BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Dương Đình Thiệu

NGHIÊN CỨU PHÁT TRIỂN CÁC GIẢI PHÁP TÍCH HỢP CÔNG CỤ THU THẬP, PHÂN TÍCH DỮ LIỆU TRONG NỀN TẢNG QUẢN LÝ VÀ CHIA SẺ DỮ LIỆU NGHIÊN CỨU KHOA HỌC VÀ CÔNG NGHỆ QUỐC GIA

LUẬN VĂN THẠC SĨ MÁY TÍNH

Ngành: Hệ thống thông tin

Mã số: **9.48.01.04**

NGƯỜI HƯỚNG DẪN KHOA HỌC :

1. PGS.TS. Nguyễn Long Giang

Hà Nội - 2024

LỜI CAM ĐOAN

Tôi xin cam đoan đề tài nghiên cứu trong luận văn này là công trình nghiên cứu của tôi dựa trên những tài liệu, số liệu do chính tôi tự tìm hiểu và nghiên cứu. Chính vì vậy, các kết quả nghiên cứu đảm bảo trung thực và khách quan nhất. Đồng thời, kết quả này chưa từng xuất hiện trong bất cứ một nghiên cứu nào. Các số liệu, kết quả nêu trong luận văn là trung thực nếu sai tôi hoàn chịu trách nhiệm trước phát luật.

LỜI CẢM ƠN

Để hoàn thành luận văn này, trước tiên, tôi xin gửi lời cảm ơn sâu sắc nhất đến thầy giáo, Phó Viện trưởng Viện Công nghệ thông tin, Viện Hàn lâm KH&CN Việt Nam, PGS.TS. Nguyễn Long Giang, người đã khơi nguồn, định hướng chuyên môn, cũng như trực tiếp hướng dẫn cho tôi trong quá trình thực hiện luận văn.

Tôi xin chân thành cảm ơn Ban Lãnh đạo Học viện đã luôn quan tâm, chỉ đạo để tạo ra một môi trường học tập và nghiên cứu chuyên nghiệp giúp tôi có thể phát huy hết khả năng của mình.

Xin cảm ơn Phòng Đào tạo và các phòng chức năng khác của học viện vì đã cung cấp những thông tin cần thiết và hỗ trợ kịp thời trong suốt quá trình học tập, nghiên cứu tiếp.

Tôi cũng xin chân thành cảm ơn anh Phạm Quang Nam đã hỗ trợ, cung cấp những tài nguyên cho nghiên cứu luận văn và hướng dẫn tôi trong suốt thời gian vừa qua.

Cuối cùng, tôi xin bày tỏ lòng kính trọng và sự biết ơn sâu sắc đến gia đình đã tạo động lực và mọi điều kiện tốt nhất để tôi có thể hoàn thành tốt mọi công việc trong quá trình thực hiện luận văn.

Mặc dù đã rất cố gắng trong quá trình thực hiện nhưng luận văn không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được sự góp ý của các thầy cô và bạn bè để tiếp tục hoàn thiện thêm nghiên cứu của mình.

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT v
DANH MỤC HÌNH ẢNH vi
MỞ ĐẦU 1
1. Lý do chọn đề tài 1
2. Mục đích nghiên cứu 1
3. Nội dung nghiên cứu 1
4. Cơ sở khoa học và tính thực tiễn của đề tài 2
5. Những đóng góp của luận văn 3
Chương 1. Tổng quan tình hình nghiên cứu 4
1.1. Tổng quan về hệ thống quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia (Openscience.vn) 4
1.1.1. Khối thu thập, tích hợp dữ liệu từ nhiều nguồn (Data Ingression) 4
1.1.2. Khối lưu trữ dữ liệu (Data stogare) 5
1.1.3. Khối xử lý, phân tích dữ liệu (data processing and analyzing) 5
1.2. Tổng quan về các giải pháp tích hợp các nền tảng mã nguồn mở phục vụ thu thập, phân tích dữ liệu và học máy trên thế giới và tại Việt Nam 6
1.2.1. Kubernetes (K8S) 6
1.2.2. CEPH 7
1.2.3. Apache Nifi 8
1.2.4. Apache Spark 10
1.2.5. Kubeflow 12
Chương 2. Xây dựng giải pháp tích hợp các nền tảng thu thập, phân tích dữ liệu vào hệ thống Openscience.vn 13
2.1. Xây dựng giải pháp tích hợp nền tảng Nifi vào hệ thống Openscience.vn 15
2.1.1. Mô tả công cụ Nifi 15
2.1.2. Thu thập dữ liệu từ tệp hệ thống (file systems) 16
2.1.3. Thu thập dữ liệu luồng từ hệ thống IoT (data stream) 20
2.1.4. Thu thập dữ liệu từ hệ thống CSDL quan hệ 23
2.1.5. Thu thập dữ liệu qua API 26
2.2. Xây dựng giải pháp tích hợp nền tảng Spark vào hệ thống Openscience.vn 27
2.2.1. Xử lý dữ liệu theo lô (Batch processing) 27
2.2.2. Xử lý dữ liệu theo luồng (Streaming processing) 31

2.3. Xây dựng giải pháp tích hợp nền tảng Kubeflow vào hệ thống Openscience.vn 35
2.3.1. Tổng quan về xây dựng một pipeline....35
2.3.2. Triển khai xây dựng pipeline ML/DL trên Openscience.vn 35
2.4. Xây dựng giải pháp đăng nhập một lần (SSO) cho Openscience.vn để truy cập vào các nền tảng 40
2.4.1. Thực hiện SSO truy cập vào Nifi qua Keycloak....40
2.4.2. Thực hiện SSO truy cập vào Kubeflow qua Keycloak 42
Chương 3. Thử nghiệm và đánh giá các giải pháp....51
3.1. Thử nghiệm, đánh giá giải pháp tích hợp Nifi 51
3.1.1. Thử nghiệm gửi và nhận dữ liệu trên Nifi 51
3.1.2. Đánh giá hoạt động 55
3.2. Thử nghiệm, đánh giá giải pháp tích hợp Spark 57
3.2.1. Thử nghiệm xử lý dữ liệu theo lô....57
3.2.2. Đánh giá xử lý dữ liệu theo lô....59
3.3. Thử nghiệm, đánh giá giải pháp tích hợp Kubeflow....60
3.3.1. Mô tả bài toán....60
3.3.2. Các bước thực hiện....60
3.3.3. Đánh giá kết quả thực hiện bài toán....63
3.4. Thử nghiệm SSO trên Openscience.vn 64
3.4.1. Kiểm tra SSO với Apache Nifi....64
3.4.2. Kiểm tra SSO với Kubeflow 65
KẾT LUẬN VÀ KIẾN NGHỊ....67
1. Kết luận 67
2. Kiến nghị 67
DANH MỤC TÀI LIỆU THAM KHẢO 68

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Chữ viết tắt	Tiếng Anh	Tiếng Việt
CSDL		Cơ sở dữ liệu
IoT	Internet of Things	Internet vạn vật
ML	Machine Learning	Học máy
DL	Deep Learning	Học sâu
SSO	Single Sign On	Đăng nhập một lần
HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
API	Application Programming Interface	Giao diện lập trình ứng dụng
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc
AI	Artificial Intelligence	Trí tuệ nhân tạo
SaaS	Software as a Service	Phần mềm dưới dạng dịch vụ
RDBMS	Relational Database Management System	Hệ quản trị cơ sở dữ liệu dạng quan hệ
FTP	File Tranfer Protocol	Giao thức truyền file
CPU	Central Processing Unit	Bộ xử lý trung tâm
GPU	Graphics Processing Unit	Bộ xử lý đồ họa

DANH MỤC HÌNH ẢNH

Hình 1. 1. Mô hình kiến trúc hệ thống Openscience.vn....4
Hình 2. 1. Mô hình kiến trúc ứng dụng Openscience.vn....13
Hình 2. 2. Các nguồn dữ liệu của hệ thống....15
Hình 2. 3. Giao diện đăng nhập chính....16
Hình 2. 4. Giao diện chính của hệ thống Openscience....17
Hình 2. 5. Cửa sổ cài đặt các thông tin truyền dữ liệu....18
Hình 2. 6. Cửa sổ cài đặt các thông tin nhận dữ liệu....19
Hình 2. 7. Kết nối hai khối truyền và nhận dữ liệu....19
Hình 2. 8. Khối điều khiển Operate....20
Hình 2. 9. Khối chức năng mới tạo....20
Hình 2. 10. Cấu hình các thuộc tính của khối nhận dữ liệu....22
Hình 2. 11. Bảng thông số của khối ExecuteSQL....24
Hình 2. 12. Kết nối các khối để nhận dữ liệu từ hệ thống CSDL quan hệ....25
Hình 2. 13. Các thuộc tính của khối InvokeHTTP....26
Hình 2. 14. Minh họa xử lý dữ liệu theo lô....28
Hình 2. 15. Giao diện để truy cập vào module xử lý dữ liệu....28
Hình 2. 16. Lệnh tạo Spark Session....29
Hình 2. 17. Sử dụng hàm để hiển thị một số dữ liệu....29
Hình 2. 18. Kiểm tra lược đồ dữ liệu và hiển thị ra màn hình....30
Hình 2. 19. Thao tác với dữ liệu đã xử lý thông qua query....30
Hình 2. 20. Minh họa quá trình xử lý dữ liệu theo luồng....32
Hình 2. 21. Tạo SparkSession cho luồng dữ liệu....32
Hình 2. 22. Đặt schemaInference sang True để truyền phát dữ liệu luồng....33
Hình 2. 23. Sử dụng select SparkSQL để đọc hết các trường dữ liệu trong JSON...33
Hình 2. 24. Làm phẳng dữ liệu trong JSON....34
Hình 2. 25. So sánh dữ liệu trước và sau khi làm phẳng....34
Hình 2. 26. Truy cập vào Kubeflow trên giao diện chính....36
Hình 2. 27. Khởi tạo một notebook mới....37
Hình 2. 28. File cấu hình Kserve trỏ vào các mô hình trên Minio....37
Hình 2. 29. File thiết lập cấp quyền KFP....38
Hình 2. 30. Truy cập dữ liệu trong cơ sở dữ liệu Minio....39
Hình 2. 31. Tạo hàm a() và b()....39
Hình 2. 32. Ví dụ một pipeline hoàn chỉnh....40
Hình 2. 33. Tạo một client mới trong Keycloak cho Apache Nifi....41
Hình 2. 34. Tạo một Client Scopes mới trong Keycloak....43
Hình 2. 35. Cấu hình file params.env....44
Hình 2. 36. Các dòng lệnh khởi tạo lại dịch vụ OIDC của Kubeflow....44
Hình 2. 37. Những file cần thiết để tạo profile cho người dùng....45
Hình 2. 38. Tệp main.py và những thư viện cần khai báo cho Flask....46
Hình 2. 39. Hàm chuẩn hóa các ký tự của chuỗi email đầu vào....46
Hình 2. 40. Hàm tạo nội dung cho tệp profile và quota....46
Hình 2. 41. Hàm apply_yaml để áp dụng vào tệp config của cụm Kubernetes....47

Hình 2. 42. Hàm API để Server nghe được các yêu cầu tạo Profile từ Internet 47
Hình 2. 43. Nội dung file requirements 48
Hình 2. 44. Nội dung file cấu hình gunicorn 48
Hình 2. 45. File cấu hình docker để tạo image 48
Hình 2. 46. Nội dung file namespace.....49
Hình 2. 47. Nội dung file deployment 49
Hình 2. 48. Nội dung file nodeport 50
Hình 2. 49. Các lệnh để triển khai API server lên Kubernetes 50
Hình 3. 1. Thư mục data chứa dữ liệu thử nghiệm 51
Hình 3. 2. Khối operate để thực hiện thao tác.....52
Hình 3. 3. Giao diện khai báo chủ đề cho dữ liệu IoT thử nghiệm.....52
Hình 3. 4. Thông tin được sử dụng để thử nghiệm dữ liệu IoT 53
Hình 3. 5. Cơ sở dữ liệu thử nghiệm dạng quan hệ 54
Hình 3. 6. Luồng gửi và nhận dữ liệu thông qua API 55
Hình 3. 7. Lịch sử trạng thái truyền nhận file 56
Hình 3. 8. Dữ liệu được hiển thị trong mục quản lý dữ liệu 57
Hình 3. 9. Câu lệnh sử dụng để tải và giải nén bộ dữ liệu 58
Hình 3. 10. Đẩy dữ liệu lên kho dữ liệu.....58
Hình 3. 11. Khởi tạo SparkSession 58
Hình 3. 12. Hiển thị dữ liệu sau khi đọc file csv xong.....58
Hình 3. 13. Phân vùng và ghi dữ liệu đã được phân vùng dưới dạng Parquet 59
Hình 3. 14. Hàm đọc và thao tác dữ liệu dạng Parquet.....59
Hình 3. 15. Tương tác với dữ liệu bằng truy vấn SQL sau khi xử lý xong.....59
Hình 3. 16. Clone mã nguồn từ Github.....60
Hình 3. 17. Đóng gói các hàm thành các component cho Kubeflow.....61
Hình 3. 18. Tạo một kubeflow pipeline 62
Hình 3. 19. Pipeline mới được tạo ra trong Kubeflow.....62
Hình 3. 20. Mô hình sau khi được huấn luyện xong.....63
Hình 3. 21. Thử nghiệm mô hình chuẩn đoán ung thư thông qua ảnh.....63
Hình 3. 22. Giao diện phần Thu thập dữ liệu trên trang chủ.....64
Hình 3. 23. Đăng nhập SSO thành công tài khoản thứ hai 65
Hình 3. 24. Giao diện phần Phân tích Dữ liệu trên trang chủ.....65
Hình 3. 25. Giao diện chính với namespace mới được tạo tự động trên Kubeflow .66

MỞ ĐẦU

1. LÝ DO CHỌN ĐỀ TÀI

Trong bối cảnh cuộc cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ, việc xây dựng và phát triển các nền tảng để xử lý và phân tích dữ liệu lớn, cũng như các nền tảng hỗ trợ phát triển mô hình học máy, đang thu hút sự quan tâm của nhiều nhà nghiên cứu và các tập đoàn công nghệ lớn. Hiện tại, với những hệ thống với lượng dữ liệu đồ sộ và tới từ nhiều nguồn sẽ cần thu thập, chuẩn hóa và lưu trữ [1] vào một nền tảng. Từ đó ta sẽ sử dụng những dữ liệu đã thu thập được để tiến hành phân tích thông qua một số nền tảng phân tích dữ liệu lớn, ví dụ như Apache Spark, Hadoop, Google Cloud BigQuery, và rất nhiều nền tảng khác [2]. Bên cạnh đó, hiện nay cũng có rất nhiều nền tảng hỗ trợ phát triển mô hình học máy từ những dữ liệu đã thu thập được như: Kubeflow, Amazon SageMaker, Microsoft Azure Machine Learning Studio, Google Cloud AI và AutoML, đều đã được áp dụng rộng rãi trong cộng đồng [3]. Tuy nhiên, các giải pháp tích hợp cả hai nền tảng phục vụ thu thập, lưu trữ, xử lý, phân tích dữ liệu lớn và nền tảng phục vụ phát triển mô hình ML vẫn còn rất nhiều hạn chế.

Dù việc tích hợp các nền tảng cho xử lý, phân tích dữ liệu lớn và phát triển học máy đã được thực hiện, nhưng việc xây dựng một khung kiến trúc thống nhất để phục vụ đồng thời cả hai nhiệm vụ trên vẫn chưa được triển khai. Do đó, đề tài đăng ký thực hiện tích hợp 05 nền tảng được mô tả ở mục 3 với hai khối: khối nền tảng thu thập, lưu trữ, xử lý dữ liệu lớn (Nifi, CEPH, Spark, Kubernetes) và khối phát triển mô hình học máy (Kubeflow) với cơ chế đăng nhập một lần SSO cho phép truy cập vào các nền tảng qua cổng dữ liệu Openscience.vn. Giải pháp này là mới, chưa có ở trên thế giới và tại Việt Nam.

2. MỤC ĐÍCH NGHIÊN CỨU

Mục tiêu của đề tài luận văn là tích hợp các nền tảng mã nguồn mở vào hệ thống Openscience.vn nhằm thực hiện các bước trong quá trình phân tích dữ liệu lớn và học máy, bao gồm:

- Nền tảng thu thập dữ liệu Nifi phục vụ thu thập dữ liệu từ bốn nguồn: cơ sở dữ liệu quan hệ (RDBMS), tệp dữ liệu (files), phần mềm dạng dịch vụ (SaaS, APIs), dữ liệu luồng (IoT)
- Nền tảng Apache Spark thực hiện tác vụ xử lý theo lô (batch processing) và xử lý theo luồng (streaming processing)
- Nền tảng Kubeflow để phát triển và thực thi các mô hình học máy (machine learning)
- Nền tảng Keycloak áp dụng công nghệ xác thực một lần cho toàn bộ hệ thống.

3. NỘI DUNG NGHIÊN CỨU

Ngoài phần mở đầu và kết luận, luận văn gồm 3 chương với nội dung như sau:

3.1. Tổng quan tình hình nghiên cứu

- Trình bày khai quát về hệ thống quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia (Openscience.vn)
- Giới thiệu về các giải pháp tích hợp các nền tảng mã nguồn mở phục vụ thu thập, phân tích dữ liệu và học máy trên thế giới và tại Việt nam

3.2. Xây dựng giải pháp tích hợp các nền tảng thu thập, phân tích dữ liệu vào hệ thống Openscience.vn

- Xây dựng giải pháp tích hợp nền tảng Nifi vào hệ thống Openscience.vn để thực hiện nhiệm vụ thu thập dữ liệu từ các nguồn dữ liệu như file, cơ sở dữ liệu dạng quang hệ, luồng IoT.
- Hỗ trợ xử lý những dữ liệu đã thu thập được thông qua giải pháp tích hợp nền tảng Spark vào hệ thống Openscience.vn
- Tiếp tục tiến hành xây dựng giải pháp tích hợp nền tảng Kubeflow vào hệ thống Openscience.vn để hỗ trợ các nhà khoa học dữ liệu quản lý và thực thi quy trình phát triển mô hình học máy
- Xây dựng giải pháp đăng nhập một lần (SSO) cho Openscience.vn để truy cập vào các nền tảng một cách đồng bộ.

3.3. Thử nghiệm và đánh giá các giải pháp

- Thử nghiệm, đánh giá các giải pháp tích hợp để kiểm tra luồng (pipeline) đã thực hiện được trên hệ thống.
- Thử nghiệm đăng nhập một lần SSO giữa các nền tảng đã tích hợp được trên Openscience.vn

4. CƠ SỞ KHOA HỌC VÀ TÍNH THỰC TIỄN CỦA ĐỀ TÀI

Giải pháp tích hợp các nền tảng mã nguồn mở: nhằm thực hiện một quy trình (pipeline) phân tích dữ liệu lớn và học máy, bao gồm các bước thu thập, lưu trữ, xử lý, phân tích dữ liệu lớn và phát triển mô hình học máy, phục vụ nghiên cứu, phát triển và ứng dụng trí tuệ nhân tạo (AI). Các nền tảng mã nguồn mở được tích hợp bao gồm:

1) Nền tảng Kubernettes (hay K8S): nhằm tự động hóa việc triển khai, cân bằng tải và quản lý các ứng dụng trên hạ tầng đám mây.

2) Nền tảng quản trị và lưu trữ dữ hạ tầng dữ liệu lớn CEPH được tích hợp trên nền tảng K8S. CEPH là công nghệ lưu trữ tiên tiến, hiện đại nhất cho phép lưu trữ phân tán, độ tin cậy và hiệu năng cao, dễ dàng mở rộng với kiến trúc Lakehouse, cung cấp các giải pháp lưu trữ đối tượng (Object storage), lưu trữ khối (block storage), tệp (file storage), ngoài ra CEPH cũng hỗ trợ cấu trúc lưu trữ S3. Ngoài ra, CEPH cũng tích hợp với lưu trữ CSDL quan hệ với PostgreSQL nhằm quản lý các thuộc tính của dữ liệu (metadata) phục vụ tra cứu, báo cáo, thống kê [4].

3) Nền tảng Apache NiFi được tích hợp với CEPH cho phép kết nối với các nguồn dữ liệu khác nhau và thực hiện tự động thu thập dữ liệu để đưa vào lưu trữ trong CEPH, bao gồm 4 nguồn dữ liệu: cơ sở dữ liệu quan hệ (RDBMS), tệp (files), phần mềm dạng dịch vụ (SaaS, APIs) và dữ liệu luồng từ các hệ thống IoT [5].

4) Nền tảng xử lý dữ liệu lớn Apache Spark được tích hợp với CEPH nhằm xử lý dữ liệu được lưu trữ trong CEPH hoặc xử lý dữ liệu luồng được thu thập trực tiếp từ Nifi. Apache Spark bao gồm hai tác vụ là xử lý dữ liệu theo lô (Batch processing) và dữ lý dữ liệu luồng (streaming processing) [6]. Kết quả xử lý dữ liệu được lưu trữ trong CEPH phục vụ các tác vụ phân tích dữ liệu tiếp theo.

5) Nền tảng quan trọng nhất, được xem là lõi của hệ thống, là nền tảng học máy Kubeflow được tích hợp trên CEPH và K8S. Kubeflow cung cấp các thư viện nền tảng như Tensorflow, PyTorch [7] cho phép các nhà khoa học dữ liệu quản lý và thực thi quy trình phát triển mô hình học máy, bao gồm xây dựng mô hình, huấn luyện, kiểm thử và quản lý mô hình và tích hợp với các công cụ triển khai phần mềm.

6) Nền tảng xác thực một lần Keycloak được tích hợp để cung cấp khả năng chuyển đổi linh hoạt giữa các hệ thống thu thập dữ liệu, phân tích dữ liệu và học máy.

Tất cả các nền tảng mã nguồn mở nêu trên được tích hợp, kết nối với nhau và được xây dựng trên nền tảng quản trị đám mây K8S, cung cấp môi trường cho các nhà khoa học dữ liệu thực hiện một luồng công việc (pipeline) từ bước thu thập, lưu trữ, xử lý và phát triển mô hình học máy.

5. NHỮNG ĐÓNG GÓP CỦA LUẬN VĂN

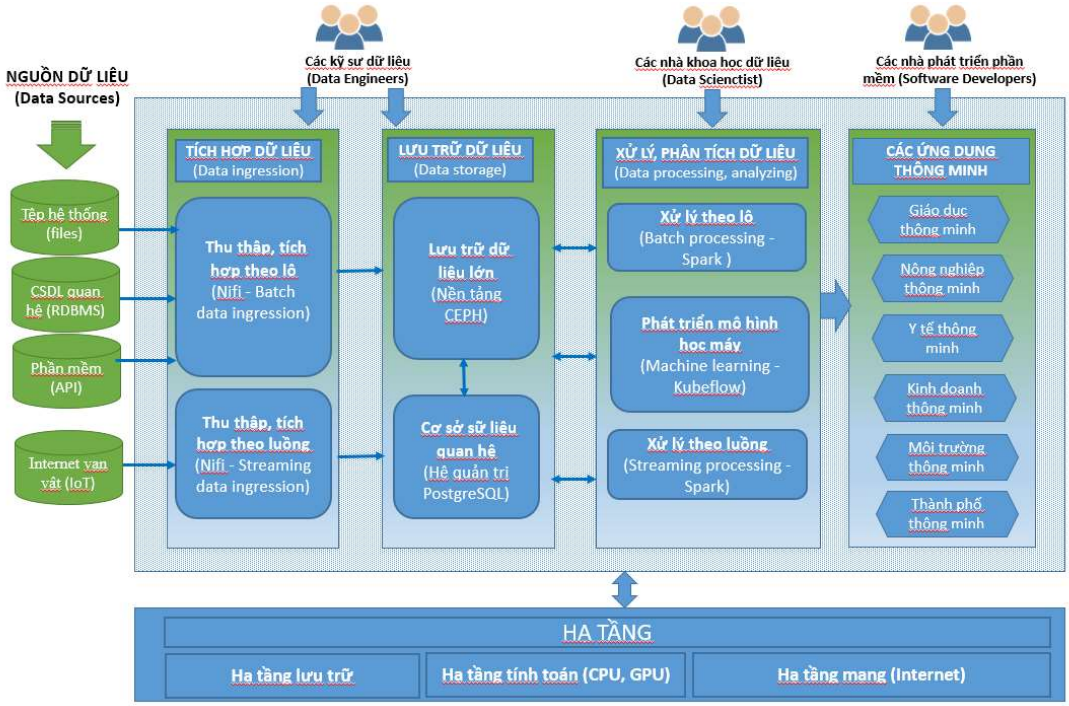
Đề tài sẽ cung cấp một giải pháp tích hợp mới mẻ và khả thi, giúp giải quyết các thách thức trong việc triển khai các hệ thống phân tích dữ liệu lớn và học máy trong thực tế. Thêm vào đó, đề tài cũng đề xuất và triển khai một giải pháp tích hợp đa nền tảng mã nguồn mở, bao gồm Apache NiFi, Apache Spark, Kubeflow, Kubernetes, và CEPH từ đó cung cấp một hệ thống pipeline tự động, bao gồm tất cả các bước chính từ thu thập dữ liệu, lưu trữ, xử lý dữ liệu lớn đến phát triển, huấn luyện và triển khai mô hình học máy. Việc này sẽ giúp giảm thiểu thời gian và công sức trong việc xử lý dữ liệu thủ công, đồng thời giúp hệ thống hoạt động liên tục và ổn định trong các môi trường có khối lượng dữ liệu lớn.

CHƯƠNG 1. TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU

1.1. TỔNG QUAN VỀ HỆ THỐNG QUẢN LÝ VÀ CHIA SẺ DỮ LIỆU NGHIÊN CỨU KHOA HỌC VÀ CÔNG NGHỆ QUỐC GIA (OPENSCIENCE.VN)

Hệ thống nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ (KH&CN) dùng chung, gọi tắt là hệ thống Openscience.vn, được xây dựng với mục tiêu thu thập, lưu trữ quản trị, chia sẻ dữ liệu nghiên cứu KH&CN, tích hợp các công cụ phát triển, thực thi mô hình phân tích dữ liệu, học máy. Đối tượng sử dụng là các nhà khoa học dữ liệu, các cán bộ nghiên cứu, giảng viên, học viên, sinh viên các trường đại học, các cán bộ quản lý trong lĩnh vực khoa học dữ liệu, học máy, trí tuệ nhân tạo (AI). Hệ thống Openscience.vn là môi trường cho phép cộng đồng đóng góp, khai thác, chia sẻ dữ liệu và xây dựng, phát triển các mô hình học máy, phân tích dữ liệu phục vụ phát triển, ứng dụng AI trong các lĩnh vực khác nhau.

Mô hình kiến trúc hệ thống Openscience.vn được mô tả ở Hình 1, bao gồm các khối như sau:



Hình 1. 1. Mô hình kiến trúc hệ thống Openscience.vn

1.1.1. Khối thu thập, tích hợp dữ liệu từ nhiều nguồn (Data Ingression)

Khối này sử dụng công nghệ thu thập dữ liệu Nifi thực hiện thu thập dữ liệu theo lô (Batch ingression) và thu thập dữ liệu luồng (streaming ingression). Dữ liệu được thu thập theo lô gồm các nguồn: các tệp dữ liệu (files), cơ sở dữ liệu quan hệ (RDBMS) và từ các hệ thống phần mềm (SaaS) qua giao tiếp API. Dữ liệu luồng gồm dữ liệu từ các thiết bị IoT hay các thiết bị truyền dữ liệu theo thời gian thực. Các dữ liệu thu thập được lưu trữ vào kho lưu trữ dạng Lakehouse với nền tảng công nghệ lưu trữ CEPH kết hợp với CSDL quan hệ [5, 8].

1.1.2. Khối lưu trữ dữ liệu (Data stogare)

Khối lưu trữ dữ liệu có nhiệm vụ lưu trữ dữ liệu được thu thập từ các nguồn phục vụ các tác vụ xử lý và phân tích dữ liệu. Khối lưu trữ sử dụng công nghệ lưu trữ dữ liệu lớn CEPH kết hợp với lưu trữ CSDL quan hệ với hệ quản trị CSDL PostgreSQL. CEPH là nền tảng lưu trữ dữ liệu lớn tiên tiến với kiến trúc quản lý dữ liệu mở Lakehouse và các cấu trúc lưu trữ như lưu trữ đối tượng (Object storage), lưu trữ khối (block storage), cấu trúc lưu trữ S3 [4].

Hệ thống quản trị cho phép người quản trị thực hiện các các tác vụ quản lý các cấu trúc lưu trữ và tương tác với dữ liệu lưu trữ trong CEPH.

1.1.3. Khối xử lý, phân tích dữ liệu (data processing and analyzing)

Khối xử lý dữ liệu: Thực hiện các tác vụ xử lý dữ liệu được lưu trữ trong CEPH, bao gồm xử lý theo lô (batch processing) và xử lý theo luồng (streaming processing). Khối xử lý theo luồng có thể xử lý các luồng dữ liệu được thu thập từ IoT qua hệ thống thu thập dữ liệu Nifi và lưu trữ kết quả trong CEPH. Khối xử lý dữ liệu sử dụng nền tảng xử lý dữ liệu lớn Spark tích hợp trên hệ thống lưu trữ CEPH.

Khối phân tích dữ liệu: là mỗi trường để xây dựng và phát triển các mô hình học máy (Machine learning) sử dụng công cụ kubeflow. Khối này cho phép các nhà khoa học dữ liệu xây dựng mô hình, kiểm tra mô hình và thực thi mô hình trên các tệp dữ liệu được lưu trữ trong nền tảng CEPH.

Các mô hình học máy sau khi phát triển được chuyển sang khối ứng dụng thông minh để tích hợp với các ứng dụng giải quyết các bài toán thực tiễn trong các lĩnh vực khác nhau như kinh tế, tài chính, ngân hàng, môi trường, du lịch…

Như vậy, kiến trúc của hệ thống openscience.vn bao gồm các khối, mỗi khối đảm nhận một chức năng khác nhau với các nền tảng công nghệ khác nhau. Các nền tảng công nghệ thành phần trong các khối được tích hợp với nhau trên một nền tảng

thống nhất Kubernetes (K8S) nhằm thực hiện một luồng công việc (pipeline), bao gồm: thu thập, lưu trữ, xử lý dữ liệu, phát triển mô hình học máy và đưa ra kết quả.

1.2. TỔNG QUAN VỀ CÁC GIẢI PHÁP TÍCH HỢP CÁC NỀN TẢNG MÃ NGUỒN MỞ PHỤC VỤ THU THẬP, PHÂN TÍCH DỮ LIỆU VÀ HỌC MÁY TRÊN THẾ GIỚI VÀ TẠI VIỆT NAM

Với sự phát triển mạnh mẽ của cuộc cách mạng công nghiệp lần thứ 4, việc xây dựng và phát triển các nền tảng xử lý, phân tích dữ liệu lớn và các nền tảng phục vụ phát triển các mô hình học máy là vấn đề đang được các nhà nghiên cứu, ứng dụng, các tập đoàn công nghệ lớn quan tâm. Cho đến nay, có rất nhiều các nền tảng phục vụ phát triển các mô hình học máy (machine learning flatform) đã được sử dụng rộng rãi trong cộng đồng như Kubeflow, Amazon SageMaker, Microsoft Azure Machine Learning Studio, Google Cloud AI và AutoML [3]. Song song với nó, nhiều nền tảng phân tích dữ liệu lớn cũng được các tập đoàn công nghệ lớn phát triển như Apache Spark, Hadoop, Google Cloud BigQuery, và rất nhiều nền tảng khác [2]. Tại Việt Nam, một số tập đoàn đã xây dựng các nền tảng phân tích dữ liệu lớn như nền tảng Bkav Big Data Platform của BKAV cho phép thu thập, lưu trữ, xử lý và phân tích dữ liệu lớn. Tuy nhiên, các giải pháp tích hợp cả hai nền tảng: 1) nền tảng phục vụ thu thập, lưu trữ, xử lý, phân tích dữ liệu lớn; 2) Nền tảng phục vụ phát triển mô hình ML nhằm triển khai đầy đủ một tiến trình (pipeline) từ thu thập, lưu trữ, xử lý dữ liệu lớn và phát triển mô hình học máy còn hạn chế và có tiềm năng ứng dụng hiệu quả.

Việc tích hợp các nền tảng phục vụ xử lý, phân tích dữ liệu lớn và phát triển học máy đã và đang được thực hiện. Tuy nhiên, việc tích hợp các nền tảng đồng thời phục vụ cả hai nhiệm vụ nêu trên trong một khung kiến trúc thống nhất chưa được thực hiện. Do đó, mục tiêu nghiên cứu của luận văn là thực hiện tích hợp 05 nền tảng được mô tả trên đây với hai khối: khối nền tảng thu thập, lưu trữ, xử lý dữ liệu lớn (Nifi, CEPH, Spark, Kubernetes) và khối phát triển mô hình học máy (Kubeflow).

Đề tài đề cập đến giải pháp tích hợp các nền tảng mã nguồn mở nhằm thực hiện một quy trình phân tích dữ liệu đầy đủ (pipeline). Bao gồm các bước thu thập, lưu trữ, xử lý, phân tích dữ liệu lớn, học máy. Trước hết, chúng tôi mô tả vắn tắt về các nền tảng được sử dụng cho giải pháp tích hợp, là các nền tảng mã nguồn mở được sử dụng rộng rãi hiện nay và có nhiều tính năng vượt trội.

1.2.1. Kubernetes (K8S)

Kubernetes là một nền tảng mã nguồn mở, khả chuyển, tự động hoá việc quản lý, khả năng mở rộng và triển khai ứng dụng dưới dạng container và service. Kubernetes ban đầu được phát triển và thiết kế bởi các kỹ sư tại Google nhằm triển khai các ứng dụng trên đám mây, đây cũng là công nghệ đằng sau các dịch vụ đám mây của Google. Có thể nói Kubernetes là môi trường triển khai ứng dụng tiên tiến nhất, với một hệ sinh thái lớn và phát triển nhanh chóng. Các chức năng chính của Kubernetes là cân bằng tải, điều chỉnh bộ nhớ, tự động cấp phát và thu hồi và quản lý cấu hình hệ thống [9]. Một số lợi ích khi sử dụng Kubernetes:

- Kubernetes cung cấp các công cụ cần thiết để phát triển ứng dụng nhanh chóng trong khi vẫn duy trì sử ổn định. Ngoài ra, Kubernetes còn sử dụng Container Image mà trong đó ứng dụng sẽ được đóng gói lập tức. Nếu ứng dụng được phát triển thêm chức năng mới sẽ tương đương với việc tạo ra một Container Image mới. Vậy nên, khi triển khai, ta chỉ cần thay thế Image cũ bằng Image mới. Và nếu có lỗi, ta có thể trở lại phiên bản ổn định trước đó ngay lập tức bằng cách sử dụng lại Image cũ.
- Application sẽ được chia nhỏ thành nhiều Service mà mỗi Service sẽ chỉ thực hiện một chức năng duy nhất (còn được gọi là microservice). Mỗi Service sẽ được duy trì bởi một nhóm microservice và có thể scale dễ dàng hơn rất nhiều so với trong hệ thống thông thường.
- Kubernetes tự động khôi phục nếu có sự cố. Khi một Container dừng hoạt động, Kubernetes sẽ tự động lên lịch để chạy một Container khác.
- Nhiều application có thể chạy trên cùng một máy mà không ảnh hưởng đến nhau.
- Tự động hóa việc phân phối các ứng dụng trên toàn cụm, đảm bảo mức độ sử dụng cao hơn so với công cụ truyền thống. Kubernetes API giúp ứng dụng có thể di động trên nhiều môi trường khác nhau.

1.2.2. CEPH

Ceph là nền tảng mã nguồn mở để xây dựng hạ tầng lưu trữ (storage) phân tán, ổn định, độ tin cậy và hiệu năng cao, dễ dàng mở rộng. Với hệ thống lưu trữ được điều khiển bằng phần mềm, Ceph cung cấp những giải pháp lưu trữ như:

- Lưu trữ theo đối tượng (Object storage): Object storage system của Ceph cung cấp một số tính năng vượt trội hơn so với nhiều hệ thống lưu trữ Object hiện nay: Ceph cung cấp giao diện File System truyền thống với POSIX. Object storage system là một cải tiến đáng kể, nhưng

chúng vẫn còn phải thực hiện nhiều hơn so với các File System truyền thống. Khi các yêu cầu về lưu trữ tăng lên cho các ứng dụng hiện tại, tổ chức có thể cấu hình các ứng dụng hiện tại để sử dụng Ceph File System. Có nghĩa là người dùng có thể chạy một Storage Cluster cho Object, Block và lưu trữ dữ liệu dựa trên File.

- Lưu trữ theo khối (Block storage): Hệ thống lưu trữ Object của Ceph không giới hạn native binding hoặc RESTful APIs. Ta có thể mount Ceph như một lớp cung ứng mỏng Block Device. Khi người dùng viết dữ liệu trên Ceph bằng cách sử dụng Block Device, Ceph tự động hóa đồng bộ và tạo bản sao dữ liệu trên Cluster, RADOS Block Device (RBD) của Ceph cũng tích hợp với Kernel Virtual Machine (KVM), mang lại việc lưu trữ ảo hóa không giới hạn tới KVM chạy trên Ceph client của người dùng.
- Lưu trữ theo tệp dữ liệu (File storage): Thư viện phần mềm của Ceph cung cấp các ứng dụng cho khách hàng với khả năng truy cập trực tiếp tới hệ thống lưu trữ dựa trên RADOS Object và cung cấp một nền tảng cho một số tính năng cao cấp của Ceph, bao gồm RADOS Block Device (RBD), RADOS Gateway và Ceph File System.

Tất cả những giải pháp nêu trên đều được tích hợp trong một nền tảng đơn nhất. Ceph chạy trên nền tảng điện toán đám mây (cloud) với các thiết bị phần cứng ổn định và tiên tiến nhất, giúp tiết kiệm chi phí và sử dụng dễ dàng [4]. Cho đến nay, Ceph là công nghệ lưu trữ tiên tiến nhất với kiến trúc lưu trữ hồ dữ liệu (Lakehouse).

1.2.3. Apache Nifi

Apache NiFi là một trong những giải pháp mã nguồn mở phổ biến cho phép kết nối với nhiều nguồn dữ liệu khác nhau và đưa dữ liệu vào nền tảng lưu trữ dữ liệu. Nifi sử dụng kiến trúc có thể cho phép tạo các trình kết nối mới bằng Java. Quá trình tích hợp dữ liệu bao gồm một số tác vụ chính như: kết nối với các nguồn dữ liệu thời gian thực (streaming data) hoặc nguồn dữ liệu dạng gói (batch data); chuyển đổi dữ liệu từ các nguồn dữ liệu sang nền tảng lưu trữ dữ liệu mà vẫn giữ nguyên nội dung và định dạng của dữ liệu (việc bảo toàn dữ liệu này rất quan trọng đối với các dữ liệu được xử lý lại sau này); ghi lại các số liệu thống kê và trạng thái của dữ liệu sau đó lưu thông tin vào khối lưu trữ dữ liệu. Nifi có thể tích hợp dữ liệu từ nhiều nguồn khác nhau, bao gồm: cơ sở dữ liệu quan hệ (RDBMS), tệp (files), phần mềm dạng dịch vụ (SaaS, APIs) và dữ liệu luồng từ các hệ thống IoT. Ngoài ra, Nifi cho

phép cấu hình để thu thập dữ liệu với dung lượng lưu trữ lớn thông qua khả năng xây dựng luồng chuyển dữ liệu tự động giữa các hệ thống từ rất nhiều kiểu nguồn và đích khác nhau như [5, 10]:

- Các loại RDBMS: Oracle, MySql, Postgre, …
- Các loại DB NoSQL: Mongo, HBase, Cassandra, …
- Từ các nguồn web như: HTTP, web-socket
- Lấy hoặc đẩy dữ liệu streaming vào Kafka
- Hay là từ: FTP, log

Ba nhóm tính năng nổi bật của Nifi bao gồm khả năng quản lý luồng dữ liệu; việc sử dụng, vận hành một cách dễ dàng; và khả năng mở rộng.

Thứ nhất là về khả năng quản lý luồng dữ liệu:

- **Đảm bảo an toàn:** Mỗi đơn vị dữ liệu trong luồng của bạn sẽ được biểu diễn bởi một Object có tên là FlowFile. Nó sẽ ghi lại tất cả các thông tin về dữ liệu trong luồng như đang được xử lý bởi khối nào, đang được chuyển đi đâu, … Lịch sử xử lý của một FlowFile lại được lưu trữ trong Provenance Repo để chúng ta truy vết. Kết hợp với cơ chế Copy-on-Write, NiFi lưu trữ lại dữ liệu tại từng bước trong luồng trước khi xử lý, giúp ta dễ dàng chạy lại dữ liệu.
- **Data Buffering:** Tính năng này giúp giải quyết vấn đề tốc độ nhận chậm hơn tốc độ truyền giữa hai hệ thống khác nhau. Nó hoạt động dựa theo cơ chế Queue giữa hai khối xử lý trong luồng. Dữ liệu này sẽ được giữ trên RAM, nhưng nếu nó vượt qua ngưỡng thiết lập thì dữ liệu sẽ được đưa xuống ổ cứng.
- **Thiết lập độ ưu tiên:** Trong một số trường hợp mà ta cần ưu tiên xử lý một loại dữ liệu nào đó. Ví dụ như log có nhãn error chẳng hạn, ta có thể thiết lập để hệ thống được xử lý ngay lập tức trước khi xử lý những log warning.
- **Hỗ trợ đánh đổi giữa tốc độ và khả năng chịu lỗi:** Có những luồng dữ liệu ta cần đảm bảo tuyệt đối về tính toàn vẹn và an toàn của dữ liệu chấp nhận độ trễ cao. Và có những luồng ta lại cần chuyển được dữ liệu tới đích trong thời gian ngắn nhất có thể. NiFi sẽ hỗ trợ ta cài đặt để cân bằng giữa hai yếu tố này.

Thứ hai là về khả năng sử dụng, vận hành một cách dễ dàng:

- Việc tạo ra một luồng dữ liệu sẽ được thực hiện hoàn toàn trên giao diện WEB, và bằng vài thao tắc kéo thả ta sẽ nhanh chóng tạo được một luồng đơn giản.
- Ngoài ra ta còn có thể tái sử dụng luồng dữ liệu đã tạo thông qua một template chứa một luồng cơ bản để sử lại khi cần.
- Theo dõi trực quan lịch sử xử lý của dữ liệu khi cần kiểm tra lỗi.
- Chạy lại được cả dữ liệu tại từng bước xử lý
- Dễ dàng lập trình được một thành phần xử lý, điều khiển, report hay UI trong NiFi khi cần. Ví dụ như một khối encode hoặc decode dữ liệu.

Cuối cùng, cần nhắc tới một tính năng quan trọng của các ứng dụng trong các hệ thống phân tán là khả năng mở rộng theo chiều ngang (thêm server vào cụm). Nếu một luồng dữ liệu trên trên một server NiFi có thể xử lý được 100MB/s , nhưng yêu cầu thực tế lại lên đến 500MB/s thì ta có thể cài đặt một cụm gồm nhiều server để xử lý dữ liệu một cách song song mà không cần nâng cấp cấu hình của server đang sử dụng.

Do đó, Nifi được xem là một trong những nền tảng thu thập dữ liệu thô tiên tiến nhất hiện nay với nhiều tính năng và hiệu quả vượt trội.

1.2.4. Apache Spark

Apache Spark là một hệ thống xử lý phân tán mã nguồn mở được sử dụng cho các khối lượng công việc xử lý dữ liệu lớn. Hệ thống này sử dụng khả năng ghi vào bộ nhớ đệm nằm trong bộ nhớ và thực thi truy vấn tối ưu hóa nhằm giúp truy vấn phân tích nhanh dữ liệu có kích thước bất kỳ [6]. Apache Spark cung cấp các API phát triển bằng ngôn ngữ Java, Scala, Python và R và hỗ trợ tái sử dụng mã trên nhiều khối lượng công việc, chẳng hạn như xử lý dữ liệu theo lô, truy vấn tương tác, xử lý dữ liệu theo luồng [11].

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

- **Spark Core** là nền tảng cho các thành phần còn lại và các thành phần này muốn khởi chạy được thì đều phải thông qua Spark Core do Spark Core đảm nhận vai trò thực hiện công việc tính toán và xử lý trong bộ

nhớ (In-memory computing) đồng thời nó cũng tham chiếu các dữ liệu được lưu trữ tại các hệ thống lưu trữ bên ngoài.

- **Spark SQL** cung cấp một kiểu data abstraction mới (SchemaRDD) nhằm hỗ trợ cho cả kiểu dữ liệu có cấu trúc (structured data) và dữ liệu nửa cấu trúc (semi-structured data – thường là dữ liệu dữ liệu có cấu trúc nhưng không đồng nhất và cấu trúc của dữ liệu phụ thuộc vào chính nội dung của dữ liệu ấy). Spark SQL hỗ trợ DSL (Domain-specific language) để thực hiện các thao tác trên DataFrames bằng ngôn ngữ Scala, Java hoặc Python và nó cũng hỗ trợ cả ngôn ngữ SQL với giao diện command-line và ODBC/JDBC server.
- **Spark Streaming** được sử dụng để thực hiện việc phân tích stream bằng việc coi stream là các mini-batches và thực hiệc kỹ thuật RDD transformation đối với các dữ liệu mini-batches này. Qua đó cho phép các đoạn code được viết cho xử lý batch có thể được tận dụng lại vào trong việc xử lý stream, làm cho việc phát triển lambda architecture được dễ dàng hơn. Tuy nhiên điều này lại tạo ra độ trễ trong xử lý dữ liệu (độ trễ chính bằng mini-batch duration) và do đó nhiều chuyên gia cho rằng Spark Streaming không thực sự là công cụ xử lý streaming giống như Storm hoặc Flink.
- **MLlib (Machine Learning Library)**: MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Theo các so sánh benchmark Spark MLlib nhanh hơn 9 lần so với phiên bản chạy trên Hadoop (Apache Mahout).
- **GraphX**: Graphx là nền tảng xử lý đồ thị dựa trên Spark. Nó cung cấp các Api để diễn tảcác tính toán trong đồ thị bằng cách sử dụng Pregel Api.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được (hay còn gọi là Spark Streaming). Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,…. Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json…) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào. Trong hệ thống mà chúng ta đang thực hiện, Spark sẽ được lấy dữ liệu từ kho lưu trữ dữ liệu để xử lý và vận hành.

1.2.5. Kubeflow

Kubeflow là nền tảng mã nguồn mở nhằm triển khai các mô hình học máy (Machine learning – ML). Nó được phát triển trên nền tảng Kubernetes giúp tự động hoá việc triển khai, mở rộng quy mô và quản lý các ứng dụng ML. Bằng cách tận dụng sức mạng của Kubernetes, Kubeflow cho phép các nhà khoa học dữ liệu xây dựng, triển khai và quản lý quy trình phát triển ML một cách dễ dàng, hiệu quả, từ đó đẩy nhanh quá trình phát triển và ứng dụng các mô hình ML vào các bài toán thực tế [12]. Kubeflow bao gồm các tính năng như cung cấp nền tảng hợp nhất với các công cụ phát triển ML như Tensorflow, PyTorch linh hoạt trong kiến trúc mô đun, tự động hoá phân bổ tài nguyên và tích hợp với các công cụ triển khai liên tục cho phép ta tự động hoá vòng đời ML, giúp giảm thời gian và công sức cần thiết để triển khai các mô hình ML [7].

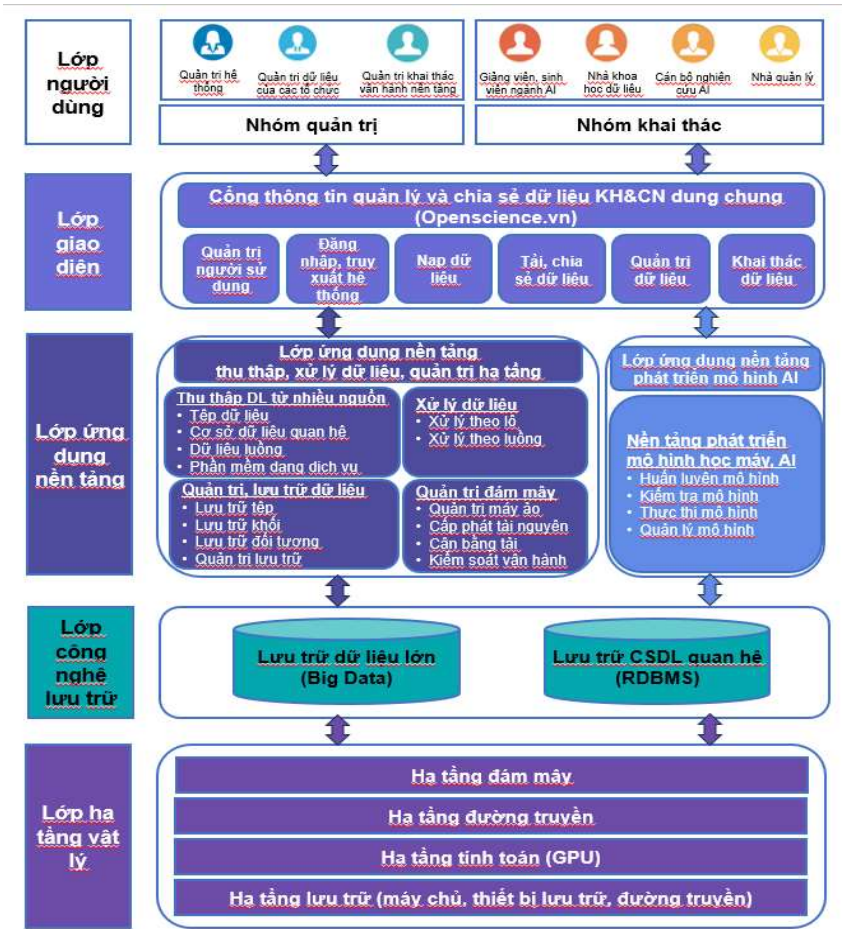
Một số ưu điểm của Kubeflow như:

- Kubeflow tương thích với các dịch vụ đám mây (AWS, GCP, Azure) và các dịch vụ tự lưu trữ.
- Nó cho phép các kỹ sư học máy tích hợp tất cả các loại framework AI để huấn luyện, tinh chỉnh, lên lịch và triển khai các mô hình.
- Nó cung cấp một bảng điều khiển tập trung để giám sát và quản lý các pipeline, chỉnh sửa mã bằng Jupyter Notebook, theo dõi thử nghiệm, registry mô hình và lưu trữ tạo tác.

Do đó, Kubeflow được xem là nền tảng phát triển mô hình ML tiên tiến nhất hiện nay.

CHƯƠNG 2. XÂY DỰNG GIẢI PHÁP TÍCH HỢP CÁC NỀN TẢNG THU THẬP, PHÂN TÍCH DỮ LIỆU VÀO HỆ THỐNG OPENSCIENCE.VN

Kiến trúc ứng dụng hệ thống Openscience.vn được mô tả ở Hình 2.1, bao gồm các lớp:



Hình 2. 1. Mô hình kiến trúc ứng dụng Openscience.vn

- Lớp người dùng: bao gồm nhóm quản trị và nhóm khai thác hệ thống. Nhóm quản trị hệ thống (thuộc NASSATI) có nhiệm vụ quản trị, phân quyền cho người sử dụng và phê duyệt, hiển thị các bộ dữ liệu trên Openscience.vn. Nhóm khai thác hệ thống bao gồm các nhà khoa học dữ liệu, cán bộ nghiên cứu, giảng viên, sinh viên đại học thuộc lĩnh vực khoa học dữ liệu, trí tuệ nhân tạo và các lĩnh vực liên quan.
- Lớp giao diện: là cổng thông tin khai thác hệ thống Openscience.vn. Lớp giao diện cho phép các lớp người dùng truy cập và khai thác các nền tảng nên trong hệ thống với cơ chế đăng nhập một lần SSO, bao gồm các khối chức năng:

quản trị hệ thống, nạp dữ liệu, tải dữ liệu, chia sẻ dữ liệu theo phân quyền và truy cập vào các nền tảng bên trong của hệ thống Openscience.vn

- Lớp ứng dụng nền tảng: bao gồm các nền tảng thập dữ liệu, xử lý dữ liệu, quản trị, lưu trữ dữ liệu, quản trị đám mây và phát triển mô hình học máy, AI.

Ngoài ra, hệ thống Openscience.vn còn bao gồm các lớp:

- Lớp lưu trữ dữ liệu: lưu trữ dữ liệu lớn bằng công nghệ CEPH và lưu trữ cơ sở dữ liệu quan hệ với hệ quản trị PostgreSQL.
- Lớp hạ tầng vật lý: bao gồm hạ tầng đám mây (Cloud), hạ tầng đường tryền (Internet), hạ tầng tính toán (CPU, GPU) và hạ tầng lưu trữ vật lý (máy chủ, thiết bị lưu trữ)

Từ các lớp trên, ta sẽ xây dựng lên những thành phần của hệ thống như sau:

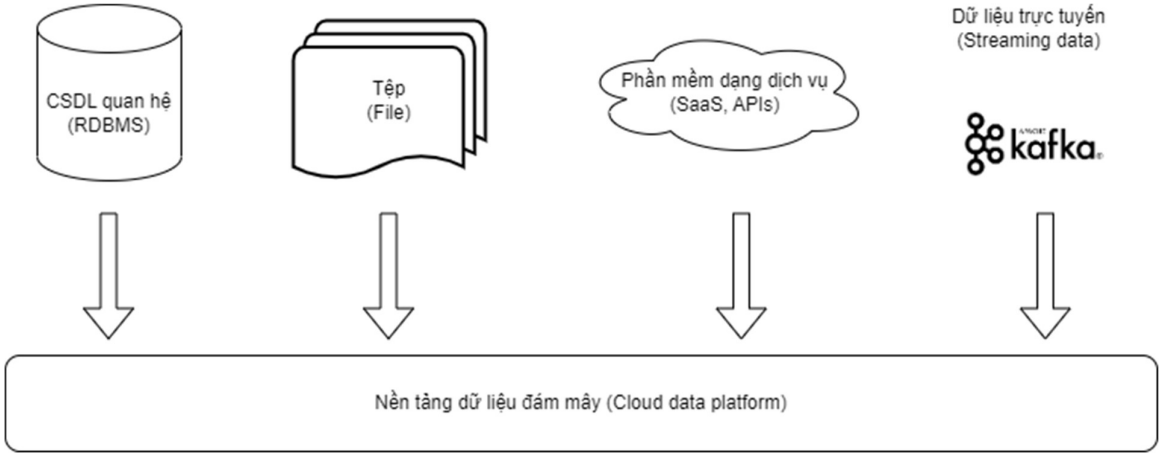
- Cổng thông tin quản lý và chia sẻ dữ liệu dùng chung Openscience.vn
- Nền tảng thu thập dữ liệu: Nền tảng cho phép người sử dụng thu thập dữ liệu từ các nguồn khác nhau và lưu trữ vào hệ thống lưu trữ dữ liệu lớn CEPH. Các nguồn dữ liệu bao gồm: tệp dữ liệu (files), cơ sở dữ liệu quan hệ (RDBMS), hệ thống phần mềm (API), dữ liệu luồng (IoT). Nền tảng sử dụng công nghệ NIFI tích hợp trên công nghệ lưu trữ CEPH.
- Nền tảng quản trị hệ thống bao gồm:
    - Quản trị lưu trữ dữ liệu lớn CEPH với các tác vụ lưu trữ tệp (files), lưu trữ khối (blocks), lưu trữ đối tượng (Objects) và quản trị lưu trữ.
    - Quản trị hạ tầng đám mây trên nền tảng K8S, bao gồm các tác vụ như quản trị máy ảo, cấp phát tài nguyên, cân bằng tải, kiểm soát vận hành.
- Nền tảng xử lý dữ liệu: Cho phép thực hiện các tác vụ xử lý dữ liệu theo lô (batch processing) và theo luồng (streaming processing) từ dữ liệu được đọc từ CEPH hoặc từ IoT sử dụng công nghệ Spark tích hợp trên nền tảng.
- Nền tảng phân tích dữ liệu: là môi trường cho phép các nhà khoa học dữ liệu huấn luyện mô hình, kiểm thử mô hình, thực thi và quản lý mô hình học máy sử dụng công nghệ Kubeflow tích hợp trên nền tảng.

Trong đó, hệ thống Openscience.vn đang hoạt động trên Kubernetes đã được cài đặt sẵn các hệ thống Apache Nifi, CEPH, Kubeflow, Apache Spark, Keycloak. Vậy nên trong chương này, chúng ta sẽ tập chung vào việc cấu hình, kết nối và phối hợp những hệ thống theo giải pháp đã đề ra.

2.1. XÂY DỰNG GIẢI PHÁP TÍCH HỢP NỀN TẢNG NIFI VÀO HỆ THỐNG OPENSCIENCE.VN

2.1.1. Mô tả công cụ Nifi

Apache NiFi là một trong những giải pháp mã nguồn mở phổ biến cho phép kết nối với nhiều nguồn dữ liệu khác nhau và đưa dữ liệu vào nền tảng dữ liệu. NiFi sử dụng kiến trúc có thể cho phép tạo các trình kết nối mới bằng Java. Quá trình tích hợp dữ liệu này bao gồm một số tác vụ chính như: kết nối với các nguồn dữ liệu thời gian thực (streaming data) hoặc nguồn dữ liệu dạng gói (batch data); chuyển đổi dữ liệu từ các nguồn dữ liệu sang nền tảng lưu trữ dữ liệu mà vẫn giữ nguyên nội dung và định dạng của dữ liệu (việc bảo toàn dữ liệu này rất quan trọng đối với các dữ liệu được xử lý lại sau này); ghi lại các số liệu thống kê và trạng thái của dữ liệu sau đó lưu thông tin vào khối lưu trữ dữ liệu.



Hình 2. 2. Các nguồn dữ liệu của hệ thống

Trong các ứng dụng thực tế có có rất nhiều nguồn dữ liệu khác nhau, tuy nhiên hệ thống chỉ tập chung sử dụng chủ yếu vào bốn nguồn dữ liệu chính là: cơ sở dữ liệu quan hệ (RDBMS), dữ liệu tệp (file), dữ liệu của các phần mềm dưới dạng dịch vụ (SaaS, API), dữ liệu trực tuyến (Streaming).

- Tích hợp dữ liệu từ CSDL quan hệ (RDBMS): Hệ thống cung cấp các phương pháp: tích hợp từ RDBMS sử dụng SQL; tích hợp từ cơ sở dữ liệu NoSQL (BigData); tích hợp siêu dữ liệu (metadata) cho RDBMS và NoSQL.
    - Tích hợp từ RDBMS sử dụng SQL: Sử dụng các câu lệnh truy vấn dữ liệu để thực hiện truy vấn dữ liệu từ CSDL nguồn và lưu trữ vào nền tảng.

- Tích hợp từ CSDL NoSQL (BigData): Sử dụng các công cụ (Tool) của các CSDL NoSQL để định nghĩa một quy trình (pipeline) tích hợp dữ liệu.
- Tích hợp các siêu dữ liệu (metadata) từ hệ thống nguồn (RDBMS, NoSQL) bằng các công cụ được xây dựng, bảo đảm tính toàn vẹn dữ liệu từ hệ thống nguồn vào nền tảng lưu trữ dữ liệu.

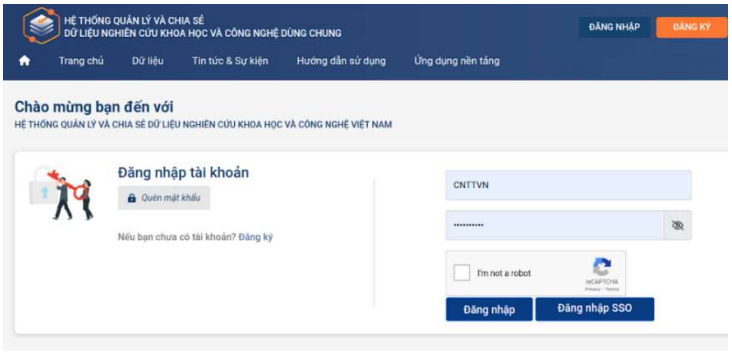
- Tích hợp dữ liệu từ dữ liệu dòng (thời gian thực): Hệ thống sử dụng giải pháp Apache Kafka thực hiện tích hợp dữ liệu dòng (streams) có yếu tố thời gian thực từ các ứng dụng, điển hình là các ứng dụng IoT.
- Tích hợp dữ liệu từ các ứng dụng SaaS: Ứng dụng SaaS ngày càng trở nên phổ biến trong giai đoạn hiện nay. Trong hệ thống, việc tích hợp với các ứng dụng SaaS được thực hiện bằng cách sử dụng API qua giao thức HTTP(s).

2.1.2. Thu thập dữ liệu từ tệp hệ thống (file systems)

Tích hợp dữ liệu từ tệp (file): Dữ liệu tệp (File) là loại dữ liệu phổ biến xuất hiện trong các nguồn dữ liệu. Hệ thống cung cấp hai phương pháp chuyển dữ liệu file vào nền tảng lưu trữ. Phương pháp thứ nhất là sử dụng giao thức truyền tệp FTP. Phương pháp thứ hai là sử dụng lưu trữ đám mây thay vì máy chủ FTP. Các tệp nguồn được lưu trữ tại một đám mây cục bộ và hệ thống sẽ thực hiện sao chép từ đám mây nguồn sang đám mây đích của nền tảng lưu trữ. Ta bắt đầu thực hiện tiến hành cấu hình thu thập dữ liệu từ những tệp hệ thống lên Apache Nifi đã được cài đặt từ trước.

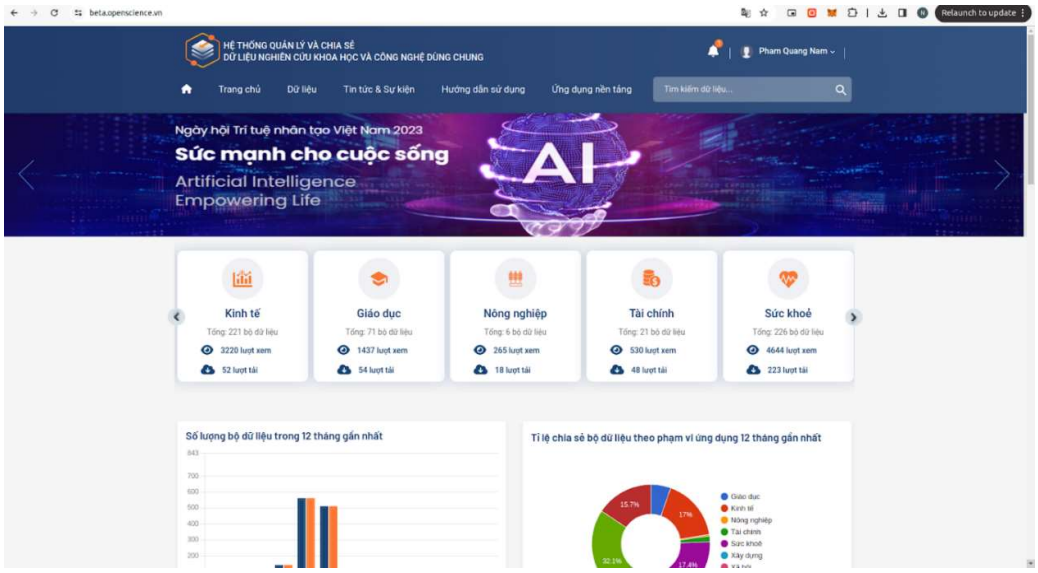
2.1.2.1. Đăng nhập hệ thống Openscience.vn

Đầu tiên, ta truy cập vào trang chủ hệ thống theo đường dẫn "**https://openscience.vn/**". Giao diện chính của hệ thống sẽ hiện ra như sau:



Hình 2. 3. Giao diện đăng nhập chính

Để đăng nhập vào hệ thống, ta nhấn vào nút Đăng nhập SSO, một màn hình đăng nhập sẽ hiện ra. Ta sử dụng tài khoản đã được cung cấp để đăng nhập vào hệ thống. Sau khi khai báo xong, ta nhấn Sign In để đăng nhập. Nếu quá trình đăng nhập diễn ra thành công, ta sẽ có thể truy cập được vào giao diện chính của hệ thống.



Hình 2. 4. Giao diện chính của hệ thống Openscience

2.1.2.2. Các bước cấu hình nguồn dữ liệu

- Khối truyền dữ liệu

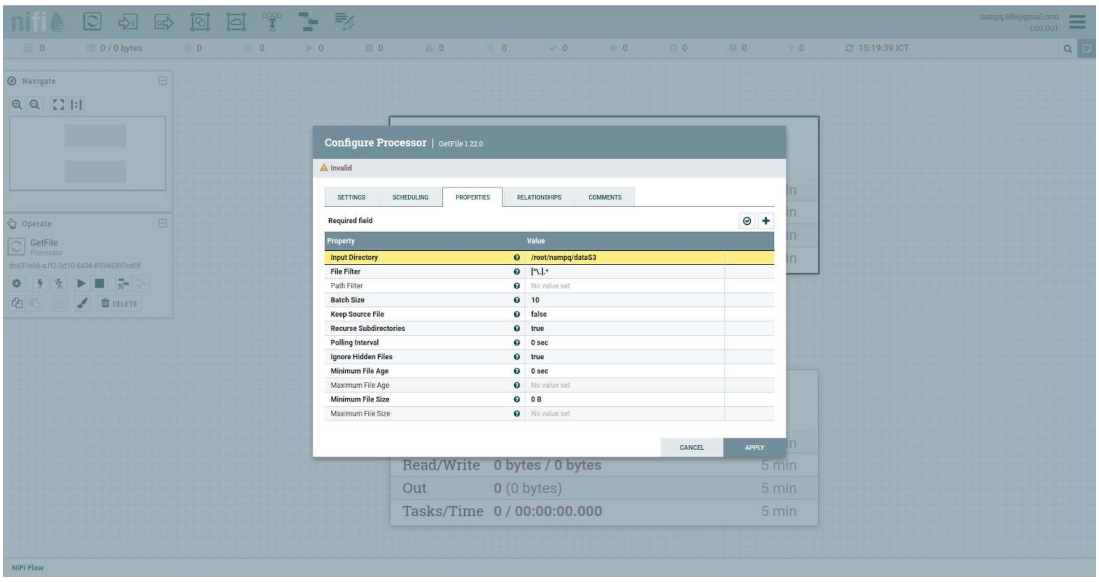
Trên thanh công cụ của Apache Nifi, nhấn chọn nút tạo một bộ xử lý mới. Từ cửa sổ hiện ra, nhập tên khối chức năng GetFile, sau đó ấn ADD. Lúc này trên màn hình làm việc chính của Apache Nifi sẽ hiện ra khối chức năng GetFile vừa được tạo. Nháy đúp chuột vào khối GetFile, lúc này một cửa sổ dùng để cài đặt các thông tin sẽ hiện ra.

Ta sẽ cần điền các thông tin quan trọng sau:

• Input Directory: đường dẫn đến thư mục chứa dữ liệu.

• File Filter: lọc các tệp muốn gửi đi dựa vào đặc điểm tên tệp. Ví dụ trong thư mục cần gửi đi có rất nhiều định dạng tệp như json, txt, word, v.v thì có thể cài đặt theo cú pháp [^\.].* để chuyển tất cả dữ liệu.

• Keep Source File: nếu ta chọn true tệp sẽ được giữ lại tại máy gốc sau khi quá trình gửi hoàn tất, ngược lại nếu ta chọn false thì tệp dữ liệu sẽ bị xóa sau khi quá trình gửi hoàn tất.



Hình 2. 5. Cửa sổ cài đặt các thông tin truyền dữ liệu

Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.

- Khối nhận dữ liệu

Thao tác tương tự như khối Truyền dữ liệu, nhập khối chức năng PutS3Object, sau đó ấn nút ADD. Lúc này trên màn hình làm việc chính của Apache Nifi sẽ hiện ra khối chức năng PutS3Object vừa được tạo. Nháy đúp chuột vào khối PutS3Object, lúc này một cửa sổ dùng để cài đặt các thông tin sẽ hiện ra.

Ta sẽ cần điền các thông tin quan trọng sau:

• Object Key: đường dẫn đến nơi lưu trữ dữ liệu trong kho dữ liệu. Ta cần tuân thủ theo cấu trúc thư mục được quy định từ trước:

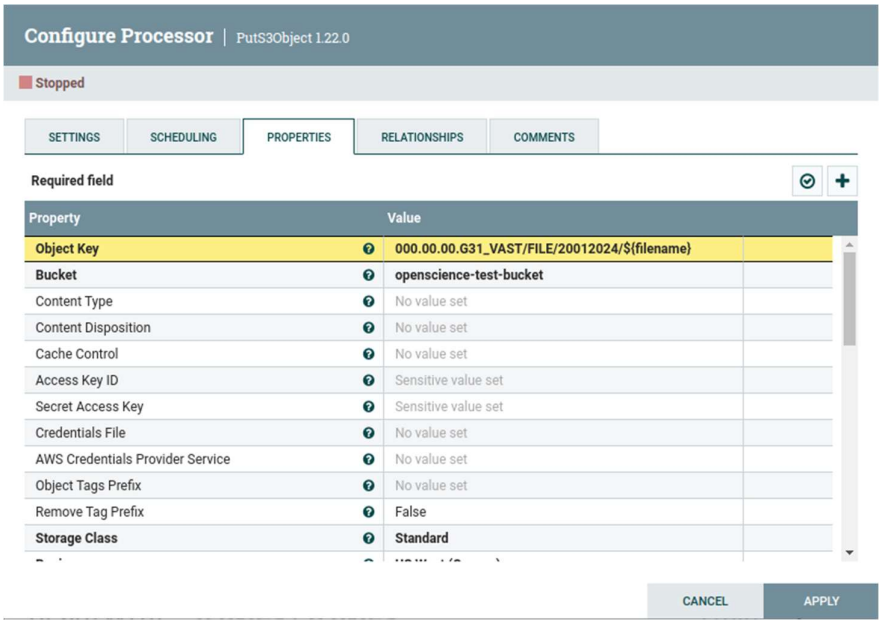
<Mã đơn vị>/FILE/<Tên bộ dữ liệu><Tên tệp dữ liệu>

• Bucket: tên bucket tại kho lưu trữ.

• Access Key ID: khóa truy cập vào kho dữ liệu.

• Secret Access Key: mật khẩu truy cập kho lưu trữ.

• Endpoint Override URL: địa chỉ truy cập đến kho lưu trữ dữ liệu.

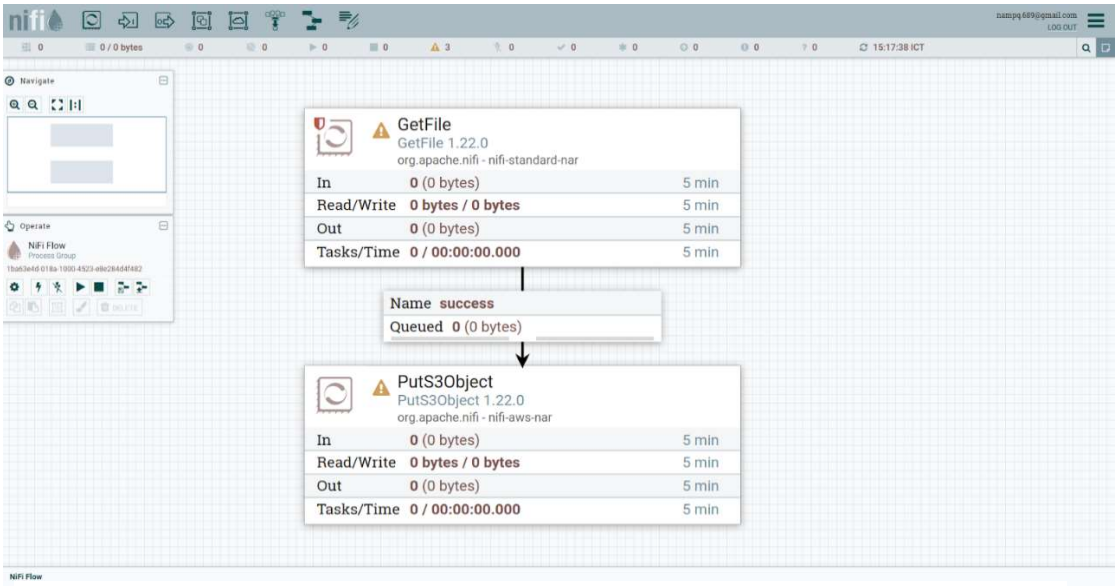


Hình 2. 6. Cửa sổ cài đặt các thông tin nhận dữ liệu

Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.

- Kết nối hai khối truyền và nhận dữ liệu

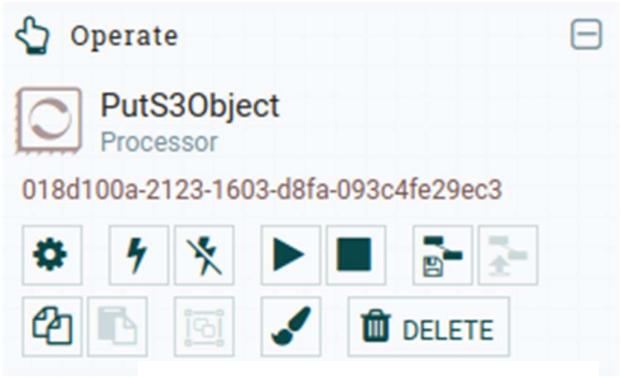
Sau khi đã tạo được hai khối truyền dữ liệu và khối nhận dữ liệu, di chuyển con trỏ chuột đến khối GetFile, kéo mũi tên hướng tới khối PutS3Object, lúc này một liên kết sẽ được hình thành.



Hình 2. 7. Kết nối hai khối truyền và nhận dữ liệu

- Thực hiện chạy thu thập dữ liệu

Sau khi đã thực hiện các thao tác cài đặt và cấu hình, tại khối Operate trên giao diện chính của Apache Nifi, nhấn chọn nút Start để thực hiện gửi và nhận dữ liệu.



Hình 2. 8. Khối điều khiển Operate

Ta có thể xem trạng thái hoạt động của các khối chức năng bằng cách nhấn chuột phải vào khối trung gian nằm giữa khối GetFile và PutS3Object. Sau đó chọn View status history và List queue để xem thông tin chi tiết.

2.1.3. Thu thập dữ liệu luồng từ hệ thống IoT (data stream)

Để bắt đầu thực hiện thu thập dữ liệu luồng từ hệ thống IoT, ta cũng sẽ cần phải đăng nhập vào hệ thống thông qua Đăng nhập SSO để có thể truy cập được vào giao diện chính của hệ thống. Sau đó ta sẽ thực hiện các bước cấu hình nguồn dữ liệu như dưới đây.

- Khối truyền dữ liệu:

Trên thanh công cụ của Apache Nifi, nhấn chọn nút tạo một bộ xử lý mới. Từ cửa sổ hiện ra, nhập tên khối chức năng ConsumeMQTT, sau đó ấn ADD. Lúc này trên màn hình làm việc chính của Apache Nifi sẽ hiện ra khối chức năng ConsumeMQTT vừa được tạo.

Hình 2. 9. Khối chức năng mới tạo

Ta nháy đúp chuột vào khối ConsumeMQTT để thiết lập các thông số quan trọng sau:

• Broker URI: Địa chỉ nhận thông tin từ thiết bị IOT.

• Topic Filter: Nhận thông tin từ tên chủ đề nhất định.

• Max Queue Size: Khi tín hiệu được gửi tới đồng thời quá nhiều cùng một lúc, ta cài đặt cho giá trị này để giữ lại thông tin của N tín hiệu gần nhất.

Trong trường hợp thử nghiệm, giá trị này được cài đặt là 100. Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.

Ngoài ra, để có thể tùy chỉnh thông tin (ví dụ như tên dữ liệu) ta có thể khi báo thêm một khối chức năng có tên UpdateAttibute bằng các thao tác tương tự như trên và nối với khối ConsumeMQTT ban đầu bằng cách nháy đúp chuột vào khối UpdateAttribute, lúc này một cửa sổ dùng để cài đặt các thông tin sẽ hiện ra. Chuyển sang tab properties để khai báo thêm thuộc tính filename và điền giá trị của thuộc tính đó theo dạng json. Sau khi hoàn thành, ta nhấn APPLY để lưu các cài đặt.

- Khối nhận dữ liệu

Thao tác tương tự như khối Truyền dữ liệu, nhập khối chức năng PutS3Object, sau đó ấn nút ADD. Lúc này trên màn hình làm việc chính của Apache Nifi sẽ hiện ra khối chức năng PutS3Object vừa được tạo. Nháy đúp chuột vào khối PutS3Object, và chuyển sang tab properties để cấu hình các thông tin cho khối nhận dữ liệu.

Ta sẽ cần điền các thông tin quan trọng sau:

• Object Key: đường dẫn đến nơi lưu trữ dữ liệu trong kho dữ liệu.

Người dùng cần tuân thủ theo cấu trúc thư mục được quy định từ trước: <Mã đơn vị>/IOT/<Tên bộ dữ liệu><Tên tệp dữ liệu>

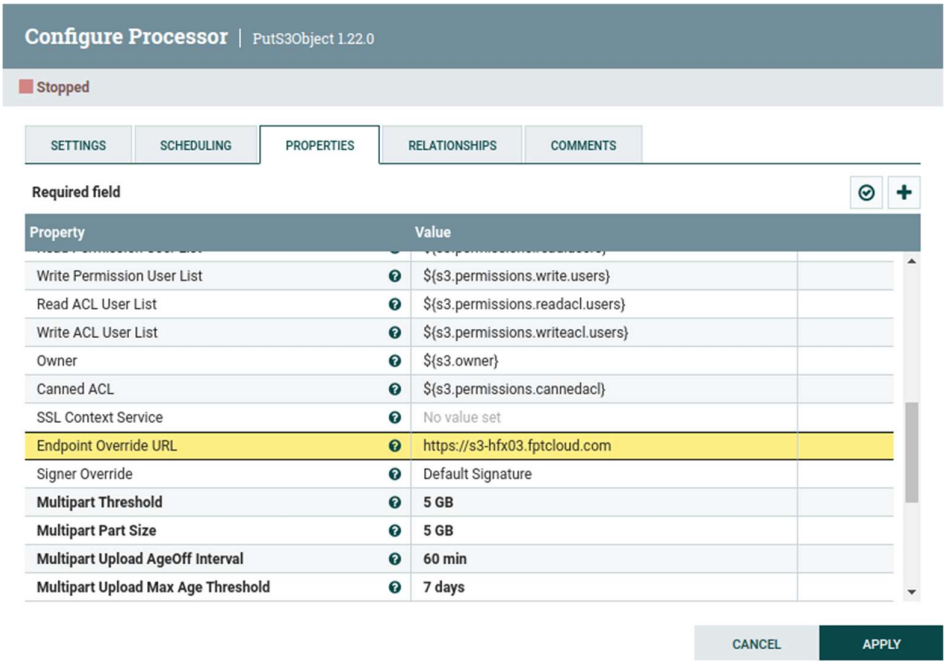
• Bucket: tên bucket tại kho lưu trữ.

• Access Key ID: khóa truy cập vào kho dữ liệu.

• Secret Access Key: mật khẩu truy cập kho lưu trữ.

• Endpoint Override URL: địa chỉ truy cập đến kho lưu trữ dữ liệu.

Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.



Hình 2. 10. Cấu hình các thuộc tính của khối nhận dữ liệu

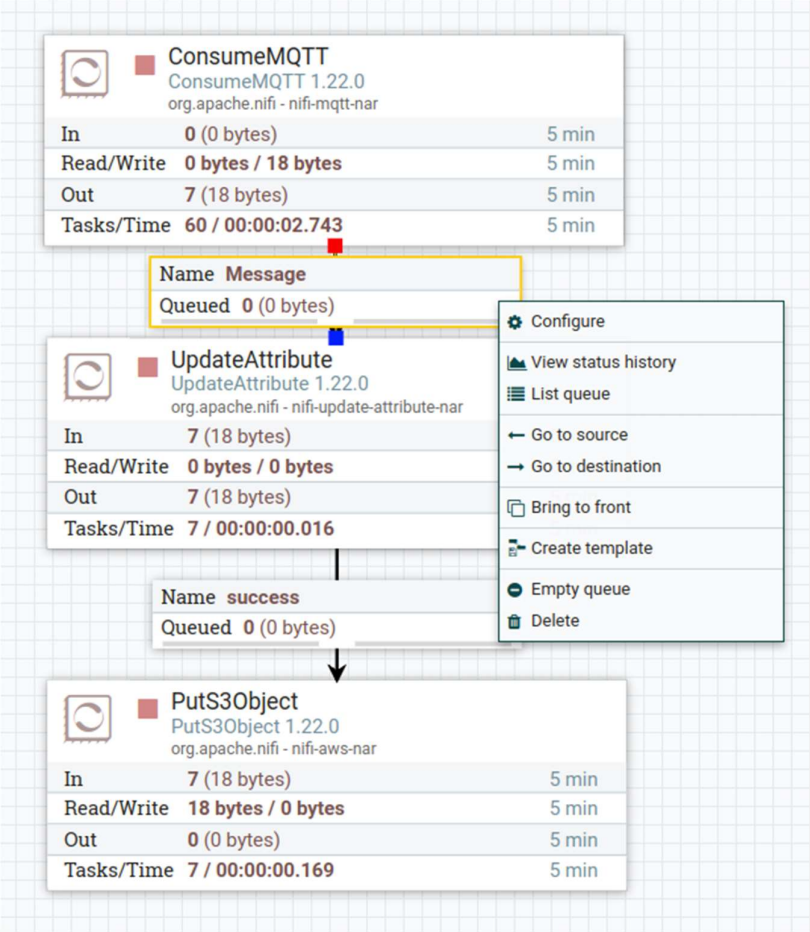
- Kết nối hai khối truyền và nhận dữ liệu

Sau khi đã tạo được hai khối truyền dữ liệu và khối nhận dữ liệu, di chuyển con trỏ chuột đến khối ConsumeMQTT, kéo mũi tên hướng tới khối UpdateAttibute, sau đó kéo tiếp mũi tên đến khối PutS3Object, lúc này một liên kết sẽ được hình thành.

- Thực hiện chạy thu thập dữ liệu

Khi đã có tín hiệu IOT gửi tới hệ thống, để tiến hành thu thập dữ liệu, ta sẽ truy cập tới Apache Nifi, tại khối Operate trên giao diện chính, nhấn chọn nút Start để thực hiện gửi và nhận dữ liệu.

Người dùng có thể xem trạng thái hoạt động của các khối chức năng bằng cách nhấn chuột phải vào khối trung gian nằm giữa các khối ConsumeMQTT, UpdateAttibute và PutS3Object. Sau đó chọn View status history và List queue để xem thông tin chi tiết.



Hình 2. 10. Luồng nhận và gửi dữ liệu IOT

2.1.4. Thu thập dữ liệu từ hệ thống CSDL quan hệ

Ta tiếp tục thực hiện đăng nhập vào hệ thống thông qua Đăng nhập SSO để có thể truy cập được vào giao diện chính của hệ thống và thực hiện các bước cấu hình nguồn dữ liệu.

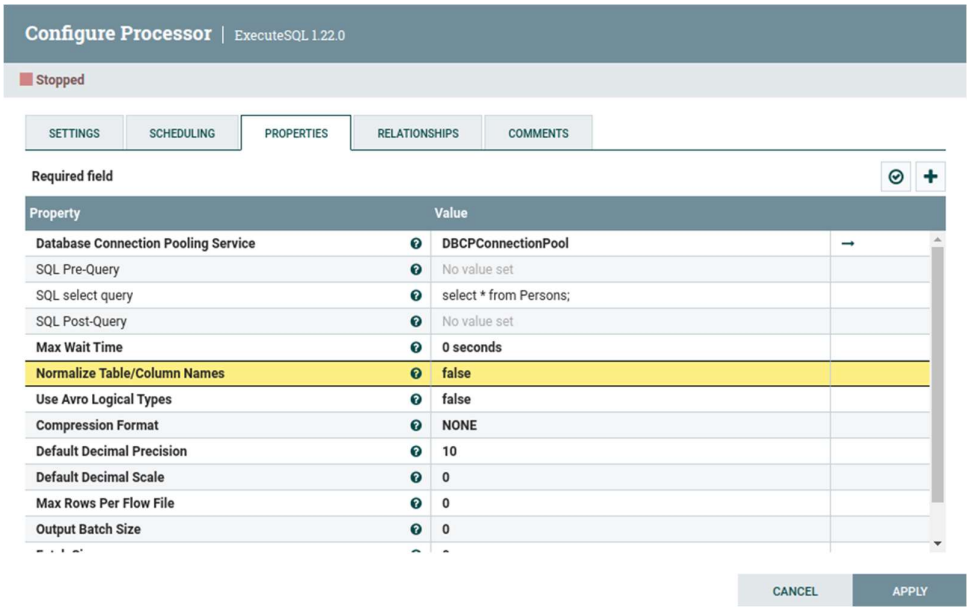
- Khối truyền dữ liệu

Trên thanh công cụ của Apache Nifi, nhấn chọn nút tạo một bộ xử lý mới. Từ cửa sổ hiện ra, nhập tên khối chức năng ExecuteSQL, sau đó ấn ADD. Lúc này trên màn hình làm việc chính của Apache Nifi sẽ hiện ra khối chức năng ExecuteSQL vừa được tạo. Ta nháy đúp chuột vào khối này và chọn tab properties để cấu hình những thông số của khối.

Ta sẽ cần điền các thông tin quan trọng sau:

• Database Connection Pooling Service: Chọn cấu hình cho dịch vụ cơ sở dữ liệu.

• SQL select query: Câu lệnh truy vấn cơ sở dữ liệu.



Hình 2. 11. Bảng thông số của khối ExecuteSQL

Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.

Để chuẩn hóa dữ liệu thu được từ cơ sở dữ liệu ta sẽ cần biến đổi chúng về dạng JSON thông qua khối ConvertAvroToJSON và đặt tên cho dữ liệu dạng JSON thu được từ cơ sở dữ liệu bằng khối UpdateAttribute. Trong đó:

• Đối với khối ConvertAvroToJSON, ta không cần cấu hình thuộc tính.

• Đối với khối UpdateAttribute, ta cần khai báo thêm thuộc tính tên là filename và điền giá trị của thuộc tính đó theo dạng json.

- Khối nhận dữ liệu

Thao tác tương tự như khối Truyền dữ liệu, ta thêm khối chức năng PutS3Object, sau đó truy cập tới thuộc tính của khối và thiết lập những thành phần quan trọng của khối như:

• Object Key: đường dẫn đến nơi lưu trữ dữ liệu trong kho dữ liệu với cấu trúc: <Mã đơn vị>/DB/<Tên bộ dữ liệu><Tên tệp dữ liệu>

• Bucket: tên bucket tại kho lưu trữ.

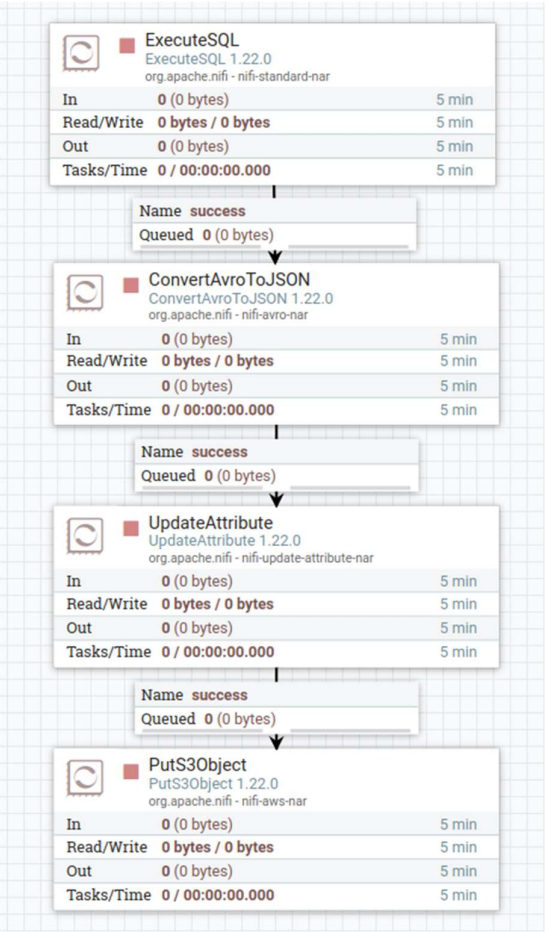
• Access Key ID: khóa truy cập vào kho dữ liệu.

• Secret Access Key: mật khẩu truy cập kho lưu trữ.

• Endpoint Override URL: địa chỉ truy cập đến kho lưu trữ dữ liệu.

Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.

- Kết nối hai khối truyền và nhận dữ liệu



Hình 2. 12. Kết nối các khối để nhận dữ liệu từ hệ thống CSDL quan hệ

Khi đã tạo được hai khối truyền dữ liệu và khối nhận dữ liệu ta sẽ liên kết các khối lại với nhau bằng cách di chuyển con trỏ chuột đến khối ExecuteSQL, kéo mũi tên hướng tới khối ConvertAvroToJSON và UpdateAttibute, sau đó kéo tiếp mũi tên đến khối PutS3Object, lúc này một liên kết sẽ được hình thành.

- Thực hiện chạy thu thập dữ liệu

Sau khi đã thực hiện các thao tác cài đặt và cấu hình, tại khối Operate trên giao diện chính của Apache Nifi, ta nhấn chọn nút Start để thực hiện gửi và nhận dữ liệu.

Ta có thể xem trạng thái hoạt động của các khối chức năng bằng cách nhấn chuột phải vào khối trung gian nằm giữa các khối ExecuteSQL,

ConvertAvroToJSON, UpdateAttibute và PutS3Object. Sau đó chọn View status history và List queue để xem thông tin chi tiết.

2.1.5. Thu thập dữ liệu qua API

Ta thực hiện đăng nhập vào hệ thống thông qua Đăng nhập SSO để có thể truy cập được vào giao diện chính của hệ thống và thực hiện các bước cấu hình nguồn dữ liệu.

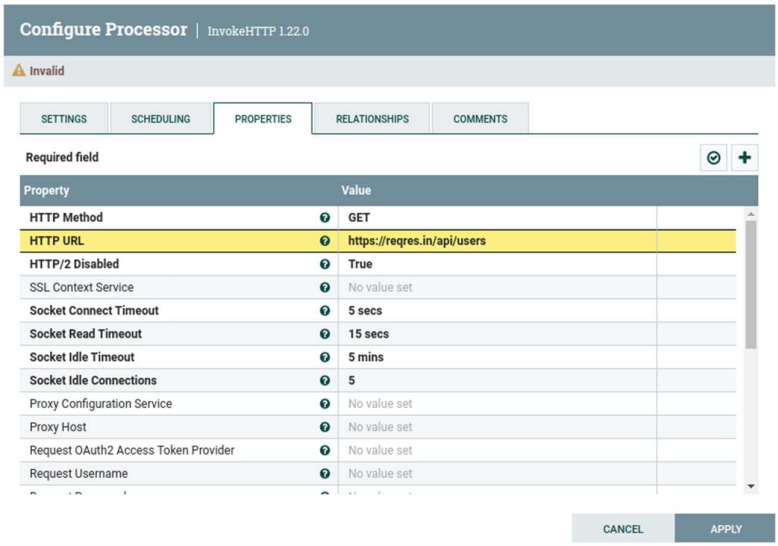
- Khối truyền dữ liệu

Trên thanh công cụ của Apache Nifi, nhấn chọn nút tạo một bộ xử lý mới. Từ cửa sổ hiện ra, nhập tên khối chức năng InvokeHTTP, sau đó ấn ADD. Ta truy cập vào khung thuộc tính của khối này, và điền những thông tin quan trọng sau:

• HTTP Method: nhập phương thức giao tiếp với API.

• HTTP URL: nhập địa chỉ kết nối tới API.

Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.



Hình 2. 13. Các thuộc tính của khối InvokeHTTP

Ngoài ra, để có thể tùy chỉnh thông tin (ví dụ như tên dữ liệu) ta có thể khi báo thêm một khối chức năng có tên UpdateAttribute bằng các thao tác tương tự như trên.

- Khối nhận dữ liệu

Thao tác tương tự như khối Truyền dữ liệu, nhập khối chức năng PutS3Object, sau đó ấn nút ADD. Khi đã thêm thành công khối này trên màn hình làm việc chính

của Apache Nifi, ta nháy đúp chuột để truy cập vào bảng thuộc tính của khối, và điền những thông tin quan trọng dưới đây:

Người dùng sẽ cần điền các thông tin quan trọng sau:

• Object Key: đường dẫn đến nơi lưu trữ dữ liệu trong kho dữ liệu tuân thủ theo cấu trúc thư mục được quy định từ trước: <Mã đơn vị>/API/<Tên bộ dữ liệu><Tên tệp dữ liệu>

• Bucket: tên bucket tại kho lưu trữ.

• Access Key ID: khóa truy cập vào kho dữ liệu.

• Secret Access Key: mật khẩu truy cập kho lưu trữ.

• Endpoint Override URL: địa chỉ truy cập đến kho lưu trữ dữ liệu.

Sau đó nhấn chọn nút APPLY để ghi nhận các cài đặt.

- Kết nối hai khối truyền và nhận dữ liệu

Sau khi đã tạo được hai khối truyền dữ liệu và khối nhận dữ liệu, di chuyển con trỏ chuột đến khối InvokeHTTP, kéo mũi tên hướng tới khối UpdateAttibute, sau đó kéo tiếp mũi tên đến khối PutS3Object để tạo liên kết truyền dữ liệu giữa các khối.

- Thực hiện chạy thu thập dữ liệu

Sau khi đã thực hiện các thao tác cài đặt và cấu hình, tại khối Operate trên giao diện chính của Apache Nifi, ta nhấn chọn nút Start để thực hiện gửi và nhận dữ liệu. Và khi cần xem thêm thông tin chi tiết, ta sẽ chọn View status history và List queue giữa các khối.

2.2. XÂY DỰNG GIẢI PHÁP TÍCH HỢP NỀN TẢNG SPARK VÀO HỆ THỐNG OPENSCIENCE.VN

2.2.1. Xử lý dữ liệu theo lô (Batch processing)

2.2.1.1. Tổng quan về xử lý dữ liệu theo lô

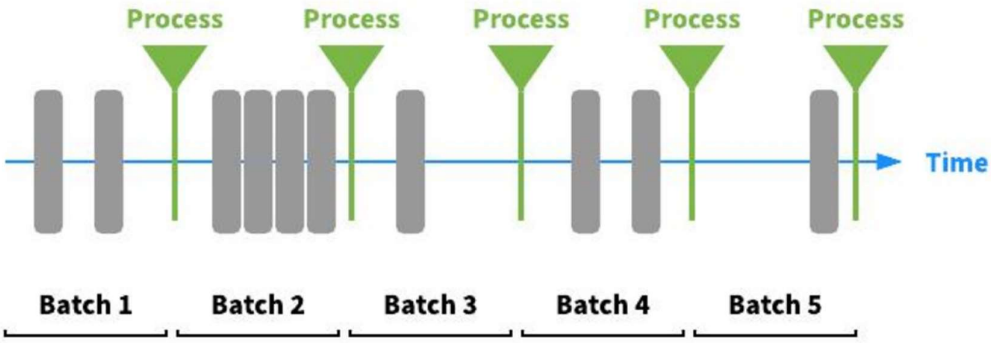
Batch Processing (Xử lý dữ liệu theo lô) được sử dụng để xử lý lượng dữ liệu khổng lồ nhằm thực hiện các tác vụ dữ liệu có khối lượng lớn và lặp lại, mỗi tác vụ thực hiện một thao tác cụ thể mà không cần sự can thiệp của người dùng. Khi có đủ tài nguyên tính toán, quy trình xử lý dữ liệu theo lô cho phép xử lý và quản lý dữ liệu mà không cần hoặc có rất ít tương tác của người dung

Kỹ thuật này đặc biệt hữu ích cho các hoạt động lặp đi lặp lại và đơn điệu để hỗ trợ một số data workflows. Vì quá trình xử lý dữ liệu theo lô tự động hóa quy trình

công việc nên nó giảm thiểu tối đa khả năng xảy ra lỗi hoặc có bất thường khi thực hiện thủ công. Do đạt được mức độ chính xác đáng kể thông qua tự động hóa, các tổ chức có thể đạt được chất lượng dữ liệu vượt trội đồng thời giảm tắc nghẽn trong các hoạt động xử lý dữ liệu [13].

Dữ liệu mới được sinh ra sẽ được gom nhóm thành các lô (batch) và sẽ được xử lý (process) sau đó. Hai cách phổ biến để xác định khi nào các batch này sẽ được xử lý là:

- Dựa trên một khoảng thời gian nhất định. Ví dụ: cứ 60 phút xử lý một lần
- Dựa trên một số điều kiện nhất định. Ví dụ: cứ thu thập đủ 50 files dữ liệu sẽ xử lý một lần, hay cứ thu thập đủ 100G dữ liệu sẽ xử lý một lần,…



Hình 2. 14. Minh họa xử lý dữ liệu theo lô

2.2.1.2. Triển khai xử lý dữ liệu theo lô trên Openscience.vn

Để sử dụng Apache Spark xử lý dữ liệu theo lô, ta cần thực hiện những bước sau:

- Đăng nhập Openscience.vn

Ta thực hiện đăng nhập vào hệ thống thông qua Đăng nhập SSO để có thể truy cập được vào giao diện chính của hệ thống và truy cập vào phần XỬ LÝ DỮ LIỆU, chọn “XỬ LÝ DỮ LIỆU (SPARK WORKER)” để mở jupyter hub kết nối đến spark cluster.

Hình 2. 15. Giao diện để truy cập vào module xử lý dữ liệu

- Chuẩn bị tệp dữ liệu lớn

Chỉ khi có dữ liệu thì Apache Spark mới có thể xử lý, nên ta sẽ cần thiết kế hoặc tìm kiếm tập dữ liệu lớn có sẵn. Khi đã có dữ liệu, ta cần chuẩn bị dữ liệu mẫu để tiến hành xử lý theo lô. Sau đó copy file dữ liệu từ máy lên hệ thống quản lý dữ liệu lớn, có thể là HDFS, S3, hoặc hệ thống file cục bộ.

- Tạo Spark Session

Để có thể làm việc với Spark, ta cần tạo một Spark session để truy vấn dữ liệu từ những hệ thống quản lý file dữ liệu lớn mà ta đã chuẩn bị từ bước trên rồi đọc dữ liệu, thực hiện các phép tính phân tán trên Spark. Một session có thể tạo bằng những dòng lệnh như ở Hình 2.16

```
spark = SparkSession.builder \
        .appName('demo') \
        .getOrCreate()
```

Hình 2. 16. Lệnh tạo Spark Session

- Phân tích dữ liệu đã nạp

Ta có thể dùng hàm read() trong Spark để đọc dữ liệu dataset trong file vừa nạp vào từ bước chuẩn bị và dùng hàm show() để hiển thị dữ liệu. Sau khi thấy bộ dữ liệu đã được nạp và hiển thị thành công, ta có thể tiếp tục thực hiện một số công tác phân tích dữ liệu

```
df.show()
```

```
+-----------------+-------------------+-------------------+-------------------+------------+------------+-------+
|hvfhs_license_num|dispatching_base_num|    pickup_datetime|   dropoff_datetime|PULocationID|DOLocationID|SR_Flag|
+-----------------+-------------------+-------------------+-------------------+------------+------------+-------+
|           HV0003|             B02682|2021-01-01 00:33:44|2021-01-01 00:49:07|         230|         166|   null|
|           HV0003|             B02682|2021-01-01 00:55:19|2021-01-01 01:18:21|         152|         167|   null|
|           HV0003|             B02764|2021-01-01 00:23:56|2021-01-01 00:38:05|         233|         142|   null|
|           HV0003|             B02764|2021-01-01 00:42:51|2021-01-01 00:45:50|         142|         143|   null|
|           HV0003|             B02764|2021-01-01 00:48:14|2021-01-01 01:08:42|         143|          78|   null|
|           HV0005|             B02510|2021-01-01 00:06:59|2021-01-01 00:43:01|          88|          42|   null|
|           HV0005|             B02510|2021-01-01 00:50:00|2021-01-01 01:04:57|          42|         151|   null|
|           HV0003|             B02764|2021-01-01 00:14:30|2021-01-01 00:50:27|          71|         226|   null|
|           HV0003|             B02875|2021-01-01 00:22:54|2021-01-01 00:30:20|         112|         255|   null|
```

Hình 2. 17. Sử dụng hàm để hiển thị một số dữ liệu

- Kiểm tra lược đồ dữ liệu của DataSet và hiển thị trực quan thông qua hàm printSchema()

```
df.printSchema()
```

```
root
 |-- hvfhs_license_num: string (nullable = true)
 |-- dispatching_base_num: string (nullable = true)
 |-- pickup_datetime: string (nullable = true)
 |-- dropoff_datetime: string (nullable = true)
 |-- PULocationID: string (nullable = true)
 |-- DOLocationID: string (nullable = true)
 |-- SR_Flag: string (nullable = true)
```

Hình 2. 18. Kiểm tra lược đồ dữ liệu và hiển thị ra màn hình

- Phân vùng lại dữ liệu thành các partition nhỏ hơn để tối ưu hiệu năng xử lý của Spark. Đối với tập dữ liệu này, chúng ta sẽ xáo trộn dữ liệu và phân vùng lại thành các partition khác nhau bằng hàm **repartition()**
- Ghi dữ liệu đã được phân vùng thành công dưới dạng Apache Parquet (Đây là một định dạng dữ liệu hướng cột giúp tăng tốc truy vấn với các bài toán cần truy vấn nhanh theo cột) bằng hàm **write.mode('overwrite').parquet()**
- Để làm việc với dữ liệu đầu ra (Các phân vùng được định dạng Parquet từ bước trên) chúng ta cần gọi hàm **spark.read.parquet()** của Spark đọc file Parquet từ bucket
- Sau cùng ta sẽ dùng hàm **createOrReplaceTempView()** của Spark để tạo một bảng dữ liệu quan hệ từ đó ta sẽ có thể thao tác với DataFrame để đọc dữ liệu dạng Apache Parquet ở bước trên.

Khi định dạng và phân tích dữ liệu xong, ta sẽ có thể tương tác với dữ liệu đã xử lý xong bằng các câu truy vấn như ở trên cơ sở dữ liệu dạng quan hệ thông thường.

```
# Question 1: taxi trips on Jan 15 = 406887
spark.sql("""
SELECT
   count(*)
FROM
   hvfhw_data
WHERE
   pickup_datetime LIKE '2021-01-01%'
""").show()
```

```
# Question 2: longest trip for each day = 16.43 Hours
spark.sql("""
SELECT
   dispatching_base_num,
   ROUND((UNIX_TIMESTAMP(dropoff_datetime)
- UNIX_TIMESTAMP(pickup_datetime)) /3600.0 , 2)
AS trip_duration_hours
FROM
   hvfhw_data
ORDER BY trip_duration_hours DESC
""").show()
```

Hình 2. 19. Thao tác với dữ liệu đã xử lý thông qua query

2.2.2. Xử lý dữ liệu theo luồng (Streaming processing)

2.2.2.1. Tổng quan về xử lý dữ liệu theo luồng

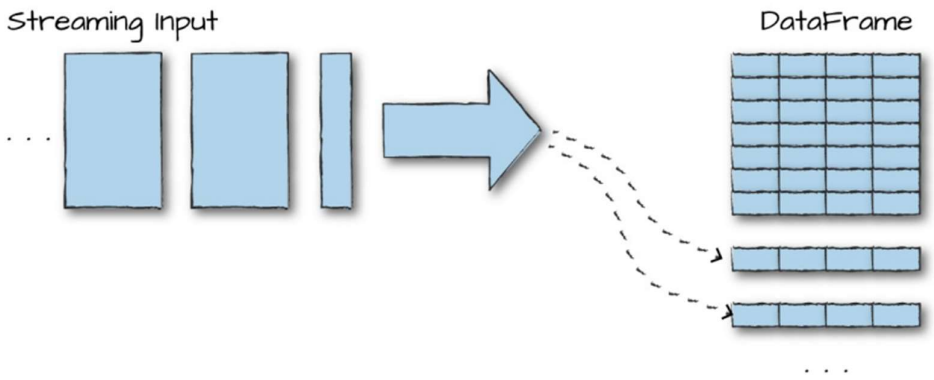
Structured Streaming là phương pháp lựa chọn cho nhu cầu xử lý dữ liệu luồng hiện đại bởi vì:

- Đây là mô hình xử lý dữ liệu luồng thực sự với chế độ xử lý liên tục, không giống như mô hình micro-batch của Spark Streaming.
- API phong phú hơn và bộ tính năng xử lý luồng dữ liệu mạnh mẽ hơn bằng cách tận dụng Spark SQL.
- Khả năng chịu lỗi và đảm bảo tính nhất quán (consistency guarantees) mạnh mẽ hơn. Việc mất dữ liệu được ngăn chặn thông qua việc kiểm tra và phục hồi.
- Hỗ trợ xử lý dựa trên thời gian của sự kiện và suy luận về dữ liệu theo thời gian.
- API dễ sử dụng hơn so với API DStreams.

Structured Streaming là một stream processing framework built trên Spark SQL engine. Thay vì tạo một API riêng biệt, Structured Streaming sử dụng các API có cấu trúc hiện có trong Spark (DataFrames, Datasets và SQL), có nghĩa là tất cả các operation bạn quen thuộc đều được hỗ trợ. Người dùng định nghĩa các hàm tính toán dữ liệu luồng giống cách họ viết hàm tính toán hàng loạt trên dữ liệu tĩnh. Sau khi xác định điều này, và xác định một điểm đến dữ liệu luồng, engine Structured Streaming sẽ chạy truy vấn của bạn một cách tăng dần và liên tục khi dữ liệu mới vào hệ thống.

Ngoài core structured processing engine, structured streaming bao gồm một số tính năng cụ thể cho việc xử lý dữ liệu luồng. Ví dụ, Structured Streaming đảm bảo xử lý end-to-end, exactly-once processin cũng như khả năng chịu lỗi thông qua checkpointing và write-ahead logs.

Ý tưởng chính đằng sau Structured Streaming là xem xét một luồng dữ liệu như một bảng mà dữ liệu được liên tục thêm vào. Spark job sau đó sẽ định kỳ kiểm tra dữ liệu đầu vào mới, xử lý nó, cập nhật một số internal state nằm trong state store nếu cần thiết, và cập nhật kết quả của mình. Một nền tảng của API là bạn không cần phải thay đổi query code của mình khi thực hiện xử lý batch hoặc streaming - bạn chỉ cần xác định liệu chạy truy vấn đó theo cách batch hay stream. Nội bộ, Structured Streaming sẽ tự động tìm ra cách "tăng dần" truy vấn của bạn, tức là cập nhật kết quả của nó một cách hiệu quả mỗi khi dữ liệu mới đến, và sẽ chạy nó một cách chịu lỗi.



Hình 2. 20. Minh họa quá trình xử lý dữ liệu theo luồng

2.2.2.2. Triển khai xử lý dữ liệu theo luồng trên Openscience.vn

Để sử dụng Apache Spark xử lý dữ liệu theo luồng, ta cần thực hiện những bước sau:

- Đăng nhập vào hệ thống

Ta thực hiện đăng nhập vào hệ thống thông qua Đăng nhập SSO để có thể truy cập được vào giao diện chính của hệ thống và truy cập vào phần XỬ LÝ DỮ LIỆU, chọn "XỬ LÝ DỮ LIỆU (SPARK WORKER)" để mở hub kết nối đến spark cluster.

- Chuẩn bị dữ liệu

Chuẩn bị dữ liệu theo luồng dạng json. Sau đó copy file dữ liệu từ máy lên hệ thống quản lý dữ liệu lớn, có thể là HDFS, S3, hoặc hệ thống file cục bộ bằng câu lệnh "!hdfs dfs -put data /path/"

- Tạo Spark Session

Sau khi đã chuẩn bị dữ liệu xong, cần tạo một Spark session để truy vấn dữ liệu theo dạng luồng từ những hệ thống quản lý. Một session có thể tạo bằng những dòng lệnh như ở Hình 2.21

```
from pyspark.sql import SparkSession
import pyspark
spark = SparkSession \
    .builder \
    .config("spark.streaming.stopGracefullyOnShutdown", True) \
    .getOrCreate()
```

Hình 2. 21. Tạo SparkSession cho luồng dữ liệu

- Xử lý dữ liệu

Tạo DataFrames dữ liệu Streaming DataFrameStreamReader. Ta cần đặt spark.sql.streaming.schemaInference thành True để cho phép truyền phát Schema như Hình 2.22 và kiểm tra Schema của dữ liệu Streaming bằng hàm **printSchema()**

```
# To allow automatic schemaInference while reading
spark.conf.set("spark.sql.streaming.schemaInference", True)

# Create the streaming_df to read from input directory
streaming_df = spark.readStream\
    .format("json") \
    .option("maxFilesPerTrigger", 1) \
    .load("hdfs:////user/tranductho1309/data/input/")
```

Hình 2. 22. Đặt schemaInference sang True để truyền phát dữ liệu luồng

Ta tiếp tục sử dụng select SparkSQL để chọn và đọc hết các trường dữ liệu rồi lấy những dữ liệu đó ra xử lý dữ liệu Streaming và làm phẳng dữ liệu JSON để nhiều thiết bị có thể đọc được. Dữ liệu trước và sau khi làm phẳng sẽ được minh họa thông qua hàm **printSchema()** như ở Hình 2.25

```
# Lets explode the data as devices contains list/array of device reading
from pyspark.sql.functions import explode, col

exploded_df = streaming_df \
    .select("customerId", "eventId", "eventOffset", "eventPublisher", "eventTime", "data") \
    .withColumn("devices", explode("data.devices")) \
    .drop("data")
```

Hình 2. 23. Sử dụng select SparkSQL để đọc hết các trường dữ liệu trong JSON

```
# Flatten the exploded df
flattened_df = exploded_df \
    .selectExpr("customerId", "eventId", "eventOffset", "eventPublisher", "eventTime",
                "devices.deviceId as deviceId", "devices.measure as measure",
                "devices.status as status", "devices.temperature as temperature")
```

Hình 2. 24. Làm phẳng dữ liệu trong JSON

Dữ liệu trước khi làm phẳng

```
# Check the schema of the exploded_df, place a sample json file and change readStream to read
exploded_df.printSchema()
```

```
root
 |-- customerId: string (nullable = true)
 |-- eventId: string (nullable = true)
 |-- eventOffset: long (nullable = true)
 |-- eventPublisher: string (nullable = true)
 |-- eventTime: string (nullable = true)
 |-- devices: struct (nullable = true)
 |    |-- deviceId: string (nullable = true)
 |    |-- measure: string (nullable = true)
 |    |-- status: string (nullable = true)
 |    |-- temperature: long (nullable = true)
```

Dữ liệu sau khi làm phẳng

```
flattened_df.printSchema()
```

```
root
 |-- customerId: string (nullable = true)
 |-- eventId: string (nullable = true)
 |-- eventOffset: long (nullable = true)
 |-- eventPublisher: string (nullable = true)
 |-- eventTime: string (nullable = true)
 |-- deviceId: string (nullable = true)
 |-- measure: string (nullable = true)
 |-- status: string (nullable = true)
 |-- temperature: long (nullable = true)
```

Hình 2. 25. So sánh dữ liệu trước và sau khi làm phẳng

Khi dữ liệu đã được làm phẳng, ta sẽ tiến hành ghi những dữ liệu luồng này vào kho dữ liệu bằng hàm **writeStream()**

2.3. XÂY DỰNG GIẢI PHÁP TÍCH HỢP NỀN TẢNG KUBEFLOW VÀO HỆ THỐNG OPENSCIENCE.VN

2.3.1. Tổng quan về xây dựng một pipeline

Việc xây dựng một pipeline deep learning (DL), machine learning(ML) trên nền tảng Kubeflow có một số mục đích quan trọng như sau:

- Tích hợp và tự động hóa quy trình: Kubeflow cung cấp một nền tảng để tổ chức và tự động hóa các quy trình ML hay DL phức tạp. Việc sử dụng pipeline ML hoặc DP trên Kubeflow giúp tạo ra một luồng làm việc rõ ràng, dễ quản lý và dễ mở rộng
- Mở rộng linh hoạt: Kubeflow cho phép triển khai pipeline trên một mạng lưới phân tán các nguồn tài nguyên tính toán, giúp tận dụng tối đa sức mạng tính toán và mở rộng linh hoạt khi cần
- Quản lý tài nguyên hiệu quả: Kubeflow tích hợp các công cụ phổ biến trong cộng đồng ML và DL như TensorFlow, PyTorch hay Sklearn, giúp dễ dàng xây dựng các pipeline sử dụng các công nghệ này
- Giảm thời gian triển khai: Bằng cách sử dụng Kubeflow, các nhóm ML hay DL có thể giảm thời gian từ khâu phát triển tới khâu triển khai sản phẩm, đồng thời tăng tính nhất quán và độ tin cậy của các quy trình ML/DL
- Tăng tính di động và tái sử dụng: Việc xây dựng pipeline ML,DL trên Kubeflow giúp tạo ra các mô hình và quy trình có thể tái sử dụng, giúp tăng hiệu suất pháp triển ML

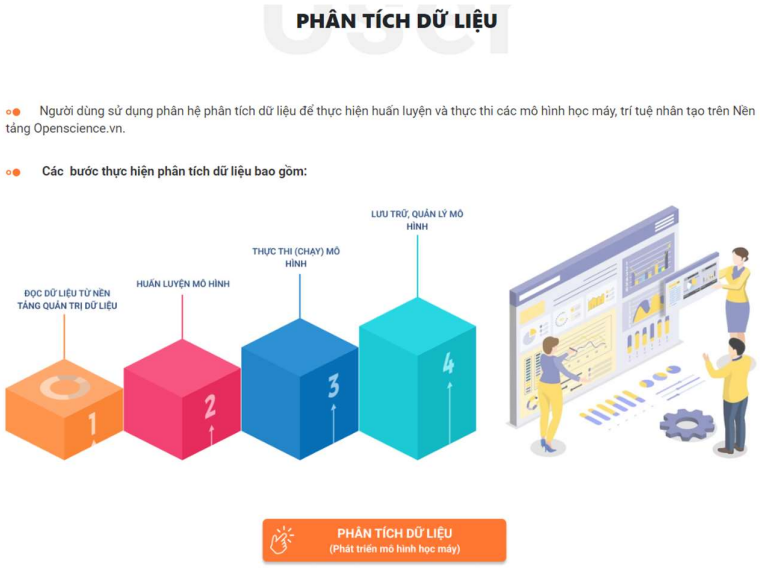
2.3.2. Triển khai xây dựng pipeline ML/DL trên Openscience.vn

Sau khi người dùng đăng ký tạo tài khoản và được cấp quyền cho phép vào sử dụng trong kubeflow. Trong phần 1), người quản trị (admin) cần cấp quyền cho một namespace có thể được có quyền để trỏ vào một kho lưu trữ để lấy dữ liệu, hoặc lưu mô hình đã huấn luyện vào để thực hiện các giai đoạn sau đó. Phần 2) hướng dẫn cho một file jupyter notebook có quyền kubeflow pipeline. Phần 3) hướng dẫn cách tạo một pipeline cơ bản. Phần 4) thiết lập quyền cho phép Kubeflow pipeline truy cập vào jupyter notebook. Phần 5) hướng dẫn cách tạo một pipeline cơ bản.

2.3.2.1. Đăng nhập hệ thống Openscience.vn

Để có thể truy cập vào mục phân tích dữ liệu kubeflow, ta cần thực hiện những bước sau đây:

- Truy cập vào web openscience.vn và chọn "ĐĂNG NHẬP"
- Chọn "Đăng nhập SSO" rồi nhập tài khoản/mật khẩu và click "Sign In"
- Sau khi đăng nhập, hệ thống chuyển đến màn hình giao diện trang chủ hệ thống. Chọn phần "Ứng dụng nền tảng"
- Ở trang Ứng dụng nền tảng, kèo xuống phần PHÂN TÍCH DỮ LIỆU và chọn "PHÂN TÍCH DỮ LIỆU(Phát triển mô hình học máy)" để mở jupyter hub kết nối đến trang làm việc của Kubeflow.

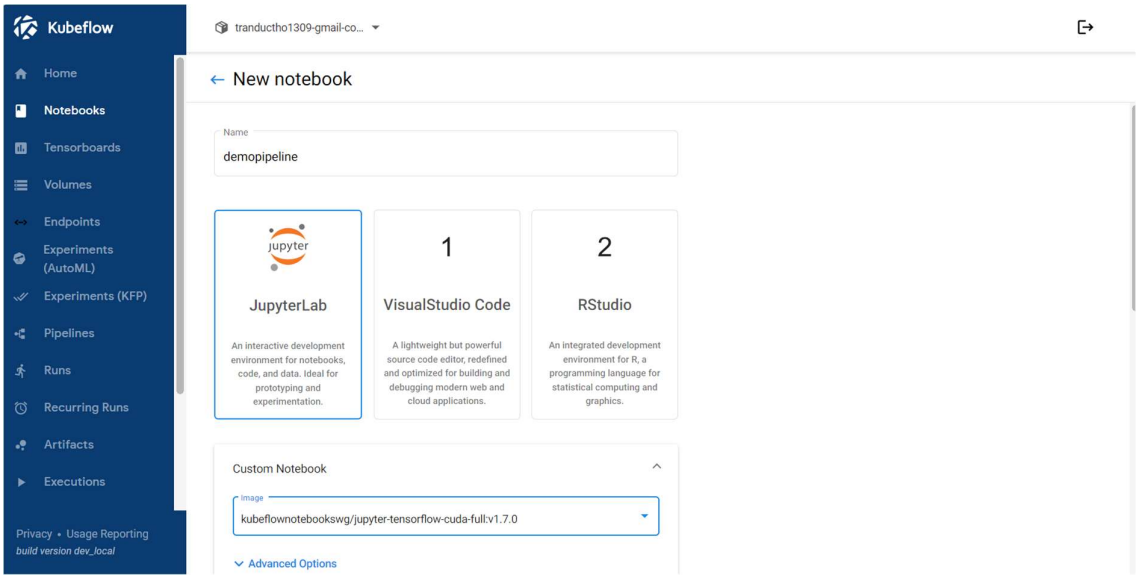


Hình 2. 26. Truy cập vào Kubeflow trên giao diện chính

2.3.2.2. Tạo một notebooks trên hệ thống Openscience

Sau khi thực hiện các bước trong mục 2.3.2.1 sẽ vào được giao diện của Kubeflow trên hệ thống. Ta sẽ cần phải khởi tạo một notebook để khám phá dữ liệu, phát triển và huấn luyện mô hình, thử nghiệm và tinh chỉnh các thuật toán, ghi chép và chia sẻ kết quả, cũng như tích hợp vào Kubeflow Pipelines để tự động hóa quy trình. Trong giao diện ấn vào phần Notebooks và tiếp tục chọn vào mục "New Notebook"

Trong giao diện khởi tạo Notebook điền tên New notebook cần khởi tạo, mục Custom Notebook chọn image "kubeflownotebookswg/jupyter-tensorflow-cuda-full:v1.7.0", tiếp đó chọn 1 CPU, 2 Gi Memory và 1 GPU NIVIDIA, mục Advanced Options trong mục Configurations lựa chọn Allow access to KFP, sau đó ấn LAUNCH để tạo một notebook mới.



Hình 2. 27. Khởi tạo một notebook mới

2.3.2.3. Thiết lập quyền truy cập để Kserve có thể truy cập vào Minio

Trước khi bước vào bước xây dựng vào pipeline bằng cách sử dụng huấn luyện, nếu dữ liệu được lưu trữ vào Minio, thì cần phải cài đặt một file cấu hình để Kserve có thể trỏ vào các mô hình được lưu trữ trên Minio và thực hiện việc suy luận trên các mô hình huấn luyện được. File cấu hình cần thay đổi namespace trùng với namespace cần khởi tạo và có thể viết như Hình 2.28 dưới đây

```
apiVersion: v1
kind: Secret
metadata:
  name: minio-kserve-secret
  namespace: tranductho1309-gmail-com
  annotations:
    serving.kserve.io/s3-endpoint: "minio-service.kubeflow:9000"
    serving.kserve.io/s3-usehttps: "0"
    serving.kserve.io/s3-useanoncredential: "false"
type: Opaque
stringData:
  AWS_ACCESS_KEY_ID: "minio"
  AWS_SECRET_ACCESS_KEY: "minio123"
---
apiVersion: v1
```

Hình 2. 28. File cấu hình Kserve trỏ vào các mô hình trên Minio

Sau đó dùng lệnh "**kubectl apply -f set-minio-kerve-secret.yaml**" để có thể apply file yaml vừa khởi tạo dưới quyền admin

2.3.2.4. Thiết lập quyền cho phép Kubeflow pipeline truy cập vào jupyter notebook

Để có thể thực hiện tùy chọn Access to KFP được tích trong bước khởi tạo jupyter notebook. Chúng ta cần cấp quyền cho phép KFP có thể truy cập. File cài đặt cấu hình sẽ như Hình 2.29 dưới đây và chạy lệnh "**kubectl apply -f access_kfp.yaml**" để có thể apply file yaml vào hệ thống

```
apiVersion: kubeflow.org/v1alpha1
kind: PodDefault
metadata:
  name: access-kf-pipeline
  namespace: tranductho1309-gmail-com
spec:
  desc: Allow access to KFP
  selector:
    matchLabels:
      access-kf-pipeline: "true"
  volumeMounts:
    - mountPath: /var/run/secrets/kubeflow/pipelines
      name: volume-kf-pipeline-token
      readOnly: true
  volumes:
    - name: volume-kf-pipeline-token
      projected:
        sources:
          - serviceAccountToken:
              path: token
              expirationSeconds: 7200
              audience: pipelines.kubeflow.org
  env:
    - name: KF_PIPELINES_SA_TOKEN_PATH
      value: /var/run/secrets/kubeflow/pipelines/token
```

Hình 2. 29. File thiết lập cấp quyền KFP

2.3.2.5. Cách tạo một pipeline cơ bản

Để có thể lấy dữ liệu trong Minio, lưu dữ liệu vào Minio trong Minio chúng ta có thể sử dụng những câu lệnh dưới Hình 2.30 như sau.

```
from minio import Minio
import numpy as np
import os

minio_client = Minio(
    "minio-service.kubeflow:9000",
    access_key="minio",
    secret_key="minio123",
    secure=False
)
minio_bucket = "mlpipeline"
```

Hình 2. 30. Truy cập dữ liệu trong cơ sở dữ liệu Minio

Trong những câu lệnh trên, ta có biến minio_client dùng để khởi tạo một client minio. Trên hệ thống openscience chúng ta có thể truy cập vào minio với access key và secret key mặc định như trên. Ví dụ như trong phần demo ở hình 2.30 ta đã tạo sẵn minio bucket là "mlpipeline". Để đọc dữ liệu chúng ta có thể dùng "**minio_client.fget_object(minio_bucket,"heart-disease/data/heart.csv","/tmp/heart.csv")**", trong đó minio_bucket là tên bucket trên minio, "heart-disease/data/heart.csv" là tên đường dẫn dữ liệu trên bucket "/tmp/heart.csv", là tên đường dẫn đích.

Để có thể tạo một pipeline cơ bản chúng ta có thể lấy ví dụ chúng ta có 2 hàm a() và b() dưới Hình 2.31.

```
def a():
    print('Đây là hàm a')
def b():
    print('Đây là hàm b')
```

Hình 2. 31. Tạo hàm a() và b()

Tiếp tục tiến hành tạo thành một pipeline đầu tiên ta cần import thư viện kfp đã có sẵn trên hệ thống, với mỗi hàm trên chúng ta có thể tạo một component cho nó, sau đó chúng ta có thể tạo nó thành một pipeline hoàn chỉnh như Hình 2.32 dưới đây và khi tạo xong ở dưới notebook sẽ xuất hiện dòng chữ màu xanh "Pipeline details" để chúng ta có thể run pipeline vừa tạo.

Hình 2. 32. Ví dụ một pipeline hoàn chỉnh

2.4. XÂY DỰNG GIẢI PHÁP ĐĂNG NHẬP MỘT LẦN (SSO) CHO OPENSCIENCE.VN ĐỂ TRUY CẬP VÀO CÁC NỀN TẢNG

Để xây dựng cơ chế đăng nhập một lần (Single Sign On – SSO) cho phép người sử dụng truy cập vào các nền tảng bên trong hệ thống từ giao diện hệ thống Openscience.vn, chúng ta sử dụng hệ thống Keycloak để đồng bộ hoá các tài khoản trong Openscience.vn với các tài khoản của các nền tảng bên trong, trên cơ sở đó xây dựng cơ chế SSO. Keycloak là một hệ thống quản lý danh tính và truy cập mã nguồn mở, được phát triển bởi Red Hat. Nó cung cấp một giải pháp an toàn và hiệu quả để quản lý xác thực và ủy quyền cho các ứng dụng web và di động. Keycloak hỗ trợ các chuẩn xác thực phổ biến như OpenID Connect, OAuth 2.0 và SAML 2.0, cho phép tích hợp dễ dàng với nhiều hệ thống khác nhau [14]. Ngoài ra, Keycloak còn cung cấp tính năng quản lý người dùng, quản lý phiên, và quản lý quyền truy cập, giúp các tổ chức dễ dàng kiểm soát quyền truy cập vào các tài nguyên và dịch vụ của mình. Với giao diện người dùng trực quan và khả năng mở rộng cao, Keycloak là giải pháp lý tưởng cho việc quản lý danh tính và truy cập trong thời đại số.

2.4.1. Thực hiện SSO truy cập vào Nifi qua Keycloak

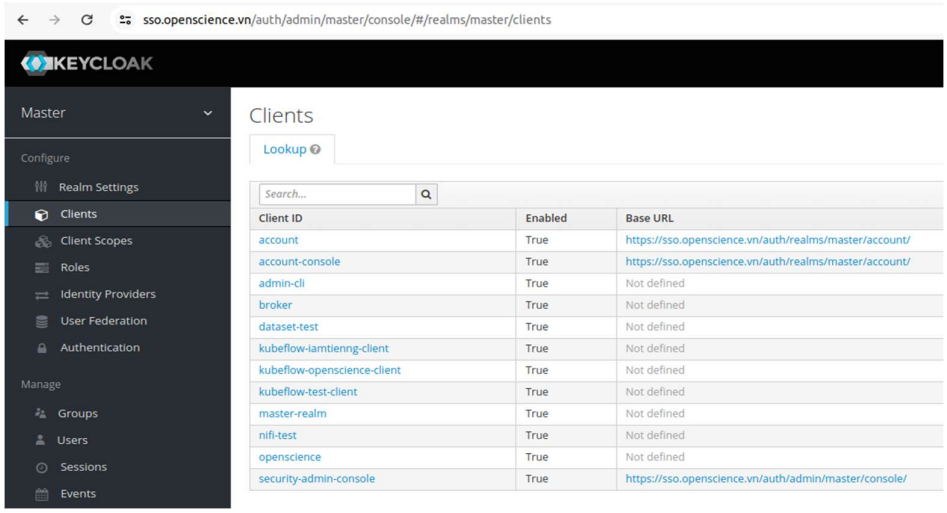
Trong hệ thống quản lý và chia sẻ dữ liệu, Apache Nifi đóng vai trò thu thập dữ liệu từ nhiều nguồn khác nhau như IOT, Saas, Database, v.v Nền tảng Apache Nifi cũng được xây dựng với cơ chế đăng nhập riêng bằng username và password.

Với mong muốn tạo ra một môi trường sử dụng thuật tiện nhất cho người dùng, hệ thống quản lý và chia sẻ dữ liệu đã tích hợp phần đăng nhập của Apache Nifi vào

phần đăng nhập chung của Hệ thống. Điều này có nghĩa là, thay vì người dùng cần đăng nhập hai lần (một lần đăng nhập vào Hệ thống, một lần đăng nhập vào Apache Nifi) thì giờ đây người dùng chỉ cần đăng nhập một lần duy nhất vào Hệ thống mà vẫn có thể truy cập được vào Apache Nifi để thu thập dữ liệu.

Và để thực hiện việc đồng bộ quá trình đăng nhập giữa Hệ thống và Apache Nifi, chúng ta sẽ cần thực hiện những công đoạn sau:

- Ta đăng nhập vào hệ thống Openscience thông qua Đăng nhập SSO rồi truy cập vào trang quản lý Keycloak thông qua đường dẫn "**https://sso.openscience.vn/auth/**"
- Thực hiện tạo một Clients có tên là **nifi-test**
- Tại phần cài đặt của Clients Nifi-test, ta thiết lập các thông tin cần thiết như:
    - Client Protocol: openid-connect
    - Access Type: public
    - Root URL: https://103.22.218.60:9443/nifi/
    - Valid Redirect URLs: *
    - Admin URL: https://103.22.218.60:9443/nifi/
    - Web Origins: https://103.22.218.60:9443
    - Backchannel Logout URL: https://103.22.218.60:9443/nifi/logout-complete



Hình 2. 33. Tạo một client mới trong Keycloak cho Apache Nifi

- Tiến hành cập nhật một số thông tin cấu hình quan trọng cần thiết để Nifi có thể kết nối đến Keycloak như sau:

- nifi.security.user.oidc.discovery.url: https://sso.openscience.vn/auth/realms/master/.well-known/openid-configuration
- nifi.security.user.oidc.client.id: nifi-test
- nifi.security.user.oidc.client.secret

2.4.2. Thực hiện SSO truy cập vào Kubeflow qua Keycloak

Không giống với Apache Nifi, để tạo tài khoản cho một người dùng mới trong Kubeflow cần rất nhiều công đoạn thủ công yêu cầu một hoặc nhiều người vận hành, đây là kịch bản tệ nhất cho một hệ thống áp dụng nhiều công nghệ như trong dự án này.

Kubeflow áp dụng Multi-Tenancy, đây là một hệ thống tự phục vụ, một người dùng mới khi tạo tài khoản nên được tạo một Profile mới một cách tự động. Để thực hiện được điều này, hệ thống yêu cầu một hệ thống xác thực để thực hiện xác định danh tính cho mỗi người dùng.

Mặc định, Kubeflow sẽ được cài đặt và tận dụng Dex, một OpenID Connect Provider. Nhưng có hai lý do để ta không dùng Dex nữa, đó là:

- Sự phức tạp và yêu cầu nhiều công đoạn thủ công của Dex.
- Openscience.vn sử dụng Keycloak.

Keycloak là một dịch vụ quản lý định danh và truy cập. Keycloak hỗ trợ tất cả những gì mà Dex có thể cung cấp, trong đó có cả OIDC Provider.

Khi người dùng tạo tài khoản trên Openscience.vn, họ sẽ tự động được tạo một tài khoản trên hệ thống của Keycloak, từ đó tạo ra Single Sign On hay SSO để truy cập vào tất cả mọi dịch vụ như Kubeflow với chỉ một lần đăng nhập. Và để thực hiện được yêu cầu này thì chúng ta sẽ phải giải quyết được hai vấn đề chính.

- Vấn đề đầu tiên đặt ra là thay thế Dex bằng Keycloak trong quá trình cài đặt Kubeflow để dịch vụ SSO trong hệ thống Openscience.vn có thể sử dụng cho Kubeflow.
- Vấn đề thứ hai là chúng ta cần một dịch vụ để tự động tạo Profile cho người dùng.

Vì Kubeflow được cài đặt trên hệ thống Kubernetes On-Premise sử dụng manifest với sự hỗ trợ của kustomize và kubectl, nên ta có thể thay đổi cài đặt OIDC ở trong folder "**manifests/common/oidc-authservice/base/**". Thiết lập Kubeflow

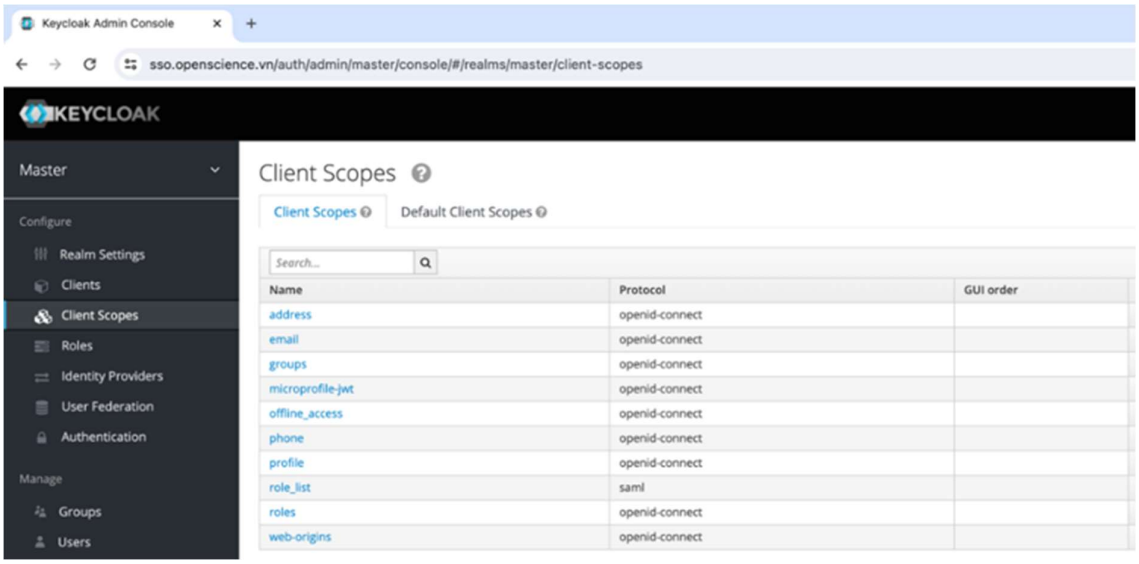
xác thực với Dex được định nghĩa trong thư mục này, nên chúng ta có thể thay thế Dex bằng Keycloak.

Tiếp đó chúng ta sẽ xây dựng một dịch vụ API với Framework Flask để tự động tạo Profile cho người dùng mới với một HTTP POST Request. Tất nhiên chúng ta sẽ đóng gói nó dưới dạng container bằng công cụ Docker và vận hành nó trên cụm Kubernetes dùng để vận hành Kubeflow.

2.4.2.1. Tạo Client trong Keycloak

Trước khi thực hiện các bước với Keycloak, ta cần chắc chắn rằng ta đã có quyền truy cập đến Admin Console của Keycloak.

Vì mặc định, Keycloak không có Client Scope tên là "groups" nên chúng ta cần tạo để có thể hoạt động với Kubeflow theo đúng kỳ vọng. Nên ta sẽ truy cập vào Client Scopes và nhấn vào nút Create, điền tên scope sau đó nhấn Save.



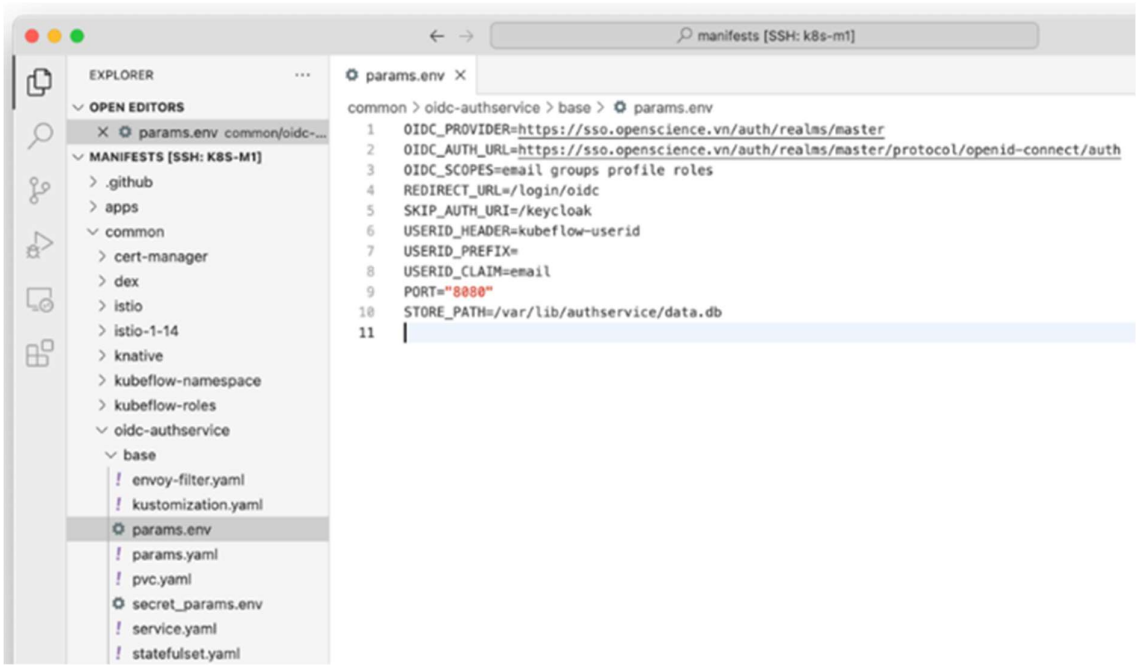
Hình 2. 34. Tạo một Client Scopes mới trong Keycloak

Để Kubeflow có thể sử dụng Keycloak làm OIDC Provider, ta cần tạo một Client trong Keycloak dành riêng cho Kubeflow. Ta tiếp tục truy cập vào phần Client trong Keycloak và nhấn vào nút Create. Trong phần thiết lập client mới, ta cần lưu ý thiết lập một số trường thông tin sau đây:

- Client ID: kubeflow-openscience-client
- Client Protocol: openid-connect
- Access Type: confidential
- Root URL: https://kubeflow.openscience.vn/login/oidc
- Valid Redirect URLs: *

2.4.2.2. Áp dụng cài đặt OIDC mới cho Kubeflow

Đầu tiên, ta thay đổi nội dung trong tệp "params.env" nằm trong thư mục "**manifests/ common/oidc-authservice/base**" với URL được phơi ra internet của Keycloak như hình 2.35 bên dưới.



Hình 2. 35. Cấu hình file params.env

Sau đó, ta tiếp tục thực hiện thay đổi nội dung trong tệp "**secret_params.env**" nằm tại "**manifests/common/oidc-authservice/base/secret_params.env**". Ở bước này, ta sẽ cần hai thông tin: Tên của Keycloak Client và giá trị của Secret trong phần Credentials từ Client ta đã tạo ở bước trên.

Khi đã thực hiện xong nội dung cần sửa đổi của hai file cấu hình trên, ta sẽ cần khởi tạo lại dịch vụ OIDC của Kubeflow bằng lần lượt hai dòng lệnh như hình 2.36 để có thể áp dụng được cấu hình mới lên hệ thống.

```
# cd manifest/common/oidc-authservice/base/
kustomize build | kubectl --kubeconfig=./config delete -f -
kustomize build | kubectl --kubeconfig=./config apply -f -
```

Hình 2. 36. Các dòng lệnh khởi tạo lại dịch vụ OIDC của Kubeflow

Với những thay đổi mới được thực hiện, lúc này, khi truy cập vào "**https://kubeflow.openscience.vn/**" ta sẽ được chuyển hướng tới trang đăng nhập của Keycloak. Tại đây, nếu người dùng đã đăng nhập trên trang Openscience.vn với tính năng SSO, họ sẽ được chuyển hướng thẳng đến Kubeflow Nếu người dùng chưa thực hiện đăng nhập trên trang Openscience với tính năng SSO, họ có thể thực hiện

đăng nhập với tài khoản và mật khẩu của họ và có thể truy cập đến các dịch vụ khác có trên Openscience.

2.4.2.3. Thiết kế Flask API Server

Sau khi áp dụng thành công thiết lập OIDC trong Kubeflow để sử dụng Keycloak thay cho Dex để làm dịch vụ xác thực người dùng, người dùng mới khi truy cập đến Kubeflow vẫn chưa thể dùng các tính năng mà Kubeflow cung cấp.

Điều này xảy ra vì người dùng vẫn chưa được khởi tạo Profile cho riêng họ. Ta sẽ cần tạo thủ công Profile cho từng người dùng cụ thể với tệp profile.yaml và áp dụng giới hạn tài nguyên với tệp quota.yaml như ở hình 2.37

profile.yaml

```yaml
apiVersion: kubeflow.org/v1beta1
kind: Profile
metadata:
  name: user-namespace
spec:
  owner:
    kind: User
    name: user@example.com

  resourceQuataSpec:
    hard:
      cpu: "1"
      memory: 2Gi
      requests.nvidia.com/gpu: "0"
      persistentvolumeclaims: "2"
      requests.storage: "15Gi"
```

quota.yaml

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
  namespace: covid-namespace
spec:
  limits:
  - default:
      memory: 10Gi
      cpu: 6000m
    defaultRequest:
      cpu: 5000m
      memory: 6Gi
    type: Container
```

Hình 2. 37. Những file cần thiết để tạo profile cho người dùng

Giải pháp được đề ra ở đây là xây dựng một API Server nhận request với đầu vào là email của người dùng mới. Từ đó, máy chủ sẽ tự động sinh ra nội dung của hai tệp profile.yaml và quota.yaml tương ứng với email và tên namespace cho người dùng đó và cũng sẽ tự động áp dụng chúng lên cụm Kubernetes với tệp config mà chúng ta chỉ định.

Để đơn giản hóa quá trình xây dựng server, chúng ta sẽ sử dụng một Framework quen thuộc là Flask để giúp việc xây dựng API Server được nhanh và bảo mật hơn.

Đầu tiên, ta tạo một tệp tên là main.py, sau đó thiết lập những thư viện cần thiết và tải về tệp config của cụm Kubernetes trên FPTCloud.

```python
import subprocess
from flask import Flask, request, jsonify
from jinja2 import Template
import re

app = Flask(__name__)
```

Hình 2. 38. Tệp main.py và những thư viện cần khai báo cho Flask

Tiếp đó, ta cần tạo một hàm chuyển email của người dùng sang dạng chuỗi phù hợp để làm tên cho Namespace trong Kubernetes.

```python
def email_to_k8s_namespace(email):
    clean_string = re.sub(r'[^a-zA-Z0-9]', '-', email.lower())
    clean_string = re.sub(r'[^a-z0-9]+', '-', clean_string)
    clean_string = clean_string[:63]
    clean_string = clean_string.strip('-')
    return clean_string
```

Hình 2. 39. Hàm chuẩn hóa các ký tự của chuỗi email đầu vào

Sau đó, ta tạo hàm **generate_profile_yaml** để tạo nội dung cho tệp profile.yaml từ email của người dùng và thực hiện tương tự với hàm **generate_limit_range_yaml**

Hàm tạo nội dung cho tệp profile.yaml

```python
def generate_profile_yaml(username, user_email):
    profile_template = """
apiVersion: kubeflow.org/v1beta1
kind: Profile
metadata:
  name: {{ username }}
spec:
  owner:
    kind: User
    name: {{ user_email }}

  resourceQuotaSpec:
   hard:
     requests.nvidia.com/gpu: "0"
     persistentvolumeclaims: "2"
     requests.storage: "15Gi"
"""

    template = Template(profile_template)
    rendered_yaml = template.render(
        username=username,
        user_email=user_email,
    )

    return rendered_yaml
```

Hàm tạo nội dung cho tệp quota.yaml

```python
def generate_limit_range_yaml(username):
    limit_range_template = """
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
  namespace: {{ username }}
spec:
  limits:
  - default:
      memory: 10Gi
      cpu: 6000m
    defaultRequest:
      cpu: 5000m
      memory: 6Gi
    type: Container
"""

    template = Template(limit_range_template)
    rendered_yaml = template.render(
        username=username
    )

    return rendered_yaml
```

Hình 2. 40. Hàm tạo nội dung cho tệp profile và quota

Sau đó ta viết hàm **apply_yaml** để áp dụng nội dung của hai tệp mà ta đã render ở trên vào cụm Kubernetes với tệp config tương ứng trong thư mục.

```
def apply_yaml(yaml_content, resource_type):
    try:
        process = subprocess.run(
            ['kubectl', '--kubeconfig=./config','apply', '-f', '-'],
            input=yaml_content,
            encoding='utf-8',  # Specify encoding
            check=True
        )
        print(f"Successfully applied {resource_type} YAML.")
    except subprocess.CalledProcessError as e:
        print(f"Error applying {resource_type} YAML: {e}")
```

Hình 2. 41. Hàm apply_yaml để áp dụng vào tệp config của cụm Kubernetes

Khi đã có đầy đủ hàm cần thiết để lấy nội dung tạo profile, ta sẽ viết hàm API để Server có thể nghe được các yêu cầu tạo Profile từ internet và viết hàm main để chạy app Flask.

```
@app.route('/generate-and-apply-yaml', methods=['POST'])
def generate_and_apply_yaml():
    data = request.json
    user_email = data.get('user_email')
    username = email_to_k8s_namespace(user_email)

    profile_yaml = generate_profile_yaml(
        username, user_email
    )
    limit_range_yaml = generate_limit_range_yaml(
        username
    )
    apply_yaml(profile_yaml, "Profile")
    apply_yaml(limit_range_yaml, "LimitRange")

    response_data = {
        'status': 'success',
        'message': 'YAML files generated and applied successfully.'
    }
    return jsonify(response_data)
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=5000, debug=True)
```

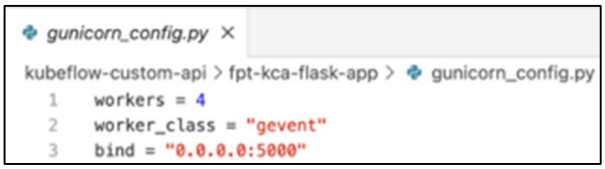
Hình 2. 42. Hàm API để Server nghe được các yêu cầu tạo Profile từ Internet

2.4.2.4. Đóng gói API Server với Docker

Đầu tiên, ta tạo một tệp với tên **requirements.txt** liệt kê tất cả các thư viện, ứng dụng Python mà api server của ta cần.

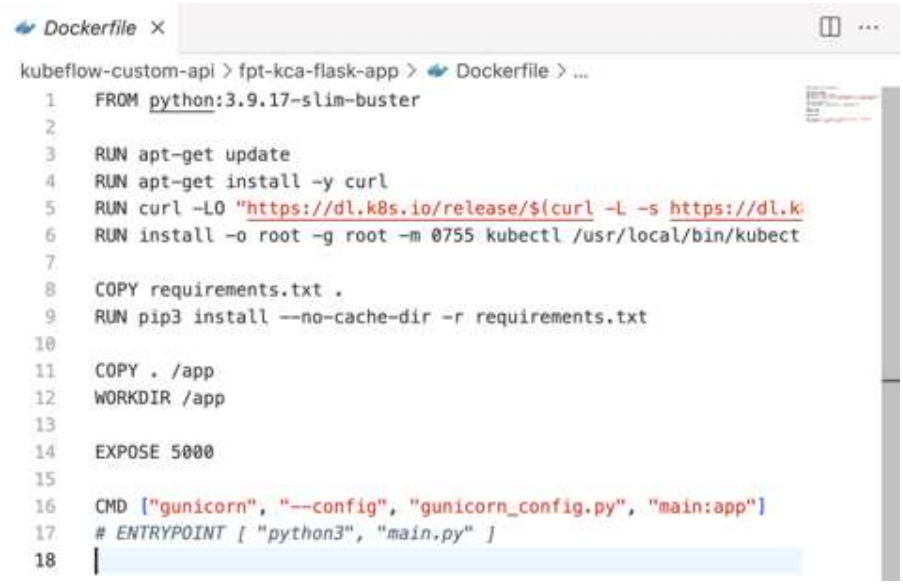
Hình 2. 43. Nội dung file requirements

Tiếp đến ta tạo tệp **gunicorn_config.py** để cấu hình các thiết lập các worker xử lý đồng thời và tăng khả năng xử lý đồng thời thông qua gevant. Worker này rất hiệu quả cho các ứng dụng có nhiều tác vụ I/O, như truy vấn cơ sở dữ liệu hoặc giao tiếp qua mạng, nhờ thông qua non-blocking I/O (nhập/xuất không chặn) và không yêu cầu xử lý CPU nặng.



Hình 2. 44. Nội dung file cấu hình gunicorn

Từ hai file đã tạo ra bên trên, ta tiến hành tạo một Dockerfile để xây dựng một image docker (thông qua lệnh build) và đóng gói lại (thông qua lệnh push) để sử dụng trên Kubernetes ở phần sau.

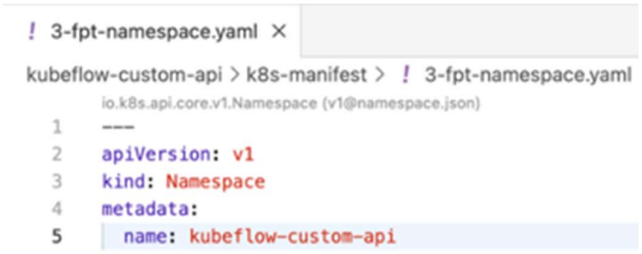


Hình 2. 45. File cấu hình docker để tạo image

2.4.2.5. Triển khai API Server trên Kubernetes

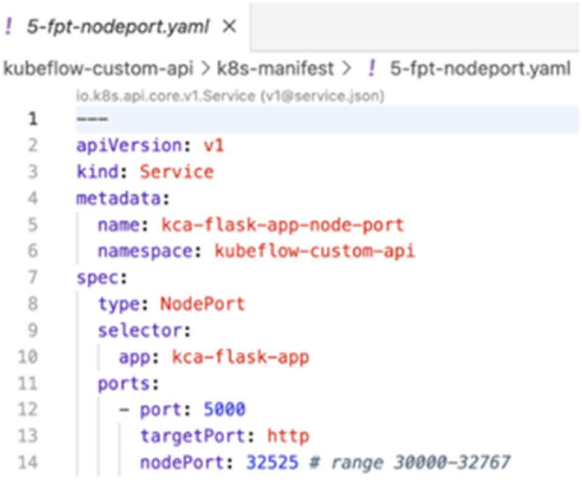
Để triển khai API Server ta đã đóng gói ở trên cụm Kubernetes ta cần tạo 3 tệp (với nội dung như ở Hình 2.45, Hình 2.46 và Hình 2.47) với mục đích lần lượt như sau:

- namespace.yaml: Dùng để tạo một không gian làm việc (namespace) riêng biệt cho API server chạy trên Kubernetes.
- deployment.yaml: Dùng để định nghĩa và quản lý việc triển khai các container (chạy từ API Server có dạng Docker image mà ta đã đóng gói ở bước trên).
- nodeport.yaml: Dùng để tạo một Service loại NodePort, cho phép API Server có thể được truy cập từ bên ngoài cluster thông qua một cổng cụ thể trên các node của Kubernetes cluster.



Hình 2. 46. Nội dung file namespace

Hình 2. 47. Nội dung file deployment



Hình 2. 48. Nội dung file nodeport

Khi đã tạo xong 3 tệp namespace, deployment và nodeport, thì ta có thể triển khai API server trên cụm Kubernetes với các dòng lệnh sau đây. Từ đó API Server sẽ có thể nghe được các yêu cầu từ máy chủ thực hiện chức năng xác thực từ Openscience.vn.

```
kubectl --kubeconfig=./config apply 3-fpt-namespace.yaml
kubectl --kubeconfig=./config apply 4-fpt-deployment.yaml
kubectl --kubeconfig=./config apply 5-fpt-nodeport.yaml
```

Hình 2. 49. Các lệnh để triển khai API server lên Kubernetes

CHƯƠNG 3. THỬ NGHIỆM VÀ ĐÁNH GIÁ CÁC GIẢI PHÁP

3.1. THỬ NGHIỆM, ĐÁNH GIÁ GIẢI PHÁP TÍCH HỢP NIFI

3.1.1. Thử nghiệm gửi và nhận dữ liệu trên Nifi

3.1.1.1. Thu thập dữ liệu từ tệp

Để có thể thử nghiệm được giải pháp thu thập dữ liệu Nifi, ta sẽ cần chuẩn bị sẵn một thư mục dữ liệu trên máy tính có chứa nhiều định dạng dữ liệu khác nhau như txt, json, word, xlsx có tên là "**data**"

```
root@vm-230914225540-h7ls5e27:~/nampq2/data# ls
dulieu.json  dulieu.txt  dulieu.word  dulieu.xlsx
```

Hình 3. 1. Thư mục data chứa dữ liệu thử nghiệm

Tiếp theo, tiến hành truy cập vào hệ thống Nifi thông qua "Đăng nhập SSO". Trên giao diện chính của Nifi ta nhấn chọn nút tạo một bộ xử lý GetFile mới và nhấn đúp vào để nhập các thông tin thuộc tính cho khối này.

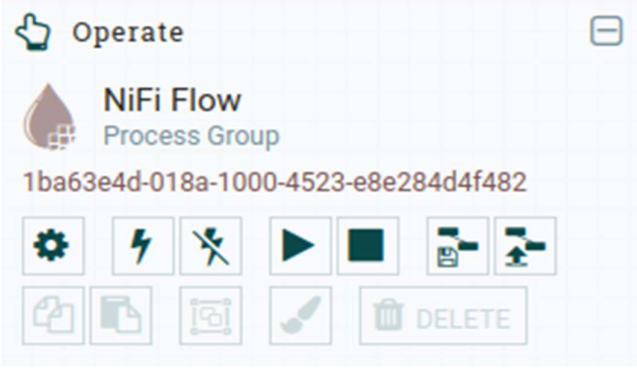
- Input Directory: đường dẫn đến thư mục chứa dữ liệu. Trong trường hợp thực nghiệm, dữ liệu được lưu tại đường dẫn "/root/nampq/dataS3"
- File Filter: Trong thư mục cần gửi đi có rất nhiều định dạng tệp như json, txt, word, v.v nên ta sẽ cài đặt theo cú pháp [^\.].* để chuyển tất cả dữ liệu.

Khi đã hoàn tất quá trình thêm và cấu hình khối Getfile, ta tiếp tục thao tác thêm và cấu hình với khối PutS3Object, trong đó ta sẽ cần phải thiết lập những thông số quan trọng sau:

- Object Key: Trong trường hợp thử nghiệm, đường dẫn được cài đặt như sau: 000.00.00.G31_VAST/FILE/20012024/${filename}
- Bucket: Bucket dữ liệu được đặt cố định là: openscience-test-bucket
- Access Key ID: khóa truy cập vào kho dữ liệu cố định: UJPZHZ35OJDLWGV11FK1
- Secret Access Key: mật khẩu truy cập kho lưu trữ cố định: KrD6hbs9COGiON6YSRXSjdkOnIJbSogLwlBUPVIK
- Endpoint Override URL: địa chỉ truy cập đến kho lưu trữ dữ liệu cố định: https://s3-hfx03.fptcloud.com

Sau khi đã tạo được hai khối truyền dữ liệu và khối nhận dữ liệu, ta di chuyển con trỏ chuột đến khối GetFile, kéo mũi tên hướng tới khối PutS3Object, để tạo liên

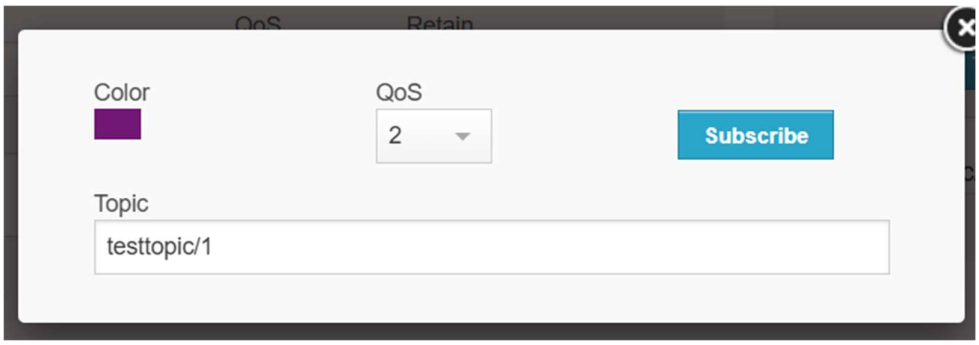
kết giữa hai khối này. Tại khối Operate trên giao diện chính của Apache Nifi, ta sẽ chọn nút Start để thực hiện việc gửi và nhận dữ liệu từ server lưu trữ vào kho dữ liệu.



Hình 3. 2. Khối operate để thực hiện thao tác

3.1.1.2. Thu thập dữ liệu luồng từ hệ thống IoT

Với những dữ liệu luồng cần được thu thập từ hệ thống IoT, ta sẽ cần tạo ra một máy chủ ảo gửi tín hiệu IOT đi, để thực hiện điều này, ta có thể truy cập vào trang WEB mô phỏng theo địa chỉ "**https://www.hivemq.com/demos/websocket-client/**". Đầu tiên cần kết nối tới máy chủ bằng cách nhấn chuột vào nút Connect (màu xanh nước biển), sau khi kết nối xong, tại mục Connection sẽ được hiển thị là "**Connected**". Tiếp theo cần tạo mới một chủ đề bằng cách nhấn vào nút "**Add New Topic Subscription**", một cửa sổ nhỏ hiện lên và người dùng thực hiện khai báo tên chủ đề và nhấn "**Subscribe**" để ghi nhận.

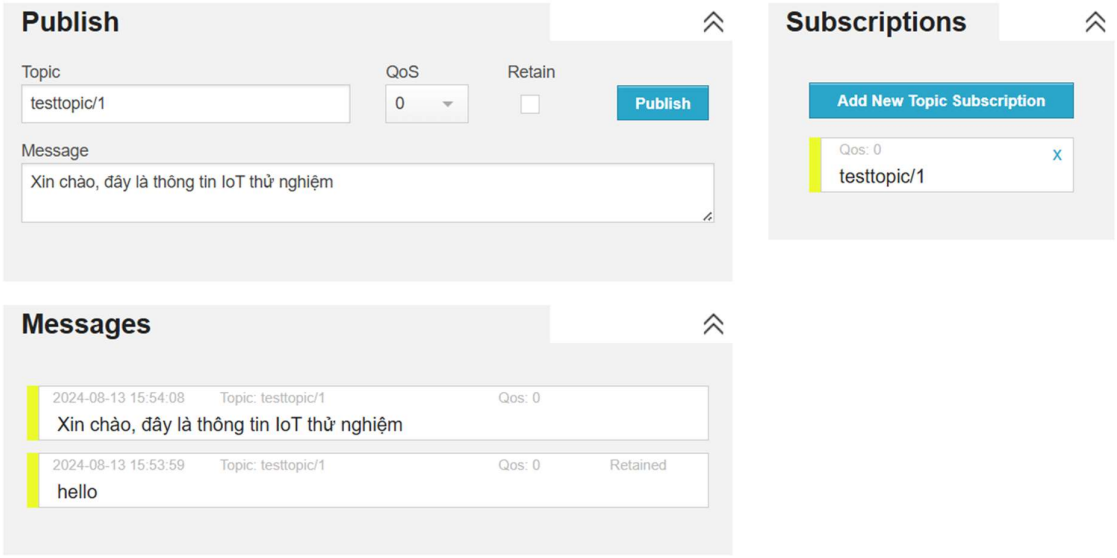


Hình 3. 3. Giao diện khai báo chủ đề cho dữ liệu IoT thử nghiệm

Tại khu vực Message, người dùng có thể nhập bất cứ nội dung gì (ví dụ đây là thông tin từ thiết bị IOT gửi đi).

Ta tiếp tục truy cập vào hệ thống thu thập dữ liệu Nifi, sau đó tạo mới khối chức năng "**ConsumeMQTT**" và thiết lập những thông tin quan trọng theo như trên trang gửi thử nghiệm thông tin IoT như sau:

- Broker URI: Trong trường hợp thử nghiệm, giá trị này được cài đặt là tcp://mqtt-dashboard.com:1883
- Topic Filter: Trong trường hợp thử nghiệm, giá trị này được cài đặt là testtopic/1
- ClientID: Ta sẽ lấy ClientID này trên trang giả lập thông tin IoT HiveMQ khi đã kết nối thành công. Ở đây là "**clientId-pgunBCksg1**"



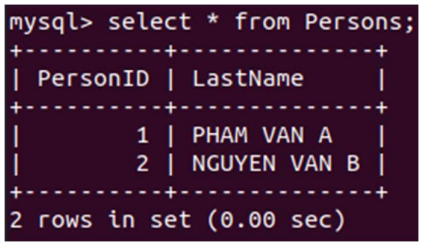
Hình 3. 4. Thông tin được sử dụng để thử nghiệm dữ liệu IoT

Khi đã tạo thành công khối "**ConsumeMQTT**", ta tiếp tục tạo mới khối "**PutS3Object"** với thông tin thuộc tính như ở phần trước và nối hai khối lại với nhau. Sau khi các khối trong Apache Nifi đã được Start và hoạt động, ta thực hiện gửi "**Messages**" thông qua trang WEB giả lập bằng cách nhập nội dung muốn gửi và nhấn "**Publish**" để bắt đầu truyền thông tin.

Ta cũng có thể xem trạng thái hoạt động của các khối chức năng bằng cách nhấn chuột phải vào khối trung gian nằm giữa các khối ConsumeMQTT và PutS3Object.

3.1.1.3. Thu thập dữ liệu từ hệ thống CSDL quan hệ

Trong trường hợp thử nghiệm, ta có một bảng cơ sở dữ liệu tên là Persons có dạng như Hình 3.5.



Hình 3. 5. Cơ sở dữ liệu thử nghiệm dạng quan hệ

Với khối truyền dữ liệu thì lần này ta sẽ sử dụng khối chức năng "**ExecuteSQL**" với các thông tin thuộc tính cần lưu tâm như sau:

- Database Connection Pooling Service: Trong trường hợp thử nghiệm này sử dụng DBCPConnectionPool
- SQL select query: Câu lệnh truy vấn cơ sở dữ liệu. Và để lấy được dữ liệu trong bảng, cần thực hiện câu lệnh "**select * from Persons**"

Ta tiếp tục chuẩn hóa dữ liệu thu được từ cơ sở dữ liệu cần biến đổi về dạng JSON. Để thực hiện điều này cần sử dụng khối chức năng có tên là "**ConvertAvroToJSON**". Ngoài ra, để có thể tùy chỉnh thông tin (ví dụ như tên dữ liệu) người dùng có thể khi báo thêm một khối chức năng "**UpdateAttibute**" bằng các thao tác tương tự như trên.

Còn đối với khối nhận dữ liệu, thì ta cũng sẽ tiếp tục sử dụng khối "**PutS3Object**" với những thông tin được sử dụng để thử nghiệm đã được nêu ở phần trên.

Sau khi đã tạo được hai khối truyền dữ liệu và khối nhận dữ liệu, ta di chuyển con trỏ chuột đến khối "**ExecuteSQL**", ta kéo mũi tên hướng tới khối "**ConvertAvroToJSON**" và "**UpdateAttibute**", sau đó kéo tiếp mũi tên đến khối "**PutS3Object**", lúc này một liên kết sẽ được hình thành.

3.1.1.4. Thu thập dữ liệu qua API

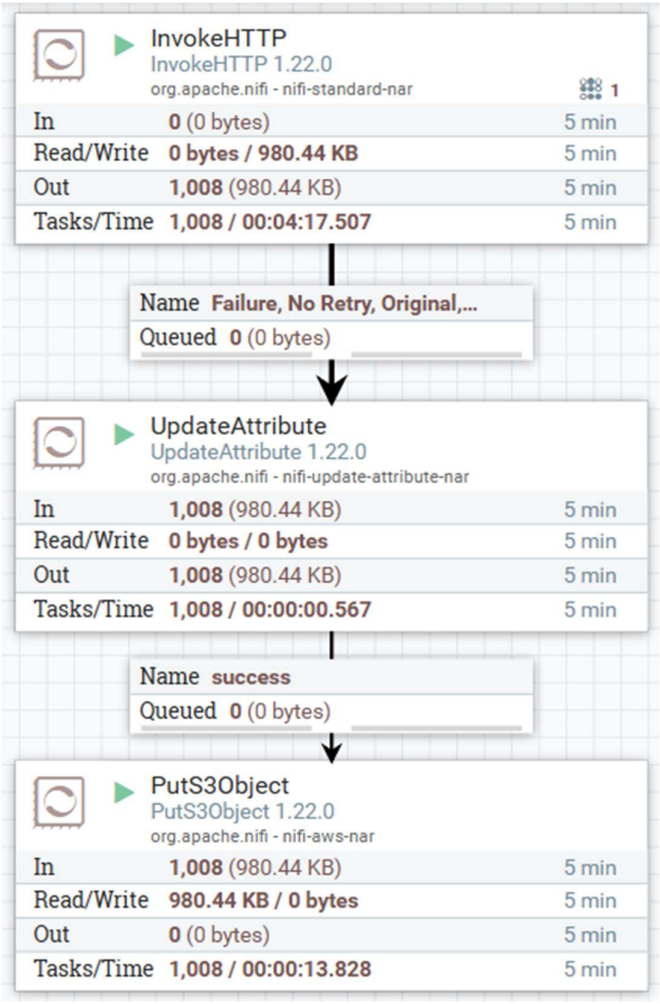
Ở trường hợp này, ta tiếp tục sử dụng một nguồn dữ liệu API chứa những thông tin người dùng thử nghiệm tại địa chỉ "**https://reqres.in/api/users**". Và trên thanh công cụ của Apache Nifi, ta nhấn chọn nút tạo một bộ xử lý mới với tên khối chức năng là InvokeHTTP. Ta cần cấu hình những thông tin quan trọng sau cho khối chức năng này:

- HTTP Method: Trong trường hợp thử nghiệm này, phương thức được sử dụng là GET để lấy dữ liệu.
- HTTP URL: Địa chỉ API được sử dụng sẽ là "**https://reqres.in/api/users**"

Ngoài ra, để có thể tùy chỉnh thông tin (ví dụ như tên dữ liệu) người dùng có thể khi báo thêm một khối chức năng có tên "**UpdateAttribute**" bằng các thao tác tương tự như trên và nhấn nút "+" để khai báo thêm thuộc tính tên là filename và điền giá trị của thuộc tính đó theo dạng json.

Đối với khối nhận dữ liệu, ta tiếp tục sử dụng tới khối "**PutS3Object**", ta có thể tạo lại khối này bằng cách thêm mới hoặc sao chép từ những luồng dữ liệu trước đó ra và chỉ cần thay đổi đường dẫn chứa tệp dữ liệu.

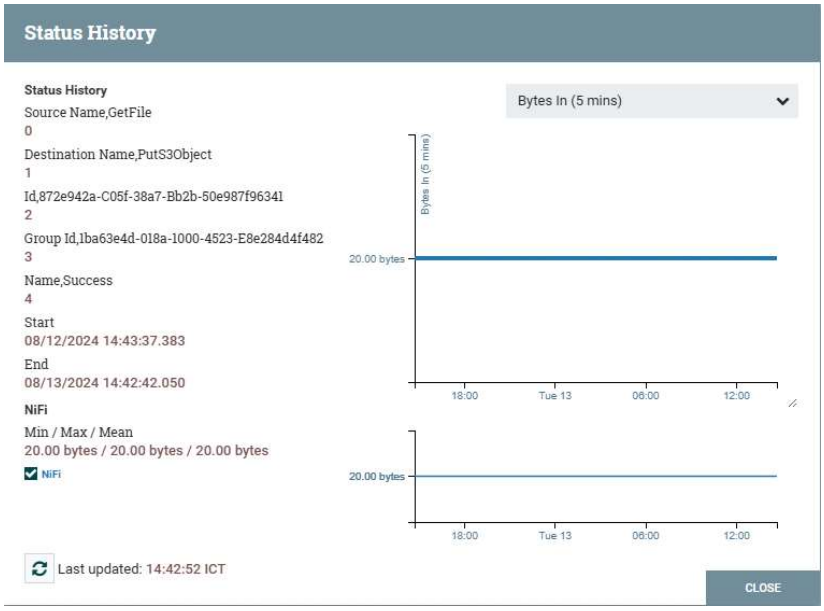
Sau khi đã có đầy đủ các khối để tạo thành một luồng nhận và gửi dữ liệu thông qua API như ở Hình 3.6.



Hình 3. 6. Luồng gửi và nhận dữ liệu thông qua API

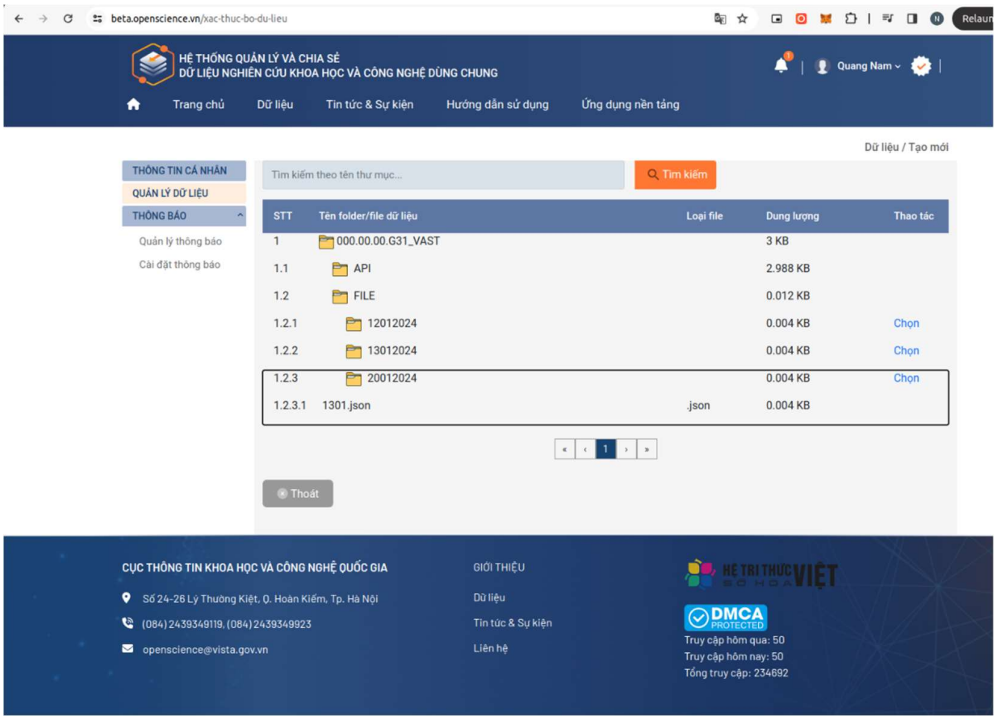
3.1.2. Đánh giá hoạt động

Để xem trạng thái hoạt động và lưu lượng dữ liệu của các khối chức năng ta có thể nhấn chuột phải vào khối trung gian nằm giữa khối gửi dữ liệu và PutS3Object. Sau đó chọn View status history và List queue để xem thông tin chi tiết.



Hình 3. 7. Lịch sử trạng thái truyền nhận file

Trong đó ta có Status History sẽ thể hiện tổng quan về số lượng FlowFiles đã được xử lý, tốc độ xử lý, và các lỗi có thể xảy ra khi tải dữ liệu lên S3. Còn List Queue sẽ được sử dụng để xem những FlowFiles nào đang chờ để được tải lên S3, kiểm tra thuộc tính của chúng để xác định xem có vấn đề gì với dữ liệu hoặc quá trình truyền tải hay không. Trong quá trình nhận dữ liệu từ API, ta cần đáp ứng được một số nội dung như: API có độ ổn định cao và ít xảy ra lỗi, downtime và lưu ý tới chính sách giới hạn tần suất của API hay còn gọi là rate limit. Nếu quá trình truyền dữ liệu không phát sinh vấn đề và hoạt động ổn định như trên hình 3.3, thì ta có thể tiến hành cập nhật bổ sung các trường dữ liệu trên hệ thống Openscience. Tại trang chủ, ta nhấn chọn mục “Dữ liệu của tôi”, lúc này màn hình quản lý danh sách các bộ dữ liệu sẽ hiện ra, tiếp tục nhấn chọn “Cập nhật bộ dữ liệu”, thông tin của bộ dữ liệu vừa gửi lên sẽ được hiển thị tại đây.



Hình 3. 8. Dữ liệu được hiển thị trong mục quản lý dữ liệu

Ta cũng có thể nhấn nút Chọn để xem bộ dữ liệu, cập nhật thông tin bộ dữ liệu và gửi yêu cầu xác thực bộ dữ liệu. Một khi đã khai báo xong, ta sẽ nhấn Lưu và Gửi duyệt để chờ quản trị viên phê duyệt bộ dữ liệu.

3.2. THỬ NGHIỆM, ĐÁNH GIÁ GIẢI PHÁP TÍCH HỢP SPARK

3.2.1. Thử nghiệm xử lý dữ liệu theo lô

Để thực hiện các tác vụ xử lý dữ liệu theo lô trên Apache Spark với dữ liệu quy mô lớn, cần thiết kế hoặc tìm kiếm tập dữ liệu lớn có sẵn. Ví dụ sau đây sử dụng tập dữ liệu mô tả chi tiết về các chuyến đi taxi và dịch vụ thuê xe tại New York, được cung cấp bởi Ủy ban taxi và xe limousine của thành phố. Tập dữ liệu này có thể được tải xuống từ nguồn dữ liệu công cộng trên trang GitHub.

Đầu tiên ta cần chuẩn bị dữ liệu bằng việc tải dữ liệu về và giải nén, với bộ dữ liệu ở đây là bộ dữ liệu chi tiết về những chuyến đi cho thuê xe số lượng lớn trong tháng 1 năm 2021 tại New York.

```
!wget https://github.com/DataTalksClub/nyc-tlc-
data/releases/download/fhvhv/fhvhv_tripdata_2021-01.csv.gz
!gzip -d 'fhvhv_tripdata_2021-1.csv.gz'
```

Hình 3. 9. Câu lệnh sử dụng để tải và giải nén bộ dữ liệu

Sau khi bộ dữ liệu đã được chuẩn bị về máy, ta tiếp tục tiến hành đẩy bộ dữ liệu lên kho dữ liệu bằng lệnh put và tạo Sparksession kết nối tới S3 Storage.

```
!hdfs dfs -put fhvhv_tripdata_2021-01.csv /user/tranductho1309
```

Hình 3. 10. Đẩy dữ liệu lên kho dữ liệu

```
spark = SparkSession.builder \
        .appName('demo') \
        .getOrCreate()
```

Hình 3. 11. Khởi tạo SparkSession

Để phân tích DataSet vừa tải về, ta cần dùng Spark và hàm read() để đọc dữ liệu trong file csv vừa giải nén thành công.

```
df.show()
```

hvfhs_license_num	dispatching_base_num	pickup_datetime	dropoff_datetime	PULocationID	DOLocationID	SR_Flag
HV0003	B02682	2021-01-01 00:33:44	2021-01-01 00:49:07	230	166	null
HV0003	B02682	2021-01-01 00:55:19	2021-01-01 01:18:21	152	167	null
HV0003	B02764	2021-01-01 00:23:56	2021-01-01 00:38:05	233	142	null
HV0003	B02764	2021-01-01 00:42:51	2021-01-01 00:45:50	142	143	null
HV0003	B02764	2021-01-01 00:48:14	2021-01-01 01:08:42	143	78	null
HV0005	B02510	2021-01-01 00:06:59	2021-01-01 00:43:01	88	42	null
HV0005	B02510	2021-01-01 00:50:00	2021-01-01 01:04:57	42	151	null
HV0003	B02764	2021-01-01 00:14:30	2021-01-01 00:50:27	71	226	null
HV0003	B02875	2021-01-01 00:22:54	2021-01-01 00:30:20	112	255	null
HV0003	B02875	2021-01-01 00:40:12	2021-01-01 00:53:31	255	232	null
HV0003	B02875	2021-01-01 00:56:45	2021-01-01 01:17:42	232	198	null
HV0003	B02835	2021-01-01 00:29:04	2021-01-01 00:36:27	113	48	null
HV0003	B02835	2021-01-01 00:48:56	2021-01-01 00:59:12	239	75	null
HV0004	B02800	2021-01-01 00:15:24	2021-01-01 00:38:31	181	237	null
HV0004	B02800	2021-01-01 00:45:00	2021-01-01 01:06:45	236	68	null
HV0003	B02682	2021-01-01 00:11:53	2021-01-01 00:18:06	256	148	null
HV0003	B02682	2021-01-01 00:28:31	2021-01-01 00:41:40	79	80	null

Hình 3. 12. Hiển thị dữ liệu sau khi đọc file csv xong

Để tối ưu hiệu năng xử lý của Spark, ta phân vùng lại dữ liệu thành các partition nhỏ hơn. Đối với tập dữ liệu này, chúng ta sẽ xáo trộn dữ liệu và phân vùng lại thành 24 partition, sau đó ghi dữ liệu đã được phân vùng thành công dưới dạng Apache Parquet để tăng tốc độ truy vấn.

```
df = df.repartition(24)
df.write.mode('overwrite').parquet('hdfs:///user/tranductho1309/fhvhv/2021/01')
```

Hình 3. 13. Phân vùng và ghi dữ liệu đã được phân vùng dưới dạng Parquet

Với dữ liệu đầu ra (24 file Apache Parquet), chúng ta cần gọi hàm read() của Spark đọc file Parquet từ bucket dữ liệu. Sau đó dùng hàm "**createOrReplaceTempView()**" của Spark để tạo một bảng dữ liệu quan hệ để có thể thao tác từ DataFrame đọc dữ liệu dạng Apache Parquet ở trên.

```
df = spark.read.parquet('hdfs:///user/tranductho1309/fhvhv/2021/01')
df.createOrReplaceTempView('hvfhw_data')
```

Hình 3. 14. Hàm đọc và thao tác dữ liệu dạng Parquet

3.2.2. Đánh giá xử lý dữ liệu theo lô

Sau khi ta xử lý dữ liệu xong, ta sẽ có thể thao tác với dữ liệu đã xử lý bằng các Query như trên cơ sở dữ liệu dạng quan hệ thông thường.

```
# Question 1: taxi trips on Jan 15 = 406887
spark.sql("""
SELECT
    count(*)
FROM
    hvfhw_data
WHERE
    pickup_datetime LIKE '2021-01-01%'
""").show()
```

```
[Stage 5:=====================================>
+--------+
|count(1)|
+--------+
|  406887|
+--------+
```

Hình 3. 15. Tương tác với dữ liệu bằng truy vấn SQL sau khi xử lý xong

3.3. THỬ NGHIỆM, ĐÁNH GIÁ GIẢI PHÁP TÍCH HỢP KUBEFLOW

3.3.1. Mô tả bài toán

- Bài toán: phân loại bệnh ung thư da cho ảnh chụp các loại ung thư da phổ biến.
- Đầu vào: đường dẫn đến file ảnh để dự đoán bệnh ung thư da
- Đầu ra: một file .json trong thư mục output_dir. Thông tin trong file kết quả bao gồm các thông tin: dự đoán và xác suất dự đoán của loại ung thư đó.

3.3.2. Các bước thực hiện

Trong bài toán này ta cần xây dựng các hàm đã có thành các component trong Kubeflow do đó chúng ta sẽ cần sử dụng tới những đoạn code có sẵn để làm đầu vào. Ở đây là mã nguồn xây dựng mô hình xác định ung thư.

```
git clone https://github.com/weedharm/deploy_openscience.git
cd deploy_openscience/skin_cancer
```

Hình 3. 16. Clone mã nguồn từ Github

Trong file code đã chứa sẵn 3 hàm: load_data(), build_and_model_() và model_serving(). Sau đó ta sử dụng kubeflow pipeline(kfp) để có thể đóng các hàm này thành các component. Và khi đã có các component cần thiết cho một pipeline, ta sẽ tiến hành tạo một pipeline hoàn chỉnh.

```
import kfp

from kfp import dsl

import kfp.components as components

comp_get_data =
components.create_component_from_func(load_data,base_image="public.ecr.aws/j1r
0q0g6/notebooks/notebook-servers/jupyter-tensorflow-full:v1.5.0")

comp_model_building =
components.create_component_from_func(build_and_train_model,base_image="publ
ic.ecr.aws/j1r0q0g6/notebooks/notebook-servers/jupyter-tensorflow-full:v1.5.0",

                                packages_to_install=['opencv-python-headless','tqdm'])

comp_model_serving =
components.create_component_from_func(model_serving,base_image="public.ecr.aw
s/j1r0q0g6/notebooks/notebook-servers/jupyter-tensorflow-full:v1.5.0",

                                        packages_to_install=['kserve==0.11.2'])

@dsl.pipeline(

   name='skin-cancer',

   description='Detect skin'

)

def output_test(n_loops,methods):

   step1 = comp_get_data()

   step2 = comp_model_building()

   step2.after(step1)

   step3 = comp_model_serving()

   step3.after(step2)
```

Hình 3. 17. Đóng gói các hàm thành các component cho Kubeflow

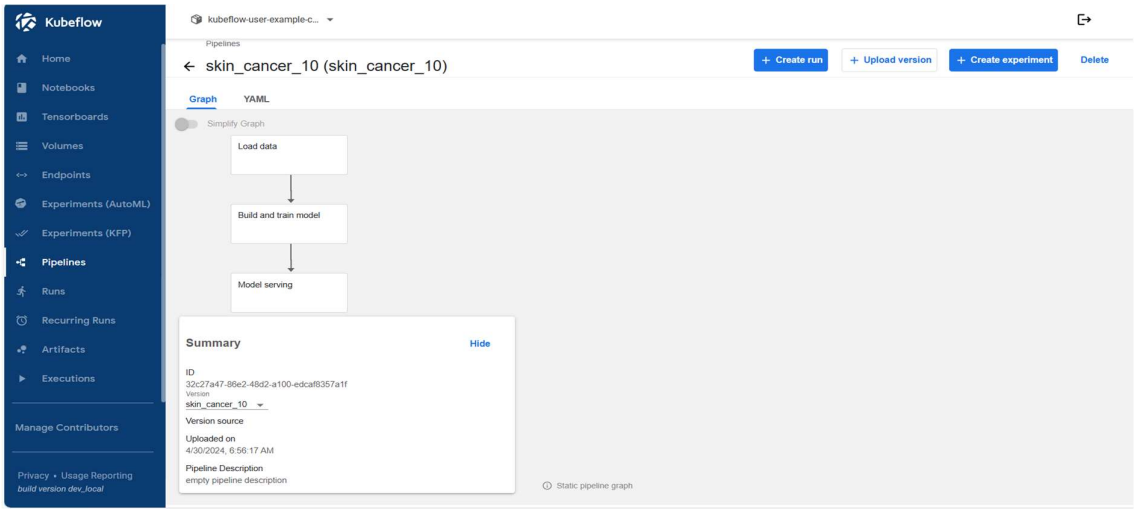
```
if __name__ == "__main__":
    client = kfp.Client()
    arguments = {
        "n_loops" : 10,
        "methods": "SVM"
    }
    run_directly = 0
    if (run_directly == 1):

client.create_run_from_pipeline_func(output_test,arguments=arguments,experiment_name="test")
    else:
        kfp.compiler.Compiler().compile(pipeline_func=output_test,package_path='output_test_1.yaml')

client.upload_pipeline(pipeline_package_path='output_test_1.yaml',pipeline_name="skin_cancer_10")
```

Hình 3. 18. Tạo một kubeflow pipeline

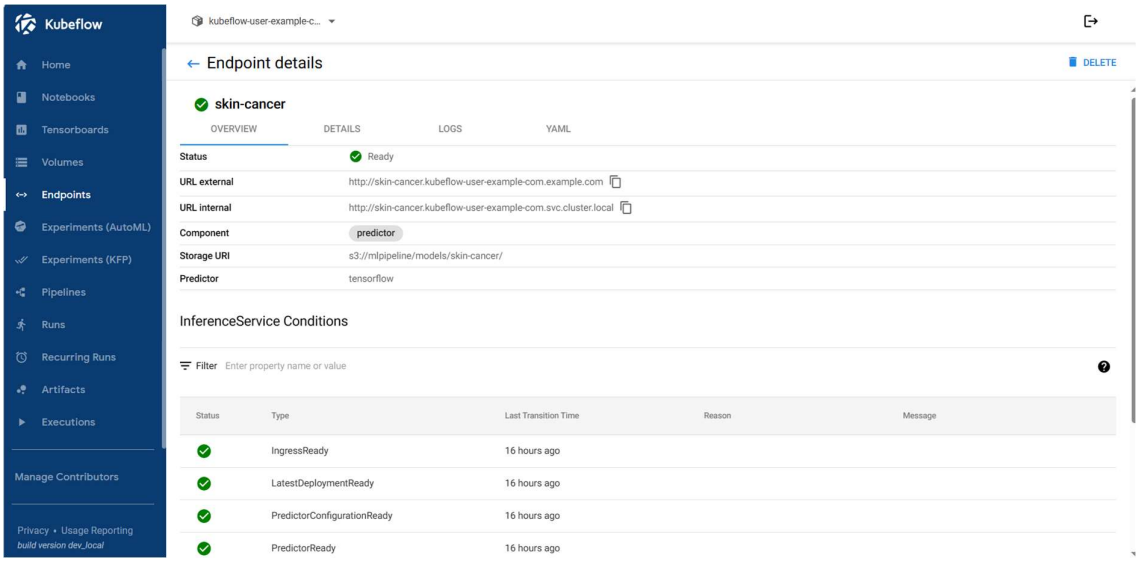
Sau khi tạo pipeline xong, ta truy cập vào trang giao diện của Kubeflow sẽ có một flow có tên tương tự với tên ta đặt trong bước tạo pipeline ở Hình 3.18. Tiếp đó ta sẽ có thể ấn vào "**Create run**" để chạy nếu đã có môi trường từ trước, còn nếu mới chạy xây dựng mô hình lần đầu tiên thì ta sẽ nhấn vào "**Create experiment**" để tạo môi trường chạy pipeline.



Hình 3. 19. Pipeline mới được tạo ra trong Kubeflow

3.3.3. Đánh giá kết quả thực hiện bài toán

Sau khi pipeline được huấn luyện xong, phần endpoint sẽ chứa mô hình mà ta đã huấn luyện được và tiến hành lưu lại URL để thử nghiệm mô hình bằng cách đưa một ảnh (dương tính) vào để mô hình dự đoán, và ta thấy kết quả dự đoán (predict) chính xác với kết quả thực tế là "1".



Hình 3. 20. Mô hình sau khi được huấn luyện xong

```
[42]: import requests
import base64
# import librosa
import io
import numpy as np
import cv2
from cv2 import imread, resize
import numpy as np
import pandas as pd
import requests
import json
def send_sample_for_prediction(image, url):
    x=[]
    output_shape = (128, 128)
    img = imread(image)
    img2 = resize(img, output_shape)
    # print(img2)
    x.append(img2)
    x = np.array(x)

    inference_input = {
      'instances': x.tolist()
    }

    response = requests.post(url, json=inference_input)


    if response.status_code == 200:
        return np.argmax(response.json()["predictions"])
    else:
        print(response.json())
        print("Error:", response.status_code)
        print(response)
        return None
```

```
[43]: x= 'ISIC_0026266.jpg'
url ="http://skin-cancer.kubeflow-user-example-com.svc.cluster.local/v1/models/skin-cancer:predict"
predict = send_sample_for_prediction(x, url)
print(predict)
```

```
1
```

Hình 3. 21. Thử nghiệm mô hình chuẩn đoán ung thư thông qua ảnh

3.4. THỬ NGHIỆM SSO TRÊN OPENSCIENCE.VN

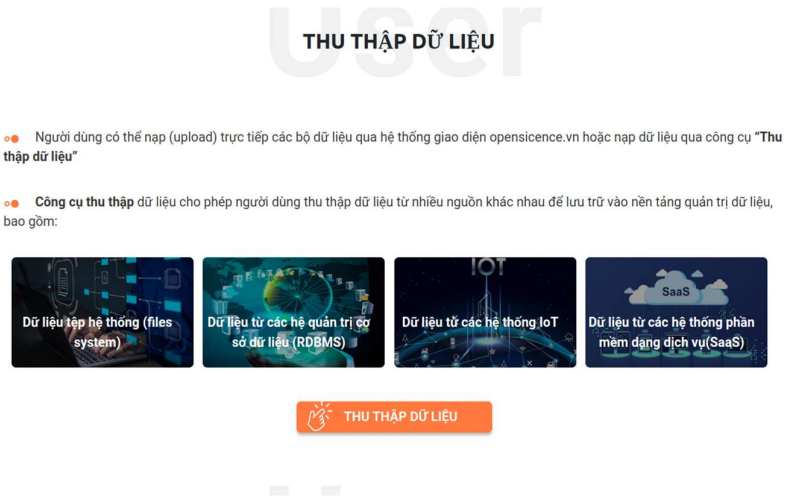
Trong phần này, chúng ta sẽ sử dụng 2 tài khoản đã được cấp quyền để truy cập vào kubeflow.

Thông tin tài khoản như sau:

- Tài khoản thứ nhất: nampq.689@gmail.com | Phamquangnam123@
- Tài khoản thứ hai: nampq.aime@gmail.com | Phamquangnam123@

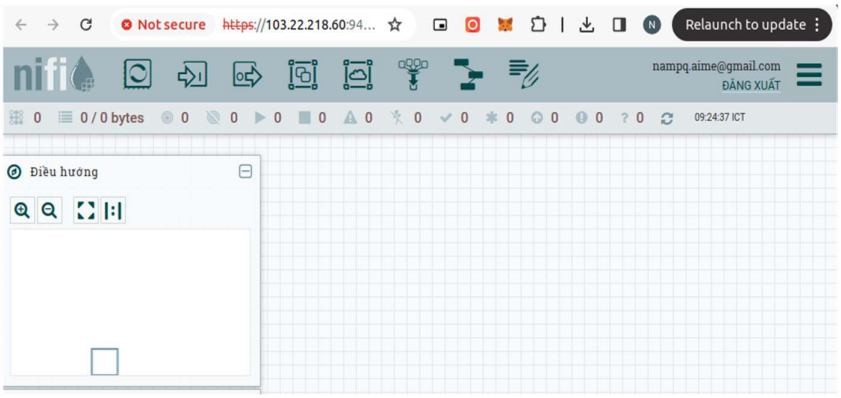
3.4.1. Kiểm tra SSO với Apache Nifi

Đầu tiên, người dùng truy cập vào trang chủ hệ thống theo đường dẫn "**https://openscience.vn/**", sau đó để đăng nhập vào hệ thống, ta chọn mục "**Đăng nhập SSO**", một màn hình đăng nhập sẽ hiện ra. Ta sử dụng tài khoản thứ nhất đã được cung cấp để đăng nhập vào hệ thống. Sau khi khai báo xong Username và Password, ta nhấn Sign In để đăng nhập. Để truy cập vào Apache Nifi, người dùng nhấn chọn Ứng dụng nền tảng, sau đó di chuyển màn hình xuống phía dưới đến mục "THU THẬP DỮ LIỆU" và truy cập vào mục này. Sau khi được chuyển sang cửa sổ của Nifi, tại góc trên cùng bên phải màn hình chính, nếu ta thấy tài khoản đăng nhập trong Apache Nifi giống với tài khoản đã đăng nhập trước đó bên trang openscience thì hệ thống đã thực hiện quá trình đăng nhập một lần thành công.



Hình 3. 22. Giao diện phần Thu thập dữ liệu trên trang chủ

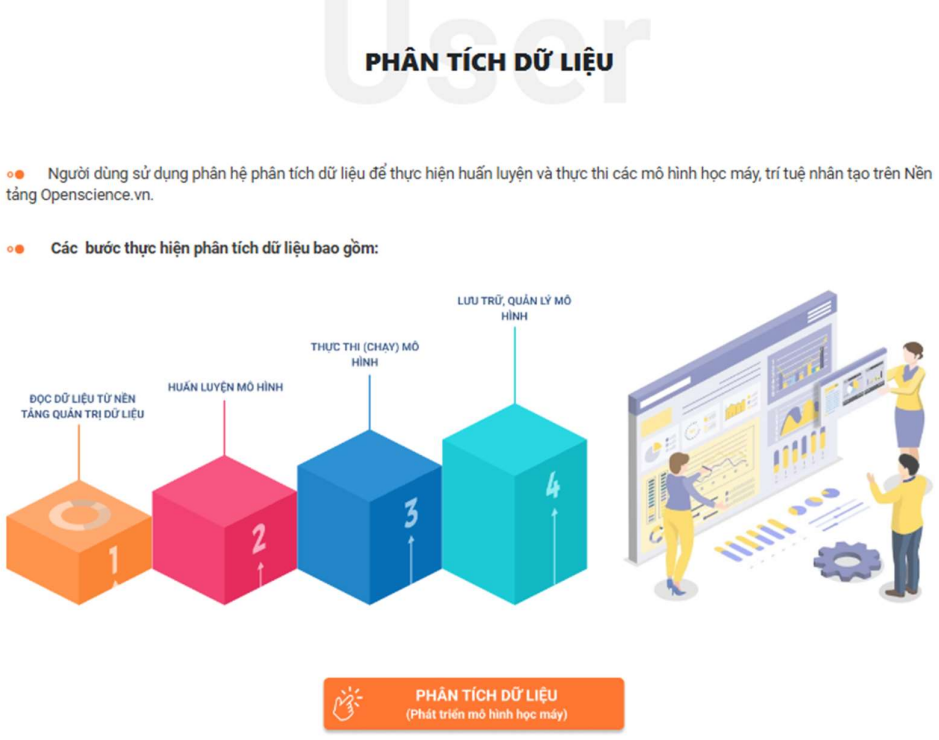
Sau đó ta quay trở lại với màn hình chính của trang chủ openscience, ta tiến hành xuất tài khoản cũ và đăng nhập bằng tài khoản thứ 2 được cung cấp và tiếp tục truy cập vào mục "Thu thập dữ liệu" trong phần "Ứng dụng nền tảng". Nếu ta tiếp tục thấy hệ thống Nifi đã được đăng nhập bằng tài khoản thứ 2, thì hệ thống đăng nhập một lần SSO đã hoạt động thành công.



Hình 3. 23. Đăng nhập SSO thành công tài khoản thứ hai

3.4.2. Kiểm tra SSO với Kubeflow

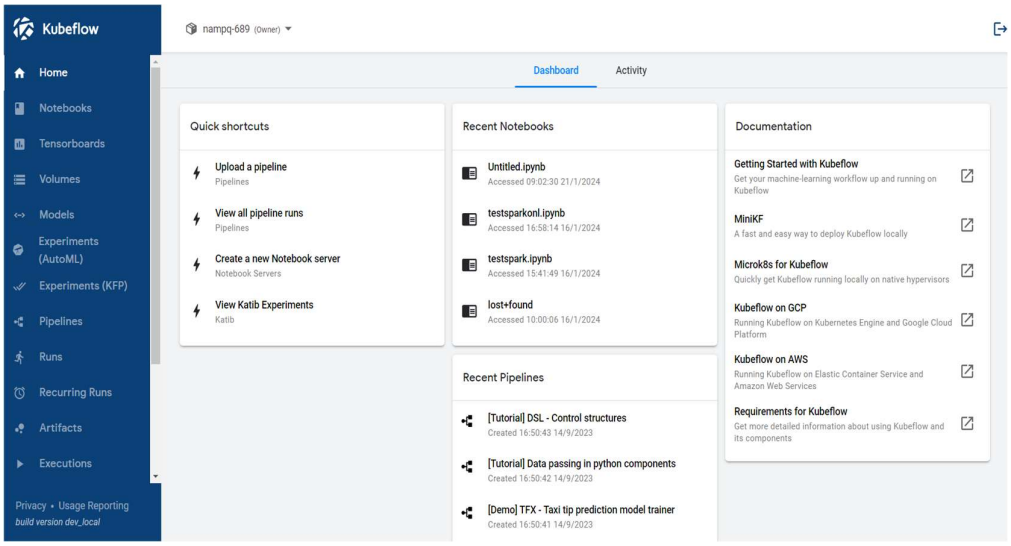
Khi kiểm tra thành công với Apache Nifi, ta tiếp tục tiến hành với Kubeflow bằng cách đăng nhập vào hệ thống Openscience thông qua tài khoản thử nghiệm đầu tiên, sau đó chuyển hướng truy cập vào "kubeflow.openscience.vn" hoặc truy cập vào phần Phân tích Dữ liệu trên giao diện chính của trang web openscience.



Hình 3. 24. Giao diện phần Phân tích Dữ liệu trên trang chủ

Lúc này, nếu ta truy cập được vào giao diện chính của Kubeflow và quan sát thấy một workspace đã được tạo tương ứng với thông tin tài khoản đã được sử dụng để đăng nhập thì hệ thống đã được cấu hình thành công.

Tiếp theo chúng ta sẽ sử dụng tài khoản thứ hai để đăng nhập vào hệ thống. Trước tiên cần thực hiện thoát khỏi tài khoản thứ nhất bằng các bước sau. Đầu tiên ta trở lại trang chủ của openscience, tại khu vực góc phải phía trên màn hình có hiển thị tên người dùng, thực hiện nhấn nút mũi tên xuống phía dưới và chọn "Đăng xuất", sau đó tiếp tục đăng nhập lại bằng tài khoản thử nghiệm thứ hai thông qua hệ thống "Đăng nhập SSO" và chuyển hướng truy cập sang hệ thống Kubeflow. Nếu ta truy cập được vào giao diện chính của Kubeflow với một namespace đã tạo tương ứng với tài khoản thử nghiệm thứ hai mà ta mới đăng nhập thì hệ thống đã hoạt động ổn định.



Hình 3. 25. Giao diện chính với namespace mới được tạo tự động trên Kubeflow

KẾT LUẬN VÀ KIẾN NGHỊ

1. KẾT LUẬN

Luận văn đã nghiên cứu và xây dựng giải pháp tích hợp các nền tảng mã nguồn mở phục vụ thu thập, lưu trữ, xử lý dữ liệu lớn và phát triển mô hình học máy trong một hệ thống thống nhất. Bằng việc tích hợp các nền tảng như NiFi, CEPH, Apache Spark, Kubeflow, và Kubernetes, đề tài đã tạo ra một quy trình (pipeline) hoàn chỉnh từ thu thập dữ liệu đến triển khai mô hình học máy.

Hệ thống Openscience.vn được nâng cấp không chỉ để hỗ trợ nghiên cứu khoa học và công nghệ mà còn cung cấp một môi trường mạnh mẽ, linh hoạt và hiệu quả cho các nhà khoa học dữ liệu có thể dễ dàng thực hiện các tác vụ phân tích dữ liệu lớn và học máy.

Luận văn đã đạt được những kết quả như:

- Hệ thống đã được cấu hình, thử nghiệm và cho thấy hiệu quả trong việc thu thập và xử lý dữ liệu từ nhiều nguồn khác nhau (cơ sở dữ liệu quan hệ, tệp, API, dữ liệu luồng từ IoT).
- Đã có thể thực hiện xử lý dữ liệu lớn, cả trong môi trường xử lý theo lô (batch processing) và xử lý theo luồng (stream processing) thông qua Apache Spark.
- Cấu hình Kubeflow và hoàn thiện quy trình việc quản lý và triển khai các mô hình học máy, từ giai đoạn huấn luyện, kiểm thử đến triển khai.
- Tích hợp thành công tính năng đăng nhập một lần (SSO) giúp hệ thống tăng cường khả năng quản lý, bảo mật và thuận tiện
- Từ những nền tảng đã được cài đặt sẵn và nội dung đề ra trong giải pháp, hệ thống đã hoàn thiện được một pipeline hoàn chỉnh từ bước thu thập dữ liệu, lưu trữ dữ liệu, và xử lý, phân tích dữ liệu.

2. KIẾN NGHỊ

Mặc dù hệ thống đã được thử nghiệm và chứng minh hiệu quả, nhưng quy mô thử nghiệm còn hạn chế. Trong tương lai, cần tiếp tục mở rộng và hoàn thiện hệ thống, đặc biệt là khả năng mở rộng quy mô để xử lý khối lượng dữ liệu lớn hơn và mô hình học máy phức tạp hơn.

Cần tiếp tục nghiên cứu, tối ưu hóa hiệu suất xử lý dữ liệu trong các nền tảng như Apache Spark và Kubeflow, đặc biệt là khi xử lý các khối lượng dữ liệu phức tạp và lớn hơn trong tương lai. Đồng thời, cần nghiên cứu các biện pháp bảo mật nâng cao như xác thực đa yếu tố và mã hóa dữ liệu, nhằm tăng cường an toàn thông tin cho hệ thống.

DANH MỤC TÀI LIỆU THAM KHẢO

1. James Warren, Nathan Marz, 2015, *Big Data: Principles and best practices of scalable realtime data systems*, Manning Publications.
2. Paul Crickard, 2020, *Data Engineering with Python: Work with massive datasets to design data models and automate data pipelines using Python*, Packt Publishing.
3. Emmanuel Ameisen, 2020, *Building Machine Learning Powered Applications: Going from Idea to Product*, O'Reilly Media.
4. Karan Singh, 2016, *Ceph Cookbook*, Packt Publishing.
5. Pontus Sj¨oberg, Lina Vilhelmsson, 2020, *Implementation and Evaluation of a Data Pipeline for Industrial IoT Using Apache NiFi*, Ph.D. Thesis, Karlstad University, Karlstad.
6. Eman Shaikh, Iman Mohiuddin, Yasmeen Alufaisan, Irum Nahvi, 2019, Apache Spark: A Big Data Processing Engine in *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, pp. 1-6.
7. Josh Patterson, Michael Katzenellenbogen, Austin Harris, 2020, *Kubeflow Operations Guide*, O'Reilly Media.
8. S. Chanthakit, P. Keeratiwintakorn, C. Rattanapoka, 2019, An IoT System Design with Real-Time Stream Processing and Data Flow Integration in *2019 Research, Invention, and Innovation Congress (RI2C)*, pp. 1-5.
9. Neylson Crepalde, 2024, *Big Data on Kubernetes: A practical guide to building efficient and scalable data solutions*, Packt Publishing.
10. Wnęk, Karol, and Piotr Boryło. 2023. A Data Processing and Distribution System Based on Apache NiFi in *Photonics*, 10(2), 210.
11. Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, Ion Stoica, 2016, Apache Spark: a unified engine for big data processing in Communications of the ACM, 59(11), pp. 56-65.
12. Doris Xin, Hui Miao, Aditya Parameswaran, Neoklis Polyzotis, 2021, Production Machine Learning Pipelines: Empirical Analysis and Optimization Opportunities in *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*, pp. 2639-2652.
13. A. Gupta, H. K. Thakur, R. Shrivastava, P. Kumar and S. Nag, 2017, A Big Data Analysis Framework Using Apache Spark and Deep Learning in *International Conference on Data Mining Workshops (ICDMW)*, pp. 9-16.
14. Stian Thorgersen, Pedro Igor Silva, 2023, *Keycloak - Identity and Access Management for Modern Applications*, Packt Publishing.

VIỆN HÀN LÂM
KHOA HỌC VÀ CÔNG NGHỆ VN
HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Số: 1115 /QĐ-HVKHCN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Hà Nội, ngày 30 tháng 09 năm 2024

QUYẾT ĐỊNH
Về việc thành lập Hội đồng đánh giá luận văn thạc sĩ

GIÁM ĐỐC
HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Căn cứ Quyết định số 303/QĐ-VHL ngày 01/3/2023 của Chủ tịch Viện Hàn lâm Khoa học và Công nghệ Việt Nam về việc ban hành Quy chế Tổ chức và hoạt động của Học viện Khoa học và Công nghệ;

Căn cứ Thông tư số 23/2021/TT-BGDĐT ngày 30/08/2021 của Bộ trưởng Bộ Giáo dục và Đào tạo về việc ban hành Quy chế đào tạo trình độ thạc sĩ;

Căn cứ Quyết định số 1966/QĐ-HVKHCN ngày 28/12/2021 của Giám đốc Học viện Khoa học và Công nghệ về việc ban hành Quy chế đào tạo trình độ thạc sĩ;

Căn cứ Quyết định số 1899/QĐ-HVKHCN ngày 15/11/2022 của Giám đốc Học viện Khoa học và Công nghệ về việc công nhận học viên cao học khóa 2022B - Đợt 2 năm 2022;

Căn cứ Quyết định số 228/QĐ-HVKHCN ngày 29/03/2024 của Giám đốc Học viện Khoa học và Công nghệ về việc công nhận đề tài và người hướng dẫn luận văn thạc sĩ;

Xét đề nghị của Trưởng khoa Khoa Công nghệ thông tin và Viễn thông, Trưởng phòng Đào tạo.

QUYẾT ĐỊNH:

Điều 1. Thành lập Hội đồng đánh giá luận văn thạc sĩ cho học viên Dương Đình Thiệu với đề tài: **Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia**

Ngành: Hệ thống thông tin Mã số: 8 48 01 04

Danh sách thành viên Hội đồng đánh giá luận văn kèm theo Quyết định này.

Điều 2. Hội đồng có trách nhiệm đánh giá luận văn thạc sĩ theo đúng quy chế hiện hành của Bộ Giáo dục và Đào tạo, Học viện Khoa học và Công nghệ. Quyết định có hiệu lực tối đa 60 ngày kể từ ngày ký và phải đảm bảo thời hạn đào tạo theo quy định của Học viện. Hội đồng tự giải thể sau khi hoàn thành nhiệm vụ.

Điều 3. Trưởng phòng Tổ chức - Hành chính và Truyền thông, Trưởng phòng Đào tạo, Trưởng phòng Kế toán, Trưởng Khoa Công nghệ thông tin và Viễn thông, các thành viên có tên trong danh sách Hội đồng và học viên cao học có tên tại Điều 1 chịu trách nhiệm thi hành Quyết định này./.

***Nơi nhận*:**
- Như Điều 3;
- Giám đốc HV (để b/c);
- Lưu hồ sơ học viên;
- Lưu: VT, ĐT, PQ.10.

KT.GIÁM ĐỐC
PHÓ GIÁM ĐỐC

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM
HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

TS. Trần Thị Phương Anh

DANH SÁCH HỘI ĐỒNG ĐÁNH GIÁ LUẬN VĂN THẠC SĨ

(Kèm theo Quyết định số 1115 /QĐ-HVKHCN ngày 30/09/2024 của Giám đốc Học viện Khoa học và Công nghệ)

Cho luận văn của học viên: Dương Đình Thiệu

Tên đề tài: **Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia**

Ngành: Hệ thống thông tin Mã số: 8 48 01 04

Người hướng dẫn: PGS.TS. Nguyễn Long Giang, Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam

TT	Họ và tên, học hàm, học vị	Ngành	Cơ quan công tác	Trách nhiệm trong Hội đồng
1.	PGS.TS. Nguyễn Việt Anh	Hệ thống thông tin	Viện Công nghệ thông tin, Viện Hàn lâm KHCN VN	Chủ tịch
2.	PGS.TS. Bùi Thu Lâm	Hệ thống thông tin	Học viện Kỹ thuật Mật mã, Ban Cơ yếu Chính phủ	Phản biện 1
3.	TS. Nguyễn Mạnh Hùng	Hệ thống thông tin	Học viện Kỹ thuật Quân sự, Bộ Quốc phòng	Phản biện 2
4.	TS. Trần Đức Nghĩa	Hệ thống thông tin	Viện Công nghệ thông tin, Viện Hàn lâm KHCN VN	Ủy viên - Thư ký
5.	PGS.TS. Nguyễn Long Giang	Hệ thống thông tin	Viện Công nghệ thông tin, Viện Hàn lâm KHCN VN	Ủy viên

Hội đồng gồm 05 thành viên./.

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT LUẬN VĂN THẠC SĨ

Họ và tên người nhận xét: Nguyễn Việt Anh. Học hàm, học vị: PGS.TS.

Chức danh trong Hội đồng: Chủ tịch.

Cơ quan công tác: Viện Công nghệ thông tin, Viện Hàn lâm KHCN VN.

Họ và tên học viên: Dương Đình Thiệu.

Tên đề tài: Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia.

Ngành: Hệ thống thông tin. Mã số: 8480104.

NỘI DUNG NHẬN XÉT

1. Tính cấp thiết, tính thời sự, ý nghĩa khoa học và thực tiễn của đề tài luận văn:

Chia sẻ dữ liệu nghiên cứu khoa học và công nghệ giúp thúc đẩy nghiên cứu và phát triển khoa học, khuyến khích tăng cường chất của dữ liệu để tạo cơ hội cấp thiết, tính thời sự

2. Sự không trùng lặp của đề tài nghiên cứu so với các công trình khoa học, luận văn đã công bố ở trong và ngoài nước; tính trung thực, rõ ràng và đầy đủ trong trích dẫn tài liệu tham khảo:

theo hiểu biết của người đọc, đề tài không trùng lặp với các công trình đã có

3. Sự phù hợp giữa tên đề tài với nội dung nghiên cứu cũng như với chuyên ngành và mã số đào tạo:

Tên đề tài phù hợp với nội dung nghiên cứu, với chuyên ngành và mã số đào tạo

4. Độ tin cậy và tính hiện đại của phương pháp nghiên cứu đã sử dụng để hoàn thành luận văn:

Phương pháp nghiên cứu có tính hiện đại và đảm bảo độ tin cậy

5. Kết quả nghiên cứu của luận văn:

- Làm rõ được mô hình tiếp cận
- Thực hiện được một số công việc liên quan đến xây dựng nền tảng

6. Những hạn chế, thiếu sót của luận văn về nội dung, hình thức và câu hỏi:

- Cần làm rõ cái đóng góp cụ thể của học viên
- Phần tổng hợp chưa thực sự tốt
- Thiếu vắng phần đánh giá kiến nghị

7. Nếu tác giả chưa viết bài báo khoa học thì nội dung của luận văn có thể được viết thành các bài báo để gửi đăng trên trên tạp chí khoa học, sách chuyên ngành hoặc tuyển tập công trình hội nghị khoa học cấp quốc gia, quốc tế hay không?

Không

8. Kết luận chung (khẳng định mức độ đáp ứng các yêu cầu đối với một luận văn Thạc sĩ; luận văn có thể đưa ra bảo vệ để nhận học vị Thạc sĩ được hay không?):

Luận văn có bản đáp ứng yêu cầu, có thể đưa ra bảo vệ để nhận học vị Thạc sĩ

....., ngày 8 tháng 10 năm 2019

Người nhận xét
(Ký, ghi rõ họ tên)

Nguyễn Việt Anh

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT LUẬN VĂN THẠC SĨ

Họ và tên người nhận xét: Bùi Thu Lâm. Học hàm, học vị: PGS.TS.

Chức danh trong Hội đồng: Phản biện 1.

Cơ quan công tác: Học viện Kỹ thuật Mật mã, Ban cơ yếu Chính phủ.

Họ và tên học viên: Dương Đình Thiệu.

Tên đề tài: Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia.

Ngành: Hệ thống thông tin. Mã số: 8480104.

NỘI DUNG NHẬN XÉT

1. **Tính cấp thiết, thời sự và ý nghĩa khoa học:**

Đề tài phù hợp với xu thế phát triển của dữ liệu lớn và trí tuệ nhân tạo. Việc tích hợp các công cụ thu thập, xử lý, và phân tích dữ liệu là một vấn đề quan trọng trong bối cảnh nghiên cứu khoa học hiện nay, đặc biệt khi dữ liệu từ nhiều nguồn cần được quản lý hiệu quả.

Đề tài có ý nghĩa khoa học trong việc tích hợp các giải pháp mã nguồn mở như Apache Spark, Nifi, và Kubeflow vào hệ thống quản lý dữ liệu khoa học quốc gia, nhằm nâng cao hiệu quả của các nghiên cứu khoa học và công nghệ.

2. **Tính không trùng lặp, trung thực và đầy đủ trong trích dẫn:**

Đề tài không trùng lặp và thể hiện sự sáng tạo trong việc tích hợp các nền tảng khác nhau. Các tài liệu tham khảo được trích dẫn rõ ràng, trung thực.

3. **Phù hợp giữa tên đề tài, nội dung và chuyên ngành:**

Tên đề tài phù hợp với nội dung nghiên cứu và chuyên ngành Hệ thống thông tin, mã số 9.48.01.04.

4. **Phương pháp nghiên cứu và độ tin cậy:**

Phương pháp nghiên cứu bao gồm việc tích hợp các nền tảng mã nguồn mở với hệ thống Openscience.vn để thu thập và phân tích dữ liệu lớn. Việc sử dụng các nền tảng hiện đại như Kubeflow, Spark và Nifi đảm bảo độ tin cậy và tính hiệu quả trong xử lý dữ liệu lớn.

5. **Kết quả nghiên cứu:**

Nội dung nghiên cứu được trình bày trong 3 chương. Kết quả nghiên cứu cho thấy hệ thống đã được tích hợp thành công, các thử nghiệm về xử lý dữ liệu và học máy đều đạt kết quả khả quan. Điều này chứng tỏ tính khả thi của hệ thống trong việc hỗ trợ nghiên cứu khoa học và công nghệ.

6. **Hạn chế, thiếu sót:**

Luận văn được trình bày quá ngắn gọn, cần bổ sung để đầy đủ hơn.

Phần mô tả nội dung nghiên cứu cần chi tiết, chứ không phải copy mục lục vào đây

Chương 1 quá ngắn (5 trang) cần bổ sung mô tả các nền tảng

Chương 2: chưa rõ các giải pháp như thế nào? Chỉ thấy mô tả cách cài đặt

Chương 3: chưa rõ tiêu chí đánh giá hệ thống.

Cần thử nghiệm thêm để đánh giá khả năng chịu tải và hiệu quả khi áp dụng thực tiễn.

7. **Khả năng viết bài báo khoa học:**

Không.

8. **Kết luận chung:**

Luận văn cơ bản đáp ứng yêu cầu của một luận văn thạc sĩ. Luận văn đủ điều kiện để đưa ra bảo vệ

Hà Nội, ngày 8 tháng 10 năm 2024

Người nhận xét
(Ký, ghi rõ họ tên)

Bùi Thu Lâm

Lưu ý:

- *Nhận xét được làm thành 02 bản, có chữ ký của người nhận xét và gửi về phòng Đào tạo 02 ngày trước buổi bảo vệ.*
- *Địa chỉ liên hệ: CV. Phạm Thị Như Quỳnh phòng Đào tạo, Học viện Khoa học và Công nghệ, Viện Hàn lâm Khoa học và Công nghệ Việt Nam. 18 Hoàng Quốc Việt, Cầu Giấy, Hà Nội. ĐT02438689977- 0916467768*

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT LUẬN VĂN THẠC SĨ

Họ và tên người nhận xét: Nguyễn Mạnh Hùng. Học hàm, học vị: TS.

Chức danh trong Hội đồng: Phản biện 2.

Cơ quan công tác: Học viện Kỹ thuật Quân sự, Bộ Quốc phòng.

Họ và tên học viên: Dương Đình Thiệu.

Tên đề tài: Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia.

Ngành: Hệ thống thông tin. Mã số: 8480104.

NỘI DUNG NHẬN XÉT

1. Tính cấp thiết, tính thời sự, ý nghĩa khoa học và thực tiễn của đề tài luận văn

Luận văn phát triển giải pháp tích hợp các nền tảng mã nguồn mở để thu thập, lưu trữ, xử lý dữ liệu lớn sử dụng các công nghệ như NiFi, CEPH, Apache Spark, Kubeflow, và Kubernetes. Hệ thống hỗ trợ việc thu thập, xử lý dữ liệu từ nhiều nguồn khác nhau, và hỗ trợ phát triển các mô hình học máy, giúp cho các nhà khoa học dữ liệu dễ dàng thực hiện các nhiệm vụ phân tích phức tạp.

2. Sự không trùng lặp của đề tài nghiên cứu so với các công trình khoa học, luận văn đã công bố ở trong và ngoài nước; tính trung thực, rõ ràng và đầy đủ trong trích dẫn tài liệu tham khảo

Đề tài và nội dung của luận văn không trùng lặp với các công trình khoa học, luận văn đã công bố ở trong và ngoài nước; Trích dẫn trung thực, rõ ràng và đầy đủ đến các tài liệu tham khảo.

3. Sự phù hợp giữa tên đề tài với nội dung nghiên cứu cũng như với chuyên ngành và mã số đào tạo

Tên đề tài phù hợp với nội dung nghiên cứu cũng như với chuyên ngành và mã số đào tạo.

4. Độ tin cậy và tính hiện đại của phương pháp nghiên cứu đã sử dụng để hoàn thành luận văn

Phương pháp nghiên cứu đảm bảo độ tin cậy và tính hiện đại. Cụ thể:

Luận văn áp dụng Kubeflow để triển khai mô hình học máy và Kubernetes để tự động hóa việc triển khai hạ tầng đám mây là những xu hướng công nghệ hiện đại, phù hợp với các yêu cầu của ngành công nghiệp 4.0. Hệ thống đã được thử nghiệm, cho phép các nhà khoa học dễ dàng phân tích dữ liệu lớn, đáp ứng được yêu cầu của các bài toán khoa học phức tạp trong các lĩnh vực khác nhau.

5. Kết quả nghiên cứu của luận văn:

Xây dựng và triển khai giải pháp đăng nhập 1 lần (SSO) cho OPENSCIENCE.VN

Xây dựng và triển khai giải pháp tích hợp: NIFI, SPARK và KUBEFLOW vào hệ thống OPENSCIENCE.VN.

Các giải pháp đề xuất góp phần hoàn thiện hệ thống OPENSCIENCE.VN gồm đầy đủ các chức năng như: thu thập, lưu trữ, xử lý dữ liệu lớn và hỗ trợ phát triển mô hình học máy trong một hệ thống thống nhất.

6. Những hạn chế, thiếu sót của luận văn về nội dung, hình thức và câu hỏi:

Các nội dung về thử nghiệm, đánh giá liên quan đến xử lý dữ liệu lớn; phân tích dữ liệu cần trình bày rõ ràng, chi tiết hơn.

Luận văn còn nhiều lỗi câu chữ, lỗi soạn thảo.

7. Nếu tác giả chưa viết bài báo khoa học thì nội dung của luận văn có thể được viết thành các bài báo để gửi đăng trên trên tạp chí khoa học, sách chuyên ngành hoặc tuyển tập công trình hội nghị khoa học cấp quốc gia, quốc tế hay không?

.....

.....

.....

8. Kết luận chung (khẳng định mức độ đáp ứng các yêu cầu đối với một luận văn Thạc sĩ; luận văn có thể đưa ra bảo vệ để nhận học vị Thạc sĩ được hay không?):

Nội dung của luận văn đáp ứng các yêu cầu đối với một luận văn Thạc sĩ; Đề nghị cơ sở đào tạo cho phép học viện được bảo vệ luận văn trước Hội đồng để nhận học vị Thạc sĩ.

Hà Nội, ngày 8 tháng 10 năm 2024

Người nhận xét

(Ký, ghi rõ họ tên)

Nguyễn Mạnh Hùng

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT LUẬN VĂN THẠC SĨ

Họ và tên người nhận xét: Nguyễn Long Giang Học hàm, học vị: PGS.TS.
Chức danh trong Hội đồng: Ủy viên
Cơ quan công tác: Viện Công nghệ thông tin, Viện Hàn lâm KHCN VN
Họ và tên học viên: Dương Đình Thiệu
Tên đề tài: **Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia**

Ngành: Hệ thống thông tin Mã số: 8 48 01 04

NỘI DUNG NHẬN XÉT

1. Tính cấp thiết, tính thời sự, ý nghĩa khoa học và thực tiễn của đề tài luận văn:

Luận văn nghiên cứu giải pháp tích hợp các nền tảng mã phục vụ phân tích dữ liệu, học máy như MLFlow, Spark, Jupyter Hub trên nền tảng Opensciencevn. Chưa thấy ý nghĩa khoa học

2. Sự không trùng lặp của đề tài nghiên cứu so với các công trình khoa học, luận văn đã công bố ở trong và ngoài nước; tính trung thực, rõ ràng và đầy đủ trong trích dẫn tài liệu tham khảo:

Luận văn không trùng lặp với các luận văn, công trình công bố trong và ngoài nước

Tài liệu trích dẫn rõ ràng, đầy đủ

3. Sự phù hợp giữa tên đề tài với nội dung nghiên cứu cũng như với chuyên ngành và mã số đào tạo:

Tên đề tài phù hợp với nội dung nghiên cứu và phù hợp với chuyên ngành và mã số đào tạo

4. Độ tin cậy và tính hiện đại của phương pháp nghiên cứu đã sử dụng để hoàn thành luận văn:

Phương pháp tích hợp các nền tảng là luận đạt, bổ sung các nền tảng công nghệ mới

- Có ý nghĩa về công nghệ sử dụng.....
.....

5. Kết quả nghiên cứu của luận văn:

.....Tích hợp các nền tảng người mới như N.L.P, spark, Kubeflow trên nền tảng Openscience v.v.....
.....Thử nghiệm thành công các nền tảng tích hợp

6. Những hạn chế, thiếu sót của luận văn về nội dung, hình thức và các câu hỏi

.....Rà soát chưa mới về tính hay, trích dẫn tài liệu tham khảo.....
.....Nên bổ sung các sơ đồ luồng (pipeline) thay vì mô tả chi tiết các mã lệnh.....
.....Chưa lại và mô tả rõ ràng các đóng góp của học viên

7. Nếu tác giả chưa viết bài báo khoa học thì nội dung của luận văn có thể được viết thành các bài báo để gửi đăng trên trên tạp chí khoa học, sách chuyên ngành hoặc tuyển tập công trình hội nghị khoa học cấp quốc gia, quốc tế hay không?

.....Không.....
.....
.....

8. Kết luận chung (khẳng định mức độ đáp ứng các yêu cầu đối với một luận văn Thạc sĩ; luận văn có thể đưa ra bảo vệ để nhận học vị Thạc sĩ được hay không?):

.....Luận văn đáp ứng yêu cầu đối với luận văn ThS
.....Luận văn có thể bảo vệ trước Hội đồng.....
.....

Hà Nội, ngày 07 tháng 10 năm 2024

Người nhận xét
(Ký, ghi rõ họ tên)

PGS.TS. Nguyễn Long Giang

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT LUẬN VĂN THẠC SĨ

Họ và tên người nhận xét: Trần Đức Nghĩa. Học hàm, học vị: TS.

Chức danh trong Hội đồng: Thư ký – Ủy viên.

Cơ quan công tác: Viện Công nghệ thông tin, Viện Hàn lâm KHCN VN.

Họ và tên học viên: Dương Đình Thiệu.

Tên đề tài: Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia.

Ngành: Hệ thống thông tin. Mã số: 8480104.

NỘI DUNG NHẬN XÉT

1. Tính cấp thiết, tính thời sự, ý nghĩa khoa học và thực tiễn của đề tài luận văn:

Có tính cấp thiết, ý nghĩa khoa học

2. Sự không trùng lặp của đề tài nghiên cứu so với các công trình khoa học, luận văn đã công bố ở trong và ngoài nước; tính trung thực, rõ ràng và đầy đủ trong trích dẫn tài liệu tham khảo:

Không trùng lặp

3. Sự phù hợp giữa tên đề tài với nội dung nghiên cứu cũng như với chuyên ngành và mã số đào tạo:

phù hợp

4. Độ tin cậy và tính hiện đại của phương pháp nghiên cứu đã sử dụng để hoàn thành luận văn:

Đáng tin cậy

5. Kết quả nghiên cứu của luận văn:

Phát triển và thử nghiệm thành công giải pháp

6. Những hạn chế, thiếu sót của luận văn về nội dung, hình thức và câu hỏi:

Làm rõ đóng góp với HV làm
Rà soát lỗi chính tả

7. Nếu tác giả chưa viết bài báo khoa học thì nội dung của luận văn có thể được viết thành các bài báo để gửi đăng trên trên tạp chí khoa học, sách chuyên ngành hoặc tuyển tập công trình hội nghị khoa học cấp quốc gia, quốc tế hay không?

Không

8. Kết luận chung (khẳng định mức độ đáp ứng các yêu cầu đối với một luận văn Thạc sĩ; luận văn có thể đưa ra bảo vệ để nhận học vị Thạc sĩ được hay không?):

Đạt yêu
Có thể đưa ra bảo vệ

....., ngày 8 tháng 10 năm 2024

Người nhận xét
(Ký, ghi rõ họ tên)

Trần Đức Nghĩa

VIỆN HÀN LÂM
KHOA HỌC VÀ CÔNG NGHỆ VN
HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Hà Nội, ngày 31 tháng 10 năm 2024

BIÊN BẢN HỌP HỘI ĐỒNG ĐÁNH GIÁ LUẬN VĂN THẠC SĨ

Thực hiện Quyết định số 1115/QĐ-HVKHCN ngày 30/09/2024 của Giám đốc Học viện Khoa học và Công nghệ về việc thành lập Hội đồng đánh giá luận văn thạc sĩ của học viên Dương Đình Thiệu.

Tên đề tài: **Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia**

Ngành: Hệ thống thông tin Mã số: 8 48 01 04

Hôm nay, ngày 31/10/2024 Hội đồng đã họp tại Phòng 1510, Học viện Khoa học và Công nghệ vào lúc 09h00, Hội đồng gồm 05 thành viên:

1. PGS.TS. Nguyễn Việt Anh	Chủ tịch hội đồng
2. TS. Trần Đức Nghĩa	Thư ký - Ủy viên
3. PGS.TS. Bùi Thu Lâm	Phản biện 1
4. TS. Nguyễn Mạnh Hùng	Phản biện 2
5. PGS.TS. Nguyễn Long Giang	Ủy viên

Thành viên vắng mặt: ..không..... (Phản biện hoặc ủy viên, đã có bản nhận xét đồng ý cho phép học viên được bảo vệ trước Hội đồng đánh giá luận văn thạc sĩ).

NỘI DUNG LÀM VIỆC

1. Đại diện cơ sở đào tạo đọc quyết định thành lập Hội đồng đánh giá luận văn
2. Chủ tịch Hội đồng, điều khiển phiên họp
3. Thư ký HĐ, đọc lí lịch khoa học và bảng điểm của học viên
4. Học viên trình bày luận văn trước Hội đồng
5. Phản biện 1:

.....Cần bổ sung nội dung LV, VD chương 1 trình bày 5 trang quá ngắn
Xem lại chương 2 ... chưa đến mức liệu để xuất giải pháp.....
Chương 3 ... Trình bày đánh giá hệ thống là như thế nào? Nên bổ sung KQ

6. Phản biện 2:

Cân bằng phần cứng? Khả năng mở rộng của hệ thống?
Khả năng đáp ứng của hệ thống sau khi mở rộng
Thông số mô hình?

7. Học viên trả lời:

Sẽ bổ sung, tìm hiểu thêm về các tiêu chí đánh giá. Mỗi Module sẽ có 1 tiêu chí đánh giá khác nhau. Tham khảo các tiêu chí đánh giá phần mềm, về thời gian hồi đáp

Có thể mở rộng, mua thuê thêm cloud.

Hiện đang thử nghiệm trên cloud

8. Các thành viên HĐ và những người tham dự nêu câu hỏi

Làm rõ đóng góp của học viên, chỉ đóng góp 1 phần trong hệ thống lớn

9. Học viên trả lời

Sẽ tiếp thu, chỉnh sửa.

10.Hội đồng họp kín và cho điểm

- Hội đồng bầu ban kiểm phiếu gồm 3 thành viên:

Trưởng ban: PGS.TS. Bùi Thu Lâm

Ủy viên: TS. Nguyễn Mạnh Hùng

Ủy viên: TS. Trần Đức Nghĩa

- Kết quả kiểm phiếu như sau:

Số phiếu phát ra: 5

Số phiếu thu về: 5

Tổng số điểm:

Điểm trung bình: ……..

Điểm thưởng công trình công bố:……..

Tổng điểm đánh giá luận văn và thưởng công trình công bố:…7,7……

- Kết luận của Hội đồng:

+ Luận văn …đạt yêu cầu…… (đạt/không đạt yêu cầu)

+ Tính không trùng lặp nội dung và tên đề tài với các công trình công bố:

……L.V. không trùng lặp nội dung và tên đề tài với……

…các công trình công bố khác……

……

11. Chủ tịch Hội đồng, công bố kết quả, yêu cầu học viên chỉnh sửa luận văn với các nội dung sau:

……

……Chỉnh sửa L.V theo góp ý của hội đồng……

……Giải thích rõ hơn 3……

……Làm rõ đóng góp……

……

……

Buổi họp đã kết thúc vào 11 giờ 00 phút ngày 31/10/2024.

Hà Nội, ngày 31 tháng 10 năm 2024

THƯ KÝ HỘI ĐỒNG	**CHỦ TỊCH HỘI ĐỒNG**
TS. Trần Đức Nghĩa	**PGS.TS. Nguyễn Việt Anh**

XÁC NHẬN CỦA HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ
KT.GIÁM ĐỐC
PHÓ GIÁM ĐỐC

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM
HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Thị Trung

VIỆN HÀN LÂM
KHOA HỌC VÀ CÔNG NGHỆ VN
HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BẢN GIẢI TRÌNH CHỈNH SỬA LUẬN VĂN THEO KẾT LUẬN CỦA HỘI ĐỒNG ĐÁNH GIÁ LUẬN VĂN THẠC SĨ

Họ tên học viên: Dương Đình Thiệu Lớp: ITT2022B

Tên đề tài luận văn: Nghiên cứu phát triển các giải pháp tích hợp công cụ thu thập, phân tích dữ liệu trong nền tảng quản lý và chia sẻ dữ liệu nghiên cứu khoa học và công nghệ quốc gia.

Chuyên ngành: Hệ thống thông tin

Mã số: 9.48.01.04

Người hướng dẫn khoa học: PGS.TS. Nguyễn Long Giang

Ngày bảo vệ luận văn: 31/10/2024

Căn cứ biên Bản họp Hội đồng đánh giá luận văn thạc sĩ, học viên đã chỉnh sửa luận văn như sau:

STT	Nội dung đề nghị bổ sung, chỉnh sửa	Nội dung đã bổ sung, chỉnh sửa
1	Viết rõ hơn về nội dung luận văn ở trang số 3	Đã viết lại để làm rõ hơn phần nội dung nghiên cứu trang số 3
2	Chương 1: Nội dung ngắn (5 trang), cần bổ sung miêu tả các nền tảng	Đã bổ sung nội dung các nền tảng. Các trang chỉnh sửa: 7, 8, 9, 10, 11, 12
3	Chương 2: Làm rõ nội dung đóng góp	Đã bổ sung nội dung để làm rõ hơn nội dung đóng góp. Các trang chỉnh sửa: 14, 16, 67
4	Chương 3: Chưa rõ tiêu chí đánh giá hệ thống	Đã bổ sung tiêu chí đánh giá. Các trang chỉnh sửa: 56, 59, 63

5	Sửa lỗi chính tả, soạn thảo	Đã sửa lỗi chính tả, soạn thảo. Các trang chỉnh sửa: 31, 44, 45, 51, 58

Lưu ý: Trong trường hợp Hội đồng yêu cầu xin ý kiến của 02 phản biện sau bảo vệ, học viên cần xin chữ ký của 02 phản biện xác nhận.

Hà Nội, ngày 13 tháng 11 năm 2024.

CHỦ TỊCH HỘI ĐỒNG | **TẬP THỂ HƯỚNG DẪN** | **HỌC VIÊN**

Nguyễn Việt Anh | Nguyễn Long Giang | Dương Đình Thuận

XÁC NHẬN CỦA CƠ SỞ ĐÀO TẠO

KT. GIÁM ĐỐC
PHÓ GIÁM ĐỐC

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM
HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Thị Trung