Big Bridge Ventes

Starting services

All the required services described in *docker-compose.yml* file. To start up all services type following command:

```
docker-compose up -d
```

In case you need for start only selected service you may use following command:

```
docker-compose up -d <name_of_service>
```

For extended use cases and related commands look at docker documentation.

Note that most of services will up automatically except *oracle* because it requires built image with proprietary source. See related link in *docker-compose.yml* file or download my own build:

```
https://1drv.ms/u/s!AqLgEs1phh2qjlKV9xn3T21Q4k5V
```

And import image using following command:

```
docker load < oracle-database-image.tar
```

Executing command inside container

For getting inside container

```
docker-compose run --rm cli
```

Importing Feedback data to Riak

Execute script inside *cli* service container:

```
python3 /app/import-riak.py -f /data/Feedback.csv
```

Data will be splitted to the different buckets according to the row *asin* field value which interpretated as primary key.

Importing data to MongoDb

Order.json file stores JSON and is ready from importing.

Execute script inside *cli* container:

```
/app/import-mongo.py -f /data/Order.json -c orders
```

Product related data splitted into two CSV files. Prepare this files for importing with the following command:

```
python3 /app/prepare-products-for-mongo.py -o /data/Product.json /data/Product.csv /data/BrandByProduct.csv
```

After that import generated JSON file:

```
python3 /app/import-mongo.py -f /data/Product.json -c products
```

Importing Social Network data to Neo4j

Execute script inside *graph* service container:

`/app/import-neo4j.sh`

Check first lines of script for actual data files location.

```
PERSON_FILE=/import/person_0_0.csv
POST_FILE=/import/post_0_0.csv
PERSON_KNOWS_PERSON_FILE=/import/person_knows_person_0_0.csv
POST_CREATED_BY_FILE=/import/post_hasCreator_person_0_0.csv
```

Importing Invoice data to Cassandra

Schema:

1. **INVOICE**
2. **ORDERLINE**

ORDERLINE contains extrafield *orderId*

Before importing data you have to prepare data.

Execute script inside *cli* service container:

`python3 /app/prepare-invoice-data.py -o /data/invoice.csv /data/orderline.csv /data/Invoice.xml`

Note that you should pass column names exactly in the same order as the header of generated CSV file.

Execute DDL inside *column* service container:

```
CREATE KEYSPACE commerce
  WITH REPLICATION = {
   'class' : 'SimpleStrategy',
   'replication_factor' : 1
  };
USE commerce;
CREATE TABLE invoice(
    OrderId TEXT PRIMARY KEY,
    OrderDate TEXT,
    PersonId TEXT,
    TotalPrice FLOAT
);
CREATE TABLE orderline(
    orderlineId TEXT PRIMARY KEY,
    asin TEXT,
    brand TEXT,
    orderId TEXT,
    price FLOAT,
```

```
    productId TEXT,
    title TEXT
);
```

After that use *cqlsh* import command inside *column* service container:

```
COPY invoice FROM '/import/invoice.csv' WITH HEADER=TRUE;
COPY orderline FROM '/import/orderline.csv' WITH HEADER=TRUE;
```