VIETTEL GROUP

Assignment Report

BUILDING SIMPLE DATA PLATFORM

Dinh Viet Thanh

dvthanh19@gmail.com

Viettel Digital Talent 2024 Program
Major: Data Engineering

HO CHI MINH CITY, JUNE 2024

Contents

1 General

1.1 Requirements

For this assignment, the task is to construct a basic data platform utilizing the following technologies: Apache Kafka, Apache NiFi, Hadoop HDFS, and Apache Spark.

Architecture

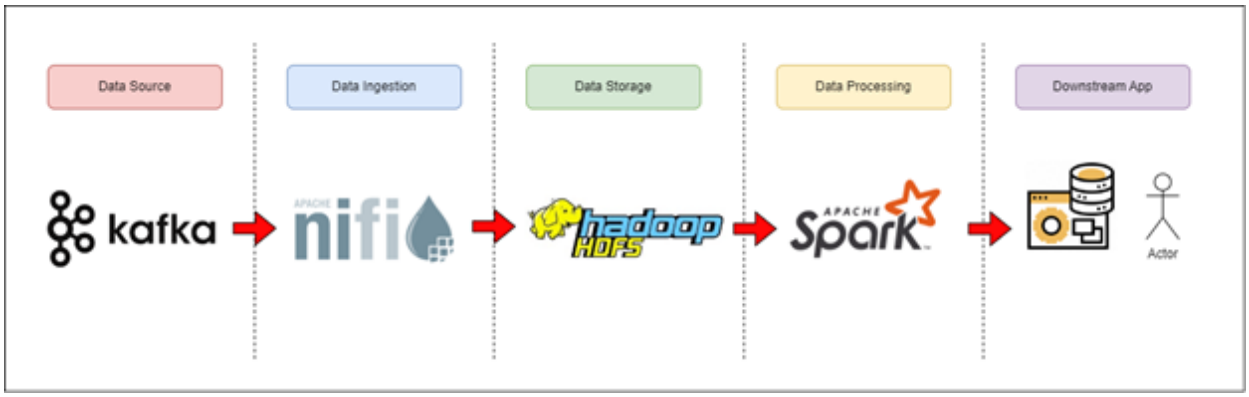

Figure 1: *Data platform architecture*

Input and Output

1. **Apache Kafka**:
   - Task: Develop a program to read from *log_action.csv* and push data to a Kafka Topic `vdt2024`, including the following field: student_code (number), activity (string), numberOfFile (number), timestamp (string)
2. **Apache NiFi**:
   - Task: Deploy Apache NiFi to pull data from the Kafka Topic `vdt2024`, process it, and store it in HDFS at the path `/raw_zone/fact/activity`
   - Data Format: Store the data in Parquet format
3. **Hadoop HDFS**:
   - Task: Store the file *danh_sach_sv_de.csv* into HDFS.
4. **Apache Spark**
   - Task: Write a program to process data stored in HDFS using Apache Spark.
   - Data Processing Requirement: Calculate the total number of files interacted with daily for each type of activity performed by each student. Save the result to an output file.

- Output File Details:
  - File Name: *student_ name.csv*
  - Format: CSV
  - Schema: date, student_code, student_name, activity, totalFile

2 Prepartion

To effectively deploy the data pipeline using Docker, we need to prepare the following components and configurations. This section outlines the necessary preparations.

2.1 Prerequisites

Before proceeding with the deployment, ensure that the following prerequisites are met:

- **Docker**: ensure Docker is installed and running on your system. Follow the official Docker installation guide for your operating system.
- **Docker Compose**: install Docker Compose, which is used to define and run multi-container Docker applications. Follow the official Docker Compose installation guide.
- **Python**: install Python, which will be used for scripting and running the Kafka producer program. Ensure you have Python 3.x installed. Follow the official Python installation guide for your operating system.
- **Docker images**: prepare the Docker images for the various technologies involved in the data pipeline.

2.2 Docker compose file

Create a *docker-compose.yml* file to define the services and their configurations. This file will specify how to run each component of the data pipeline as a container. There are several points that we need to notices:

- Image's name and version, dependent images.
- Ports and port mapping.
- Volume, mounted volume and networks.

You can see detail in my repository.

3 Deployment

3.1 Run the docker compose up

Start the Docker daemon and run the docker compose up:

```
docker-compose -f docker-compose.yml -p demo up -d
```

Then you will wait for it to install the image (if it is not available yet) and build.

Figure 2: *Docker compose up*

It is seen that all containers run successfully. You can also check in Docker application

3.2 Pipeline

Kafka and NiFi Firstly, go to Kafka UI at `localhost:8080` to check Kafka's status:

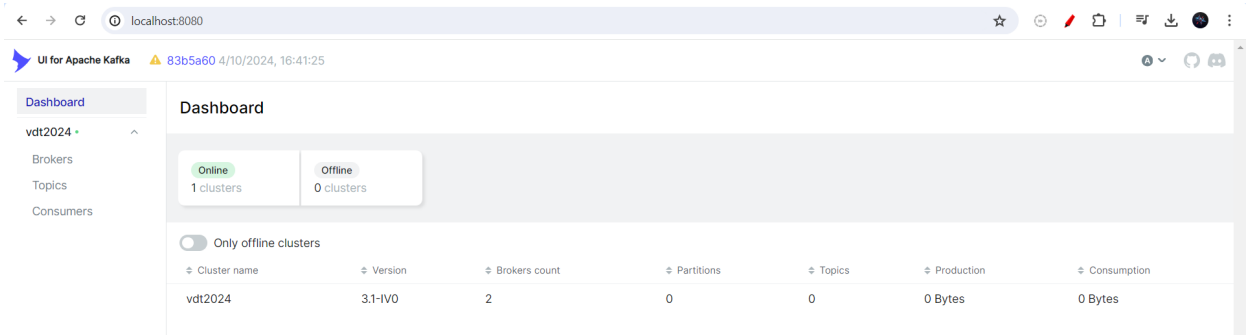

Figure 3: *Apache Kafka interface*

Second, go to NiFi UI at `localhost:8082/nfi` to check NiFi's status:

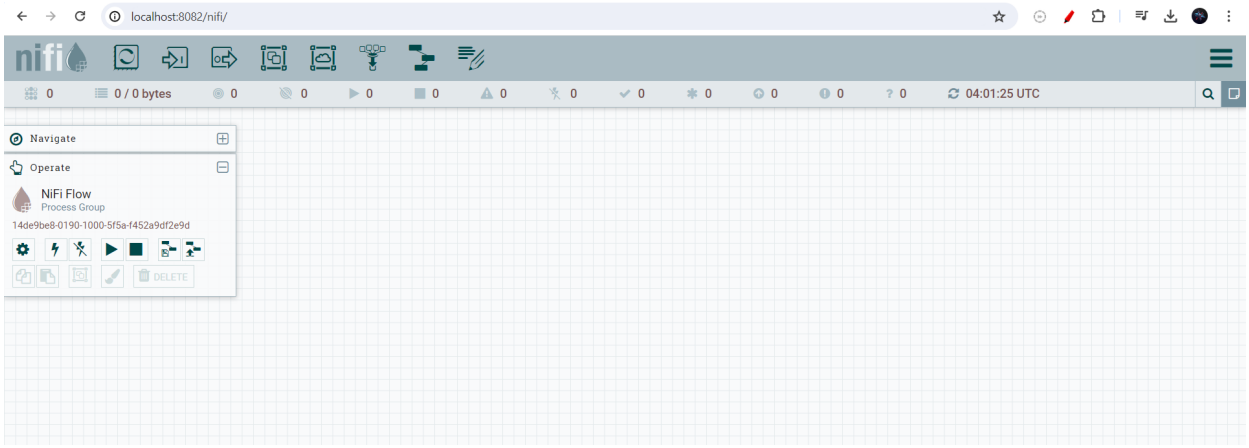

Figure 4: *Apache Kafka interface*

Then we will configure the NiFi to ingest data from Kafka topic `vdt2024` and put to HDFS. We will choose 2 processors: *ConsumeKafkaRecord_2.0* and *PutHDFS*.

Because the producer send by JSON format, so I will choose to read by JSON format and write in parquet format following the requirements. Moreover, because NiFi and Kafka use the same *default-network*, so we need to config `broker01:9093,broker02:9095` for the Kafka Broker fields.

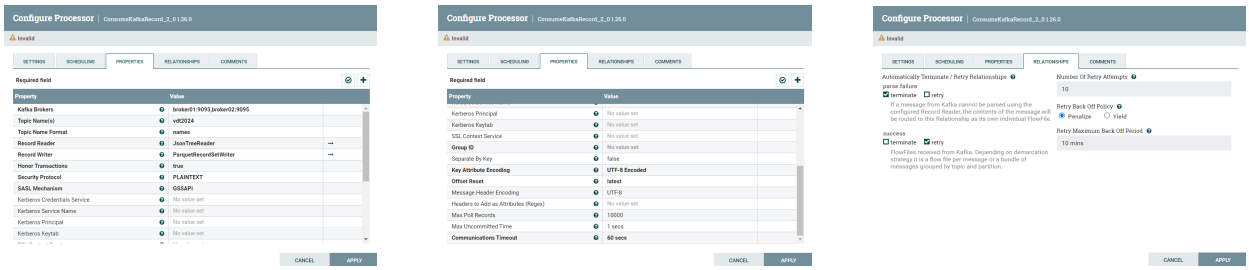

Figure 5: *Configuration for processor ConsumeKafkaRecord*

With processor PutHDFS, we need to configure the field *Hadoop Configuration Resources* with directory of *core-site.xml*, *hdfs-site.xml* for NiFi get the information about the HDFS which can be easily find on website with a little appropriate modification. And it is necessary to fill the field directory that NiFi will write into in HDFS, which is `/raw_zone/fact/activity`.

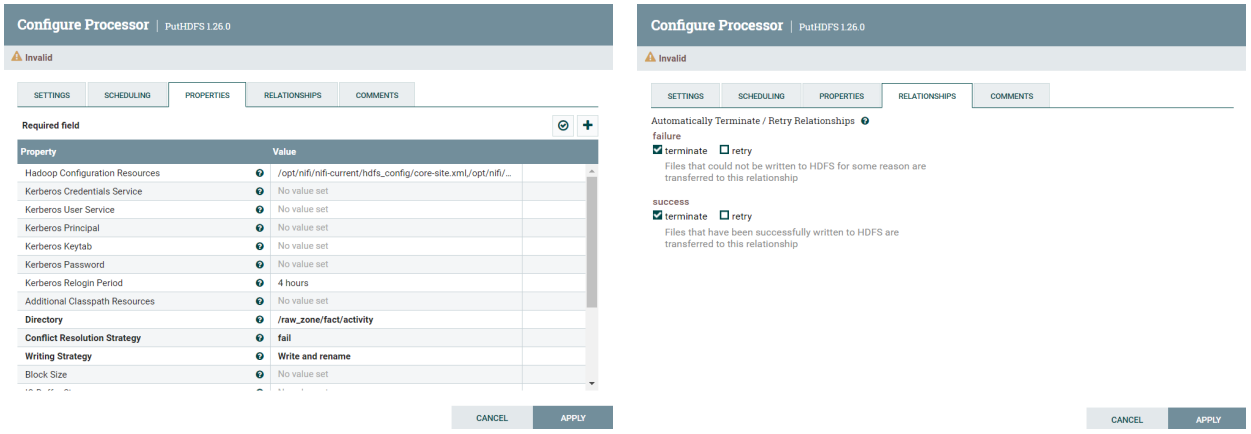

Figure 6: *Configuration for processor PutHDFS*

Then I connect these 2 processor and configure for this connection

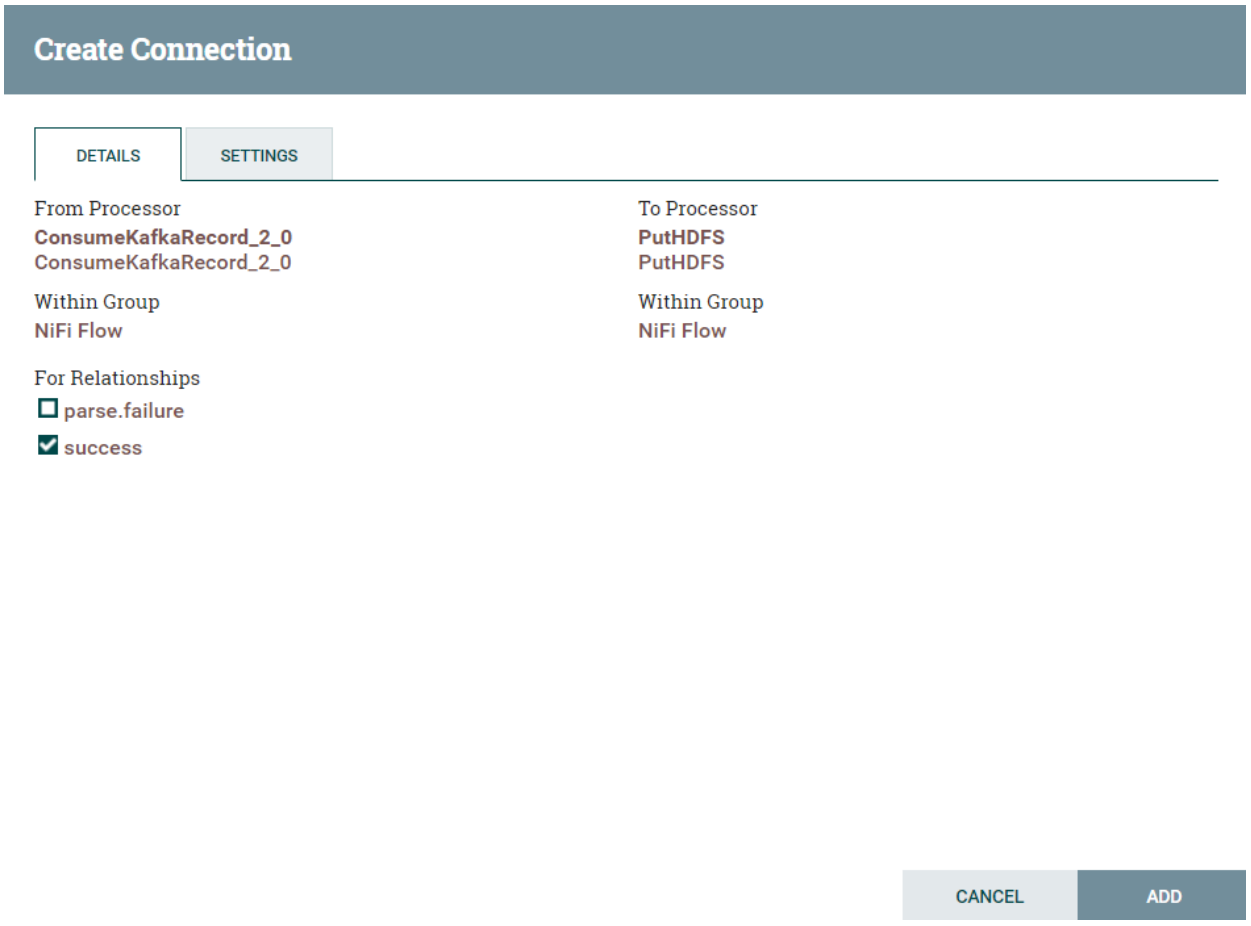

Figure 7: *Configuration for connections among 2 processors*

Enable all controller services in NiFi Flow and run it.

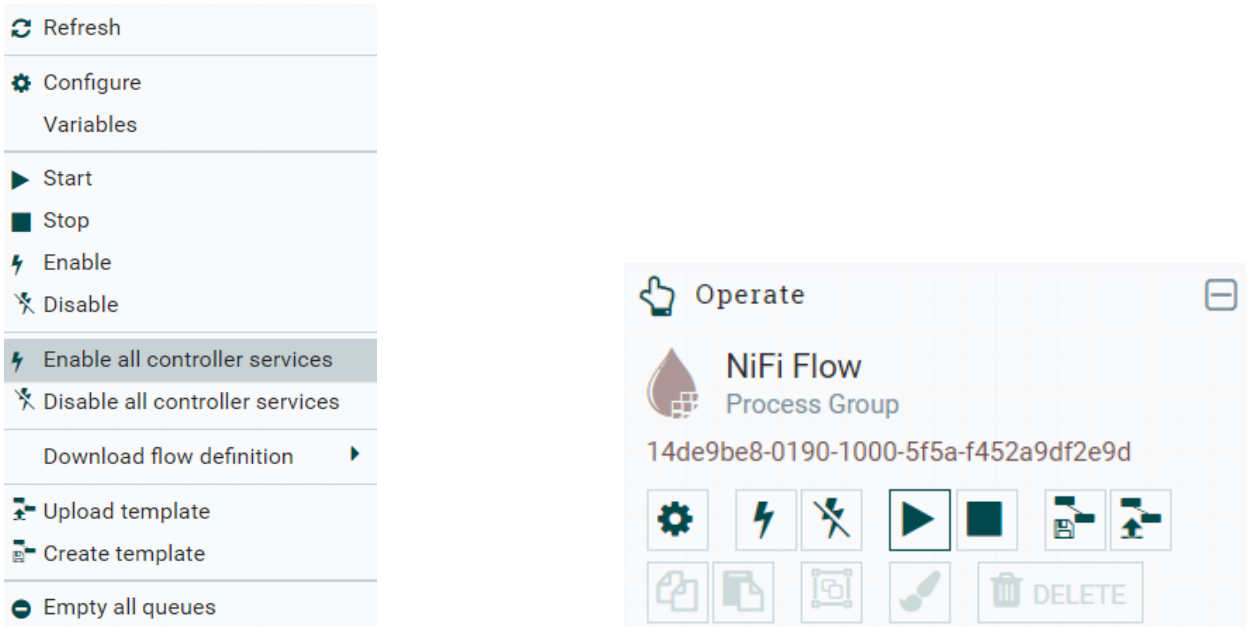

Figure 8: *Enable controller services and Run the NiFi flow*

You can see the NiFi flow has run already.

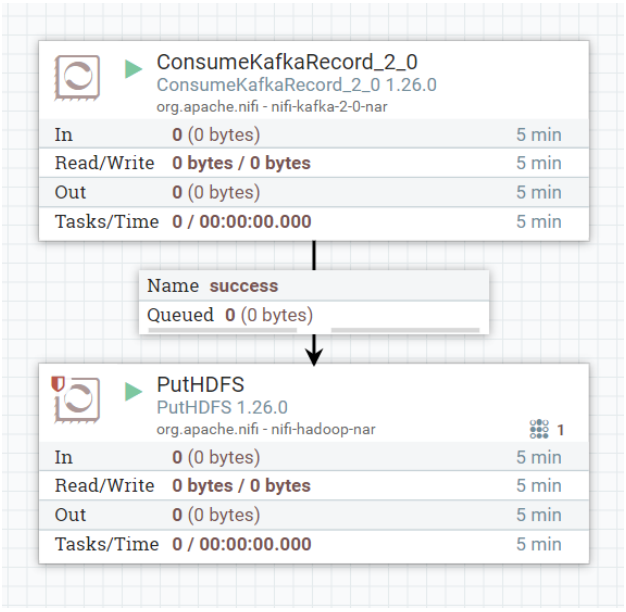

Figure 9: *NiFi processors run*

Next step is running the local consumer *kafka_consumer.py* to check if the producer send successfully. Moreover, you also can check by Kafka UI and NiFI UI. Now, start to stream the data to Kafka by Kafka producer *kafka_producer.py*

Figure 10: *Local consumer consume message*

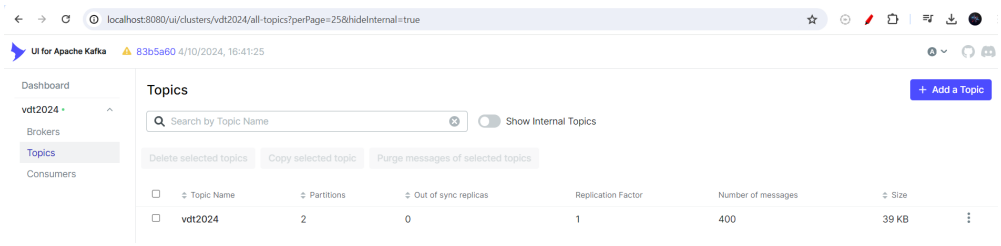

Figure 11: *Check by Kafka UI*

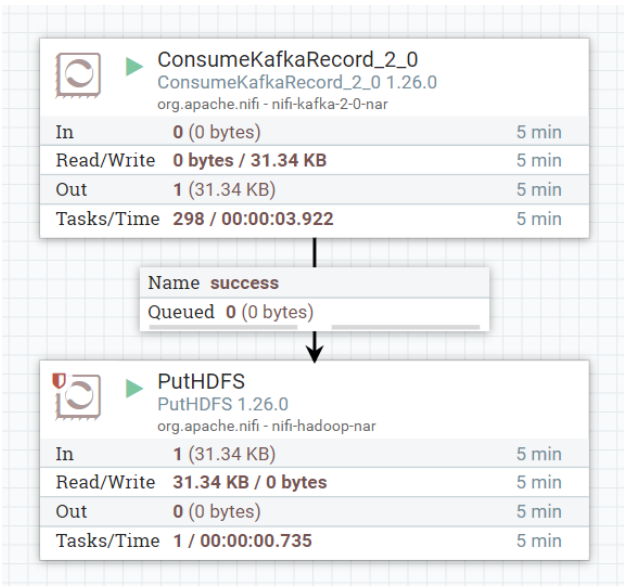

Figure 12: *NiFi ingest the data from Kafka*

HDFS and Spark First, it is needed to check whether the parquet was written by NiFi to HDFS by execute some command inside `nodemanager` container

Figure 13: *Check parquet file put in HDFS from NiFi*

Now we are going to move the file *danh_sach_sv_de.csv* into the HDFS by a command inside `namenode`

Figure 14: *Put file danh_sach_sv_de.csv into HDFS*

Then we go inside `sparkworker` container to execute command in order to process the data stored in HDFS.

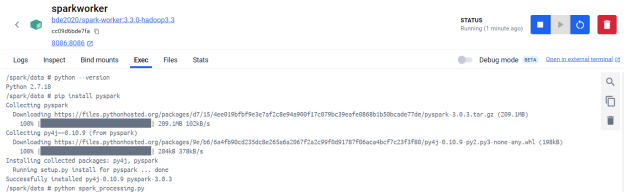

Figure 15: *Run spark_processing.py in Spark container*

And this is the result. Furthermore, the result is also written down in the output directory

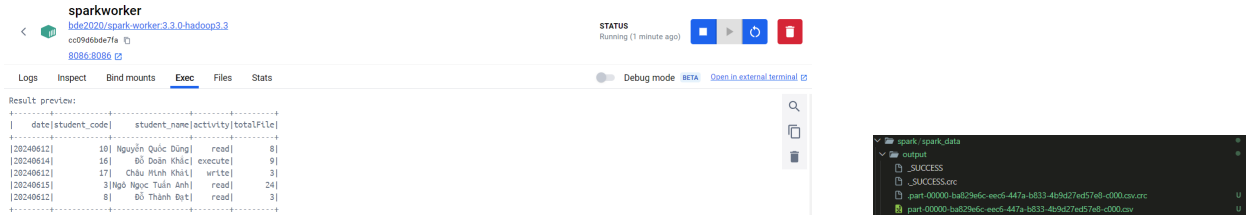

Figure 16: *Spark output*

4 Sourse Code

Follow the deployment on this repository. All the code is in directory *src*.

This is the link of my repository: github.com/dvthanh19/vdt_data_pipeline