**BÁO CÁO ASSIGNMENT**

**TRẦN PHÚC MẠNH LINH**

STT: 21

**Data Engineering**

**VDT2024**

Source Code: [megaheart/viettel-assignment-21-linh-tpm-1 (github.com)](https://github.com/megaheart/viettel-assignment-21-linh-tpm-1)

Sinh viên thực hiện

Ký và ghi rõ họ tên

**MỤC LỤC**

[CHƯƠNG 1. CÁC QUI ĐỊNH CHUNG 1](#_Toc21499666)

[1.1 Giới thiệu chung 1](#_Toc21499667)

[1.2 Sử dụng các định dạng văn bản theo qui định 1](#_Toc21499668)

[1.2.1 Qui định về căn lề văn bản 1](#_Toc21499669)

[1.2.1 Tạo lề cho văn bản in 2 mặt 3](#_Toc21499670)

[1.2.2 Tạo chương mới 3](#_Toc21499671)

[1.2.3 Tạo tiêu đề các cấp 3](#_Toc21499672)

[1.2.4 Định dạng phần nội dung các chương, mục 4](#_Toc21499673)

[1.2.5 Hình vẽ - Đồ thị 4](#_Toc21499674)

[1.2.6 Bảng biểu 6](#_Toc21499675)

[1.2.7 Phương trình 8](#_Toc21499676)

[1.3 Tạo tham chiếu chéo giữa các đoạn văn bản 11](#_Toc21499677)

[1.4 Tạo danh mục tài liệu tham khảo 11](#_Toc21499678)

[1.5 Cập nhật lại các chú thích và tham chiếu 15](#_Toc21499679)

[1.6 Tạo danh mục hình vẽ 15](#_Toc21499680)

[1.7 Tạo danh mục bảng biểu 16](#_Toc21499681)

[1.8 Tạo trang mục lục 16](#_Toc21499682)

[1.9 Qui cách đóng quyển 17](#_Toc21499683)

[CHƯƠNG 2. SỬ DỤNG CÁC BIỂU ĐỒ 19](#_Toc21499684)

[2.1 Giới thiệu về biểu diễn bằng đồ thị 19](#_Toc21499685)

[2.2 Đồ thị kiểu bánh 19](#_Toc21499686)

[2.3 Đồ thị kiểu thanh ngang 20](#_Toc21499687)

[2.4 Đồ thị kiểu cột đứng 20](#_Toc21499688)

[2.5 Đồ thị kiểu đường 21](#_Toc21499689)

[2.6 Đồ thị kiểu diện tích 21](#_Toc21499690)

[CHƯƠNG 3. KẾT LUẬN 23](#_Toc21499691)

[3.1 Kết luận 23](#_Toc21499692)

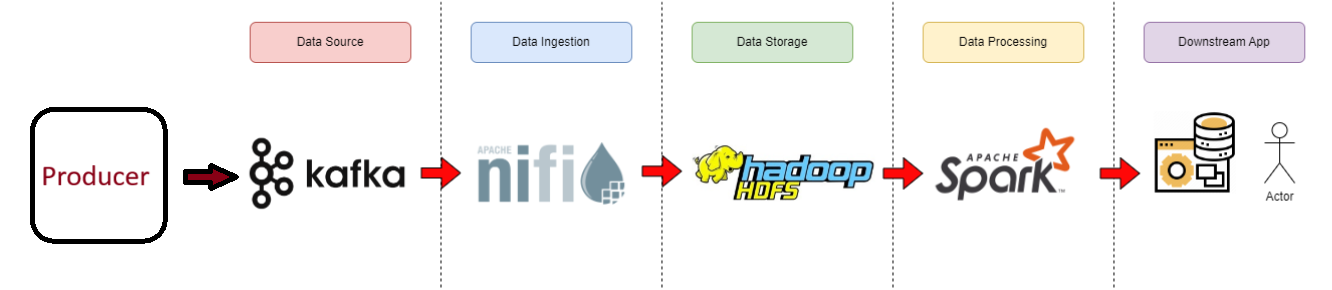
[3.2 Hướng phát triển của đồ án trong tương lai 23](#_Toc21499693)

[TÀI LIỆU THAM KHẢO 24](#_Toc21499694)

[PHỤ LỤC 25](#_Toc21499695)

# CÁCH THỨC TRIỂN KHAI

## Tổng quan các thành phần



Hệ thống sẽ bao gồm các thành phần

* **Kafka Producer**: Cung cấp các bản ghi từ log\_action.csv đẩy lên Kafka topic *vdt2024*
* **Kafka Broker**: Trung gian điều phối việc pub/sub của các publisher/consumer và lưu trữ các topic, trong đó có topic *vdt2024*
* **Nifi Flow**: Phụ trách việc lấy dữ liệu từ topic *vdt2024* và lưu lại vào trong HDFS với đường dẫn */raw\_zone/fact/activity* với format parquet.
* **HDFS**: Lưu trữ các action và file *danh\_sach\_sv\_de.csv*
* **Spark**: Xử lý dữ liệu lưu dưới HDFS và xuất ra output là tổng số file được tương tác hàng ngày theo mỗi loại activity mà mỗi sinh viên thực hiện.

## Kafka Producer

Đoạn mã thực hiện Kafka Producer được viết bằng Python yêu cầu thư viện *confluent\_kafka* để chạy, file chương trình nằm ở *kafka\producer\_kafka.py*

A screen shot of a computer program

Description automatically generated

## Cấu hình Kafka

Kafka sẽ gồm 2 container với cấu hình khởi chạy như sau:

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

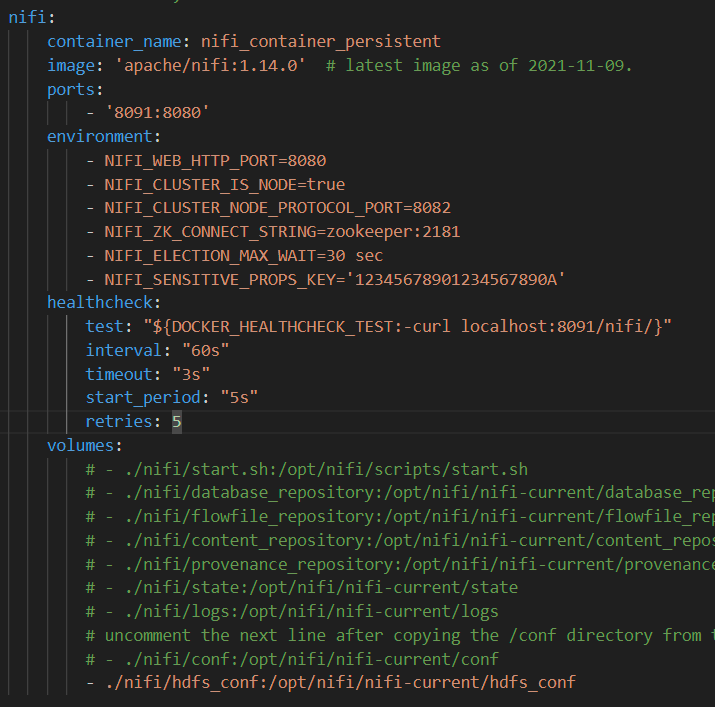
Description automatically generated

Trong đó zookeeper phụ trách quản lý các metadata (các topic, partition, offset và cấu hình cluster), các leader election, tình trạng các broker và đồng bộ hóa producer và consumer. Còn broker01 sẽ phụ trách lưu trữ các topic trên các partition cũng như cho phép các producer/consumer pub/sub dữ liệu.

Broker ở đây được cấu hình để có thể lắng nghe các yêu cầu pub/sub đến domain localhost:9092 hoặc broker01:9093 với giao thức PLAINTEXT. Do chỉ chạy 1 node nên em thiết lập chỉ dùng 1 partition.

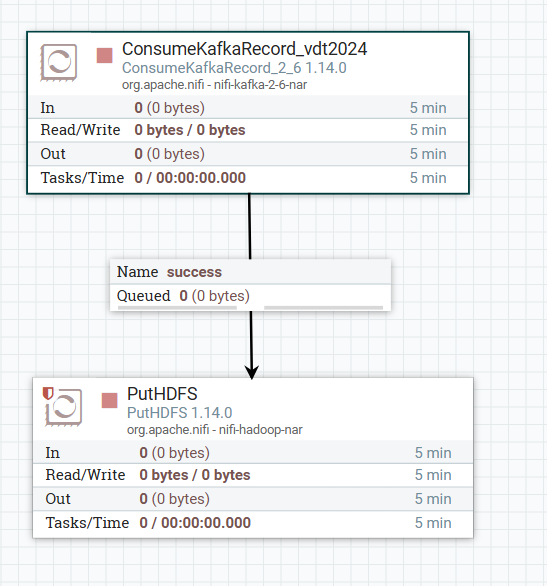
## Cấu hình Nifi

Nifi sẽ được chạy dưới 1 container với thiết lập như sau:



Có thể truy cập trang Web từ http://localhost:8091/nifi/ để chỉnh sửa Flow.

Nifi Flow sẽ được cấu hình như sau:



Processor **ConsumeKafkaRecord\_vdt2024** sẽ phụ trách lấy dữ liệu từ topic *vdt2024* và chuyển đổi về dạng parquet còn processor **PutHDFS** sẽ phụ trách đẩy dữ liệu parquet xuống HDFS với đường dẫn */raw\_zone/fact/activity*

Có thể import Template của Flow trên từ file *nifi\kafka2hdfs.xml*

## Cấu hình HDFS

HDFS sẽ được chạy trên 1 namenode container và 2 datanode container:



A computer screen shot of a program

Description automatically generated

Trong đó cổng 9000 trên namenode sẽ dùng để kết nối phục vụ cho việc đọc ghi trên HDFS, còn cổng 9870 thì phục vụ cho trang web giúp dễ dàng theo dõi trạng thái HDFS.

Ngoài ra một số cấu hình Hadoop khác được thiết lập trên file *hadoop/hadoop.env*:

A computer screen shot of a computer program

Description automatically generated

Các thiết lập liên quan đến HDFS bao gồm việc cho phép truy cập HDFS qua HTTP (WebHDFS), vô hiệu hóa kiểm tra quyền trên HDFS, và tắt kiểm tra IP/tên máy chủ trong quá trình đăng ký datanode với namenode. Ngoài ra, hệ số nhân bản được thiết lập là 1 (vì chỉ có 2 data node).

## Thiết lập Spark Notebook

## Spark Notebook sẽ được chạy trong một container được thiết lập như sau:

## A screen shot of a computer Description automatically generated

## Việc này giúp cho Spark được thực thi trong cùng 1 network với các container của HDFS, từ đó có thể kết nối được với HDFS dễ dàng hơn.

## File cấu hình của notebook là *spark\jupyter\_notebook\_config.py*. Bên trong file là cấu hình mật khẩu của Spark Notebook

## File Notebook nằm ở *spark\process\_data\_spark.ipynb*, để thực hiện file này ta cần kết nối đến Jupiter-server thông qua domain <http://localhost:8888> và mật khẩu là 123456

# THỰC HIỆN XỬ LÝ

## Khởi chạy Kafka Producer

Thực thi file chương trình nằm ở *kafka\producer\_kafka.py*, chương trình sẽ đẩy từng dòng trong file *log\_action.csv* lên kafka topic *vdt2024* theo từng giây.

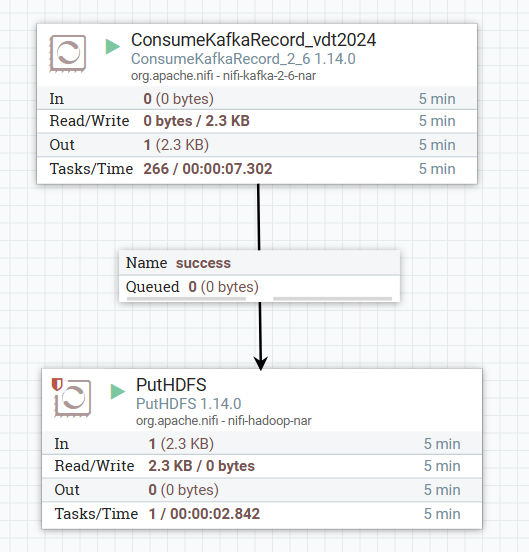
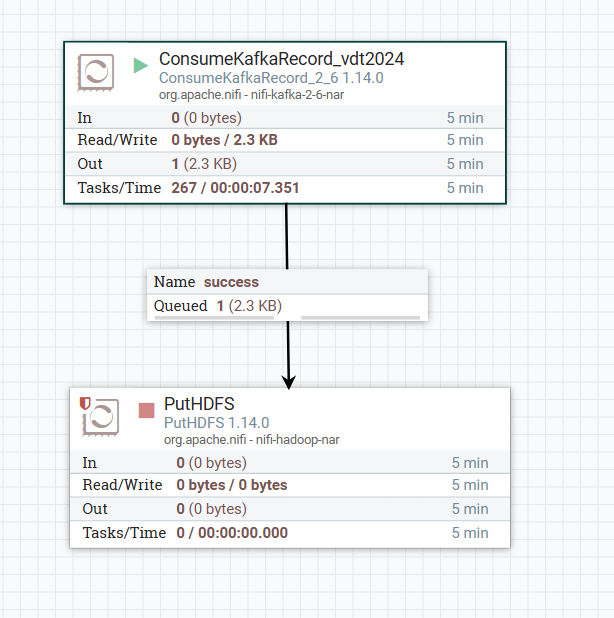
A screenshot of a computer

Description automatically generated

## Lấy dữ liệu từ Kafka lưu vào HDFS

Tiến hành lấy dữ liệu từ topic *vdt2024* và lưu lại vào trong HDFS với đường dẫn */raw\_zone/fact/activity* với format parquet bằng Nifi.

Vào trang Web <http://localhost:8091/nifi/> sau đó khởi chạy ConsumeKafkaRecord\_vdt2024 và sau đó là PutHDFS.



Kết thúc thì file HDFS với đường dẫn */raw\_zone/fact/activity* đã được ghi thành công.

A close up of a computer screen

Description automatically generated

## Lưu trữ file “danh\_sach\_sv\_de.csv” xuống HDFS

Mở file Notebook nằm ở *spark\process\_data\_spark.ipynb*, để thực hiện file này ta cần kết nối đến Jupiter-server thông qua domain <http://localhost:8888> và mật khẩu là 123456.

Đầu tiên ta tạo một Spark Context và Spark Session

A screen shot of a computer

Description automatically generated

Sau đó ta tạo cấu trúc dữ liệu cho Action và SV\_DE:

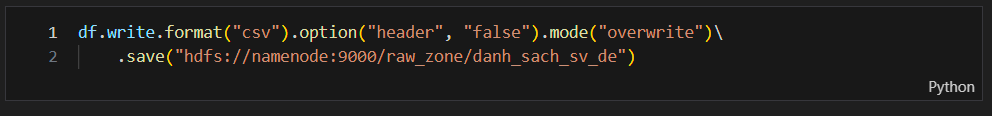
A screen shot of a computer code

Description automatically generated

Tiếp theo ta đọc file *danh\_sach\_sv\_de.csv* rồi sau đó lưu xuống HDFS:

A screen shot of a computer

Description automatically generated



Ta mở [http://localhost:9870/](http://localhost:9870/explorer.html#/raw_zone/danh_sach_sv_de) lên để kiểm tra kết quả. File đã được lưu xuống HDFS.

A screenshot of a computer

Description automatically generated

## Xử lý dữ liệu lưu trữ dưới HDFS, sử dụng Apache Spark

Trong notebook, ta thực thi đoạn lệnh đọc danh sách sinh viên DE từ HDFS:

A screen shot of a computer

Description automatically generated

Tiếp tục đọc */raw\_zone/fact/activity* từ HDFS:

A screen shot of a computer

Description automatically generated

Sau đó tiến hành tính tổng số file được tương tác hàng ngày theo mỗi loại activity

mà sinh viên đó thực hiện. Trong quá trình này thực hiện join với Dataframe của SV DE để lấy họ tên luôn. Quá trình này do chỉ toàn Transformations nên Spark chưa thực hiện Job.

A screen shot of a computer code

Description automatically generated

Tiếp theo ta sắp xếp lại các cột cho đúng thứ tự đề bài yêu cầu và lưu ra file csv kết quả:

A screen shot of a computer code

Description automatically generated

Do lưu bằng Spark write nó sẽ không ra một file duy nhất nên em phải lưu bằng file io của python.