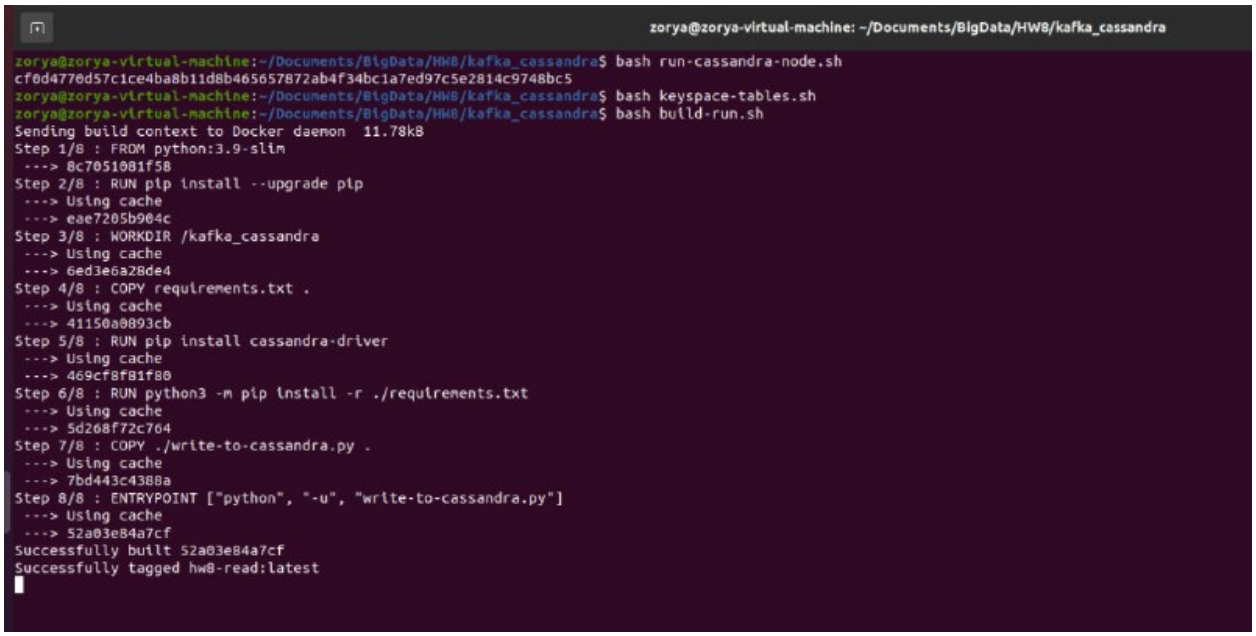
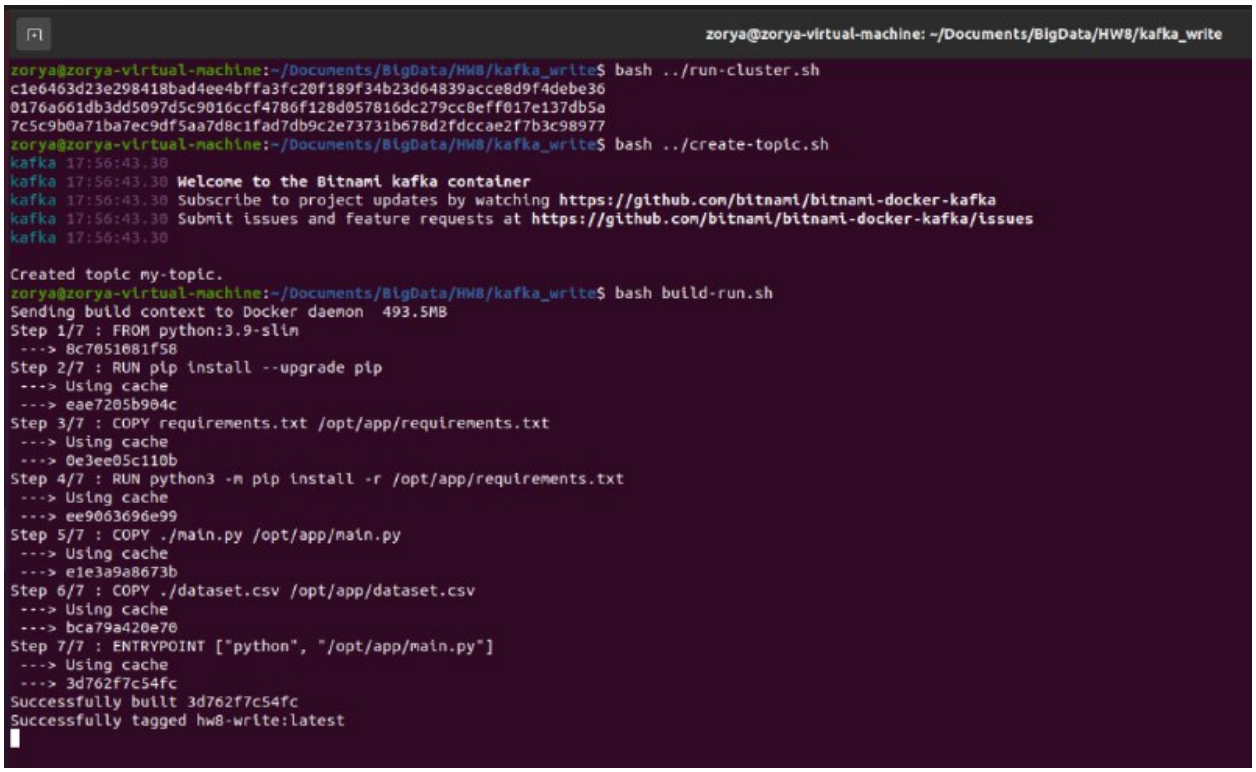
Спершу, запустимо kafka producer та kafka consumer - cassandra writer:





Перевіримо працюючі контейнери:

```
zorya@zorya-virtual-machine:~/Documents/BigData/HW8$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                                                                                  NAMES
9ee781d37fe8   hw8-read                 "python -u write-to-…"   18 minutes ago   Up 18 minutes                                                                                          kafka-consumer
b5b1eba62072   hw8-write                "python /opt/app/mai…"   18 minutes ago   Up 18 minutes                                                                                          kafka-producer
cf0d4770d57c   cassandra:latest         "docker-entrypoint.s…"   20 minutes ago   Up 20 minutes   7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp, :::9042->9042/tcp   node1
7c5c9b0a71ba   bitnami/kafka:latest     "/opt/bitnami/script…"   21 minutes ago   Up 21 minutes   9092/tcp                                                                               kafka-server
0176a661db3d   bitnami/zookeeper:latest "/opt/bitnami/script…"   21 minutes ago   Up 21 minutes   2181/tcp, 2888/tcp, 3888/tcp, 8080/tcp                                                 zookeeper-server
zorya@zorya-virtual-machine:~/Documents/BigData/HW8$
```

Даємо системі попрацювати 30 хв та зупиняємо продюсера та консюмера.

Після цього, через cassandra node перевіримо скільки записів зараз є в таблиці:

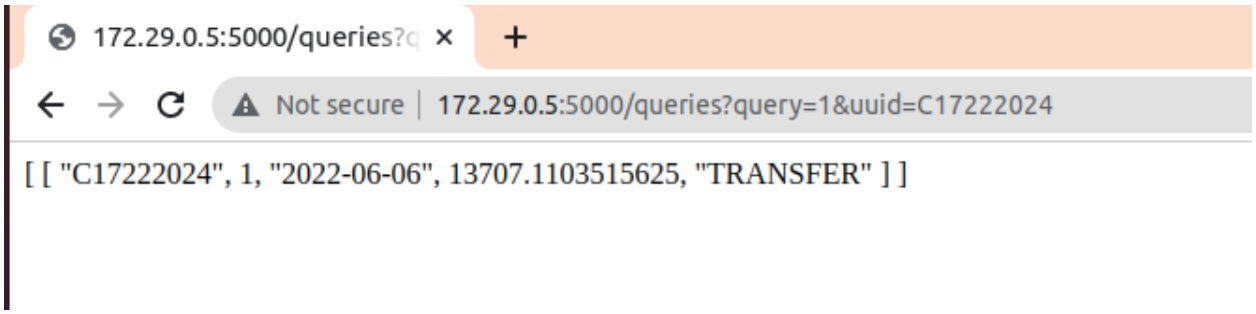
```
zorya@zorya-virtual-machine:~/Documents/BigData/HW8$ docker exec -it node1 cqlsh

Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.4 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh>
cqlsh> use my_keyspace;
cqlsh:my_keyspace> select count(*) from transactions;

 count
-------
 18314

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:my_keyspace> exit;
zorya@zorya-virtual-machine:~/Documents/BigData/HW8$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                                                                                  NAMES
cf0d4770d57c   cassandra:latest         "docker-entrypoint.s…"   38 minutes ago   Up 38 minutes   7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp, :::9042->9042/tcp   node1
7c5c9b0a71ba   bitnami/kafka:latest     "/opt/bitnami/script…"   39 minutes ago   Up 39 minutes   9092/tcp                                                                               kafka-server
0176a661db3d   bitnami/zookeeper:latest "/opt/bitnami/script…"   39 minutes ago   Up 39 minutes   2181/tcp, 2888/tcp, 3888/tcp, 8080/tcp                                                 zookeeper-server
zorya@zorya-virtual-machine:~/Documents/BigData/HW8$
```

Тепер, запустимо контейнер з апкою:

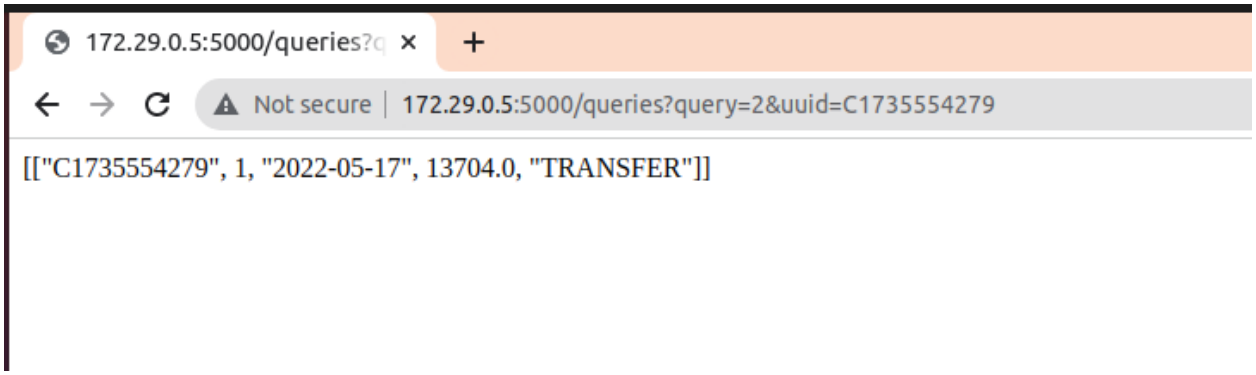
```
zorya@zorya-virtual-machine: ~/Documents/BigData/HW8/cassandra_app
zorya@zorya-virtual-machine:~/Documents/BigData/HW8/cassandra_app$ bash build-run.sh
Sending build context to Docker daemon  8.704kB
Step 1/9 : FROM python:3.9-slim
 ---> 8c7051081f58
Step 2/9 : RUN pip install --upgrade pip
 ---> Using cache
 ---> eae7205b904c
Step 3/9 : WORKDIR /cassandra_app
 ---> Using cache
 ---> 804cdbe7a695
Step 4/9 : COPY requirements.txt .
 ---> Using cache
 ---> 153d15248584
Step 5/9 : RUN pip install cassandra-driver
 ---> Using cache
 ---> f30408ec2c7e
Step 6/9 : RUN python3 -m pip install -r ./requirements.txt
 ---> Using cache
 ---> 3419b7b5e148
Step 7/9 : COPY ./app.py .
 ---> Using cache
 ---> 23d59865d08c
Step 8/9 : COPY write_to_cassandra.py .
 ---> 95a04650c813
Step 9/9 : ENTRYPOINT ["python", "-u", "app.py"]
 ---> Running in 6f0412f925c2
Removing intermediate container 6f0412f925c2
 ---> e4c15d2c4bc8
Successfully built e4c15d2c4bc8
Successfully tagged my-app:latest
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
 * Running on http://172.29.0.5:5000 (Press CTRL+C to quit)
```

Викличемо по одній з кожних queries (я зробила перевірку і так як всі uuid в нашому файлі унікальні відповідно результати всіх запитів будуть складатись лише з одного стовпця):

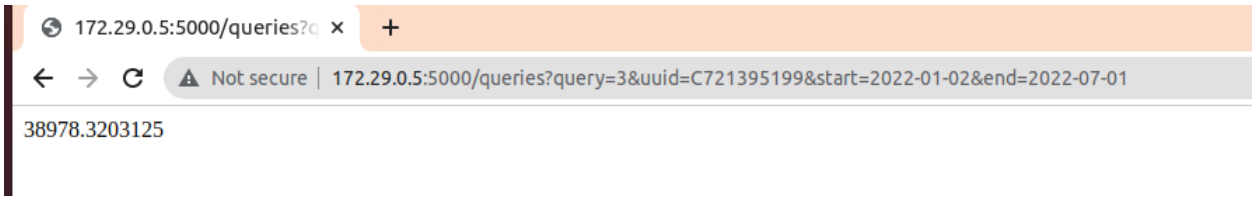
http://172.29.0.5:5000/queries?query=1&uuid=C17222024 :



http://172.29.0.5:5000/queries?query=2&uuid=C1735554279 :



http://172.29.0.5:5000/queries?query=3&uuid=C721395199&start=2022-01-02&end=2022-07-01 :



Перевіримо також консольний вивід:

```
Successfully tagged my-app:latest
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
 * Running on http://172.29.0.5:5000 (Press CTRL+C to quit)
172.29.0.1 - - [10/Jun/2022 18:49:58] "GET /queries?query=1&uuid=C17222024 HTTP/1.1" 200 -
172.29.0.1 - - [10/Jun/2022 18:50:23] "GET /queries?query=2&uuid=C1735554279 HTTP/1.1" 200 -
172.29.0.1 - - [10/Jun/2022 18:50:51] "GET /queries?query=3&uuid=C721395199&start=2022-01-02&end=2022-07-01 HTTP/1.1" 200 -
```