# Technical task

Application consists of two microservices:

1. Dataloader - reads data from local directory and loads into Kafka topics, scheduled to run every 24 hours
2. REST service - lets request customer by id, top N, customers/products count

## Technology

* Java 17
* SpringBoot 2.6.6
* Kafka, ksqldb (tested on confluent-7.1.0)
* ksqldb-api-client 7.1.0

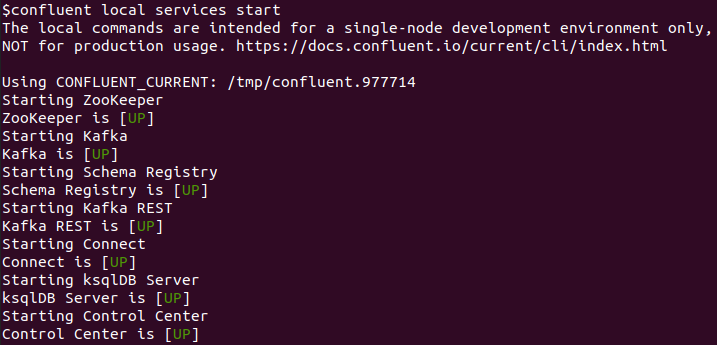
## Services

1. http://localhost:8902/dataloader - manualy run Dataloader from REST interface (Dataloader runs also as Spring scheduled task)
2. http://localhost:8902/customer?id=6003 - get customer by id
3. http://localhost:8902/customer/top?n=3 - get top N last registered customers
4. http://localhost:8902/customer/count - get number of customers
5. http://localhost:8902/product/count - get number of products

## Test steps

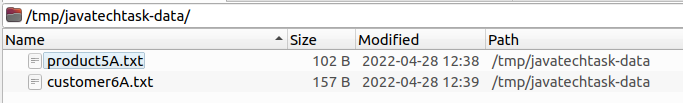
1. Start Kafka and etc. with command:

confluent local services start

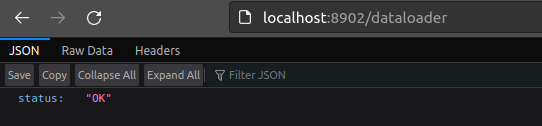
2. Create Kafka topic bank-products with command from file “kafka/src/main/resources/doc/topics/bank-products.topic”

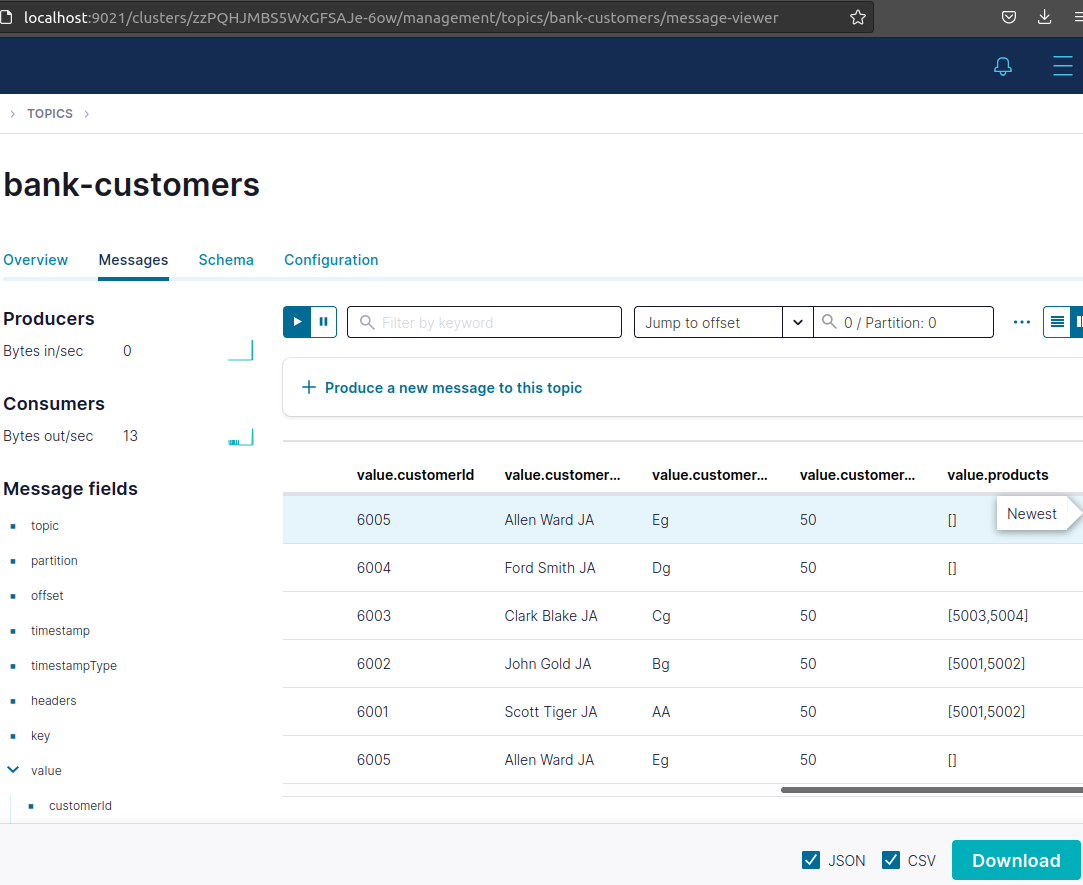
3. Create Kafka topic bank-customers with command from file “kafka/src/main/resources/doc/topics/bank-customers.topic”

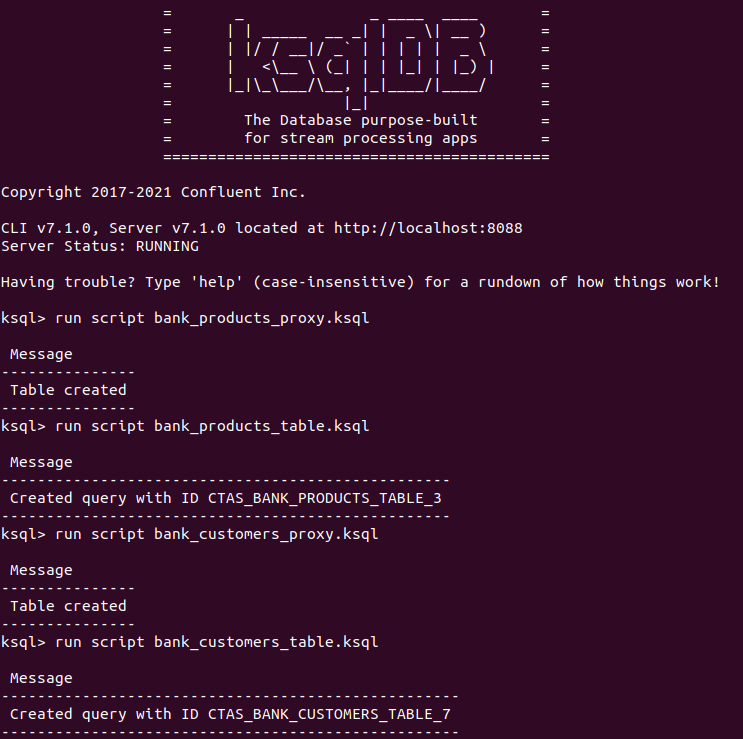
4. Copy sample data files from “kafka/src/main/resources/doc/sample-data” to local directory specified in application property “dataloader.base.path”

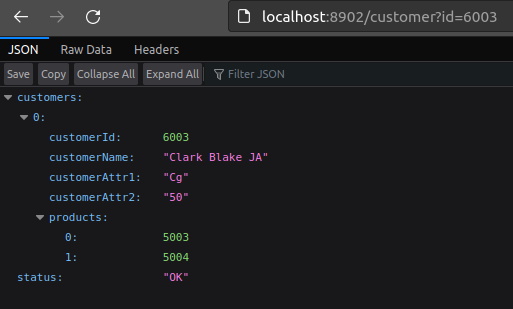
5. Run Application

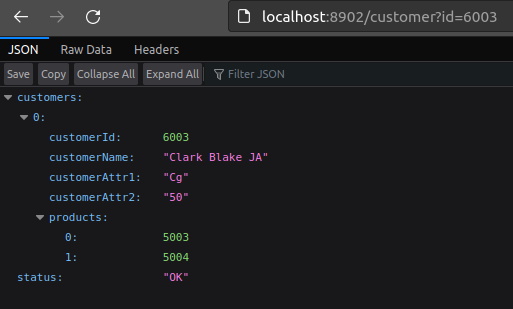
6. Start Dataloader manualy

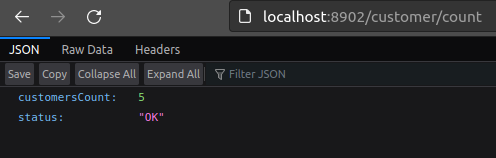
7. Browse topic data in confluent control center

8. Create KSQL tables by running scripts from directory “kafka/src/main/resources/doc/ksql/scripts” in KSQL tool

9. Test REST http://localhost:8902/customer?id=6003

10. Test REST http://localhost:8902/customer/top?n=2

10. Test REST http://localhost:8902/customer/count

11. Test REST http://localhost:8902/product/count

## Data format

1. Data is in delimited format, with fields separator “;”.
2. The data for each object is stored in a single line. A single file can contain many lines, but of same entity – customer or product.
3. What entity data is stored in the file is recognized by file prefix “product” or “customer”.
4. Mentioned parameters are controlled by Application properties.
5. Field can be multi-valued with separator “,”.

## How does the Dataloader work?

1. Finds files named customer\*.txt, product\*.txt in base path.
2. Creates new sub-directory for load session (batch)
3. Moves the found files to a new directory
4. Reads lines from moved files
5. Creates product or customer objects
6. Sends objects into Kafka topics

## Run

mvn spring-boot:run