

Kakfa-connect-workshop



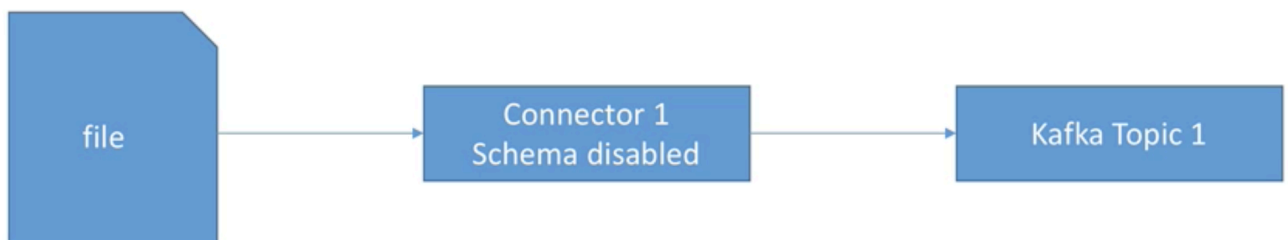
Requirements:

Github repo: <https://github.com/rvanrijn/workshop-apache-kafka-connect>

Lab 1) FileStreamSourceConnector in standalone mode

Goal:

- Read a file and load the content directly into Kafka.
- Run a connector in standalone mode (useful for development).



Learning:

- Understand how to configure a connector in standalone mode.

Steps:

Go to the repository and start the kafka cluster

\$ **docker-compose up kafka-cluster**

Kakfa-connect-workshop



Use one of these based on your OS:

Linux / Mac

```
docker run --rm -it -v "$(pwd)":/tutorial --net=host landoop/fast-data-dev bash
```

Windows Command Line:

```
docker run --rm -it -v %cd%:/tutorial --net=host landoop/fast-data-dev bash
```

Windows Powershell:

```
docker run --rm -it -v ${PWD}:/tutorial --net=host landoop/fast-data-dev bash
```

we launch the kafka connector in standalone mode:

```
cd /workshop-apache-connect/source/demo-1
```

This directory contains three files:

- **worker.properties**
- **file-stream-demo-standalone.properties**
- **demo-file.txt**

create the topic we write to with 3 partitions

```
$ kafka-topics --create --topic demo-1-standalone --partitions 3 --replication-factor 1 --zookeeper 127.0.0.1:2181
```

Check if Topics are visible in the browser, visit: <http://localhost:3030/kafka-topics-ui/#/>

Now go to to the next page.

Kakfa-connect-workshop



3 TOPICS

☐ System Topics

Search topics:

demo-1-standalone

3 Partitions × 1 Replication

...

logs-broker

1 Partitions × 1 Replication

...

position-reports

5 Partitions × 1 Replication | 1configs

avro

Powered by [Landoop](#)

The newly created topic should be visible in the topics list.

Within the `/workshop-apache-connect/source/demo-1` run:

`$ connect-standalone worker.properties file-stream-demo-standalone.properties`

write some data to the demo-file.txt! and check if the data is written the topic! Check:

<http://localhost:3030/kafka-topics-ui/#/cluster/fast-data-dev/topic/n/demo-1-standalone/>

demo-1-standalone

DATA

PARTITIONS 3

CONFIGURATION

Total Messages Fetched: 1. Data type: json

filter

All partitions

⏮ ⏭

TOPIC

TABLE

RAW DATA

⏴

Key:

Value: Hi

Offset: 0
Partition: 0

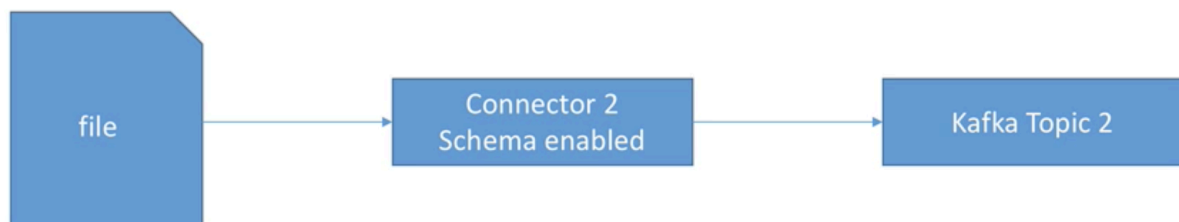
Kakfa-connect-workshop



Lab 2) FileStreamSourceConnector in distributed mode:

Goal:

- Read a file and load the content directly into Kafka
- Run in distributed mode on our Kafka Connect Cluster



Learning:

- Understand how to configure a connector in distributed mode
- Get a first feel for Kafka Connect Cluster
- Understand the schema configuration option

2. FileStreamSourceConnector in distributed mode:

create the topic we're going to write to

```
docker run --rm -it --net=host landoop/fast-data-dev bash
```

```
kafka-topics --create --topic demo-2-distributed --partitions 3 --replication-factor 1 --zookeeper 127.0.0.1:2181
```

you can now close the new shell

head over to 127.0.0.1:3030 -> Connect UI

Kakfa-connect-workshop



Create a new connector -> File Source

New Connector

Search

Sources

Twitter
Use the Twitter API to stream data into Kafka

Yahoo Finance
Stream stock and currency exchange rates into Kafka

File
Tail files or folders and stream data into Kafka

Ftp
Tail remote FTP folders and bring messages in Kafka

Blockchain
Get blockchain.info data into Kafka

Jdbc
Stream data from SQL server into Kafka

Sinks

Elastic Search
Write data from Kafka to Elastic Search

Cassandra
Store Kafka data into Cassandra

InfluxDB
Store Kafka data into InfluxDB

MongoDB
Write Kafka data into MongoDB

HazelCast
Store Kafka data into HazelCast (RingBuffer)

Jdbc
Store Kafka data into SQL

Paste the configuration from source/demo-2/file-stream-demo-distributed.properties

Create New Connector

File → Kafka
class: org.apache.kafka.connect.file.FileStreamSourceConnector

HINTS

```
1 # These are standard kafka connect parameters, need for ALL connectors
2 name=file-stream-demo-distributed
3 connector.class=org.apache.kafka.connect.file.FileStreamSourceConnector
4 tasks.max=1
5 # Parameters can be found here: https://github.com/apache/kafka/blob/trunk/connect/file/src/main/java/org/apache/kafka/connect/file
  /FileStreamSourceConnector.java
6 file=demo-file.txt
7 topic=demo-2-distributed
8 # Added configuration for the distributed mode:
9 key.converter=org.apache.kafka.connect.json.JsonConverter
10 value.converter=org.apache.kafka.connect.json.JsonConverter
11 |
```

☐ Show cURL command ☐ Show Optional fields

CREATE

Kafka Connect: Configuration is valid.

Hit CREATE

Now that the configuration is launched, we need to create the file demo-file.txt

\$ docker ps

\$ docker exec -it <containerId> bash

\$ touch demo-file.txt

\$ echo "hi" >> demo-file.txt

Kakfa-connect-workshop



```
$ echo "hello" >> demo-file.txt
```

```
$ echo "from the other side" >> demo-file.txt
```

observe we now have json as an output, even though the input was text!

The screenshot shows the Kafka Connect UI for the topic 'demo-2-distributed'. The 'DATA' tab is active, displaying a table of messages. The first message has a key of null and a value of 'hi'. The second message has a key of null and a value of 'hi test'. The 'PARTITIONS' tab shows 3 partitions. The 'CONFIGURATION' tab is also visible.

TOPIC	TABLE	RAW DATA
		<p>► Key: { schema: null, payload: null }</p> <p>▼ Value:</p> <ul style="list-style-type: none">► schema: { type: string, optional: false }► payload: hi <p>Offset: 0 Partition: 0</p>
		<p>► Key: { schema: null, payload: null }</p> <p>▼ Value:</p> <ul style="list-style-type: none">► schema: { type: string, optional: false }► payload: hi test <p>Offset: 1 Partition: 0</p>

Read the topic data from the cli.

```
docker run --rm -it --net=host landoop/fast-data-dev bash
```

```
kafka-console-consumer --topic demo-2-distributed --from-beginning --bootstrap-server  
127.0.0.1:9092
```

Again, observe we now have json as an output, even though the input was text!

Kakfa-connect-workshop



Lab 3) TwitterSourceConnector in distributed mode:

Goal:

- Gather data from Twitter in Kafka Connect Distributed mode



Learning:

- Gather real data using: <https://github.com/Eneco/kafka-connect-twitter>

C) TwitterSourceConnector in distributed mode:

create the topic we're going to write to

```
$ docker run --rm -it --net=host landoop/fast-data-dev bash
```

```
$ kafka-topics --create --topic demo-3-twitter --partitions 3 --replication-factor 1 --zookeeper 127.0.0.1:2181
```

Start a console consumer on that topic

```
$ kafka-console-consumer --topic demo-3-twitter --bootstrap-server 127.0.0.1:9092
```

Follow the instructions at: <https://github.com/Eneco/kafka-connect-twitter#creating-a-twitter-application>

To obtain the required keys, visit <https://apps.twitter.com/> and Create a New App. Fill in an application name & description & web site and accept the developer agreement. Click on Create my access token and populate a file `twitter-source.properties` with consumer key & secret and the access token & token secret using the example file to begin with.


Kakfa-connect-workshop



Setup instructions for the connector are at: <https://github.com/Eneco/kafka-connect-twitter#setup>

fill in the required information at demo-3/source-twitter-distributed.properties

Create New Connector

 **Twitter → Kafka**
class: com.eneco.trading.kafka.connect.twitter.TwitterSourceConnector

HINTS

```
1 name=TwitterSourceConnector
2 connector.class=com.eneco.trading.kafka.connect.twitter.TwitterSourceConnector
3 tasks.max=1
4 twitter.token=
5 twitter.secret=
6 twitter.consumersecret=
7 twitter.consumerkey=
```

☐ Show cURL command ☐ Show Optional fields

CREATE

Config "twitter.token" requires a value
Config "twitter.secret" requires a value
Config "twitter.consumersecret" requires a value
Config "twitter.consumerkey" requires a value
Required config "topic" is not there

Launch the connector and start seeing the data flowing in!

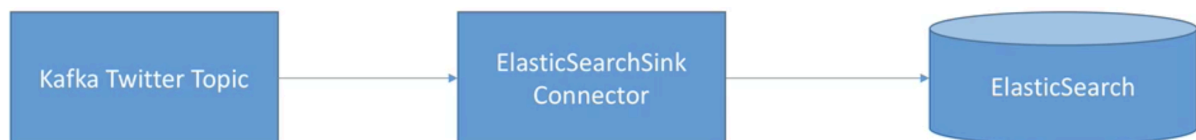
Kakfa-connect-workshop



Lab 4) ElasticSearchSinkConnector in distributed mode:

Goal:

- Start an ElasticSearch instance using Docker
- Sink a topic with multiple partitions to ElasticSearch
- Run in distributed mode with multiple tasks



Learning:

- Learn about the tasks.max parameter
- Understand how Sink Connectors work

1) Source connectors

Start our kafka cluster

\$ docker-compose up kafka-cluster elasticsearch postgres

Wait 2 minutes for the kafka cluster to be started

A) ElasticSearch Sink

Info here: http://docs.confluent.io/3.2.0/connect/connect-elasticsearch/docs/elasticsearch_connector.html

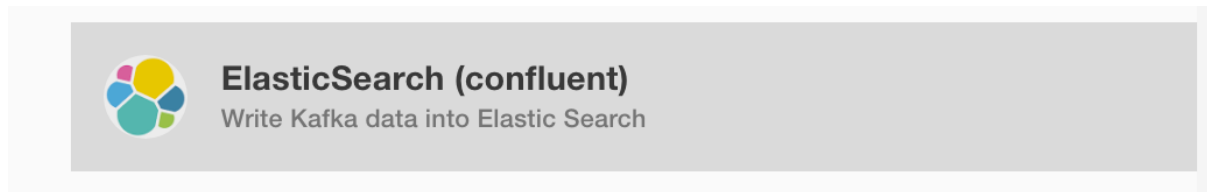
We make sure elasticsearch is working. Replace 127.0.0.1 by 192.168.99.100 if needed.

Check: <http://127.0.0.1:9200/>

Kakfa-connect-workshop



Go to the connect UI and create a new Source connector – ElasticSearch (confluent):



and apply the configuration: sink/demo-elastic/sink-elastic-twitter-distributed.properties

Visualize the data at:

http://127.0.0.1:9200/_plugin/dejavu The type: kafka-connect should appear. See image below:



http://docs.confluent.io/3.1.1/connect/connect-elasticsearch/docs/configuration_options.html

Counting the number of tweets:

http://127.0.0.1:9200/demo-3-twitter/_count

You can download the data from the UI to see what it looks like

We can query elasticsearch for users who have a lot of friends, see query-high-friends.json

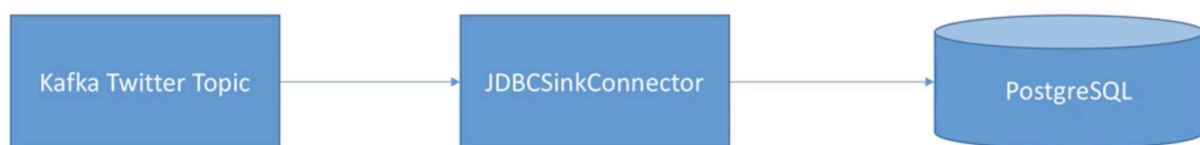
Kakfa-connect-workshop



Lab 5) JdbcSinkConnector in distributed mode:

GOAL:

- Start a PostgreSQL instance using Docker
- Run in distributed mode with multiple tasks



Learning:

- Learn about the JDBC Sink Connector

Create a Sink – Jdbc Connector and copy-paste the sink/demo-postgres/ sink-postgres-twitter-distributed.properties

New Connector

Search

Sources



Twitter

Use the Twitter API to stream data into Kafka



Yahoo Finance

Stream stock and currency exchange rates into Kafka



File

Tail files or folders and stream data into Kafka



Ftp

Tail remote FTP folders and bring messages in Kafka



Blockchain

Get blockchain.info data into Kafka



Jdbc

Stream data from SQL server into Kafka

Sinks



Elastic Search

Write data from Kafka to Elastic Search



Cassandra

Store Kafka data into Cassandra



InfluxDB

Store Kafka data into InfluxDB



MongoDB

Write Kafka data into MongoDB



HazelCast

Store Kafka data into HazelCast (RingBuffer)



Jdbc

Store Kafka data into SQL

Kakfa-connect-workshop



If all went well the twitter feed should sink(ing) into PostgreSQL. To check if the data is there. Open another CLI and run the following:

```
$ docker run -it --rm --net=host jbergknoff/postgresql-client
postgres://postgres:postgres@127.0.0.1:5432/postgres
```

And check the following table:

```
1. psql (9.3.11, server 9.5.7)
2. WARNING: psql major version 9.3, server major version 9.5.
3.      Some psql features might not work.
4. Type "help" for help.
5. postgres=# \dt
6.          List of relations
7.  Schema |      Name      | Type | Owner
8.  -----+-----+-----+-----
9.  public | demo-3-twitter | table | postgres
10. (1 row)
11. postgres=# select count(*) from "demo-3-twitter";
12. count
13. -----
14.      17
15. (1 row)
16. postgres=# \q
```