Big Data - Milestone 2 - Tests

Adrianna Klimczak, Adam Kowalczyk, Wojciech Replin and Marcel Wenka

November 29, 2021

1 Functional tests

1.1 Full system test

Test objective: having set up and enabled the whole system wait for the data to be processed and query the hive server to verify the data saved on HDFS.

Test steps:

1. Build and start the docker containers by running `docker-compose up -d` in the project root directory.
2. In a web browser go to `localhost:8443/nifi/` and log in using credentials present in `docker-compose.yml` file.
3. Upload and add `nifi-template.xml` template.
4. Enable all components.
5. Wait approximately 3 to 5 minutes for the data to be fetched, filtered, transformed and put to HDFS.
6. Open Hive Server command line interface by opening Docker Desktop and pressing the button shown in Figure 1.
7. Connect to Hive using the following command `beeline -u jdbc:hive2://`.
8. Verify the data by running HiveQL queries to all tables (e.g. `select * from pollutionwarsaw;`). Table names are in format `pollution<cityname>` and `weather<cityname>` (Figure 2).

Expected result: the data is present in the database
Actual result (pass) can be seen on figure 2

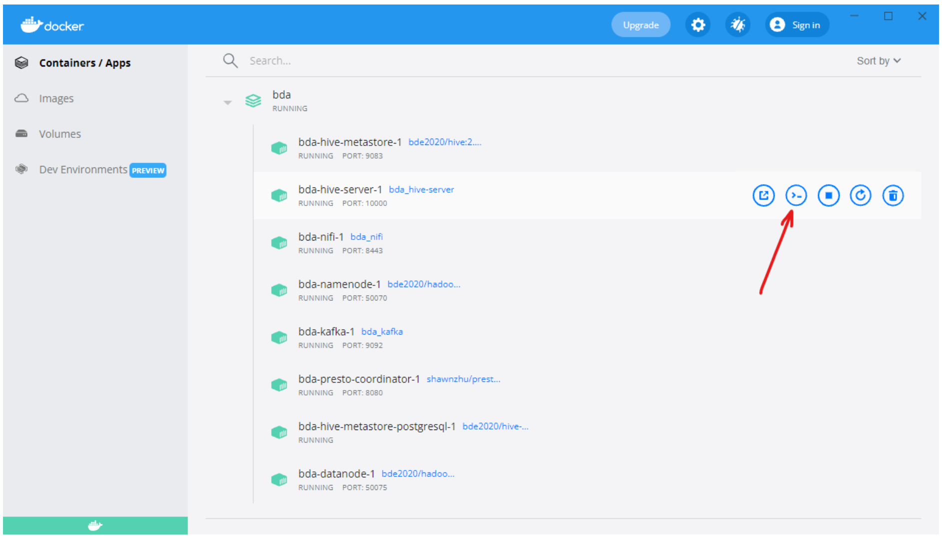

Figure 1: Means of accessing the Hive Server CLI

```
0: jdbc:hive2://> select * from pollutionwarsaw;
```

pollutionwarsaw.id	pollutionwarsaw.ts	pollutionwarsaw.aqi	pollutionwarsaw.lon	pollutionwarsaw.lat	pollutionwarsaw.measured
a997935c-36c1-44e6-893c-2680d8f5eeee	2021-11-26 11:51:49.837	87	52.22517	21.014784	1637924400
6c0e5ac5-1dde-4585-a5df-b9223db6dbf7	2021-11-26 11:51:27.59	87	52.22517	21.014784	1637924400
54f257e7-0110-446a-bba9-9e10ae508cb1	2021-11-27 19:47:45.616	87	52.22517	21.014784	1637924400
8ff24762-6848-408d-ab47-51aef6515ed5	2021-11-27 19:47:45.623	87	52.22517	21.014784	1637924400
2f78de6b-e038-4583-a546-f2b3c8b30761	2021-11-27 19:47:45.73	87	52.22517	21.014784	1637924400
5f1244e2-c182-45ec-ab83-b9dafd6c87df	2021-11-27 19:47:45.795	87	52.22517	21.014784	1637924400
b87dd8fa-4efe-4a6e-a4f1-12c7f3bbd124	2021-11-27 19:47:45.835	87	52.22517	21.014784	1637924400
68bef5fc-e3b8-4277-bf13-e61d08b0af36	2021-11-27 19:47:45.671	87	52.22517	21.014784	1637924400
9757c838-de92-4e3e-a17d-8ffaee0bd963	2021-11-27 19:47:45.79	87	52.22517	21.014784	1637924400
35e04446-171c-44d8-b060-0ef35b40a916	2021-11-27 19:47:45.793	87	52.22517	21.014784	1637924400
58d8ad5b-97da-4250-bcdf-0e0dcdf620bd	2021-11-27 19:47:45.817	87	52.22517	21.014784	1637924400
8f4621c8-0a44-4e05-9f37-de079b8367da	2021-11-27 19:47:45.841	87	52.22517	21.014784	1637924400
b4edb402-3fd5-4eaf-bc9b-cb4cb8c2ad4e	2021-11-27 19:47:45.853	87	52.22517	21.014784	1637924400
4751cc45-0c1e-428d-8cc4-fc1ddce5953c	2021-11-27 19:47:45.882	74	52.22517	21.014784	1638039600
d481250e-1798-4bbe-957a-72a8ffeaef7e	2021-11-27 19:47:45.53	87	52.22517	21.014784	1637924400
2d169e00-f153-4c73-bd4b-2dae70caac88	2021-11-27 19:47:45.593	87	52.22517	21.014784	1637924400
d234e47b-2dde-4ec0-8b88-fa50f453654d	2021-11-27 19:47:45.604	87	52.22517	21.014784	1637924400
56c70a20-63ed-40df-8dc3-9d23a0a8bd82	2021-11-27 19:47:45.607	87	52.22517	21.014784	1637924400
3ab583c7-87d5-4f0d-ac5f-8300bf63a6c8	2021-11-27 19:47:45.72	87	52.22517	21.014784	1637924400
51036054-49e2-4478-a31e-f37c520ab1c1	2021-11-27 19:47:45.767	87	52.22517	21.014784	1637924400
d317647d-0279-48ce-89a9-f3b1c5c87f5c	2021-11-27 19:47:45.8	87	52.22517	21.014784	1637924400
bc5e9497-0493-4f8d-ac6b-d775570366e2	2021-11-27 19:47:45.82	87	52.22517	21.014784	1637924400
1ef8cc9e-f0c9-4c77-b541-13ea0f3009a2	2021-11-27 19:47:45.824	87	52.22517	21.014784	1637924400

Figure 2: Example test output for pollutionwarsaw table

1.2 System failure email test

Test objective: simulate system failure and check whether an email is sent to notify the user about a failure.

Test steps:

1. Follow steps 1-4 from the test 1.1 (setting up the environment and running all NIFI components).
2. Simulate a system failure. For instance, open configuration window for any JoltTransformJSON processor (e.g. Pollution Data Aquisition / Warsaw / Transform Pollution JSON). Stop the processor and invalidate the Jolt Specification (Figure 3).
3. Log into the email account that gets notified about failures (weather.pollution.alert@gmail.com) and verify whether and email was received (Figure 4).

Expected result: the email is sent
Actual result (pass) can be seen on figure 2

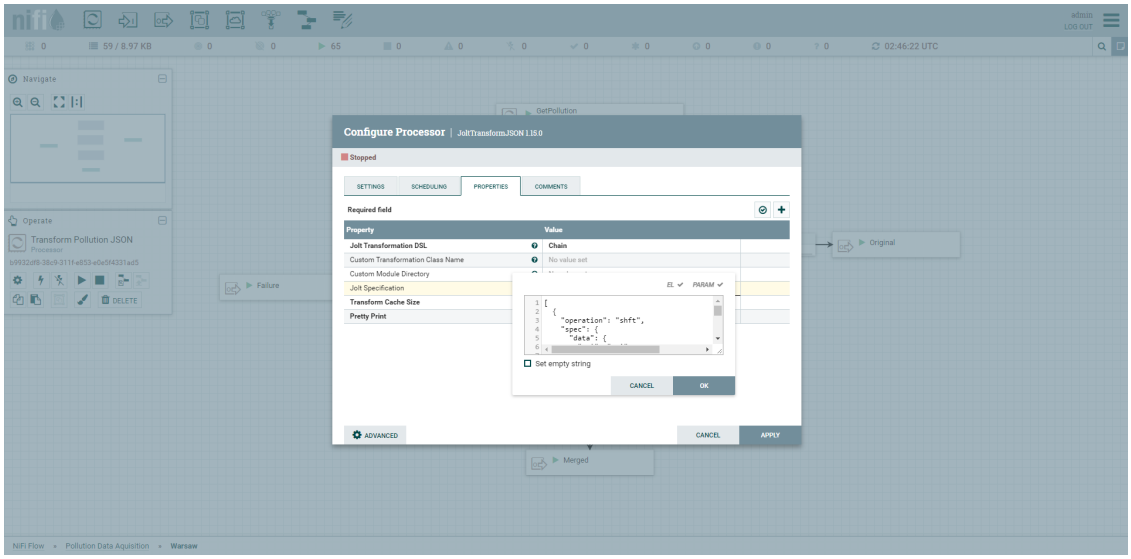

Figure 3: Invalid jolt specification

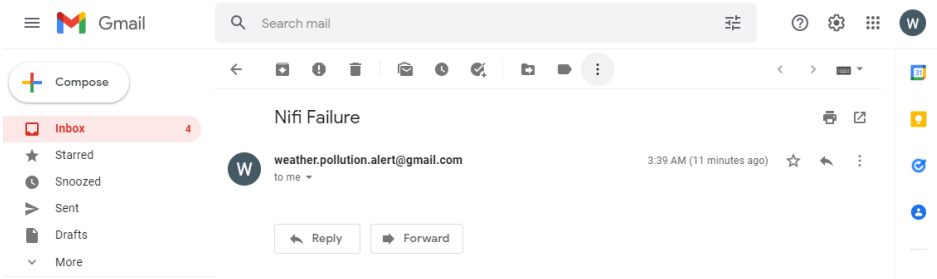

Figure 4: System failure email

1.3 Kafka queues test

Test objective: verify that Kafka producers are gathering data properly and without errors.
Test steps:

1. Follow steps 1-2 from the test 1.1 (setting up the environment).
2. Create `ConsumeKafka_2_6` processor
3. Create `LogAttribute` processor
4. Disable `LogAttribute` processor
5. Connect `ConsumeKafka_2_6` to `LogAttribute`
6. Configure `ConsumeKafka_2_6` processor as follows:
   (a) `Kafka Brokers`: `kafka:9092`
   (b) `Topic name(s)`: any of the available topic e.g. `pollutionwarsaw`
   (c) `Group ID`: `grid1`
   (d) `Offset Reset`: `earliest`
7. Run `ConsumeKafka_2_6` processor and wait 20 seconds

8. Verify that messages are queueing up to `LogAttribute` processor
9. Copy and paste `ConsumeKafka_2_6` processor
10. Configure new `ConsumeKafka_2_6` processor as follows:
    (a) `Topic name(s)`: error
11. Connect new `ConsumeKafka_2_6` to `LogAttribute`
12. Run new `ConsumeKafka_2_6` processor and wait 20 seconds
13. Verify that messages are not queueing up to `LogAttribute` processor from new `ConsumeKafka_2_6` (Figure 5)

Expected result: messages are queued up without errors
Actual result (pass) can be seen on figure 5

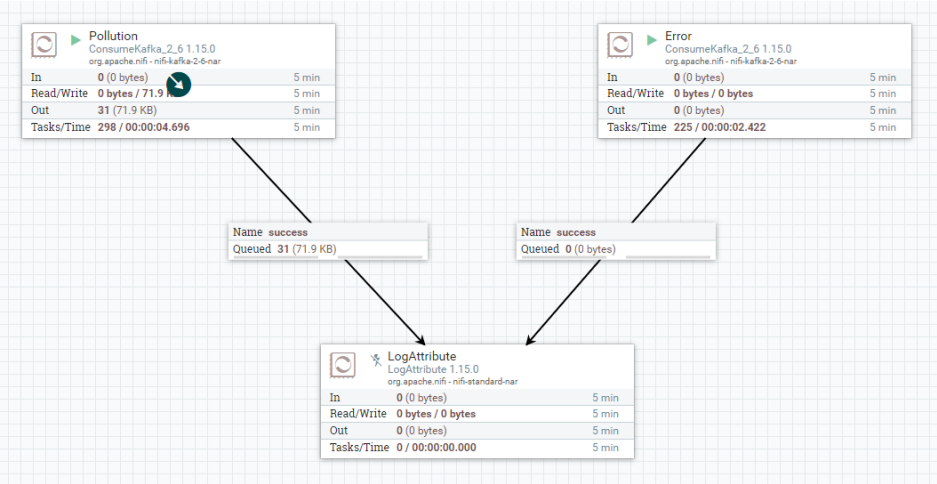

Figure 5: Kafka queues test test result

1.4 Data persistency test

Test objective: verify that the data is stored in a non-volatile memory.
Test steps:

1. Follow all steps from 1.1
2. Stop the environment by running the command `docker-compose stop`
3. Follow steps 1,5-8 from 1.1

Expected result: messages are queued up without errors
Actual result (pass) is identical to 2