq

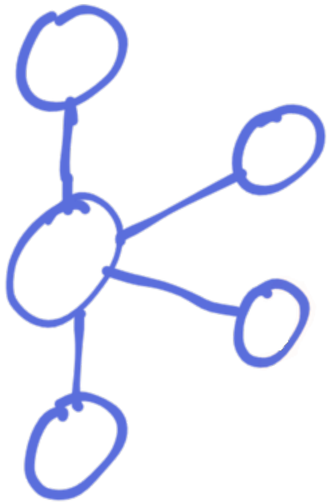
Fail Fast

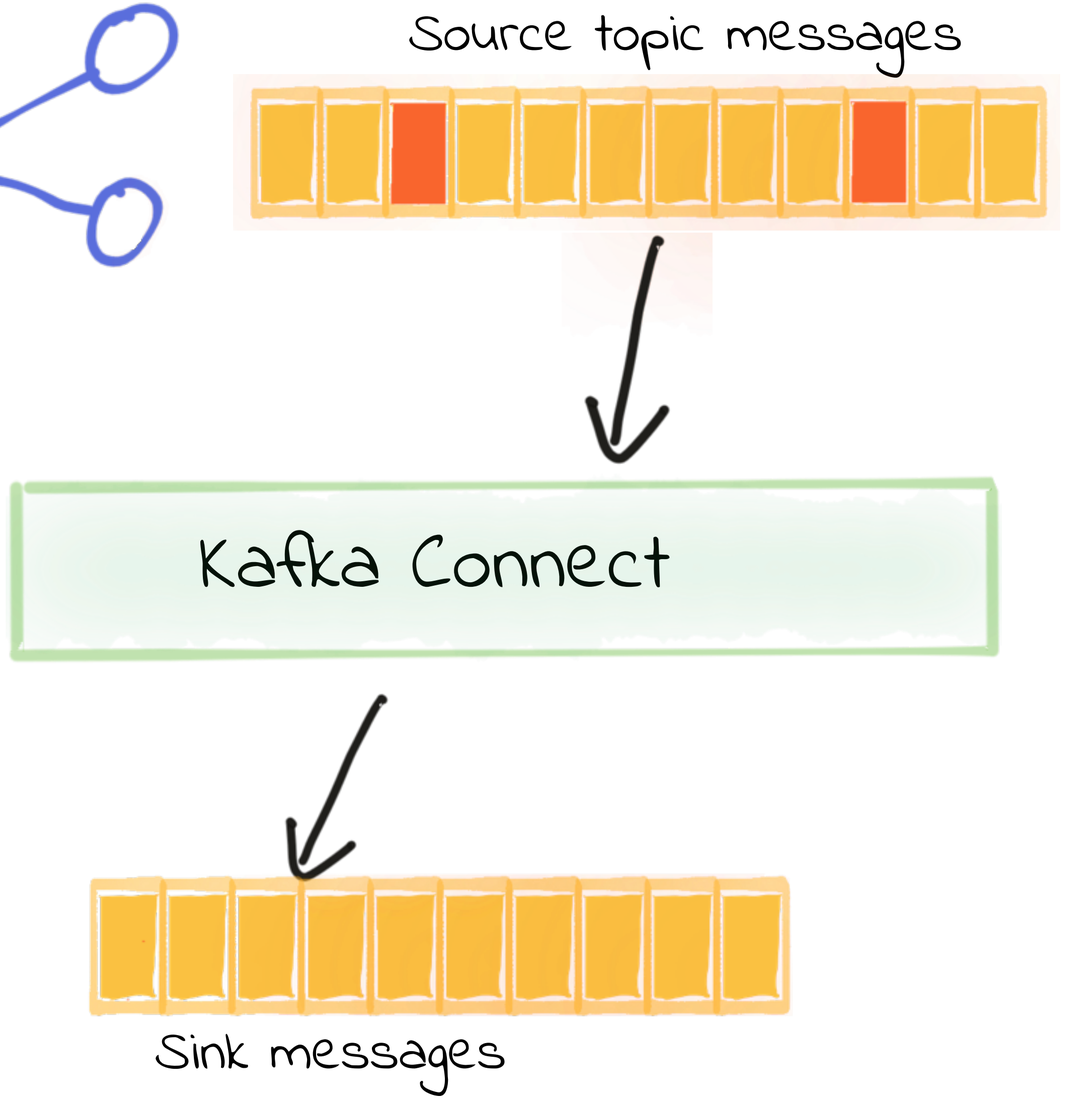


https://cnfl.io/connect-dlq

confluent

YOLO ¯_(ツ)_/¯



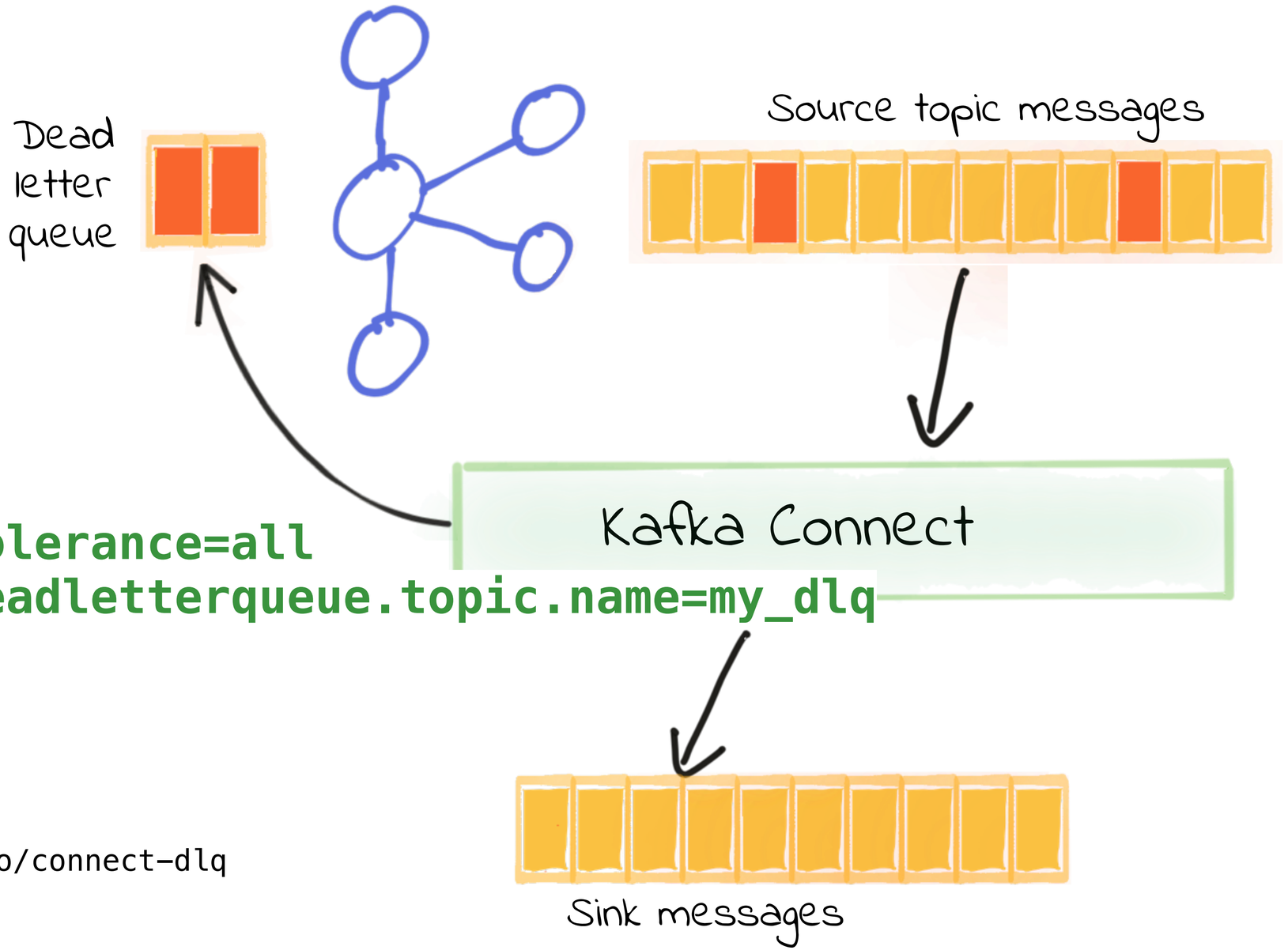



```
errors.tolerance=all
```

https://cnfl.io/connect-dlq

confluent

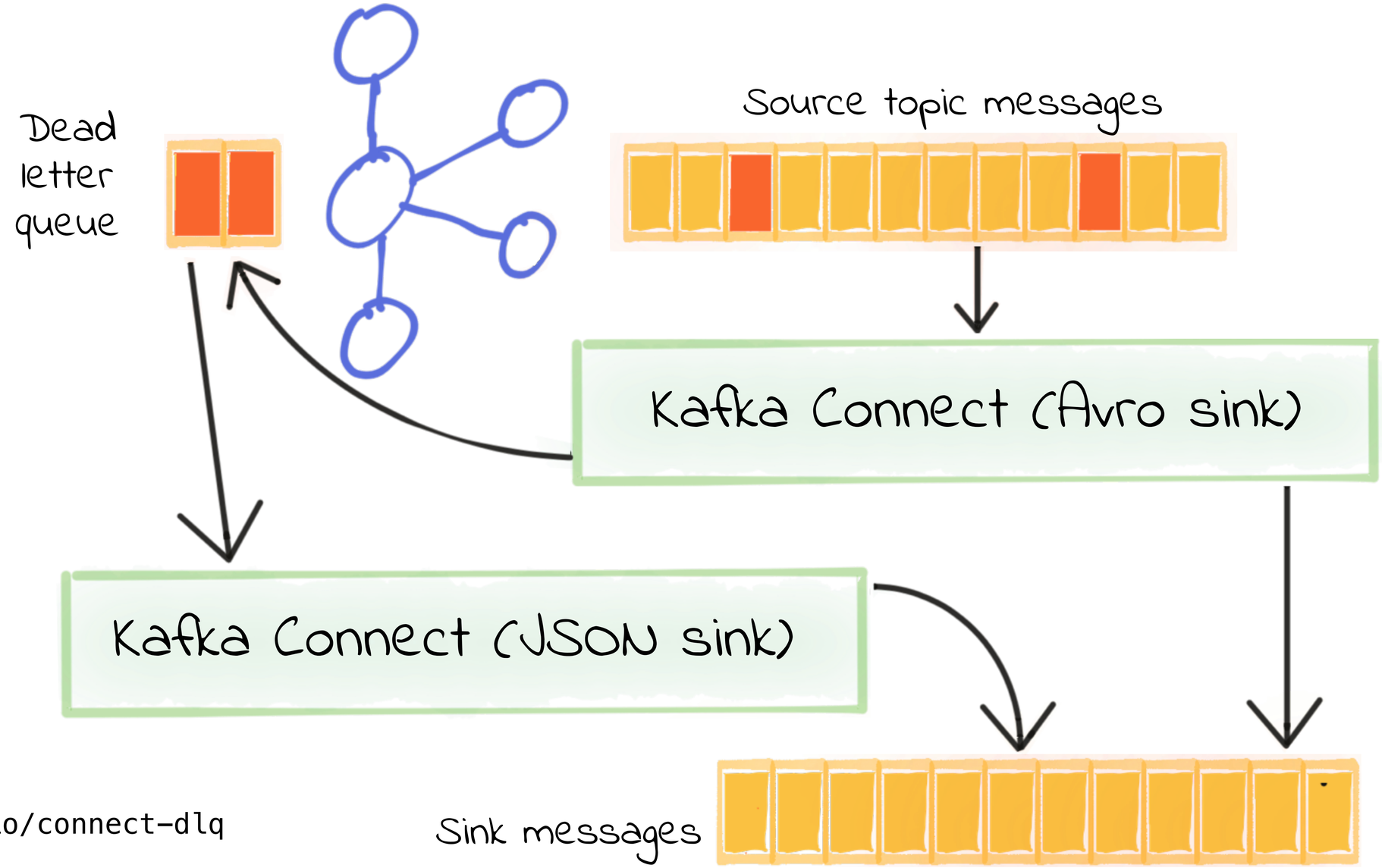
Dead Letter Queue



https://cnfl.io/connect-dlq

confluent

Re-processing the Dead Letter Queue



https://cnfl.io/connect-dlq

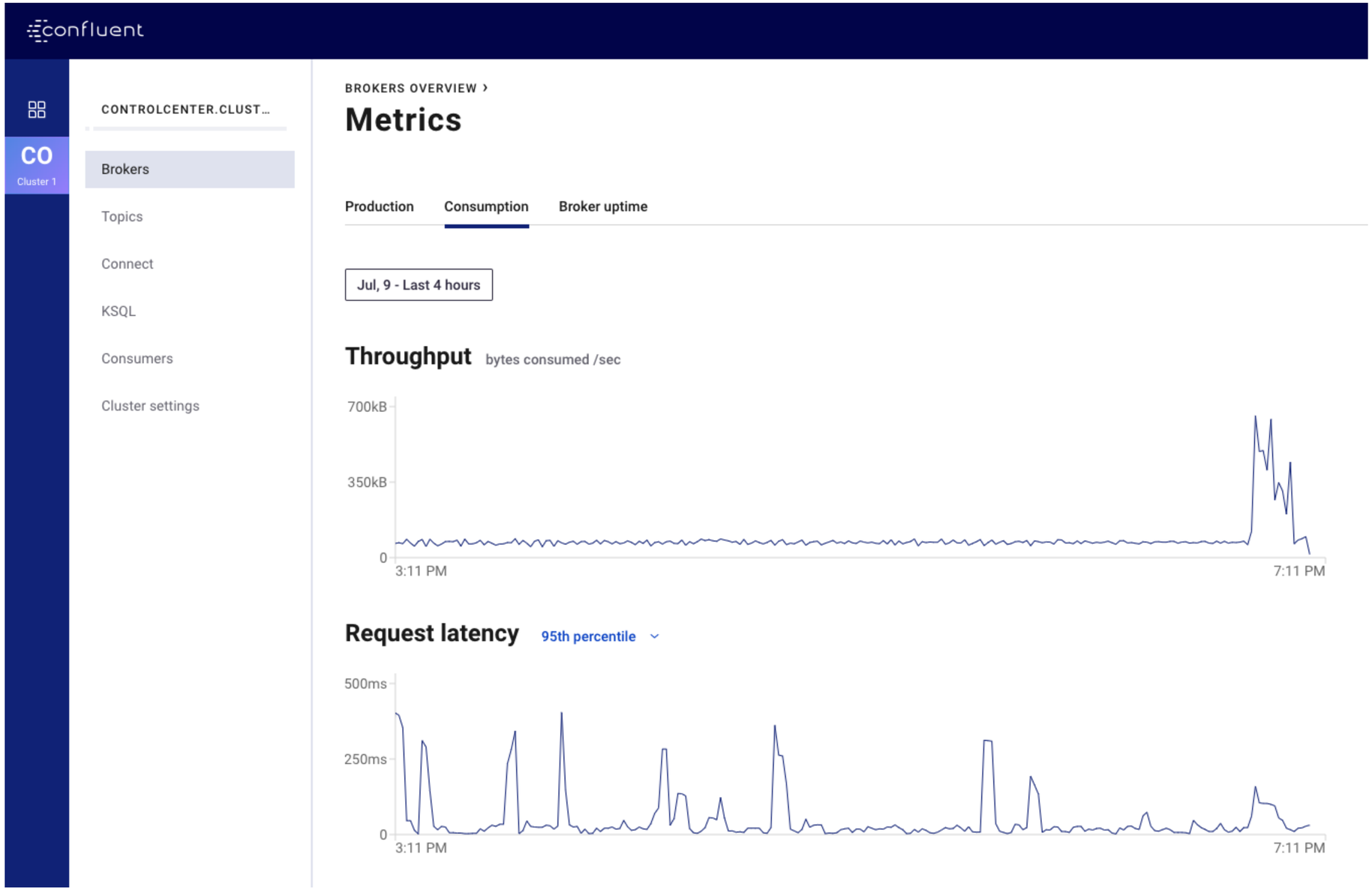
Metrics and Monitoring

REST API

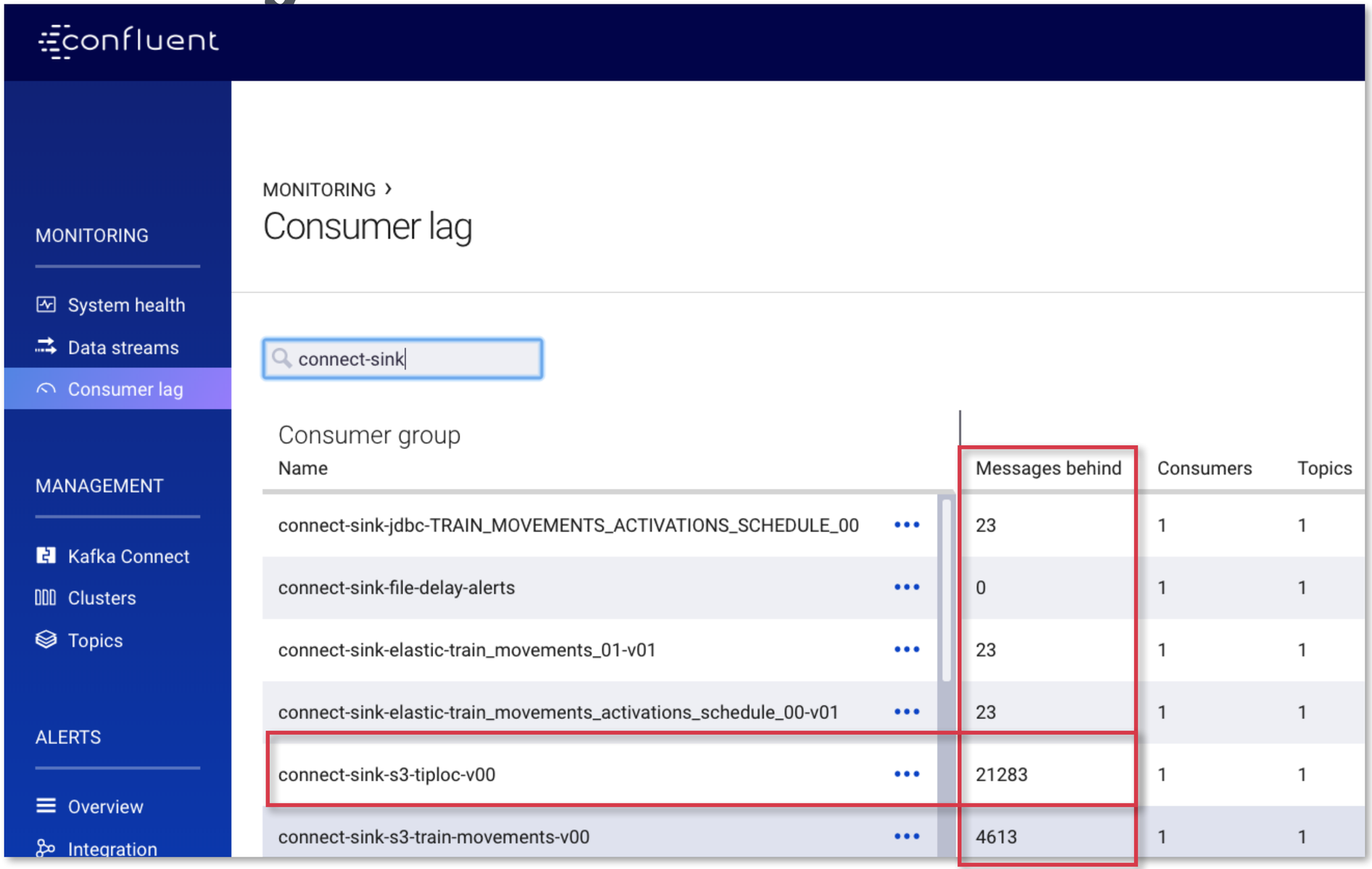
```
[~] Robin@asgard02-2.local
$ curl -s "http://localhost:8083/connectors"| \
    jq '.[]'| \
    xargs -I{connector_name} curl -s "http://localhost:8083/connectors/"{connector_name}"/status" | \
    jq -c -M '[.name,.connector.state,.tasks[].state]|join(":|:")' | \
    column -s : -t | \
    sed 's/\"//g' | \
    sort

sink-elastic-orders-00      |  RUNNING  |  RUNNING
sink-elastic-orders-01      |  RUNNING  |  RUNNING
source-debezium-orders-00   |  RUNNING  |  RUNNING
[~] Robin@asgard02-2.local
$
```

http://go.rmoff.net/connector-status

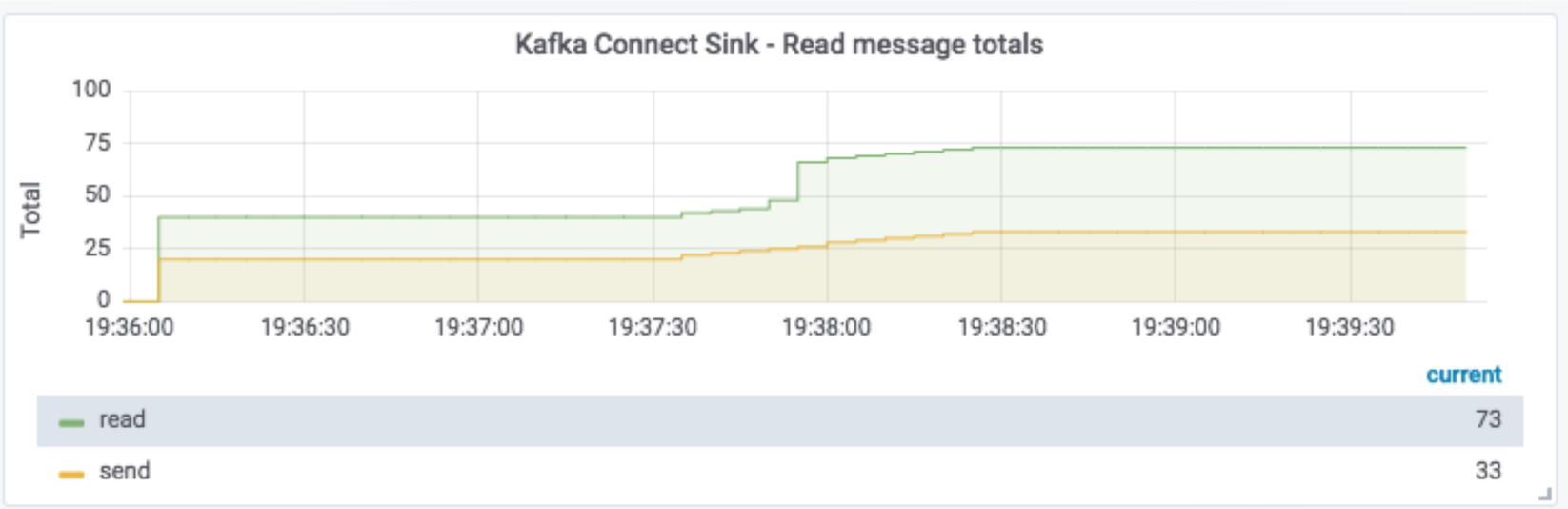
Confluent Control Center

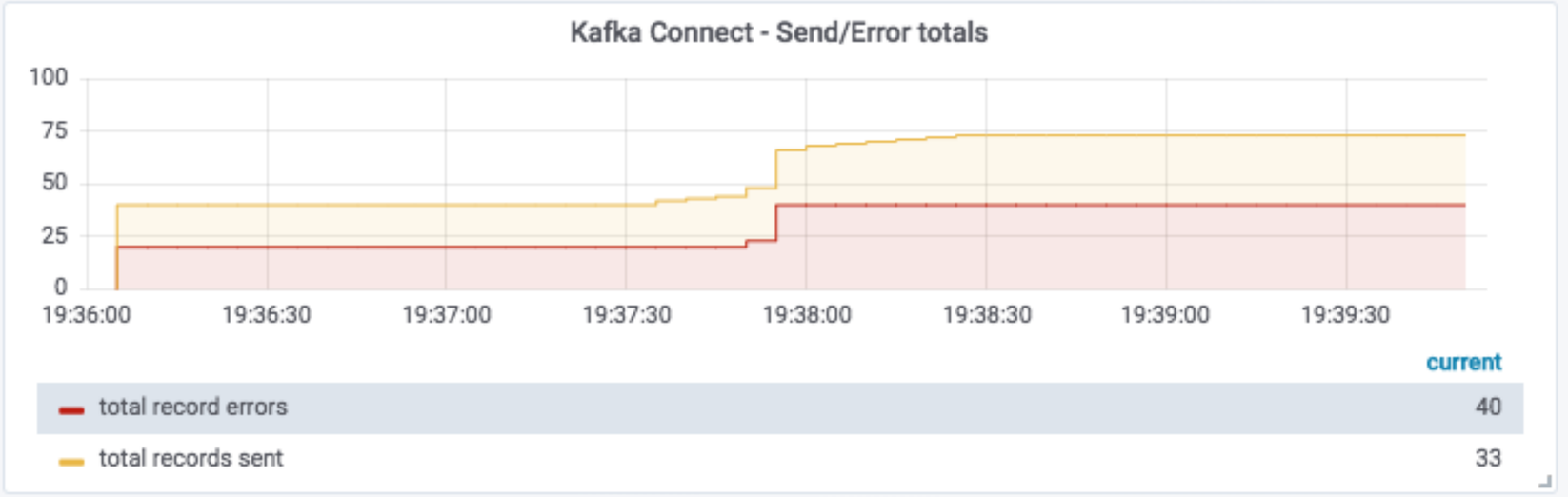


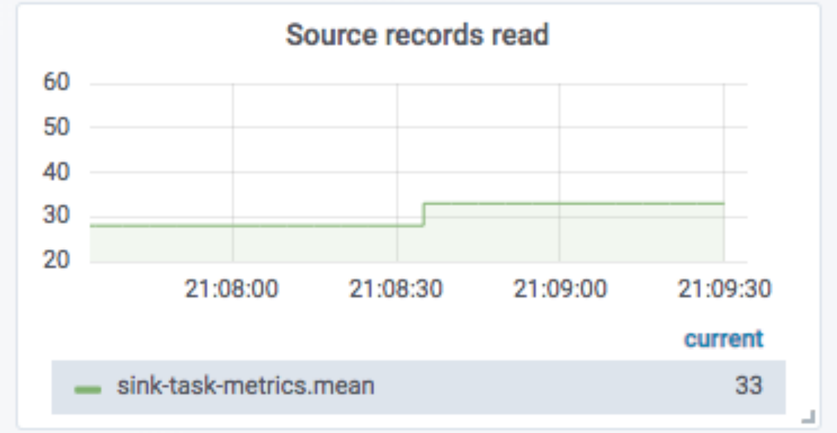
Consumer lag

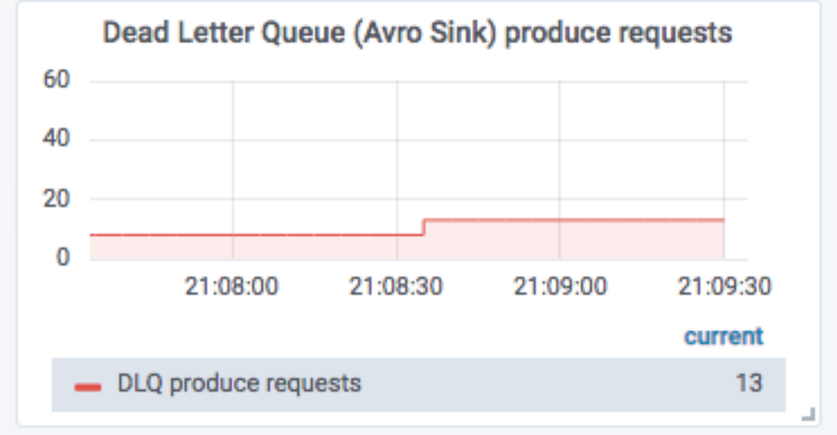


JMX


Kafka Connect Sink - Read message totals
Total
100
75
50
25
0
19:36:00
19:36:30
19:37:00
19:37:30
19:38:00
19:38:30
19:39:00
19:39:30
current
read 73
send 33


Kafka Connect - Send/Error totals
100
75
50
25
0
19:36:00
19:36:30
19:37:00
19:37:30
19:38:00
19:38:30
19:39:00
19:39:30
current
total record errors 40
total records sent 33

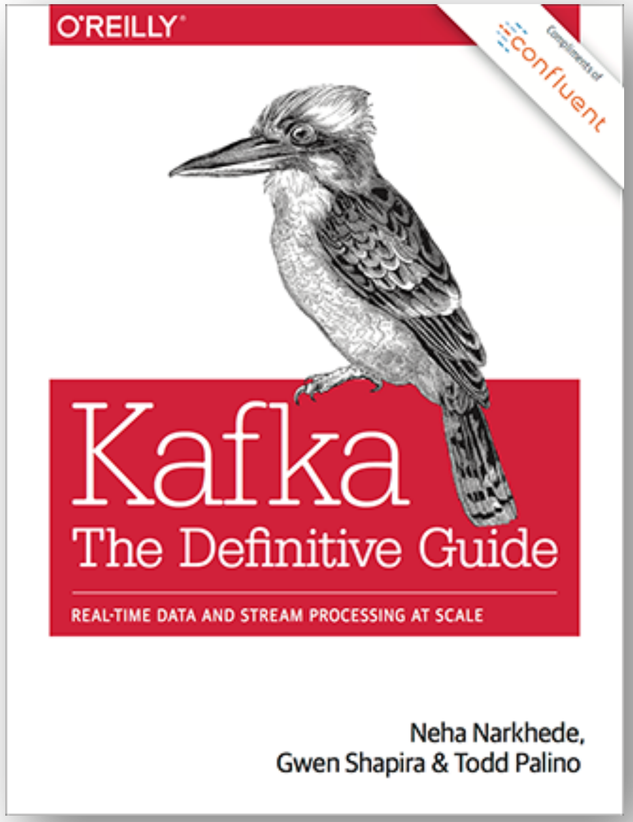

Source records read
60
50
40
30
20
21:08:00
21:08:30
21:09:00
21:09:30
current
sink-task-metrics.mean 33

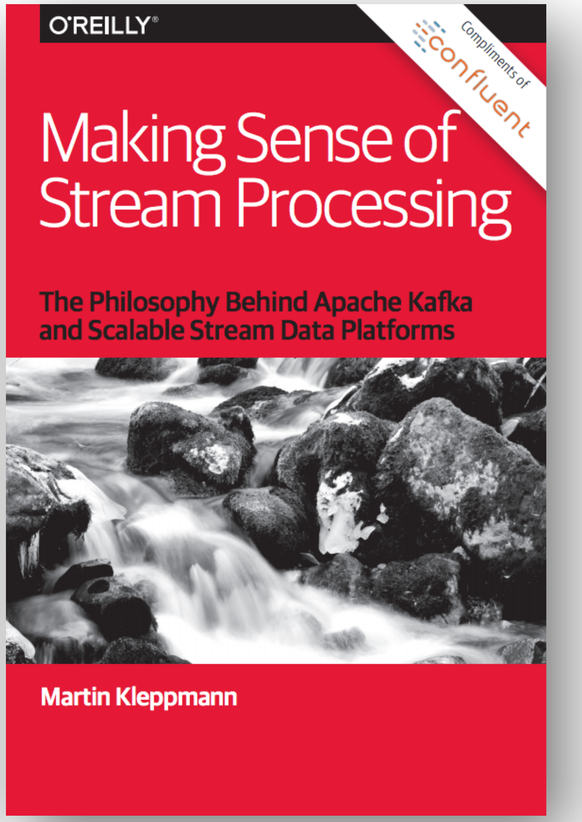

Dead Letter Queue (Avro Sink) produce requests
60
40
20
0
21:08:00
21:08:30
21:09:00
21:09:30
current
DLQ produce requests 13

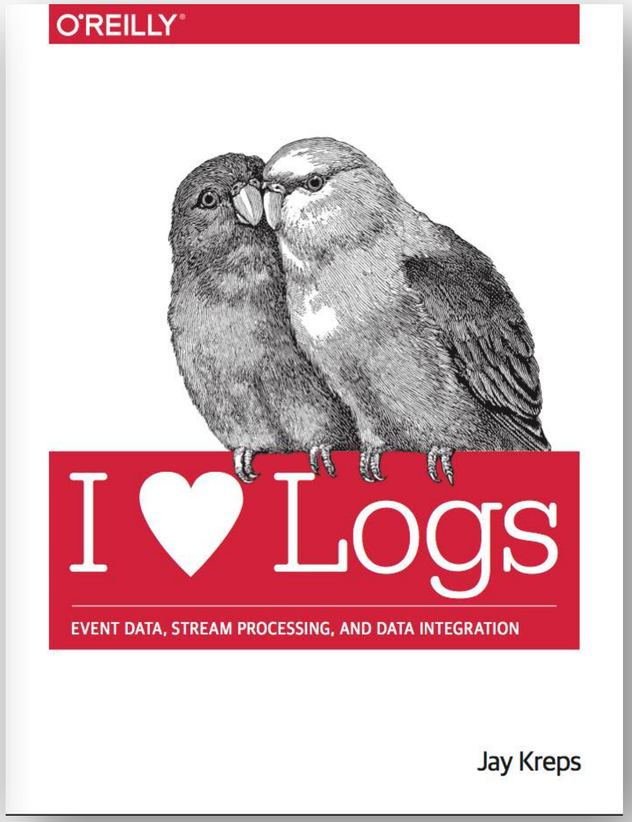
Fully Managed Kafka as a Service

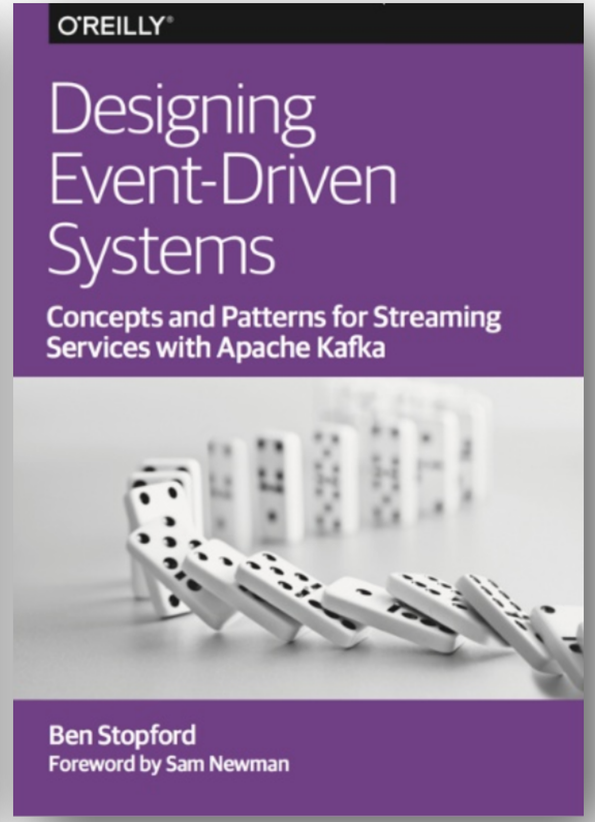
Free Books!

http://cnfl.io/book-bundle









@rmoff #KafkaMeetup

#EOF

💬 Join the Confluent Community Slack group at http://cnfl.io/slack

https://talks.rmoff.net