|  |  |
| --- | --- |
|  |  |
| **NGUYỄN HOÀNG LONG** | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC** |
| **NGÀNH CÔNG NGHỆ THÔNG TIN** |
| **ĐỂ TÀI XÂY DỰNG HỆ THỐNG XỬ LÝ**  **DỮ LIỆU REAL-TIME SỬ DỤNG CONFLUENT PLATFORM** |
|  |
|  |
| **CBHD :TS. Phạm Văn Hà** |
| **CÔNG NGHỆ THÔNG TIN** | **Sinh viên : Nguyễn Hoàng Long** |
| **Mã sinh viên : 2020600528** |
|  |
|  |
|  |
| **Hà Nội – Năm 2024** |

# LỜI CẢM ƠN

Sau 4 năm học tập và rèn luyện tại Trường Đại học Công Nghiệp Hà Nội, đồ án tốt nghiệp này là một dấu ấn quan trọng đánh dấu việc em - một sinh viên đã hoàn thành nhiệm vụ của mình trên ghế giảng đường Đại học. Đầu tiên, với tình cảm sâu sắc, chân thành, cho phép em được bày tỏ lòng biết ơn sâu sắc đến gia đình, các anh chị, bạn bè đã luôn động viên, khích lệ, tạo điều kiện giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài.

Em cũng xin gửi tới các thầy các cô khoa Công Nghệ Thông Tin Trường Đại học Công Nghiệp Hà Nội lời chào trân trọng, lời chúc sức khỏe và lời cảm ơn sâu sắc. Với sự quan tâm, dạy dỗ, chỉ bảo tận tình chu đáo của thầy cô, đến nay em đã có thể hoàn thành đồ án tốt nghiệp, đề tài: “Xây dựng hệ thống xử lý dữ liệu real-time sử dụng Confluent Platform”.

Đặc biệt em xin gửi lời cảm ơn chân thành đến thầy TS. Phạm Văn Hà người đã tận tình giúp và hướng dẫn em hoàn thành đề tài đồ án.

Đồng thời, em xin bày tỏ lòng biết ơn đến lãnh đạo Trường Đại học Công Nghiệp Hà Nội, các Khoa, Phòng ban chức năng đã tạo điều kiện cho em được học tập tại nơi mà em yêu thích, cho em bước vào đời sống thực tế và áp dụng những kiến thức em đã học tại trường và môi trường làm việc mới của em. Qua quá trình học tập em đã tích lũy được rất nhiều kiến thức để chuẩn bị cho công việc sau này cũng như để phát triển thêm bản thân.

Trong quá trình hoàn thành đồ án tốt nghiệp không thể tránh khỏi thiếu sót, kính mong có sự góp ý từ thầy cô.

*Em xin chân thành cảm ơn!*

Nguyễn Hoàng Long.

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc167743545)

[DANH MỤC BẢNG 6](#_Toc167743546)

[DANH MỤC HÌNH VẼ 7](#_Toc167743547)

[MỞ ĐẦU 10](#_Toc167743548)

[1. Tên đề tài 10](#_Toc167743549)

[2. Lý do chọn đề tài 10](#_Toc167743550)

[3. Mục tiêu của đề tài 11](#_Toc167743551)

[4. Phạm vi đề tài 11](#_Toc167743552)

[5. Bố cục của đề tài 11](#_Toc167743553)

[CHƯƠNG 1: TỔNG QUAN CÔNG NGHỆ 12](#_Toc167743554)

[1.1 Confluent Platform (Enterprise) 12](#_Toc167743555)

[1.1.1 Tại sao doanh nghiệp, tổ chức lại cần Confluent Platform 12](#_Toc167743556)

[1.1.2 Tổng quan Confluent Platform 14](#_Toc167743557)

[1.1.3 Ưu và nhược điểm của Confluent Platform 21](#_Toc167743558)

[1.2 Tổng quan MinIO 22](#_Toc167743559)

[1.2.1 Lý do tích hợp MinIO vào hệ thống Confluent Platform 22](#_Toc167743560)

[1.2.2 Giới thiệu MinIO 22](#_Toc167743561)

[1.3 Tổng quan Grafana 23](#_Toc167743562)

[1.3.1 Lý do tích hợp Grafana vào hệ thống Confluent Platform 23](#_Toc167743563)

[1.3.2 Giới thiệu Grafana 23](#_Toc167743564)

[1.4 Tổng kết Chương 1 23](#_Toc167743565)

[CHƯƠNG 2: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG XỬ LÝ DỮ LIỆU REAL-TIME CONFLUENT PLATFORM 24](#_Toc167743566)

[2.1 Khảo sát yêu cầu hệ thống 24](#_Toc167743567)

[2.2 Phân tích yêu cầu hệ thống 25](#_Toc167743568)

[2.3 Mô hình tổng quan hệ thống streaming, xử lý dữ liệu real-time Confluent Platform 28](#_Toc167743569)

[2.4 Tổng kết chương 2: 29](#_Toc167743570)

[CHƯƠNG 3: TRIỂN KHAI CÀI ĐẶT HỆ THỐNG XỬ LÝ DỮ LIỆU REAL-TIME CONFLUENT PLATFORM. 30](#_Toc167743571)

[3.1 Thông tin cài đặt 30](#_Toc167743572)

[3.1.1 Thông tin máy chủ 30](#_Toc167743573)

[3.1.2 Thông tin kết nối 30](#_Toc167743574)

[3.1.3 Các gói cài đặt 31](#_Toc167743575)

[3.1.4 Tổng quan quá trình cài đặt hệ thống Confluent Platform 32](#_Toc167743576)

[3.2 Triển khai cài đặt hệ thống Confluent Platform 33](#_Toc167743577)

[3.2.1 Cài đặt package 33](#_Toc167743578)

[3.2.2 Kiểm tra java version 33](#_Toc167743579)

[3.2.3 Khai báo biến môi trường vào file .bash\_profile 34](#_Toc167743580)

[3.2.4 Cấu hình cho Confluent Zookeeper 36](#_Toc167743581)

[3.2.5 Cấu hình cho Confluent Server 40](#_Toc167743582)

[3.2.6 Cấu hình Schema Registry 52](#_Toc167743583)

[3.2.7 Cấu hình Kafka Connect 57](#_Toc167743584)

[3.2.8 Cấu hình KsqlDB 64](#_Toc167743585)

[3.2.9 Cấu hình Control Center 70](#_Toc167743586)

[3.3 Tích hợp MinIO 77](#_Toc167743587)

[3.3.1 Thông tin kết nối máy chủ MinIO 77](#_Toc167743588)

[3.3.2 Quy trình triển khai tích hợp MinIO 77](#_Toc167743589)

[3.4 Tích hợp Prometheus và Grafana 79](#_Toc167743590)

[3.4.1 Triển khai Prometheus 79](#_Toc167743591)

[3.4.2 Triển khai Grafana: 82](#_Toc167743592)

[3.5 Tạo luồng Streaming và xử lý dữ liệu 83](#_Toc167743593)

[3.5.1 Mô hình luồng streaming 83](#_Toc167743594)

[3.5.2 Mô tả cách hoạt động của luồng 83](#_Toc167743595)

[3.5.3 Giải thích cơ chế hoạt động của Confluent Connector 84](#_Toc167743596)

[3.5.4 Các bước thực hiện tạo luồng Streaming và xử lý dữ liệu. 85](#_Toc167743597)

[3.6 Confluent Platform xử lý dữ liệu: 94](#_Toc167743598)

[3.7 Tổng kết chương 3 97](#_Toc167743599)

[CHƯƠNG 4: KẾT QUẢ VÀ KIỂM THỬ 98](#_Toc167743600)

[4.1 Kết quả đạt được 98](#_Toc167743601)

[4.1.1 Đối với hệ thống Confluent Platform 98](#_Toc167743602)

[4.1.2 Giao diện website của Confluent Control Center: 99](#_Toc167743603)

[4.1.3 Công nghệ đã tích hợp thành công: 103](#_Toc167743604)

[4.2 Kiểm thử hệ thống: 106](#_Toc167743605)

[4.3 Kiểm thử luồng streaming và xử lý dữ liệu 107](#_Toc167743606)

[4.3.1 Tiêu chí 107](#_Toc167743607)

[4.3.2 Câu lệnh truy vấn ở cơ sở dữ liệu nguồn EKA (Oracle) 107](#_Toc167743608)

[4.3.3 Kết quả dữ liệu ở cơ sở dữ liệu đích LONGCV (Oracle) 109](#_Toc167743609)

[4.4 Tổng kết chương 4 112](#_Toc167743610)

[KẾT LUẬN 113](#_Toc167743611)

[TÀI LIỆU THAM KHẢO 114](#_Toc167743612)

**DANH MỤC CÁC CHỮ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| STT | Từ viết tắt | Giải thích |
| 1 | Admin | Quản trị viên |
| 2 | Database | Cơ sở dữ liệu |
| 3 | Connector | Trình kết nối |
| 4 | WebUI | Giao diện website |
| 5 | Log | Nhật ký |
| 6 | Cloud | Lưu trữ đám mây |
| 7 | Produce | Cung cấp, sản xuất |
| 8 | Consume | Tiêu thụ, sử dụng |
| 9 | Message | Gói dữ liệu |
| 10 | EnterPrise | Bản dùng cho doanh nghiệp |
| 11 | Open source | Phần mềm mã nguồn mở |
| 12 | Data Source | Dữ liệu ở hệ thống nguồn |
| 13 | Data Sink | Dữ liệu ở hệ thống đích |

# DANH MỤC BẢNG

[Bảng 3. 1 Bảng thông tin máy chủ 30](#_Toc167669491)

[Bảng 3. 2 Gói cài đặt Conflfunet 7.6 31](#_Toc167669492)

[Bảng 3. 3 Bảng khởi chạy ConfluentZookeeper 39](#_Toc167669493)

[Bảng 3. 4 Khởi chạy server 51](#_Toc167669494)

[Bảng 3. 5 Bảng khởi động Confluent Schema Registry 56](#_Toc167669495)

[Bảng 3. 6 Bảng khởi chạy Confluent Kafka Connect 64](#_Toc167669496)

[Bảng 3. 7 Bảng khởi chạy KsqlDB 69](#_Toc167669497)

[Bảng 3. 8 Bảng khởi chạy Confluent Control Center 75](#_Toc167669498)

[Bảng 4. 1 Bảng kiểm thử hệ thống 106](#_Toc167669499)

[Bảng 4. 2 Bảng tính toán độ trễ thời gian streaming 109](#_Toc167669500)

# DANH MỤC HÌNH VẼ

[Hình 1. 1 Các luồng dữ liệu chồng chéo 12](#_Toc167669547)

[Hình 1. 2 Confluent xử lý luồng dữ liệu 12](#_Toc167669548)

[Hình 1. 3 Nguồn document của Confluent Platform 13](#_Toc167669549)

[Hình 1. 4 Kiến trúc cơ bản của Apache Kafka 13](#_Toc167669550)

[Hình 1. 5 Cấu trúc của Kafka Message 14](#_Toc167669551)

[Hình 1. 6 Mô hình publish Kafka Producer 14](#_Toc167669552)

[Hình 1. 7 Mô hình subcribe của Kafka Consumer 15](#_Toc167669553)

[Hình 1. 8 Topic chia thành 3 Partitions 16](#_Toc167669554)

[Hình 1. 9 Kafka replication 17](#_Toc167669555)

[Hình 2. 1 Thành phần Confluent cung cấp 24](#_Toc167669556)

[Hình 2. 2 Đề xuất các thành phần trong hệ thống Confluent Platfrom 26](#_Toc167669557)

[Hình 2. 3 Công nghệ tích hợp đề xuất 26](#_Toc167669558)

[Hình 2. 4 Mô hình Confluent Platform 27](#_Toc167669559)

[Hình 3. 1 Kết quả kiểm tra version java 33](#_Toc167669560)

[Hình 3. 2 Chọn version java 34](#_Toc167669561)

[Hình 3. 3 Kết quả cấu hình file log4j 37](#_Toc167669562)

[Hình 3. 4 Trạng thái service Zookeeper 40](#_Toc167669563)

[Hình 3. 5 Kiểm tra quyền cho thư mục Kafka 40](#_Toc167669564)

[Hình 3. 6 Kiểm tra quyền thư mục certs của broker 42](#_Toc167669565)

[Hình 3. 7 Trạng thái service Server 52](#_Toc167669566)

[Hình 3. 8 Kiểm tra quyền thư mục schema 52](#_Toc167669567)

[Hình 3. 9 Kiểm tra quyền thư mục certs của schema registry 54](#_Toc167669568)

[Hình 3. 10 Trạng thái bật thành công service Schema Registry 57](#_Toc167669569)

[Hình 3. 11 Kiểm tra quyền thư mục connect 57](#_Toc167669570)

[Hình 3. 12 Kiểm tra quyền thư mục certs của kafka connect 59](#_Toc167669571)

[Hình 3. 13 Trạng thái bật thành công service Kafka Connect 64](#_Toc167669572)

[Hình 3. 14 Kiểm tra quyền thư mục ksqldb 65](#_Toc167669573)

[Hình 3. 15 Kiểm tra quyền thư mục certs của ksqldb 66](#_Toc167669574)

[Hình 3. 16 Trạng thái bật thành công service KsqlDB 69](#_Toc167669575)

[Hình 3. 17 Kiểm tra quyền thư mục control-center 70](#_Toc167669576)

[Hình 3. 18 Kiểm tra quyền thư mục certs của control center 71](#_Toc167669577)

[Hình 3. 19 Trạng thái bật thành công service Control Center 76](#_Toc167669578)

[Hình 3. 20 Trạng thái bật thành công của service Prometheus 81](#_Toc167669579)

[Hình 3. 21 Trạng thái bật thành công của service Grafana 82](#_Toc167669580)

[Hình 3. 22 Mô hình luồng stream 83](#_Toc167669581)

[Hình 3. 23 Giao diện tạo source connector 86](#_Toc167669582)

[Hình 3. 24 Giao diện trạng thái của connector trên Control Center 89](#_Toc167669583)

[Hình 3. 25 Create stream TRANSACTIONS\_STREAM 90](#_Toc167669584)

[Hình 3. 26 Create stream TRANSACTIONS\_STREAM\_SINK 90](#_Toc167669585)

[Hình 3. 27 Giao diện tạo connector trên Control Center 91](#_Toc167669586)

[Hình 3. 28 Giao diện trạng thái của connector trên Control Center 93](#_Toc167669587)

[Hình 3. 29 Dữ liệu trong stream TRANSACTIONS\_STREAM 95](#_Toc167669588)

[Hình 3. 30 Dữ liệu trong stream TRANSACTIONS\_SINK\_STREAM 95](#_Toc167669589)

[Hình 4. 1 Trạng thái service Zookeeper 98](#_Toc167669590)

[Hình 4. 2 Trạng thái service Server 98](#_Toc167669591)

[Hình 4. 3 Trạng thái service Schema Registry 98](#_Toc167669592)

[Hình 4. 4 Trạng thái service Kafka Connect 99](#_Toc167669593)

[Hình 4. 5 Trạng thái service KsqlDB 99](#_Toc167669594)

[Hình 4. 6 Trạng thái service Control Center 99](#_Toc167669595)

[Hình 4. 7 Giao diện đăng nhập Control Center 100](#_Toc167669596)

[Hình 4. 8 Giao diện tính năng Health+ 100](#_Toc167669597)

[Hình 4. 9 Giao diện Overview của Cluster 101](#_Toc167669598)

[Hình 4. 10 Giao diện overrview Broker 101](#_Toc167669599)

[Hình 4. 11 Giao diện Topics 102](#_Toc167669600)

[Hình 4. 12 Giao diện Kafka Connect 102](#_Toc167669601)

[Hình 4. 13 Giao diện KsqlDB 103](#_Toc167669602)

[Hình 4. 14 Giao diện đăng nhập MinIO 103](#_Toc167669603)

[Hình 4. 15 Giao diện chính MinIO 104](#_Toc167669604)

[Hình 4. 16 Giao diện Control Center đã tích hợp MinIO 104](#_Toc167669605)

[Hình 4. 17 Giao diện Grafana 105](#_Toc167669606)

[Hình 4. 18 Giao diện Grafana 105](#_Toc167669607)

[Hình 4. 19 Kết quả dữ liệu ở LONGCV 109](#_Toc167669608)

# MỞ ĐẦU

## 1. Tên đề tài

Đề tài “**Xây dựng hệ thống xử lý dữ liệu real-time sử dụng Confluent Platform**”.

## 2. Lý do chọn đề tài

Trong thời đại công nghệ 4.0 hiện nay, dữ liệu thời gian thực (real-time) có giá trị đặc biệt quan trong đối với các doanh nghiệp và tổ chức. Các doanh nghiệp sử dụng dữ liệu real-time để cải thiện dịch vụ khách hàng, quản lý sản phẩm và tối ưu hóa vận hành, theo dõi và duy trì cơ sở hạ tầng công nghệ thông tin. Nhận biết được những lợi ích trên, nhu cầu thu thập dữ liệu real-time ngày càng tăng ở khắp các doanh nghiệp trên thế giới. Chính vì thế, những công cụ có khả năng thu thập, lưu trữ, xử lý dữ liệu thời gian thực ngày càng trở nên được "săn đón", Confluent chính là một trong những công cụ đó. Hơn nữa Confluent đóng vai trò là trung gian kết nối các hệ thống nguồn và hệ thống đích giúp phân luồng dữ liệu, xử lý các định dạng dữ liệu không đồng nhất từ nhiều nguồn khác nhau như XML, CSV, JSON,... hay các giao thức truyền tải dữ liệu TCP, HTTP... Nhìn thấy Confluent Platform mang lại nhiều lợi ích cho doanh nghiệp và tổ chức , nên đã thúc đẩy em tìm hiểu, nghiên cứu và chọn công nghệ Confluent Platform với mục tiêu xây dựng hệ thống xử lý dữ liệu real-time làm đề tài cho đồ án tốt nghiệp của mình.

Đề tài này có tính ứng dụng cao, gần gữi với thực tế, việc xây dựng một hệ thống xử lý dữ liệu real-time sử dụng Confluent không chỉ là một dự án thực hành mà còn là cơ hội giúp em rèn luyện kĩ năng làm việc với hệ điều hành Linux, phân tích, thiết kế, xây dựng triển khai một hệ thống tạo tiền đề định hướng cho công việc tương lai sau khi ra trường.

## 3. Mục tiêu của đề tài

* Phân tích, thiết kế, triển khai cài đặt hệ thống Confluent Platform
* Xây dựng luồng streaming xử lý dữ liệu real-time giữa hai cơ sở dữ liệu.
* Tích hợp công nghệ MinIO, Grafana.
* Rèn luyện kĩ năng làm việc với hệ điều hành Linux.

## 4. Phạm vi đề tài

Phạm vi thời gian: Đồ án này được thực hiện từ ngày 18/03/2024 đến ngày 18/05/2024.

Phạm vi nội dung: Xây dựng hệ thống xử lý dữ liệu real-time sử dụng Confluent Platform. Tích hợp công nghệ MinIO, Grafana.

## 5. Bố cục của đề tài

Trong báo cáo đồ án này, cấu trúc được chia thành 3 chương:

Chương 1: Trình bày lý do doanh nghiệp nên sử dụng Confluent Platform. Tổng quan các công nghệ sử dụng MinIO, Grafana và lý do tích hợp vào hệ thống này.

Chương 2: Phân tích, thiết kế, hệ thống xử lý dữ liệu Confluent Platform.

Chương 3: Triển khai cài đặt, tích hợp công nghệ cho hệ thống xử lý dữ liệu Confluent Platform.

Chương 4: Trình bày các kết quả đạt được, kiểm thử hệ thống, kiểm thử luồng streaming và kết luận.

# CHƯƠNG 1: TỔNG QUAN CÔNG NGHỆ

* 1. **Confluent Platform (Enterprise)**

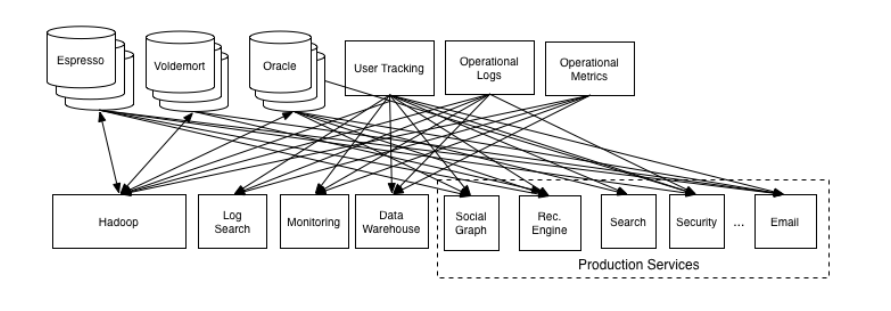
### 1.1.1 Tại sao doanh nghiệp, tổ chức lại cần Confluent Platform

Thứ nhất, doanh nghiệp, tổ chức yêu cầu tính thời gian thực của dữ liệu chẳng hạn như các lĩnh vực:

* Dịch vụ tài chính: Thanh toán thời gian thực, giao dịch chứng khoán, cập nhật giá thị trường cổ phiếu...
* Bán lẻ, thương mại điện tử: Quản lý chuỗi cung ứng theo thời gian thực (từ nhà máy, nhà kho, phân phối đến giao hàng), thu thập dữ liệu tìm kiếm đề xuất xu hướng bán hàng.
* IoT (Internet of Things): Thu thập dữ liệu theo thời thực từ cảm biến, dây chuyền sản xuất… đưa dự đoán, bảo trì.

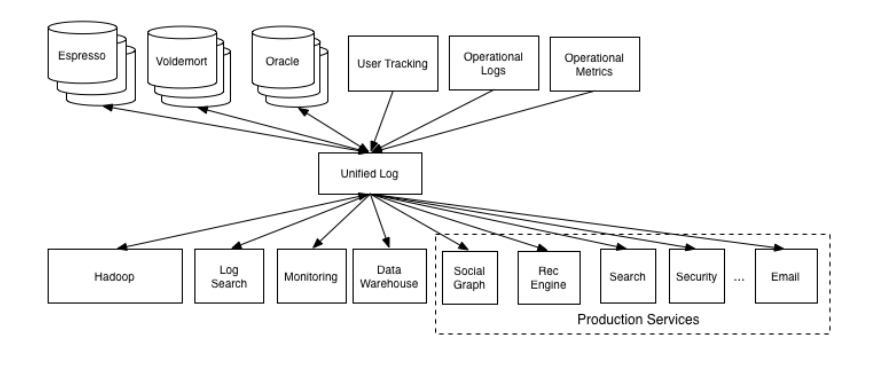
Thứ hai, doanh nghiệp, tổ chức có rất nhiều hệ thống dữ liệu và kho dữ liệu việc xây dựng luồng kết nối chúng khiến hệ thống trở nên rất phức tạp. Hơn nữa, dữ liệu thường chảy theo hai hướng vì nhiều hệ thống vừa là nguồn, vừa là đích để truyền dữ liệu nên khiến các luồng chồng chéo rất khó quản lý.

Thứ ba, nhiều nguồn dữ liệu đồng nghĩa với việc nhiều định dạng khác nhau như XML, JSON, CSV... và các giao thức truyển tải dữ liệu (TCP – Transmission Control Protocol, HTTP – Hypertext Transfer Protocol…) để xử lý chúng cũng là một thử thách đối với doanh nghiệp, tổ chức.



Hình 1. 1 Các luồng dữ liệu chồng chéo

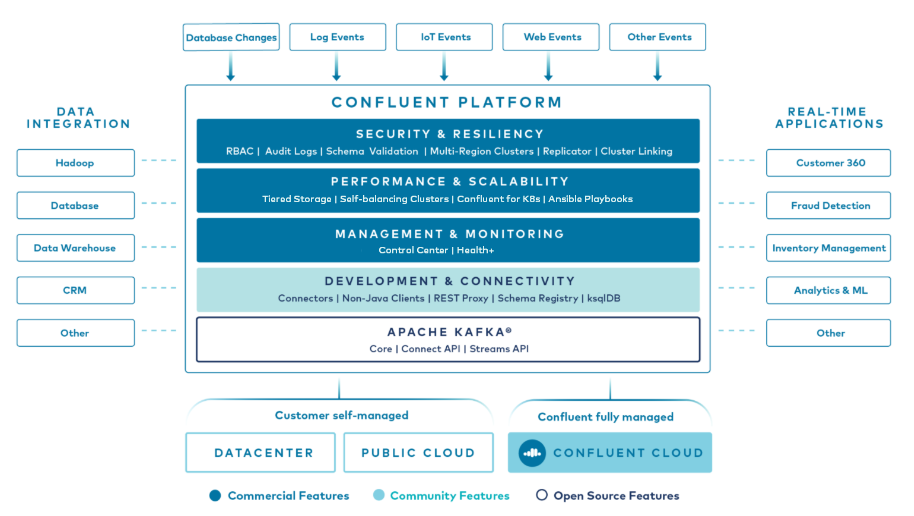
Confluent Platform ra đời giải quyết toàn bộ vấn đề nêu trên mà doanh nghiệp, tổ chức gặp phải. Confluent Platform cho phép xây dựng các luồng streaming, có thể thu thập, lưu trữ, xử lý dữ liệu thời gian thực, làm trung gian tách riêng luồng dữ liệu nguồn và hệ thống đích cung cấp tính năng xử lý các định đạng dữ liệu khác nhau, các hệ thống dữ liệu nguồn gửi dữ liệu lên Confluent, các hệ thống dữ liệu đích sẽ lấy dữ liệu từ Confluent.



Hình 1. 2 Confluent xử lý luồng dữ liệu

### 1.1.2 Tổng quan Confluent Platform

Confluent Platform là một nền tảng streaming phân tán được xây dựng dựa trên Apache Kafka, Confluent Platform là bản hoàn thiện, nâng cấp của Kafka dành cho doanh nghiệp với tất cả các tính năng của Apache Kafka và các tính năng độc quyền được thiết kế để giúp tăng tốc phát triển và kết nối ứng dụng.



Hình 1. 3 Nguồn document của Confluent Platform

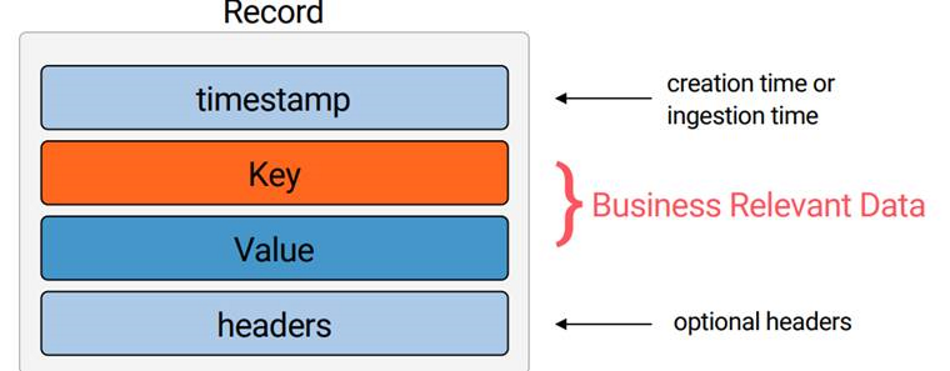
**1.1.2.1 Các thành phần chính của Apache Kafka**



Hình 1. 4 Kiến trúc cơ bản của Apache Kafka

1. Kafka Event

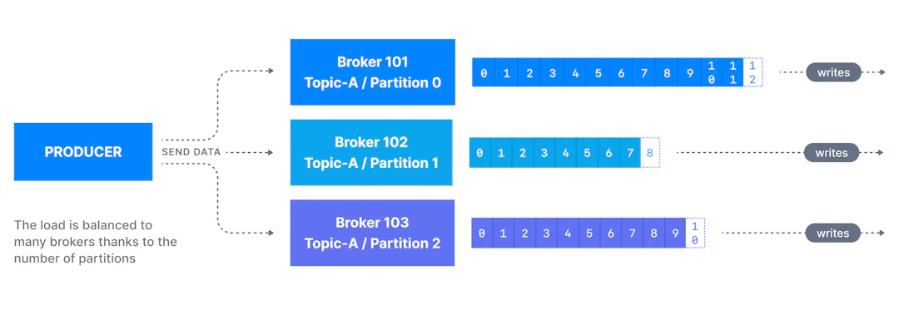
Một Kafka Event (sự kiện) ghi lại thực tế “điều gì đó đang xảy ra” trong hệ thống của doanh nghiệp. Nó còn được gọi là Record (bản ghi) hoặc Message (thông điệp) 3 thuật ngữ này đều mang nghĩa tương đương nhau.



Hình 1. 5 Cấu trúc của Kafka Message

2. Kafka Producer

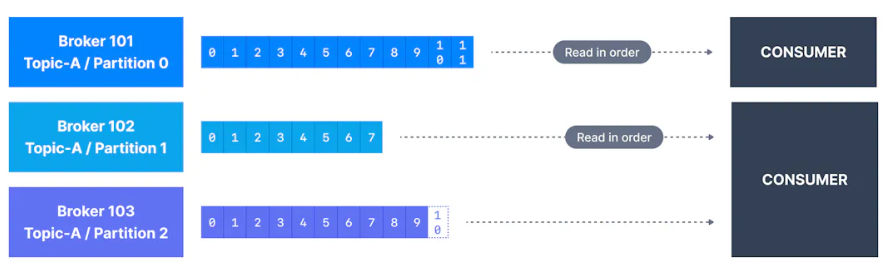
Kafka Producer là những ứng dụng publish – xuất bản Message tới một Kafka Topic được định dạng gửi đi dưới dạng mảng Byte tới Kafka Broker.



Hình 1. 6 Mô hình publish Kafka Producer

3. Kafka Consumer

Các ứng dụng subcribe (đăng ký) một hoặc nhiều Kafka Topic được gọi là Consumer. Kafka Consumer có thể đọc từ một đến nhiều Partition cùng một lúc và dữ liệu được đọc theo thứ tự từ thấp đến cao của mỗi phân vùng, không thể đọc ngược lại vì dó cách triển khai của Apache Kafka và Consumer.



Hình 1. 7 Mô hình subcribe của Kafka Consumer

4. Kafka Cluster

Kafka Cluster là một nhóm các máy chủ (server) Kafka Broker hoạt động cùng nhau để lưu trữ và xử lý dữ liệu. Mỗi Cluster có ít nhất một Kafka Broker.

5. Kafka Broker

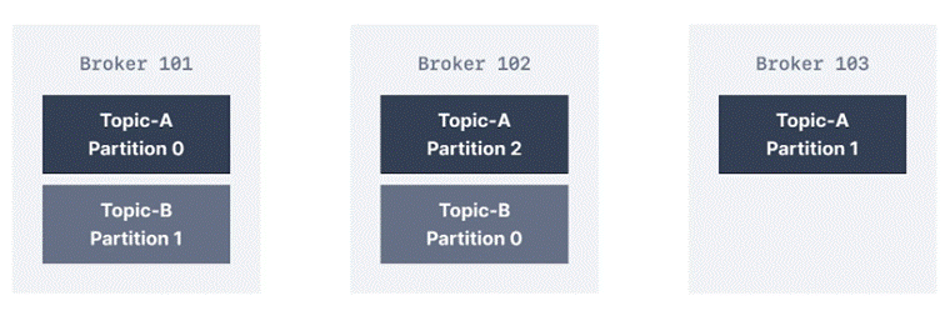
Một Kafka Broker là một server trong Kafka, đóng vai trò là cầu nối giữa Producer và Consumer để hai thành phần này có thể trao đổi Message với nhau.

6. Kafka Topic

Cụm Kafka tổ chức và lưu trữ Message theo các danh mục được gọi là Topic. Một Topics là một log (nhật ký) khi một Message mới được ghi vào một Topic, Message sẽ được ghi vào cuối của log theo thứ tự và không thể chỉnh sửa.

7. Kafka Partitions

Topic được chia thành một hoặc nhiều Partitions (phân vùng), là đơn vị cơ bản của tính song song và phân phối trong Kafka. Bằng cách này việc lưu trữ, produce Message, consume Message có sẵn được phân chia trên nhiều máy chủ trong cụm. Việc phân tán tăng khả năng mở rộng, giúp thao tác đọc ghi dữ liệu từ nhiều ứng dụng thực hiện cùng cùng một lúc.

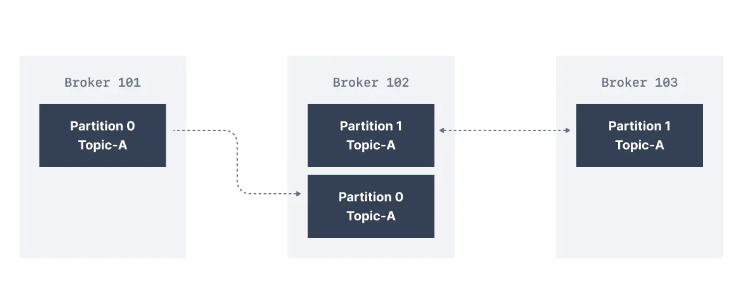


Hình 1. 8 Topic chia thành 3 Partitions

8. Kafka Replication

Một trong những lí do chính khiến Kafka trở nên nối tiếng là khả năng phục hồi - Recovery. Kafka được thiết kế với tính năng sao chép là tính năng cốt lõi để chống lại những lỗi, duy trì thời gian hoạt động và độ chính xác dữ liệu.

Kafka đảm bảo độ bền của dữ liệu bằng cách sao chép dữ liệu (replication) trên nhiều Brokers. Mỗi Partition có thể có một hoặc nhiều replica (bản sao) trên các Broker khác nhau, ngăn ngừa mất dữ liệu trong trường hợp Broker lỗi.



Hình 1. 9 Kafka replication

9. Zookeeper

Zookeeper là một dịch vụ điều phối phân tán mã nguồn mở thuộc Apache Software Foundation, được sử dụng để quản lý Kafka Broker, Kafka Topic, Metadata, điều phối các Consumer, gửi thông báo đến Kafka trong trường hợp có thay đổi. (Kafka Broker lỗi, có Topic mới,...).

#### 1.1.2.2 Tính năng của Apache Kafka

Kafka cung cấp tính năng bảo mật như mã hõa dữ liệu giữa Producer, Consumer và Broker sử dụng SSL/TLS (Secure Socket Layout/ Transport Layer Secure) và ủy quyền truy cập ACLs( Access Control Lists).

Kafka cung cấp Java APIs:

* Producer API cho phép ứng dụng gửi Message tới Kafka.
* Consumer API cho phép các ứng dụng đăng ký đọc một hoặc nhiều Topic.
* Kafka Connect - một thành phần dùng để truyền dữ liệu giữa Kafka và các hệ thống dữ liệu khác.
* AdminAPI cung cấp khả năng tạo, xóa, xem chi tiết, quản lý Topics, Brokers, ACLs và mốt số đối tượng Kafka khác.

#### 1.1.2.3 Tính năng kết nối và phát triển của Confluent

* Confluent Connectors: Tận dụng Kafka Connect API kết nối tới các hệ thống như database, hệ thống file. Confluent Hub cho phép tải các Connectors cho các Data Sources và Data Sinks phổ biến.
* REST Proxy: tích hợp RESTfull HTTP service cho Kafka Cluster, giúp Admin API có thể dễ dàng làm việc với Kafka từ mọi ngôn ngữ. Hỗ trợ tất cả các chức năng cơ bản của Admin: tạo Message, đọc Message,... REST Proxy cũng tích hợp với Schema Registry, hỗ trợ tự động dịch dữ liệu JSON sang Avro.
* Confluent Schema Registry: là một hệ thống giúp quản lý và xác thực các định dạng Schema – Cấu trúc dữ liệu của Message. Khi các dịch vụ khác tương tác với nhau, chúng cần đồng bộ 1 định dạng chung để truyền nhận thông tin (Một số định dạng cấu trúc dữ liệu phổ biến trong Confluent hỗ trợ có thể kể đến JSON, Avro, Protobuf), Schema Registry giúp tạo ra một kho lưu trữ các định dạng này.
* Confluent KsqlDB: Xây dựng dựa trên ngôn ngữ SQL. Nó cung cấp một giao diện SQL thực hiện tương tác xử lý dữ liệu, các luồng trên Kafka.

#### 1.1.2.4 Tính năng quản lý và giám sát của Confluent

Confluent Platform cung cấp một số tính năng bổ sung Admin API:

* Confluent Control Center: Một giao diện web giúp quản lý và giám sát Kafka. Dễ dàng quản lý Kafka Connnect: tạo, sửa, xóa kết nối tới các hệ thống khác. Control Center cho phép giám sát luồng dữ liệu từ Producer tới Consumer đảm bảo mọi Message đều được gửi và đo lường thời gian gửi tin nhắn.
* Health+: Được tích hợp với Control Center, công cụ giúp đảm bảo tình trạng hoạt động, cảnh báo thông minh, hỗ trợ chủ động phát hiện lỗi cho cụm Kafka.
* Metrics reporter: Thu thập các số liệu khác nhau từ cụm Kafka như thông lượng, độ trễ, thông số của hệ thống…

#### 1.1.2.5 Tính bảo mật và khả năng phục hồi của Confluent

Confluent Platform cũng cung cấp một số tính năng được xây dựng dựa trên các tính năng bảo mật của Kafka để giúp đảm bảo quá trình triển khai của bạn luôn an toàn và linh hoạt:

* Thiết lập ủy quyền theo vai trò với Confluent Role-based Access Control (RBAC) và ủy quyền truy cập Access Control Lists (ACLs).
* Cluster Linking: cho phép kết nối trực tiếp các cụm Kafka Cluster và dịch chuyển các topic giữa các Kafka Cluster khác nhau giúp việc xây dựng triển khai nhiều Data Center - Trung tâm dữ liệu.

#### 1.1.2.6 Hiệu suất và khả năng mở rộng của Confluent

Confluent Platform cung cấp một số tính năng mở rộng quy mô một cách hiệu quả và đặt được hiệu suất tối đa:

* Tính năng Tired Storage – Lưu trữ theo tầng: Tự động gửi dữ liệu lên Object Storage – Đối tượng lưu trữ được.
* Manage Self-Balancing Kafka Cluster – Quản lý tự cân bằng cụm Kafka: Giúp tự động cân bằng tải khi quá tải, phát hiện lỗi, tự phục hồi cho cụm. Nó hỗ trợ thêm hoặc ngừng hoạt động của Broker khi cần thiết mà không cần điều chỉnh thủ công.

### 1.1.3 Ưu và nhược điểm của Confluent Platform

Ưu điểm:

* Thông lượng cao : Có thể xử lý dữ liệu lớn theo thời gian thực lên tới hàng triệu Message mỗi giây với độ trễ thấp.
* Khả năng mở rộng: Không giới hạn số Kafka Broker giúp tăng khả năng lưu trữ. Công nghệ Cluster Linking liên kết các cụm Kafka Cluster giúp mở rộng các trung tâm dữ liệu.
* Tính linh hoạt: Confluent tương thích với nhiều nguồn dữ liệu và tích hợp được nhiều công nghệ mới.
* Trình kết nối có sẵn: Khoảng 100 trình kết nối được dựng sẵn từ Confluent giúp việc tích hợp dữ liệu với nhiều hệ thống khác nhau như cơ sở dữ liệu, dịch vụ Cloud, tiện ích IoT.
* Confluent Platform cung cấp các tính năng độc quyền.
* Tính sẵn sàng cao, khả năng khôi phục, chịu lỗi.
* Hỗ trợ trực tiếp từ đội ngũ kĩ thuật của hãng.

Nhược điểm:

* Độ phức tạp: các công cụ và tính năng phức tạp của Confluent Platform khó triển khai, định cấu hình và bảo trì phức tạp, quản lý cần có kiến thức chuyên sâu.
* Bảo trì: Yêu cầu kiểm tra bảo trì liên tục để duy trì tính khả dụng và hiệu suất tốt. Tốn kém chi phí xây dựng đội nhân sự quản lý.
* Phụ thuộc vào các công nghệ: Phụ thuộc vào công nghệ Zookeeper để hoạt động, khiến triển khai trở nên phức tạp.
* Chi phí: trả chi phí cao cho Confluent Platform (Enterprise) để sử dụng nhiều tính năng độc quyền, dịch vụ hỗ trợ.
* Cơ sở hạ tầng: Yêu cầu cơ sở hạ tầng lớn.

## 1.2 Tổng quan MinIO

### 1.2.1 Lý do tích hợp MinIO vào hệ thống Confluent Platform

* Giảm thời gian tái cân bằng: MinIO hỗ trợ lưu trữ các dữ liệu cũ giúp Kafka Broker chỉ lưu những dữ liệu mới. Khi 1 node Broker lỗi, chỉ cần tái cân bằng dữ liệu mới.
* Giảm chi phí: Tiết kiệm chi phí và tài nguyên hơn khi lưu trữ dưới thiết bị lưu trữ vật lý.
* Đáp ứng nhu cầu lưu trữ lớn: MinIO cung cấp khả năng lưu trữ dữ liệu phân tán trên nhiều nút, cho phép mở rộng dễ dàng tăng khả năng lưu trữ.

### 1.2.2 Giới thiệu MinIO

Minio là một giải pháp mã nguồn mở được xây dựng để cung cấp dịch vụ máy chủ lưu trữ đám mây (Object Storage Server). Dịch vụ giúp dữ liệu dưới dạng Object (Đối tượng) nên có thể sử dụng để lưu trữ những dữ liệu không định dạng (unstructured data) như hình ảnh, video, log files,...

MinIO sử dụng giao thức lưu trữ đối tượng tương tự AWS S3 – Amazon Simple Storage Service, giao thức S3 cung cấp phương thức GET, POST, DELETE để đọc , cập nhật, xóa các đối tượng lưu trữ trên MinIO Server.

MinIO khác với AWS S3 1 điểm là bạn có thể cài đặt và host MinIO trên một máy tính cục bộ, tạo và quản lý một hệ thống lưu trữ đám mây trên máy tính cá nhân mang lại sự linh hoạt. Bạn có thể kiểm soát hoàn toàn dữ liệu, kiểm soát quyền truy cập và bảo mật dữ liệu.

## 1.3 Tổng quan Grafana

### 1.3.1 Lý do tích hợp Grafana vào hệ thống Confluent Platform

* Grafana là hệ thống mà nhiều tổ chức đang sử dụng và có thể tích hợp giám sát nhiều hệ thống vào cùng một bảng điều khiển.
* Grafana có thể tùy chỉnh để giám sát một số thông số mà Confluent Control Center không có.

### 1.3.2 Giới thiệu Grafana

Grafana là một công cụ mã nguồn mở và phổ biến để trực quan hóa và giám sát dữ liệu từ các nguồn khác nhau. Grafana hiển thị số liệu (metrics) dưới dạng biểu đồ hoặc đồ thị được tập hợp lại thành dashboard có tính tùy biến cao.

Grafana hỗ trợ đa dạng nguồn dữ liệu khác nhau như cơ sở dữ liệu SQL, hệ thống lưu trữ đối tượng - Object Storage, dịch vụ đám mây - Cloud Services, và các hệ thống giám sát như Prometheus, Elasticsearch việc này giúp người dùng tập trung dữ liệu từ nhiều nguồn vào một bảng điều khiển.

## 1.4 Tổng kết Chương 1

Trong chương 1, em đã trình bày về tổng quan Confluent Platform, tính năng, ưu điểm, nhược điểm, giải quyết vấn đề của doanh nghiệp và tổng quan công nghệ MinIO, Grafana, lý do cần tích hợp 2 công nghệ này vào hệ thống Confluent Platform.

# CHƯƠNG 2: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG XỬ LÝ DỮ LIỆU REAL-TIME CONFLUENT PLATFORM

## 2.1 Khảo sát yêu cầu hệ thống

Trong thực tế, để có thể đưa ra một thiết kê mô hình hệ thống streaming, xử lý dữ liệu real-time Confluent Platform cho doanh nghiệp cần phải khảo sát thông tin về hệ thống của doanh nghiệp để thu thập được các thông tin như

* Thông tin dữ liệu: TPS (Transaction per second) – Số giao dịch trong một giây, Message size – kích thước của thông điệp, Throughtput – thông lượng...
* Thông tin về cấu hình của máy chủ được cấp như RAM, Disk, CPU...

Tuy nhiên để đảm bảo phạm vi đồ án nên từ vấn đề doanh nghiệp gặp phải, em sẽ đưa ra phân tích dựa trên kinh nghiệm và đề xuất của hãng để có thể đưa ra một mô hình hệ thống phù hợp.

Khảo sát yêu cầu hệ thống:

1.Tích hợp dữ liệu từ nhiều hệ thống nguồn và hệ thống đích.

2.Chuẩn hóa dữ liệu để hệ thống đích có thể sử dụng mà không cần phải xử lý

3. Đảm bảo tính sẵn sàng (High Available – HA) của hệ thống.

4. Có khả năng thu thập, lưu trữ, xử lý dữ liệu thời gian thực.

5. Có hệ thống giám sát, quản lý.

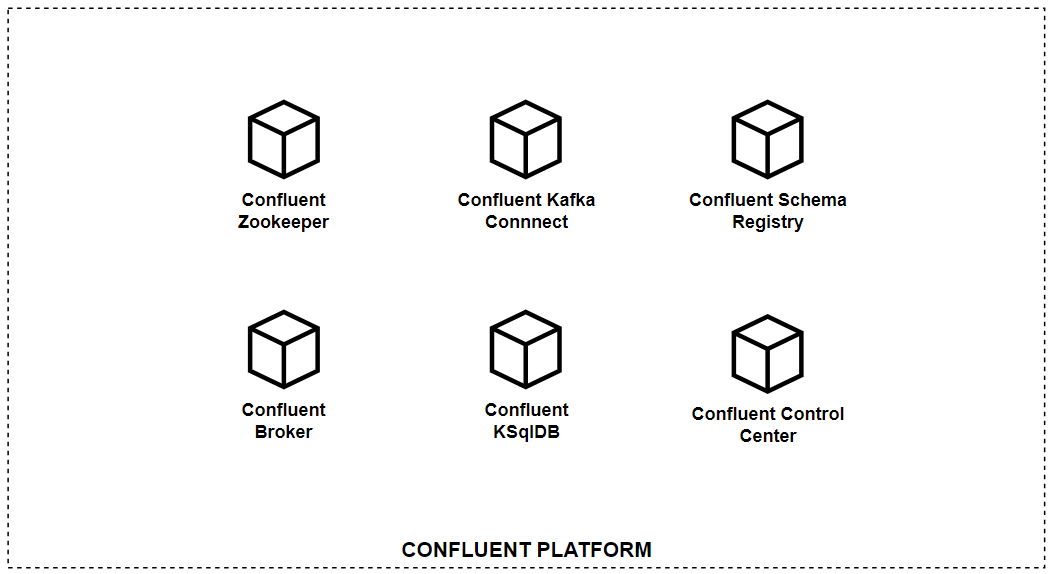
6. Tối ưu chi phi, tiết kiệm tài nguyên.

7. Giảm thời gian tái cân bằng (rebalance) khi gặp sự cố.

8. Đảm bảo tính toàn vẹn, chính xác của dữ liệu.

## 2.2 Phân tích yêu cầu hệ thống

Các thành phần hãng cung cấp:



Hình 2. 1 Thành phần Confluent cung cấp

Từ các yêu cầu hệ thống khảo sát được đề xuất :

1. 3 nodes (nút) Zookeeper vì:

* Yêu cầu số Zookeeper trong cụm là số lẻ do thuật toán đồng thuận ZAB được sử dụng để đạt được sự đồng thuận trong quá trình bầu lại Control Broker, Leader Partition.
* Khuyến nghị của hãng: chỉ nên sử dụng quần thể có 3-5 server để đảm bảo tính sẵn sàng của hệ thống và duy trì tính nhất quán của dữ liệu.

2. 3 nodes Broker vì:

* Đảm bảo tính sẵn sàng của hệ thống.
* Tăng hiệu suất xử lý dữ liệu
* Tăng khả năng lưu trữ
* 3 nodes tạo ra cấu trúc cân bằng tải tự nhiên.
* Tài nguyên được cấp chỉ cho phép 3 nodes.

3. 1 node Schema Registry vì

* Vẫn đảm bảo chức năng cơ bản của Schema Registry: điểm tập trung quản lý, đăng ký và lưu trữ các phiên bản schema (schema – đề cập đến cấu trúc dữ liệu của Message).
* Tài nguyên được cấp hạn chế.

4. 1 node Kafka Connect vì

* Đảm bảo hệ thống Confluent có thể kết nối và lấy dữ liệu từ nhiều hệ thống nguồn và đích.
* Cung cấp các Connector (trình kết nối)
* Quản lý và theo dõi các connector.

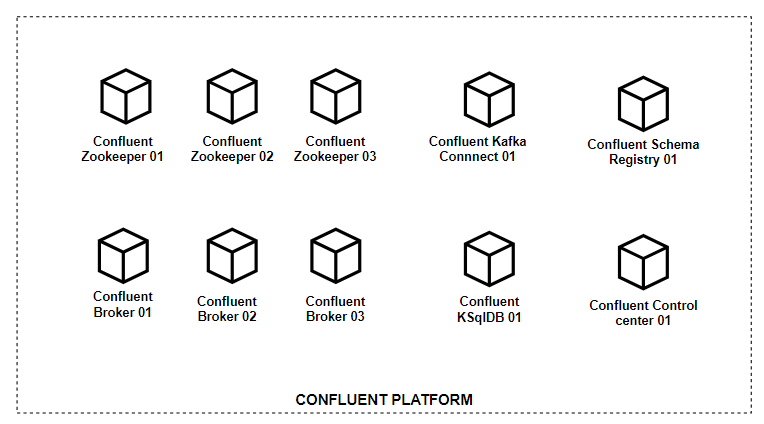
5. 1 node KsqlDB vì

* Đảm bảo hệ thống có thể truy vấn dữ liệu theo thời gian thực.
* Tạo các luồng streaming
* Kết hợp với Confluent Schema Registry quản lý schema và thực hiện truy vấn và biến đổi.

6. 1 node Control Center vì

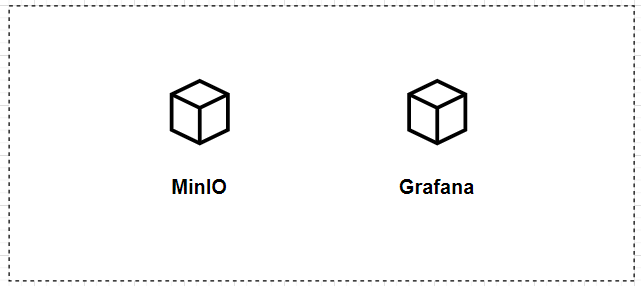
* Quản lý giám sát trạng thái, hiệu suất cụm Kafka
* Cung cấp công cụ và giao diện Web UI để quản lý bảo mật, kiểm soát truy cập cụm Kafka
* Quản lý theo dõi các connector, Stream, các Consumer
* Thông báo về các sự cố hoặc tình trạng quan trọng

Đề xuất thành phần trong hệ thống:



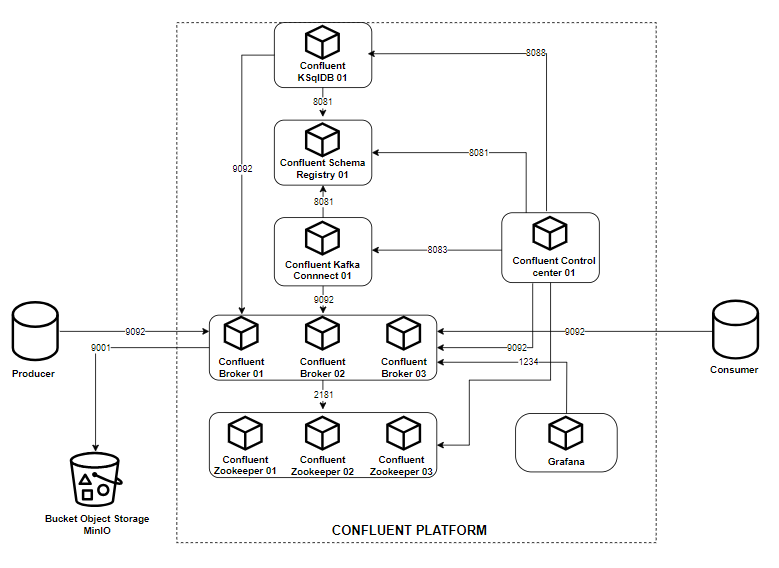
Hình 2. 2 Đề xuất các thành phần trong hệ thống Confluent Platfrom

Đề xuất thêm 2 công nghệ MinIO, Grafana:

****

Hình 2. 3 Công nghệ tích hợp đề xuất

## 2.3 Mô hình tổng quan hệ thống streaming, xử lý dữ liệu real-time Confluent Platform



Hình 2. 4 Mô hình Confluent Platform

Mô tả chức năng của các thành phần tương ứng:

1. Confluent Kafka Broker: Là các server cài đặt Confluent Kafka Broker, thành phần chính của Confluent Platform, có nhiệm vụ lưu trữ các Message của các Topic. Dữ liệu từ các Topic có thể publish hoặc subscribe giữa Confluent Platform sang các hệ thống khác.
2. Confluent Zookeeper: Zookeeper được sử dụng để quản lý và điều phối Confluent Kafka Broker. Zookeeper chủ yếu được sử dụng để thông báo cho Producer và Consumer về tính sẵn sàng của bất kì Broker nào trong hệ thống Confluent Kafka hoặc trong trường hợp các Broker lỗi trong hệ thống cụm Confluent Kafka. Từ các thông báo của Zookeeper về hiện trạng của các Broker, Producer và Consumer sẽ đưa ra quyết định và bắt đầu thực hiện các tác vụ tương ứng.
3. Confluent Kafka Connect: Để kết nối và streaming dữ liệu từ hệ thống Confluent Kafka với các hệ thống khác.
4. Confluent ksqlDB: là một công cụ giúp nhà phát triển có thể tạo các ứng dụng bằng cách sử dụng câu lệnh SQL đơn giản, có khả năng ETL(Extract – Transform – Loading) dữ liệu của Kafka theo thời gian thực sử dụng các câu lệnh SQL hoặc UDF (User Defined Function) để biến đổi dữ liệu.
5. Confluent Schema Registry: Xử lý việc lưu trữ, đồng bộ hóa các Schema đảm bảo tính tương thích cho Producer và Consumer.
6. Confluent Control Center: Là một công cụ WebUI, giúp quản lý và giám sát Confluent Platform, cung cấp giao diện người dùng cho phép theo dõi hiện trạng Cluster, quản lý các Topic, Message, Kafka Connector, đường truyền dữ liệu, thông số hệ thống…
7. Producer: Cơ sở dữ liệu nguồn gửi Message vào các server Confluent Kafka Broker.
8. Consumer: Cơ sở dữ liệu đích đọc Message từ các server Confluent Kafka Broker.
9. MinIO: Hỗ trợ lưu trữ dữ liệu lâu dài trên Cloud.
10. Prometheus: Lấy dữ liệu số liệu của Confluent Platform và hệ thống.
11. Grafana: Sử dụng nguồn số liệu từ Prometheus cung cấp, giám sát và quản lý cụm Confluent Platform bằng biểu đồ trực quan.

## 2.4 Tổng kết chương 2:

Trong chương 2, em đã trình bày phân tích yêu cầu hệ thống. Thiết kế mô hình hệ thống streaming, xử lý dữ liệu real-time Confluent Platform phù hợp.

# CHƯƠNG 3: TRIỂN KHAI CÀI ĐẶT HỆ THỐNG XỬ LÝ DỮ LIỆU REAL-TIME CONFLUENT PLATFORM.

**3.1 Thông tin cài đặt**

**3.1.1 Thông tin máy chủ**

Bảng 3. 1 Bảng thông tin máy chủ

|  |  |
| --- | --- |
| Hệ thống | Máy chủ/IP |
| Confluent-zookeeper-01 | 10.10.12.241 |
| Confluent-zookeeper-02 | 10.10.12.242 |
| Confluent-zookeeper-03 | 10.10.12.244 |
| Confluent-server-01 | 10.10.12.241 |
| Confluent-server-02 | 10.10.12.242 |
| Confluent-server-03 | 10.10.12.244 |
| Confluent-schema-registry-01 | 10.10.12.241 |
| Confluent-kafka-connect-01 | 10.10.12.241 |
| Confluent-ksqldb-01 | 10.10.12.241 |
| Confluent-control-center-01 | 10.10.12.241 |

### 3.1.2 Thông tin kết nối

Đường dẫn kết nối tới giao diện website Confluent Control Center:

<http://10.10.12.241:9021>

### 3.1.3 Các gói cài đặt

Bảng 3. 2 Gói cài đặt Conflfunet 7.6

|  |
| --- |
| https://packages.confluent.io/rpm/7.6/confluent-ce-kafka-http-server-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-common-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-control-center-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-control-center-fe-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-hub-client-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-kafka-connect-replicator-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-kafka-mqtt-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-kafka-rest-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-ksqldb-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-metadata-service-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-platform-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-rebalancer-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-rest-utils-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-schema-registry-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-schema-registry-plugins-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-security-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-server-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-server-rest-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-telemetry-7.6.0-1.noarch.rpm |
| https://packages.confluent.io/rpm/7.6/confluent-cli-7.6.0-1.x86\_64.rpm |

**3.1.4** **Tổng quan quá trình cài đặt hệ thống Confluent Platform**

1. Cài đặt packages trên từng server
2. Cấu hình Confluent Zookeeper
3. Khởi động Confluent Zookeeper
4. Cấu hình Confluent Server
5. Khởi động Confluent Server
6. Cấu hình Confluent Schema Registry
7. Khởi động Confluent Schema Registry
8. Cấu hình Confluent Kafka Connect
9. Khởi động Confluent Kafka Connect
10. Cấu hình Confluent KSQLDB
11. Khởi động Confluent KSQLDB
12. Cấu hình Confluent Control Center
13. Khởi động Confluent Control Center
    1. **Triển khai cài đặt hệ thống Confluent Platform**

**3.2.1 Cài đặt package**

1. Tạo thư mục để lưu trữ gói cài đặt cho Confluent:

|  |
| --- |
| mkdir -p /data/confluent/package |

(Sao chép tất cả các file .rpm của Confluent vào thư mục đã tạo.)

1. Thực hiện cài đặt:

|  |
| --- |
| cd /data/confluent/package |

|  |
| --- |
| rpm -ivh \*.rpm |

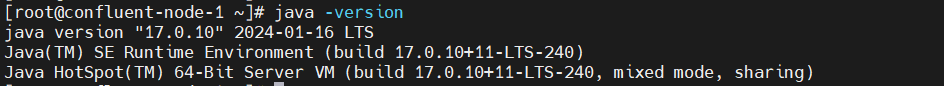
* + 1. **Kiểm tra java version**

Confluent recommend java version 17 cho Confluent Platform 7.6

1. Kiểm tra Java Version

|  |
| --- |
| java -version |

1. Kết quả



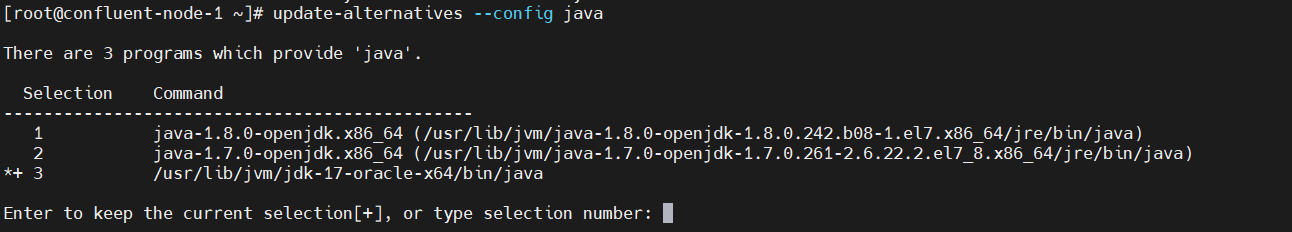
Hình 3. 1 Kết quả kiểm tra version java

1. Nếu kết quả trả về là các phiên bản khác, cần phải thực hiện lệnh sau để thay đổi cấu hình java:

|  |
| --- |
| wget [https://download.oracle.com/java/17/archive/jdk-17.0.10\_linux-x64\_bin.rpm](https://download.oracle.com/java/17/archive/jdk-17.0.10_linux-x64_bin.rpm%20) |

|  |
| --- |
| rpm -ivh jdk-17.0.10\_linux-x64\_bin.rpm |

|  |
| --- |
| update-alternatives -config java |



Hình 3. 2 Chọn version java

Nhập số 3 để chọn version java.

* + 1. **Khai báo biến môi trường vào file .bash\_profile**

1. Mở tệp .bash\_profile

|  |
| --- |
| vi .bash\_profile |

1. Thêm đoạn mã sau vào tệp .bash\_profile

|  |
| --- |
| export ZOOKEEPER\_ROOT\_DIR=/data/confluent/apps/zookeeper  export ZOOKEEPER\_ROOT\_CONFIG\_DIR=/etc/kafka  export BASE\_DIR=/bin  export BROKER\_ROOT\_DIR=/data/confluent/apps/kafka  export BROKER\_ROOT\_CONFIG\_DIR=/etc/kafka  export BASE\_DIR=/bin  export SCHEMA\_ROOT\_DIR=/data/confluent/apps/schema-registry  export SCHEMA\_ROOT\_CONFIG\_DIR=/etc/schema-registry  export BASE\_DIR=/bin  export CONNECT\_ROOT\_DIR=/data/confluent/apps/connect  export CONNECT\_ROOT\_CONFIG\_DIR=/etc/kafka  export BASE\_DIR=/bin  export KSQLDB\_ROOT\_DIR=/data/confluent/apps/ksqldb  export KSQLDB\_ROOT\_CONFIG\_DIR=/etc/ksqldb  export BASE\_DIR=/bin  export C3\_ROOT\_DIR=/data/confluent/apps/control-center  export C3\_ROOT\_CONFIG\_DIR=/etc/confluent-control-center  export BASE\_DIR=/bin |

1. Làm mới lại tệp .bash\_profile để nhận các biến môi trường vừa thêm ở bước trên:

|  |
| --- |
| . ~/.bash\_profile |

1. Tạo nhóm Confluent

|  |
| --- |
| groupadd confluent |

* + 1. **Cấu hình cho Confluent Zookeeper**

Bước 1: Chuẩn bị thư mục lưu log và dữ liệu cho Zookeeper

1. Tạo thư mục dùng để chứa log và dữ liệu Zookeeper

|  |
| --- |
| mkdir -p $ZOOKEEPER\_ROOT\_DIR/{data,logs} |

1. Gán quyền read/write cho user cp-kafka vào thư mục vừa tạo

|  |
| --- |
| chown -R cp-kafka:confluent $ZOOKEEPER\_ROOT\_DIR |

1. Tạo tệp myid trong phân vùng chứa dữ liệu của Zookeeper, trong tệp chứa id của Zookeeper

|  |
| --- |
| echo 1 > $ZOOKEEPER\_ROOT\_DIR/data/myid |

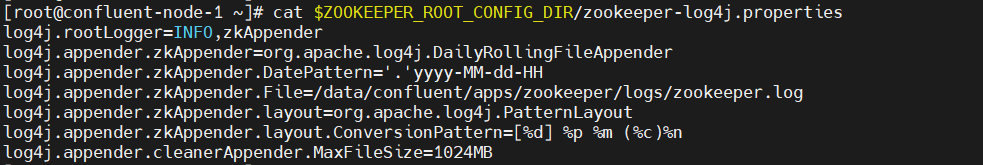
Bước 2: Cấu hình log4j cho Confluent Zookeeper

1. Thiết lập cấu hình log4j cho Confluent Zookeeper

|  |
| --- |
| echo "log4j.rootLogger=INFO,zkAppender  log4j.appender.zkAppender=org.apache.log4j.DailyRollingFileAppender  log4j.appender.zkAppender.DatePattern='.'yyyy-MM-dd-HH  log4j.appender.zkAppender.File="$ZOOKEEPER\_ROOT\_DIR"/logs/zookeeper.log  log4j.appender.zkAppender.layout=org.apache.log4j.PatternLayout  log4j.appender.zkAppender.layout.ConversionPattern=[%d] %p %m (%c)%n  log4j.appender.cleanerAppender.MaxFileSize=1024MB"> $ZOOKEEPER\_ROOT\_CONFIG\_DIR/zookeeper-log4j.properties |

1. Kiểm tra lại cấu hình:

|  |
| --- |
| cat $ZOOKEEPER\_ROOT\_CONFIG\_DIR/zookeeper-log4j.properties |



Hình 3. 3 Kết quả cấu hình file log4j

Bước 3: Cấu hình lại tệp service cho Zookeeper:

1. Backup file confluent-zookeeper-service

|  |
| --- |
| cp /usr/lib/systemd/system/confluent-zookeeper.service /usr/lib/systemd/system/confluent-zookeeper.service.backup |

1. Thêm cấu hình vào tệp service cho Zookeeper

|  |
| --- |
| sed -i '/^Group=confluent/cGroup=confluent\nEnvironment="LOG\_DIR=’$ZOOKEEPER\_ROOT\_DIR’/logs" "KAFKA\_LOG4J\_OPTS=-Dlog4j.configuration=file:/etc/kafka/zookeeper-log4j.properties" "KAFKA\_HEAP\_OPTS=-Xmx2G -Xms2G"' /usr/lib/systemd/system/confluent-zookeeper.service |

1. Reload lại file service

|  |
| --- |
| systemctl daemon-reload |

Bước 4: Cấu hình Confluent Zookeeper

1. Backup cấu hình default của Confluent Zookeeper

|  |
| --- |
| mv $ZOOKEEPER\_ROOT\_CONFIG\_DIR/zookeeper.properties $ZOOKEEPER\_ROOT\_CONFIG\_DIR/zookeeper.properties.backup |

1. Tạo file cấu hình mới cho Zookeeper

|  |
| --- |
| mkdir -p $ZOOKEEPER\_ROOT\_CONFIG\_DIR/zookeeper.properties |

1. Thêm cấu hình vào tệp Zookeeper

|  |
| --- |
| echo "dataDir=/data/confluent/apps/zookeeper/data  clientPort=2181  maxClientCnxns=60  admin.enableServer=false  server.1=10.10.12.241:2888:3888  server.2=10.10.12.242:2888:3888  server.3=10.10.12.244:2888:3888  initLimit=10  syncLimit=5  autopurge.snapRetainCount=3  autopurge.purgeInterval=24" > $ZOOKEEPER\_ROOT\_CONFIG\_DIR/zookeeper.properties |

**Giải thích cấu hình:**

* dataDir : cấu hình thư mục lưu data cho Confluent Zookeeper.
* clientPort: khai báo port cho các Client kết nối tới Confluent Zookeeper
* maxClientCnxns: Chỉ số lượng connection tối đa mà Client kết nối.
* admin.enableServer: bật giao diện admin cho Confluent Server.
* server.x : Khai báo địa chỉ ip của node Confluent Zookeeper

|  |  |
| --- | --- |
| Default | Update |
| **“”** | server.1=10.10.12.241:2888:3888  server.2=10.10.12.242:2888:3888  server.3=10.10.12.244:2888:3888 |

* initLimit: thời gian tối đa các follower kết nối đến Node Leader.
* autopurge.snapRetainCount: giữ lại các bản snapshot, xóa những bản còn lại trong dataDir và dataLogDir.
* autopurge.purgeInterval: Thời gian làm sạch tự động được kích hoạt.

Tương tự với confluent-node-2 và confluent-node-3.

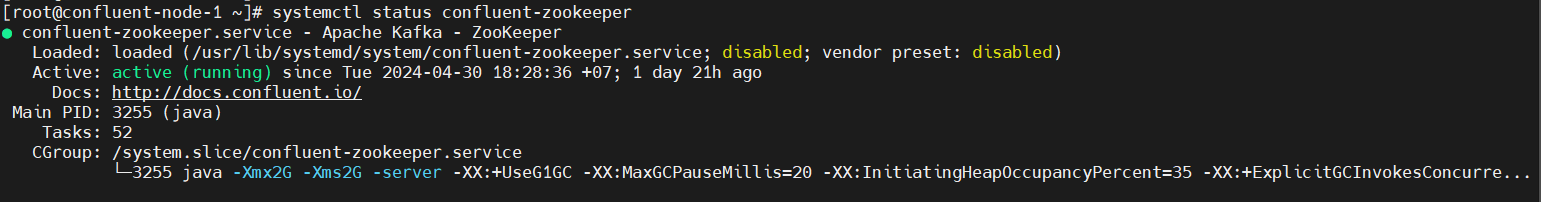
BƯỚC 5: Khởi chạy Confluent Zookeeper

Bảng 3. 3 Bảng khởi chạy ConfluentZookeeper

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ | Câu lệnh systemctl |
| Khởi động Confluent Zookeeper | confluent-zookeeper-01 | systemctl start confluent-zookeeper.service |
| confluent-zookeeper-02 | systemctl start confluent-zookeeper.service |
| confluent-zookeeper-03 | systemctl start confluent-zookeeper.service |
| Kiểm tra log |  | tail -1000f /data/confluent/apps/zookeeper/logs/zookeeper.log |

Kiểm tra trạng thái của service:

|  |
| --- |
| systemctl status confluent-zookeeper.service |



Hình 3. 4 Trạng thái service Zookeeper

### 3.2.5 Cấu hình cho Confluent Server

Bước 1: Chuẩn bị thư mục lưu log và dữ liệu cho Server

1. Tạo thư mục dùng để chứa log và dữ liệu Server

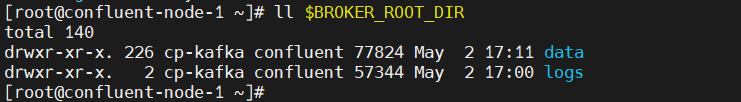
|  |
| --- |
| mkdir -p $BROKER\_ROOT\_DIR/{data,logs} |

1. Gán quyền read/write cho user cp-kafka vào thư mục vừa tạo

|  |
| --- |
| chown -R cp-kafka:confluent $BROKER\_ROOT\_DIR |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $BROKER\_ROOT\_DIR |



Hình 3. 5 Kiểm tra quyền cho thư mục Kafka

Bước 2: Cấu hình lại tệp service cho Server:

1. Backup tệp confluent-server-service

|  |
| --- |
| cp /usr/lib/systemd/system/confluent-server.service /usr/lib/systemd/system/confluent-server.service.backup |

1. Thêm cấu hình vào tệp service cho Server

|  |
| --- |
| sed -i '/^Group=confluent/cGroup=confluent\nEnvironment="LOG\_DIR='$BROKER\_ROOT\_DIR'/logs" "KAFKA\_LOG4J\_OPTS=-Dlog4j.configuration=file:'$BROKER\_ROOT\_CONFIG\_DIR'/log4j.properties" "KAFKA\_HEAP\_OPTS=-Xms6g -Xmx6g -XX:MetaspaceSize=96m -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:G1HeapRegionSize=16M -XX:MinMetaspaceFreeRatio=50 -XX:MaxMetaspaceFreeRatio=80"' /usr/lib/systemd/system/confluent-server.service |

1. Reload lại tệp service

|  |
| --- |
| systemctl daemon-reload |

Bước 3: Tạo Private key và Public key

1. Tạo thư mục chứa private key và public key

|  |
| --- |
| mkdir -p $BROKER\_ROOT\_CONFIG\_DIR/certs |

1. Tạo tệp private key ở thư mục $BROKER\_ROOT\_CONFIG\_DIR/certs

|  |
| --- |
| openssl genrsa -out $BROKER\_ROOT\_CONFIG\_DIR/certs/private.pem 2048 |

1. Tạo tệp public key ở thư mục $BROKER\_ROOT\_CONFIG\_DIR/certs

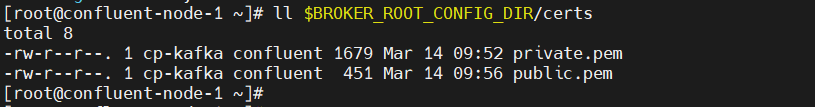
|  |
| --- |
| openssl rsa -in $BROKER\_ROOT\_CONFIG\_DIR/certs/private.pem -outform PEM -pubout -out $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem |

1. Grant quyền cho user cp-kafka

|  |
| --- |
| chown -R cp-kafka: $BROKER\_ROOT\_CONFIG\_DIR/certs |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $BROKER\_ROOT\_CONFIG\_DIR/certs |



Hình 3. 6 Kiểm tra quyền thư mục certs của broker

Bước 4: Cấu hình Confluent Server

1. Backup cấu hình default của Confluent Server

|  |
| --- |
| mv $BROKER\_ROOT\_CONFIG\_DIR/server.properties $BROKER\_ROOT\_CONFIG\_DIR/server.properties.backup |

1. Tạo tệp cấu hình mới cho Server

|  |
| --- |
| vi $BROKER\_ROOT\_CONFIG\_DIR/server.properties |

1. Thêm cấu hình vào tệp Server

Cấu hình sử dụng cho Confluent Server Mix Authentication

|  |
| --- |
| echo "broker.id=1  listeners=OAUTH://10.10.12.241:9091,EXTERNAL://10.10.12.241:9092,RBAC://10.10.12.241:9093,ACLS://10.10.12.241:9094,INTERNAL://10.10.12.241:9095  #advertised.listeners=SASL\_PLAINTEXT://10.10.12.241:9092,EXTERNAL://10.10.12.241:9093,OAUTH://10.10.12.241:9091  listener.security.protocol.map=OAUTH:SASL\_PLAINTEXT,EXTERNAL:PLAINTEXT,RBAC:SASL\_PLAINTEXT,ACLS:SASL\_PLAINTEXT,INTERNAL:PLAINTEXT  inter.broker.listener.name=INTERNAL  sasl.enabled.mechanisms=PLAIN,OAUTHBEARER,SCRAM-SHA-256  sasl.mechanism.inter.broker.protocol=PLAIN  num.network.threads=3  num.io.threads=8  socket.send.buffer.bytes=102400  socket.receive.buffer.bytes=102400  socket.request.max.bytes=104857600  log.dirs=/data/confluent/apps/kafka/data  num.partitions=3  num.recovery.threads.per.data.dir=1  offsets.topic.replication.factor=2  transaction.state.log.replication.factor=2  transaction.state.log.min.isr=2  log.retention.hours=168  log.segment.bytes=1073741824  log.retention.check.interval.ms=300000  zookeeper.connect=10.10.12.241:2181,10.10.12.242:2181,10.10.12.244:2181  zookeeper.connection.timeout.ms=18000  #Metric-reporter  metric.reporters=io.confluent.metrics.reporter.ConfluentMetricsReporter  confluent.metrics.reporter.bootstrap.servers=10.10.12.241:9093,10.10.12.242:9093,10.10.12.244:9093  confluent.metrics.reporter.security.protocol=SASL\_PLAINTEXT  confluent.metrics.reporter.sasl.mechanism=PLAIN  confluent.metrics.reporter.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="denodo" password="vertica\_4U";  confluent.metrics.reporter.topic.replicas=3  #Group-coordinator-settings  group.initial.rebalance.delay.ms=0  #Confluent-Authorization  confluent.license=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJjb250cm9sLWNlbnRlciIsIm5iNCI6IjE2OTU2MTM1ODEiLCJtb25pdG9yaW5nIjp0cnVlLCJsaWNlbnNlVHlwZSI6IkVudGVycHJpc2UiLCJpc3MiOiJDb25mbHVlbnQiLCJpYXQiOjE2OTM1NTE2MDAsImV4cCI6MTc5NTkzNTYwMCwiYXVkIjoiMDA2NFUwMDAwMHNxallQUUFZIn0=.Rjm0el5v7BKbC\_jcu\_lmuHQeoAhKQ3xP4MM1KSl8wo7859OogL954nO2BbNos3OwplXbvuu7\_IsSFyf8UBFK2zUtHV8elCfcIcAxu0uE-pMhYZoBuq\_OXePvrBfRs8TaJOzXq6CcdSrkekL-Ym7epXOO4YOWQzJXLU6GzVcOVDv8bTMEYtVHw-lE3nOYa7rLci9UiKKWex5u4hLdCJMvSNIFhOz5xm5WcGhhSv8mpZJapaVfcxJ5585\_xQ-w\_0QnG-mcLKDhNOkBfG97DId93KC4MIKQe1Cgs-zFpGXqgNajg75aqNxj7WgPcqrrbJSWIirFgD1IKR2oXaXR2m8FW  confluent.license.topic.replication.factor=3  confluent.metadata.topic.replication.factor=3  confluent.security.event.logger.exporter.kafka.topic.replicas=3  confluent.balancer.enable=true  confluent.balancer.topic.replication.factor=3  default.replication.factor=3  min.insync.replica=2  message.max.bytes=10485760  # Search groups for group-based authorization.  ldap.group.name.attribute=sAMAccountName  ldap.group.object.class=group  ldap.group.member.attribute=member  ldap.group.member.attribute.pattern=CN=(.\*),cn=Users,dc=bnh,dc=vn  ldap.group.search.base=cn=Builtin,dc=bnh,dc=vn  ldap.user.search.scope=2  # ldap.group.search.filter=(|(CN=denodo)(CN=users))  # Kafka authenticates to the directory service with the bind user.  ldap.java.naming.provider.url=ldap://10.10.12.186:389  ldap.java.naming.security.authentication=simple  ldap.java.naming.security.credentials=vertica\_4U  ldap.java.naming.security.principal=cn=denodo,cn=Users,dc=bnh,dc=vn  # Locate users. Make sure that these attributes and object classes match what is in your directory service.  ldap.user.name.attribute=sAMAccountName  ldap.user.object.class=organizationalPerson  ldap.user.search.base=cn=Users,dc=bnh,dc=vn  listener.name.rbac.sasl.enabled.mechanisms=PLAIN  listener.name.rbac.plain.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="denodo" password="vertica\_4U";  listener.name.rbac.plain.sasl.server.callback.handler.class=io.confluent.security.auth.provider.ldap.LdapAuthenticateCallbackHandler  listener.name.oauth.sasl.enabled.mechanisms=OAUTHBEARER  listener.name.oauth.oauthbearer.sasl.login.callback.handler.class=io.confluent.kafka.server.plugins.auth.token.TokenBearerServerLoginCallbackHandler  listener.name.oauth.oauthbearer.sasl.server.callback.handler.class=io.confluent.kafka.server.plugins.auth.token.TokenBearerValidatorCallbackHandler  listener.name.oauth.oauthbearer.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required publicKeyPath="/etc/kafka/certs/public.pem";  authorizer.class.name=io.confluent.kafka.security.authorizer.ConfluentServerAuthorizer  confluent.authorizer.access.rule.providers=CONFLUENT,ZK\_ACL  confluent.metadata.server.advertised.listeners=http://10.10.12.241:8091  confluent.metadata.server.authentication.method=BEARER  confluent.metadata.server.listeners=http://0.0.0.0:8091  confluent.metadata.server.token.key.path=/etc/kafka/certs/private.pem  super.users=User:denodo;User:ANONYMOUS;User:localhost  confluent.balancer.num.concurrent.partition.movements.per.broker=50  ########ACL#####  listener.name.acls.sasl.enabled.mechanisms=SCRAM-SHA-512  listener.name.acls.scram-sha-512.sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required username="denodo" password="vertica\_4U";  authorizer.class.name=kafka.security.authorizer.AclAuthorizer  authorizer.class.name=io.confluent.kafka.security.authorizer.ConfluentServerAuthorizer  allow.everyone.if.no.acl.found=false " > $BROKER\_ROOT\_CONFIG\_DIR/server.properties |

**Giải thích cấu hình**

* broker.id : id broker hay còn gọi là Confluent Server cho máy chủ này.

|  |  |
| --- | --- |
| Default | Update |
| -1 | 1 |

* listeners: danh sách cổng và tên phương thức của các listener.

|  |  |
| --- | --- |
| Default | Update |
| PLAINTTEXT://9092 | OAUTH://10.10.12.241:9091,EXTERNAL://10.10.12.241:9092,RBAC://10.10.12.241:9093,ACLS://10.10.12.241:9094,INTERNAL://10.10.12.241:9095 |

* listener.security.protocol.map : Định nghĩa cùng một giao thức bảo bật có thể sử dụng ở nhiều cổng hoặc địa chỉ ip khác nhau.
* inter.broker.listener.name: Tên listener được sử dụng để giao tiếp giữa các broker.

|  |  |
| --- | --- |
| Default | Update |
| Null | INTERNAL |

* sasl.enable.mechanisms: cơ chế SASL được kích hoạt trong Confluent Server.
* sasl.mechanism.inter.broker.protocol: cơ chế SASL được sử dụng để giao tiếp giữa các broker.
* num.network.threads: số lượng luồng mà máy chủ sử dụng để nhận các yêu cầu từ mạng và gửi phản hổi đến mạng.
* socket.send.buffer.bytes: Bộ đệm SO\_SNDBUF của các socket.
* socket.request.max.bytes: số byte tối đa trong một yêu cầu socket
* log.dirs: Địa chỉ lưu trữ dữ liệu

|  |  |
| --- | --- |
| Default | Update |
| Null | /data/confluent/apps/kafka/data |

* num.partitions: giá trị mặc định số partitions của Topic trong cụm.

|  |  |
| --- | --- |
| Default | Update |
| 1 | 3 |

* num.recovery.threads.per.data.dir: số lượng tiến trình sử dụng để khôi phục dữ liệu khi khởi động hoặc tắt service

|  |  |
| --- | --- |
| Default | Update |
| 1 | 3 |

* offset.topic.replication.factor: Số bản replication cho topics offset
* log.retension.hours: thời gian lưu log mặc định của topic. Quá thời gian dữ liệu trong topic sẽ tự động xóa.
* log.segment.bytes: dung lượng tối đa của một file log trong topic được lưu dưới disk.
* zookeeper.connect: địa chỉ các node Confluent Zookeeper.
* zookeeper.connection.timeout.ms: thời gian server kết nối tới zookeeper.
* Metric.reporters: danh sách các class đưucọ sử dụng để thực hiện báo cáo metric.

|  |  |
| --- | --- |
| Default | Update |
| “” | io.confluent.metrics.reporter.ConfluentMetricsReporter |

* Confluent.metrics.repoerter.bootstrap.server: danh sách Confluent Server để lưu trữ dữ liệu cho topic metric dùng để báo cáo jmx.

|  |  |
| --- | --- |
| Default | Update |
| “” | 10.10.12.241:9093,10.10.12.242:9093,10.10.12.244:9093 |

* Confluent.metrics.reporter.security.protocol: giao thức cho các tiến trình báo cáo metrics sử dụng

|  |  |
| --- | --- |
| Default | Update |
| Null | SASL\_PLAINTEXT |

* Confluent.metrics.reporter.sasl.mechanism: cơ chế SASL được sử dụng cho tiến trình báo cáo metric
* Confluent.metric.reporter.sasl.jaas.config: tham số đăng nhập JAAS cho kết nối SASL.
* Confluent metrics.reporter.topic.replicas: số bản sao cho topic metrics.
* Group.initial.rebalance.delay.ms: thời gian chờ để tái cân bằng.
* Confluent.license: giấy phép enterprise của confluent platform.
* Confluent.metadata.topic.replication.factor: số bản sao của topic metadata

|  |  |
| --- | --- |
| Default | Update |
| 1 | 3 |

* Confluent.balancer..enable: cấu hình bật tính năng self balance
* Confluent.balancer.topic.replication.factor:số bản sao topic balancer.
* Default.replication.factor: số bản replication cho các topic được tạo.

|  |  |
| --- | --- |
| Default | Update |
| 1 | 3 |

* min.insync.replica: cấu hình số insync replica tối thiểu của cụm

|  |  |
| --- | --- |
| Default | Update |
| 1 | 2 |

* ldap.group.name.attribute:cấu hình thuộc tính của group AD/LDAP
* ldap.group.member.attribute.pattern: user AD/LDAP.
* Listener.name.oauth.sasl.enable.mechanisms: cấu hình listener OAUTH.

|  |  |
| --- | --- |
| Default | Update |
| “” | OAUTHBEARER |

* Confluent.metadata.server.advertised.listener: cấu hình metadata service để các client kết nối đến.
* Confluent.banlacer.num.concurrent.partition.movement.per.broker: số partition được chuyển trong khi tái cân bằng.
* Allow.everyone.if.no.acl.found: cho phép super user có thể truy cập đến tất cả tài nguyên của confluent.
* auto.create.topics.enables: cấu hình cho phép client không thể tự tạo topic.

Làm tương tự với confluent-node-2 và confluent-node-3.

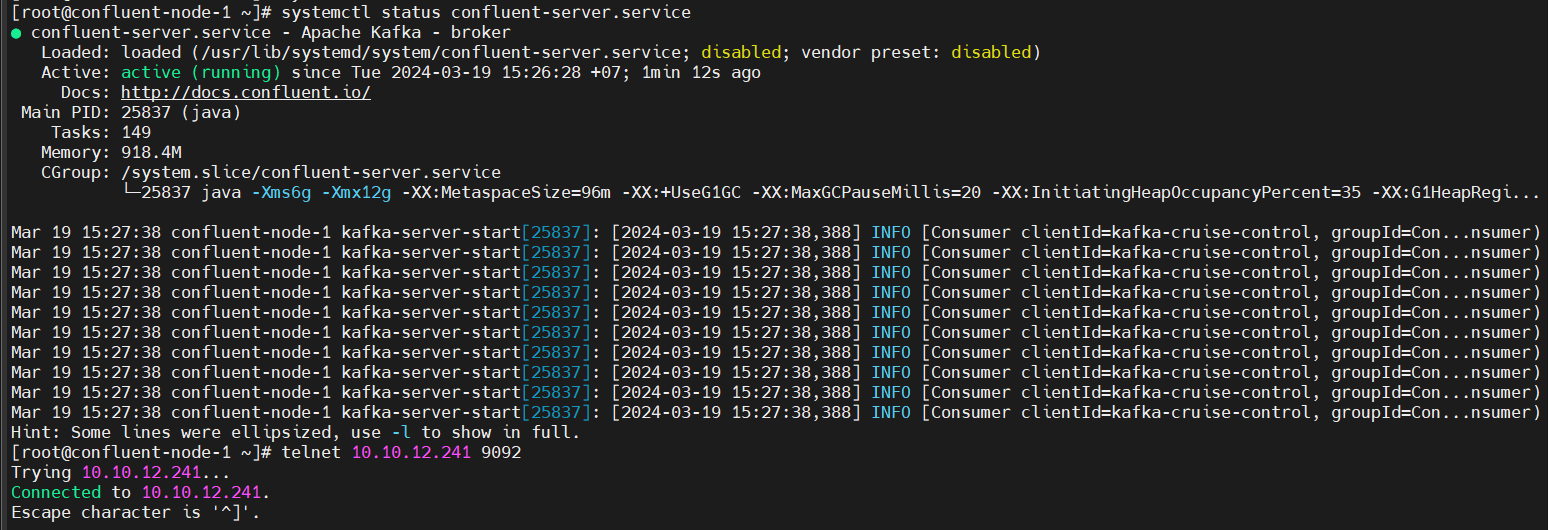
Bước 5: Khởi chạy Confluent Server

Bảng 3. 4 Khởi chạy server

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ | Câu lệnh systemctl |
| Khởi động Confluent Server | 10.10.12.241 | systemctl start confluent-server.service |
| 10.10.12.242 | systemctl start confluent-server.service |
| 10.10.12.244 | systemctl start confluent-server.service |
| Kiểm tra log confluent server |  | tail -1000f /data/confluent/apps/server /logs/server.log |

Kiểm tra trạng thái của service:

|  |
| --- |
| systemctl status confluent-server.service |



Hình 3. 7 Trạng thái service Server

### 3.2.6 Cấu hình Schema Registry

Bước 1: Chuẩn bị thư mục lưu log và dữ liệu cho Schema Registry

1. Tạo thư mục dùng để chứa log Schema Registry

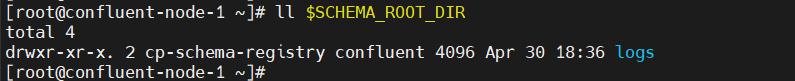
|  |
| --- |
| mkdir -p $SCHEMA\_ROOT\_DIR/logs |

1. Gán quyền read/write cho user cp-schema-registry vào thư mục vừa tạo

|  |
| --- |
| chown -R cp-schema-registry:confluent $SCHEMA\_ROOT\_DIR |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $SCHEMA\_ROOT\_DIR |



Hình 3. 8 Kiểm tra quyền thư mục schema

Bước 2: Cấu hình lại tệp service cho schema-registry

1. Backup file confluent-schema-registry-service

|  |
| --- |
| cp /usr/lib/systemd/system/confluent-schema-registry.service /usr/lib/systemd/system/ confluent-schema-registry.service.backup |

1. Thêm cấu hình vào tệp service cho Schema Registry service

|  |
| --- |
| sed -i 's|^Env.\*|Environment="LOG\_DIR='$SCHEMA\_ROOT\_DIR'/logs" "SCHEMA\_REGISTRY\_HEAP\_OPTS=-Xmx2G"|' /usr/lib/systemd/system/confluent-schema-registry.service |

1. Reload lại tệp service

|  |
| --- |
| systemctl daemon-reload |

Bước 3: Tạo Private key và public key

1. Tạo thư mục chứa private key và public key

|  |
| --- |
| mkdir -p $SCHEMA\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép tệp private key:

|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/private.pem $SCHEMA\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép tệp public key:

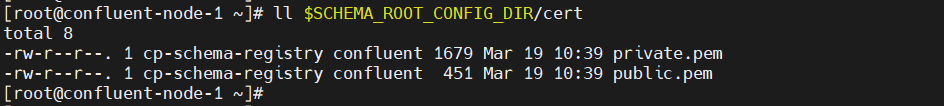
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $SCHEMA\_ROOT\_CONFIG\_DIR/certs |

1. Grant quyền cho user cp-schema-registry

|  |
| --- |
| chown -R cp-schema-registry: $SCHEMA\_ROOT\_CONFIG\_DIR/certs |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $SCHEMA\_ROOT\_CONFIG\_DIR/certs |



Hình 3. 9 Kiểm tra quyền thư mục certs của schema registry

Bước 4: Cấu hình Confluent Schema Registry

1. Backup cấu hình default của Confluent Schema Registry

|  |
| --- |
| mv $SCHEMA\_ROOT\_CONFIG\_DIR/schema-registry.properties $SCHEMA\_ROOT\_CONFIG\_DIR/schema-registry.properties.backup |

1. Tạo tệp cấu hình mới cho Schema Registry

|  |
| --- |
| vi $SCHEMA\_ROOT\_CONFIG\_DIR/schema-registry.properties |

1. Thêm cấu hình vào tệp Schema Registry

Cấu hình tích hợp với AD/LDAP

|  |
| --- |
| ####CONFIG LDAP####  listeners=http://0.0.0.0:8081  kafkastore.bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  kafkastore.topic=\_schemas  debug=false  kafkastore.security.protocol=SASL\_PLAINTEXT  kafkastore.sasl.mechanism=OAUTHBEARER  kafkastore.sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.TokenUserLoginCallbackHandler  kafkastore.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \  metadataServerUrls="http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091" \  username="denodo" password="vertica\_4U" \  publicKeyPath="/etc/schema-registry/cert/public.pem";  resource.extension.class=io.confluent.kafka.schemaregistry.security.SchemaRegistrySecurityResourceExtension  confluent.schema.registry.authorizer.class=io.confluent.kafka.schemaregistry.security.authorizer.rbac.RbacAuthorizer  rest.servlet.initializor.classes=io.confluent.common.security.jetty.initializer.InstallBearerOrBasicSecurityHandler  confluent.metadata.bootstrap.server.urls=http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091  confluent.metadata.basic.auth.user.info=denodo:vertica\_4U  confluent.metadata.http.auth.credentials.provider=BASIC  public.key.path=/etc/schema-registry/cert/public.pem  confluent.schema.registry.auth.mechanism=JETTY\_AUTH |

**Giải thích cấu hình:**

* Listener: thông tin cổng để client kết nối đến Confluent Schema Registry

|  |  |
| --- | --- |
| Default | Update |
| <http://8081> | http://0.0.0.0:8081 |

* Kafkastore.bootstrap.server: cấu hình bootstrap server để kết nối đến cụm Confluent Server đã bật RBAC.

|  |  |
| --- | --- |
| Default | Update |
| localhost:9092 | 10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091 |

* Kafkastore.topic: tên topic sử dụng bởi Confluent Schema Registry
* Kafkastore.security.protocol: giao thức giao tiếp với Broker
* Kafkastore.sasl.jaas.config: chứa user truy cập kết nối tới Broker
* Resource.extension.class: tên class có thể sử dụng tính năng như filter,logging,security.
* Rest.servlet.initializor.classes: tên class sử dụng tính năng RBAC cho Confluent Schema Registry.
* Confluent.metadata.bootstrap.server.url: cấu hình địa chỉ IP của confluent metadata service
* Confluent.schema.registry.auth.mechanism: phương thức bảo mật của shema registry.

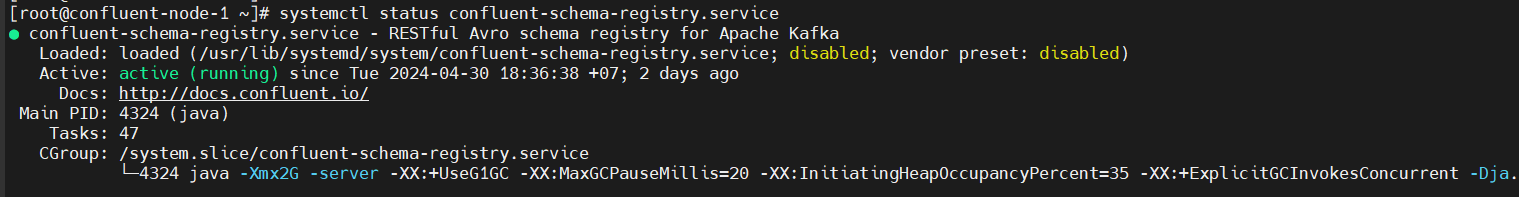
Bước 5: Khởi động Confluent Schema Registry Service:

Bảng 3. 5 Bảng khởi động Confluent Schema Registry

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ | Câu lệnh systemctl |
| Khởi động Confluent Schema Registry | 10.10.12.241 | systemctl start confluent-schema-registry.service |
| Kiểm tra log Confluent Schema Registry | 10.10.12.241 | tail -1000f /data/confluent/apps/schema/logs/schema-registry-log |

Kiểm tra trạng thái của Confluent Schema Registry service:

|  |
| --- |
| systemctl status confluent-schema-registry.service |



Hình 3. 10 Trạng thái bật thành công service Schema Registry

### 3.2.7 Cấu hình Kafka Connect

Bước 1: Chuẩn bị thư mục lưu log và connector cho Kafka Connect

1. Tạo thư mục dùng để chứa log và connector

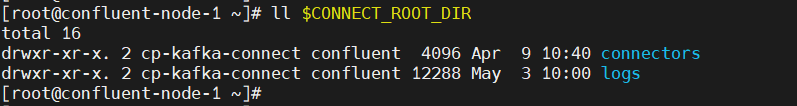
|  |
| --- |
| mkdir -p $BROKER\_ROOT\_DIR/{connector,logs} |

1. Gán quyền read/write cho user cp-kafka-connect vào thư mục vừa tạo

|  |
| --- |
| chown -R cp-kafka-connect:confluent $CONNECT\_ROOT\_DIR |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $CONNECT\_ROOT\_DIR |



Hình 3. 11 Kiểm tra quyền thư mục connect

Bước 2: Cấu hình lại tệp service cho Confluent Kafka Connect:

1. Backup file confluent-kafka-connect-service

|  |
| --- |
| cp /usr/lib/systemd/system/confluent-kafka-connect.service /usr/lib/systemd/system/confluent-kafka-connect.service.backup |

1. Thêm cấu hình vào tệp service cho Kafka Connect

|  |
| --- |
| sed -i '/^Group=confluent/cGroup=confluent\nEnvironment="LOG\_DIR='$CONNECT\_ROOT\_DIR'/logs" "KAFKA\_LOG4J\_OPTS=-Dlog4j.configuration=file:'$CONNECT\_ROOT\_CONFIG\_DIR'/connect-log4j.properties" "KAFKA\_HEAP\_OPTS=-Xmsg -Xmx2g"' /usr/lib/systemd/system/confluent-kafka-connect.service |

1. Reload lại tệp service

|  |
| --- |
| systemctl daemon-reload |

Bước 3: Tạo Private key và public key

1. Tạo thư mục chứa private key và public key

|  |
| --- |
| mkdir -p $CONNECT\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép private key ở thư mục

|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/private.pem $CONNECT\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép public key ở thư mục

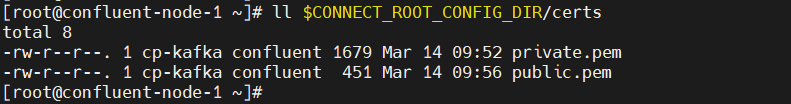
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $CONNECT\_ROOT\_CONFIG\_DIR/certs |

1. Grant quyền cho user cp-kafka

|  |
| --- |
| chown -R cp-kafka-connect: $CONNECT\_ROOT\_CONFIG\_DIR/certs |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $CONNECT\_ROOT\_CONFIG\_DIR/certs |



Hình 3. 12 Kiểm tra quyền thư mục certs của kafka connect

Bước 4: Cấu hình Confluent Kafka Connect

1. Backup cấu hình default của Confluent Kafka Connect

|  |
| --- |
| mv $CONNECT\_ROOT\_CONFIG\_DIR/connect-distributed.properties $CONNECT\_ROOT\_CONFIG\_DIR/connect-distributed.properties.backup |

1. Tạo tệp cấu hình mới cho Confluent Kafka Connect

|  |
| --- |
| vi $CONNECT\_ROOT\_CONFIG\_DIR/connect-distributed.properties |

1. Thêm cấu hình vào tệp Confluent Kafka Connect

|  |
| --- |
| ###CONFIG RBAC####  key.converter.schemas.enable=true  value.converter.schemas.enable=true  offset.flush.interval.ms=10000  plugin.path=/usr/share/java,/usr/share/confluent-hub-components,/data/confluent/apps/connect/connectors  admin.bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  admin.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \  metadataServerUrls="http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091" \  username="denodo" password="vertica\_4U" \  publicKeyPath="/etc/kafka/certs/public.pem";  admin.sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.TokenUserLoginCallbackHandler  admin.sasl.mechanism=OAUTHBEARER  admin.security.protocol=SASL\_PLAINTEXT  bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  config.storage.replication.factor=1  config.storage.topic=connect-configs  confluent.metadata.basic.auth.user.info=denodo:vertica\_4U  confluent.metadata.bootstrap.server.urls=http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091  confluent.metadata.http.auth.credentials.provider=BASIC  confluent.topic.replication.factor=1  consumer.bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  consumer.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \  metadataServerUrls="http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091" \  username="denodo" password="vertica\_4U" \  publicKeyPath="/etc/kafka/certs/public.pem";  consumer.sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.TokenUserLoginCallbackHandler  consumer.sasl.mechanism=OAUTHBEARER  consumer.security.protocol=SASL\_PLAINTEXT  group.id=connect-cluster  ############### KEY CONVERTER#######  #key.converter=org.apache.kafka.connect.json.JsonConverter  key.converter=io.confluent.connect.avro.AvroConverter  key.converter.schema.registry.basic.auth.credentials.source=denodo  key.converter.schema.registry.basic.auth.user.info=denodo:vertica\_4U  key.converter.schema.registry.url=http://10.10.12.241:8081  key.converter.schemas.enable=false  listeners=http://10.10.12.241:8083  offset.flush.interval.ms=10000  offset.storage.replication.factor=1  offset.storage.topic=connect-offsets  producer.bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  producer.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \  metadataServerUrls="http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091" \  username="denodo" password="vertica\_4U" \  publicKeyPath="/etc/kafka/certs/public.pem";  producer.sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.TokenUserLoginCallbackHandler  producer.sasl.mechanism=OAUTHBEARER  producer.security.protocol=SASL\_PLAINTEXT  public.key.path=/etc/kafka/certs/public.pem  request.timeout.ms=20000  rest.extension.classes=io.confluent.connect.security.ConnectSecurityExtension  rest.servlet.initializor.classes=io.confluent.common.security.jetty.initializer.InstallBearerOrBasicSecurityHandler  retry.backoff.ms=500  sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \  metadataServerUrls="http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091" \  username="denodo" password="vertica\_4U" \  publicKeyPath="/etc/kafka/certs/public.pem";  sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.TokenUserLoginCallbackHandler  sasl.mechanism=OAUTHBEARER  security.protocol=SASL\_PLAINTEXT  status.storage.replication.factor=1  status.storage.topic=connect-status  ###############VALUE CONVERTER########  #value.converter=org.apache.kafka.connect.json.JsonConverter  value.converter=io.confluent.connect.avro.AvroConverter  value.converter.schema.registry.basic.auth.credentials.source=denodo  value.converter.schema.registry.basic.auth.user.info=denodo:vertica\_4U  value.converter.schema.registry.url=http://10.10.12.241:8081  value.converter.schemas.enable=false  rest.advertised.host.port=8083 |

**Giải thích cấu hình:**

* Key.convert.schema.enable: để chuyển đổi giữa định dạng Kafka Connect và định dạng được ghi vào Kafka. Kiểm soát Key message.

|  |  |
| --- | --- |
| Default | Update |
| True | True |

* Value.converter.schema.enable: để chuyển đổi giữa định dạng Kafka Connect và định dạng được ghi vào Kafka. Kiểm soát Value message.

|  |  |
| --- | --- |
| Default | Update |
| True | True |

* Offset.flush.interval.ms: Khoảng thời gian để Confluent Kafka Connect commit offset trên các task.
* Plugin.path: đường dẫn tới thư mục chứa các thư viện, connector…
* Admin.bootstrap.server: IP của confluent server. Chú ý Kafka Connect kết nối đến cụm confluent RBAC cần kết nối tới port OAuth.
* Admin.sasl.jaas.config: Cấu hình jaas cho Confluent Kafka Connect để xác thực được với Confluent Server.
* Admin.secuvity.protocol: phương thức bảo mật giữa Kafka Connect và Confluent Server.

|  |  |
| --- | --- |
| Default | Update |
| “” | SASL\_PLAINTEXT |

* Config.storage.replication.factor: cấu hình bản sao topic Kafka Connect.
* Confluent.metadata.basic.auth.user.info:user và password sử dụng để kết nối đến Confluent Server.
* Rest.advertised.host.port: khai báo port sử dụng adventise listener.

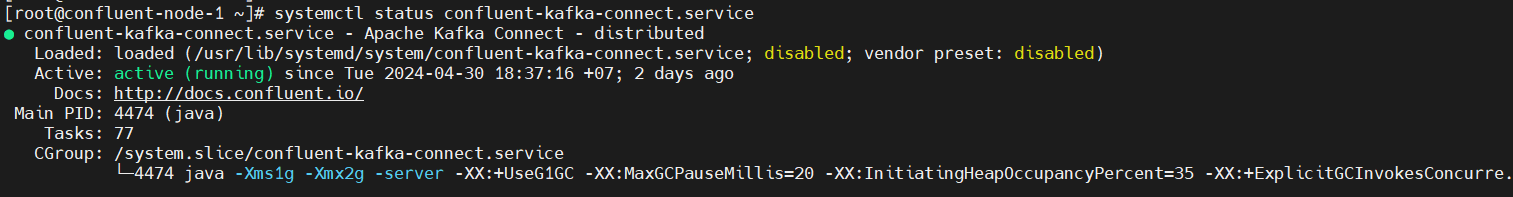
Bước 5: Khởi động Confluent Kafka Connect

Bảng 3. 6 Bảng khởi chạy Confluent Kafka Connect

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ | Câu lệnh systemctl |
| Khởi động Confluent Kafka Connect | 10.10.12.241 | systemctl start confluent-kafka-connect.service |
| Kiểm tra log | 10.10.12.241 | tail -1000f /data/confluent/apps/connect/logs/connect.log |

Kiểm tra trạng thái của Confluent Kafka Connect:

|  |
| --- |
| systemctl status confluent-kafka-connect.service |



Hình 3. 13 Trạng thái bật thành công service Kafka Connect

### 3.2.8 Cấu hình KsqlDB

Bước 1: Chuẩn bị thư mục lưu log và dữ liệu cho Confluent KsqlDB

1. Tạo thư mục dùng để chứa log và dữ liệu Confluent KsqlDB

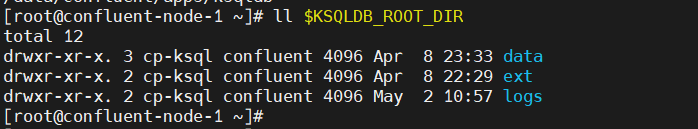
|  |
| --- |
| mkdir -p $KSQLDB\_ROOT\_DIR/{data,logs,ext} |

1. Gán quyền read/write cho user cp-ksql vào thư mục vừa tạo

|  |
| --- |
| chown -R cp-ksql:confluent $KSQLDB\_ROOT\_DIR |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $KSQLDB\_ROOT\_DIR |



Hình 3. 14 Kiểm tra quyền thư mục ksqldb

Bước 2: Cấu hình lại file service cho Confluent KsqlDB:

1. Backup tệp confluent-ksqldb-service

|  |
| --- |
| cp /usr/lib/systemd/system/confluent-ksqldb.service /usr/lib/systemd/system/confluent-ksqldb.service.backup |

1. Thêm cấu hình vào tệp service cho Server

|  |
| --- |
| sed -i 's|^Env.\*|Environment="LOG\_DIR='$KSQLDB\_ROOT\_DIR'/logs" "KSQL\_HEAP\_OPTS=-Xmx15G"|' /usr/lib/systemd/system/confluent-ksqldb.service |

1. Reload lại tệp service

|  |
| --- |
| systemctl daemon-reload |

Bước 3: Tạo Private key và public key

1. Tạo thư mục chứa private key và public key

|  |
| --- |
| mkdir -p $KSQLDB\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép tệp private key

|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/private.pem $KSQLDB\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép tệp public key

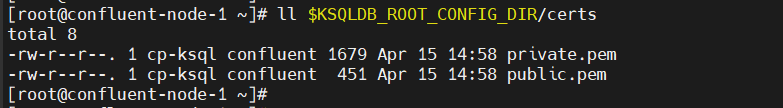
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $KSQLDB\_ROOT\_CONFIG\_DIR/certs |

1. Grant quyền cho user cp-kafka

|  |
| --- |
| chown -R cp-kafka: $BROKER\_ROOT\_CONFIG\_DIR/certs |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $KSQLDB\_ROOT\_CONFIG\_DIR/certs |



Hình 3. 15 Kiểm tra quyền thư mục certs của ksqldb

Bước 4: Cấu hình Confluent Ksqldb

1. Backup cấu hình default của Confluent Ksqldb

|  |
| --- |
| mv $KSQLDB\_ROOT\_CONFIG\_DIR/ksql-server.properties $KSQLDB\_ROOT\_CONFIG\_DIR/ksql-server.properties.backup |

1. Tạo file cấu hình mới cho Server

|  |
| --- |
| vi $KSQLDB\_ROOT\_CONFIG\_DIR/ksql-server.properties |

1. Thêm cấu hình vào tệp Server

|  |
| --- |
| ###CONFIG RBAC####  listeners=http://0.0.0.0:8088  ksql.logging.processing.topic.auto.create=true  ksql.logging.processing.stream.auto.create=true  producer.interceptor.classes=io.confluent.monitoring.clients.interceptor.MonitoringProducerInterceptor  consumer.interceptor.classes=io.confluent.monitoring.clients.interceptor.MonitoringConsumerInterceptor  state.dir=/data/confluent/apps/ksqldb/data/kafka-streams  bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  security.protocol=SASL\_PLAINTEXT  sasl.mechanism=OAUTHBEARER  sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.TokenUserLoginCallbackHandler  sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required \  metadataServerUrls="http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091" \  username="denodo" password="vertica\_4U" \  publicKeyPath="/etc/ksqldb/certs/public.pem";  ksql.security.extension.class=io.confluent.ksql.security.KsqlConfluentSecurityExtension  public.key.path=/etc/ksqldb/certs/public.pem  confluent.metadata.bootstrap.server.urls=http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091  confluent.metadata.basic.auth.user.info=denodo:vertica\_4U  confluent.metadata.http.auth.credentials.provider=BASIC  ksql.schema.registry.url=http://10.10.12.241:8081  ksql.schema.registry.basic.auth.credentials.source=USER\_INFO  ksql.schema.registry.basic.auth.user.info=denodo:vertica\_4U  ksql.extension.dir=/usr/share/ksqldb\_udf  ksql.functions.\_global\_.schema.registry.url=http://10.10.12.241:8081  ksql.functions.\_global\_.schema.registry.basic.auth.credentials.source=USER\_INFO  ksql.functions.\_global\_.schema.registry.basic.auth.user.info=denodo:vertica\_4U  ksql.authentication.plugin.class=io.confluent.ksql.security.VertxBearerOrBasicAuthenticationPlugin  ksql.access.validator.enable=on  ksql.connect.url=http://10.10.12.241:8083 |

**Giải thích cấu hình:**

* Listeners: cấu hình listencer cho Confluent KsqlDB

|  |  |
| --- | --- |
| Default | Update |
| “” | http://0.0.0.0:8088 |

* Ksql.logging.processing.stream.auto.create: tự tạo Topic Stream
* Producer.interceptor.classes: cấu hình class interceptor để monitor cho Confluent Control Center
* State.dir:đường dẫn lưu dữ liệu Confluent KsqlDB

|  |  |
| --- | --- |
| Default | Update |
| /tmp/kafka-streams | /data/confluent/apps/ksqldb/data |

* Security.protocol: phương thức bảo mật giữa Confluent Ksqldb và Confluent Server
* Sasl.mechanism: cơ chế SASL giữa các kết nối của Confluent KsqlDB tới Confluent Server
* Ksql.extension.dir: đường dẫn lưu dữ liệu Confluent KsqlDB
* Ksql.function.\_global\_.schema.registry.url: địa chỉ kết nối đến Confluent Schema Registry
* Ksql.function.\_global\_.schema.registry.basic.auth.user.info: cấu hình username:password kết nối đến Confluent Schema Registry.
* Ksql.access.validator.enable:cấu hình kết nối tới cụm Confluent Server

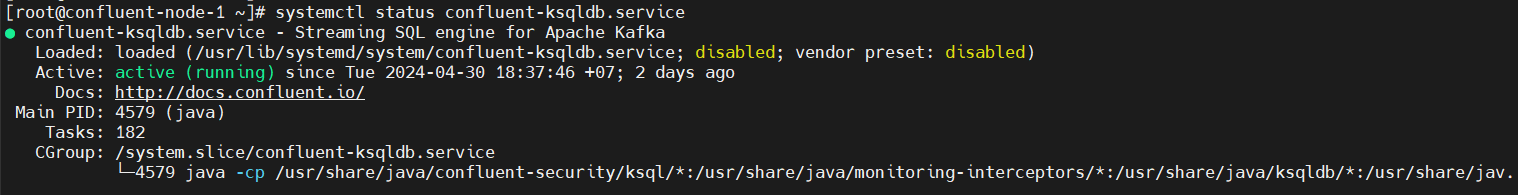
Bước 5: Khởi chạy Confluent KsqlDB service:

Bảng 3. 7 Bảng khởi chạy KsqlDB

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ | Câu lệnh |
| Khởi động Confluent Ksqldb | 10.10.12.241 | systemctl start confluent-ksqldb.service |
| Kiểm tra log Confluent ksqldb | 10.10.12.241 | tail -1000f /data/confluent/apps/ksqldb/log/ksqldb.log |

Kiểm tra trạng thái của Confluent KsqlDB service:

|  |
| --- |
| systemctl status confluent-ksqldb.service |



Hình 3. 16 Trạng thái bật thành công service KsqlDB

### 3.2.9 Cấu hình Control Center

Bước 1: Chuẩn bị thư mục lưu log và dữ liệu cho Confluent Control Center

1. Tạo thư mục dùng để chứa log và dữ liệu Confluent Control Center

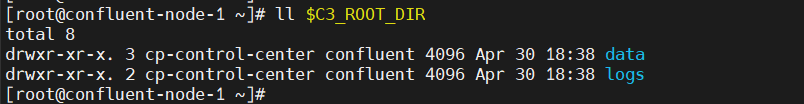
|  |
| --- |
| mkdir -p $C3\_ROOT\_DIR/{data,logs} |

1. Gán quyền read/write cho user cp-control-center vào thư mục vừa tạo

|  |
| --- |
| chown -R cp-control-center:confluent $C3\_ROOT\_DIR |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $C3\_ROOT\_DIR |



Hình 3. 17 Kiểm tra quyền thư mục control-center

Bước 2: Cấu hình lại file service cho Confluent Control Center:

1. Backup file confluent-control-center-service

|  |
| --- |
| cp /usr/lib/systemd/system/confluent-control-center.service /usr/lib/systemd/system/confluent-control-center.service.backup |

1. Thêm cấu hình vào file service cho Server

|  |
| --- |
| sed -i 's|LOG\_DIR=.\*" |LOG\_DIR='$C3\_ROOT\_DIR'/logs" |' /usr/lib/systemd/system/confluent-control-center.service |

1. Reload lại file service

|  |
| --- |
| systemctl daemon-reload |

Bước 3: Tạo Private key và public key

1. Tạo thư mục chứa private key và public key

|  |
| --- |
| mkdir -p $C3\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép file private key

|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/private.pem $C3\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép file public key

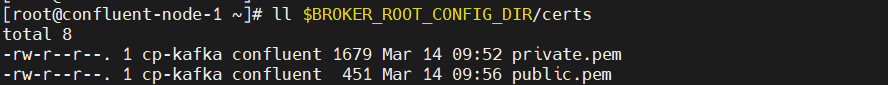
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $C3\_ROOT\_CONFIG\_DIR/certs |

1. Grant quyền cho user cp-control-center

|  |
| --- |
| chown -R cp-control-center: $3\_ROOT\_CONFIG\_DIR/certs |

1. Kiểm tra lại quyền

|  |
| --- |
| ll $BROKER\_ROOT\_CONFIG\_DIR/certs |



Hình 3. 18 Kiểm tra quyền thư mục certs của control center

Bước 4: Cấu hình Confluent Control Center

1. Backup cấu hình default của Confluent Control Center

|  |
| --- |
| mv $C3\_ROOT\_CONFIG\_DIR/control-center-production.properties $C3\_ROOT\_CONFIG\_DIR/control-center-production.properties.backup |

1. Tạo file cấu hình mới cho Confluent Control Center

|  |
| --- |
| vi $C3\_ROOT\_CONFIG\_DIR/control-center-production.properties |

1. Thêm cấu hình vào file Confluent Control Center

|  |
| --- |
| #CONFIG RBAC####  config.providers=file  config.providers.file.class=org.apache.kafka.common.config.provider.FileConfigProvider  bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  confluent.controlcenter.ksql.ksql-cluster.url=http://10.10.12.241:8088  confluent.controlcenter.auth.bearer.public.key.path=/etc/confluent-control-center/certs/public.pem  confluent.controlcenter.command.topic.partitions=1  confluent.controlcenter.command.topic.replication=1  confluent.controlcenter.data.dir=/data/confluent/apps/control-center/data  confluent.controlcenter.id=bnh  confluent.controlcenter.internal.topics.partitions=1  confluent.controlcenter.internal.topics.replication=1  confluent.controlcenter.metadata.password=${file:/etc/confluent-control-center/secret.properties:cluster.super.user.credentials}  confluent.controlcenter.metadata.urls=http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091  confluent.controlcenter.metadata.username=${file:/etc/confluent-control-center/secret.properties:cluster.super.user.username}  confluent.controlcenter.rest.authentication.method=BEARER  confluent.controlcenter.schema.registry.enable=true  confluent.controlcenter.schema.registry.url=http://10.10.12.241:8081  confluent.controlcenter.schema.registry.basic.auth.user.info=denodo:vertica\_4U  confluent.controlcenter.streams.num.stream.threads=1  confluent.controlcenter.streams.sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required metadataServerUrls="http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091" username="${file:/etc/confluent-control-center/secret.properties:cluster.super.user.username}" password="${file:/etc/confluent-control-center/secret.properties:cluster.super.user.credentials}" publicKeyPath="/etc/confluent-control-center/certs/public.pem";  confluent.controlcenter.streams.sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.TokenUserLoginCallbackHandler  confluent.controlcenter.streams.sasl.mechanism=OAUTHBEARER  confluent.controlcenter.streams.security.protocol=SASL\_PLAINTEXT  confluent.controlcenter.ui.autoupdate.enable=false  confluent.controlcenter.usage.data.collection.enable=false  confluent.license="${file:/etc/confluent-control-center/secret.properties:confluent.license}"  confluent.metadata.basic.auth.user.info=${file:/etc/confluent-control-center/secret.properties:user.info}  confluent.metadata.bootstrap.server.urls=http://10.10.12.241:8091,http://10.10.12.242:8091,http://10.10.12.244:8091  confluent.metrics.topic.partitions=1  confluent.metrics.topic.replication=1  confluent.metrics.topic=\_confluent-metrics  public.key.path=/etc/confluent-control-center/certs/public.pem  zookeeper.connect=10.10.12.241:2181,10.10.12.242:2181,10.10.12.244:2181  confluent.controlcenter.connect.connect-cluster.cluster=http://10.10.12.241:8083  confluent.telemetry.enabled=false |

**Giải thích cấu hình:**

* Config.providers: cấu hình đọc file config cho Confluent Control Center

|  |  |
| --- | --- |
| Default | Update |
| “” | file |

* Bootstrap.servers: Địa chỉ IP của Confluent Control Center
* Confluent.controlcenter.ksql.ksql-cluster.url: cấu hình địa chỉ IP Confluent KsqlDB

|  |  |
| --- | --- |
| Default | Update |
| “” | http://10.10.12.241:8088 |

* Confluent.controlcenter.command.topic.replication: Số bản sao cho topic command của Confluent Control Center

|  |  |
| --- | --- |
| Default | Update |
| 1 | 3 |

* Confluent.controlcenter.data.dir: đường dẫn lưu dữ liệu
* Confluent.controlcenter.id: cấu hình tên id cho Confluent Control Center
* Confluent.controlcenter.internal.topics.partitions: cấu hình partition nội tại của Confluent Control Center

1. Tạo file chứa cấu hình RBAC cho Confluent Control Center

|  |
| --- |
| Vi $C3\_ROOT\_CONFIG\_DIR/secret.properties |

1. Thêm nội dung sau vào file secret.properties

|  |
| --- |
| cluster.super.user.credentials=denodo  cluster.super.user.username=vertica\_4U  user.info=denodo:vertica\_4U  confluent.license=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJjb250cm9sLWNlbnRlciIsIm5iNCI6IjE2OTU2MTM1ODEiLCJtb25pdG9yaW5nIjp0cnVlLCJsaWNlbnNlVHlwZSI6IkVudGVycHJpc2UiLCJpc3MiOiJDb25mbHVlbnQiLCJpYXQiOjE2OTM1NTE2MDAsImV4cCI6MTc5NTkzNTYwMCwiYXVkIjoiMDA2NFUwMDAwMHNxallQUUFZIn0=.Rjm0el5v7BKbC\_jcu\_lmuHQeoAhKQ3xP4MM1KSl8wo7859OogL954nO2BbNos3OwplXbvuu7\_IsSFyf8UBFK2zUtHV8elCfcIcAxu0uE-pMhYZoBuq\_OXePvrBfRs8TaJOzXq6CcdSrkekL-Ym7epXOO4YOWQzJXLU6GzVcOVDv8bTMEYtVHw-lE3nOYa7rLci9UiKKWex5u4hLdCJMvSNIFhOz5xm5WcGhhSv8mpZJapaVfcxJ5585\_xQ-w\_0QnG-mcLKDhNOkBfG97DId93KC4MIKQe1Cgs-zFpGXqgNajg75aqNxj7WgPcqrrbJSWIirFgD1IKR2oXaXR2m8FWw |

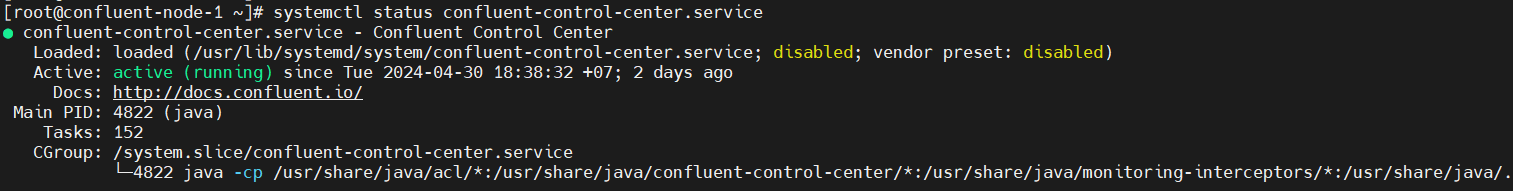
Bước 5: Khởi chạy Confluent Control Center Service:

Bảng 3. 8 Bảng khởi chạy Confluent Control Center

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ | Câu lệnh |
| Khởi chạy confluent control-center | 10.10.12.241 | systemctl start confluent-control-center.service |
| Kiểm tra log control center | 10.10.12.241 | tail -1000f /data/confluent/apps/control-center/logs/control-center.log |

Kiểm tra trạng thái của Confluent Control Center Service:

|  |
| --- |
| systemctl status confluent-control-center.service |



Hình 3. 19 Trạng thái bật thành công service Control Center

Bước 6: Tạo Super User

|  |
| --- |
| kafka-configs --zookeeper 10.10.12.241 --alter --add-config 'SCRAM-SHA-256=[password=vertica\_4U],SCRAM-SHA-512=[password=vertica\_4U]' --entity-type users --entity-name denodo |

|  |
| --- |
| kafka-configs --bootstrap-server 10.10.12.241:9092 --alter --add-config 'SCRAM-SHA-256=[password=vertica\_4U],SCRAM-SHA-512=[password=vertica\_4U]' --entity-type users --entity-name denodo |

Bước 7: Grant quyền cho User denodo:

1. Login metadata service

|  |
| --- |
| confluent login --url http://10.10.12.241:8091 |

1. Grant quyền cho user AD/LDAP

|  |
| --- |
| confluent iam rbac role-binding create --principal User:denodo --role SystemAdmin --kafka-cluster $kafka\_cluster --ksql-cluster connect-cluster |

|  |
| --- |
| confluent iam rbac role-binding create --principal User:denodo --role SystemAdmin --kafka-cluster $kafka\_cluster --connect-cluster default\_ |

|  |
| --- |
| confluent iam rbac role-binding create --principal User:denodo --role SystemAdmin --kafka-cluster $kafka\_cluster --schema-registry-cluster schema-registry |

## 3.3 Tích hợp MinIO

### 3.3.1 Thông tin kết nối máy chủ MinIO

<http://10.10.12.241:9001>

### 3.3.2 Quy trình triển khai tích hợp MinIO

Bước 1: Tải và cài đặt

1. Tạo thư mục chứa tệp cài đặt

|  |
| --- |
| mkdir /data/minio |

1. Tải tệp cài đặt minio và giải nén

|  |
| --- |
| wget [https://dl.min.io/server/minio/release/linux-amd64/archive/minio-20240330094156.0.0-1.x86\_64.rpm -O minio.rpm](https://dl.min.io/server/minio/release/linux-amd64/archive/minio-20240330094156.0.0-1.x86_64.rpm%20-O%20minio.rpm) |

|  |
| --- |
| rpm -ivh minio.rpm |

Bước 2: Cấu hình tích hợp vào Confluent Platform

1. Thêm cấu hình MinIO vào tệp server.properties

|  |
| --- |
| confluent.tier.feature=true  confluent.tier.enable=true  confluent.tier.backend=S3  confluent.tier.s3.bucket=confluent  confluent.tier.s3.region=ap-southeast-1  confluent.tier.s3.aws.endpoint.override=http://10.10.12.241:9000  confluent.tier.s3.force.path.style.access=true  confluent.tier.s3.cred.file.path=/etc/kafka/s3/keyfile  confluent.tier.s3.sse.algorithm=none |

1. Tạo file chứa khóa truy cập

|  |
| --- |
| mkdir -p /etc/kafka/s3/keyfile |

1. Thêm cấu hình cho tệp keyfile

|  |
| --- |
| [default]  accessKey=FCaXQ4Oi9ZB7XpHKeLdd  secretKey=b55sme2YVS2XAwgFT5v8P4f6yMH9Ol81NG0TkZf5 |

Bước 3: Khởi chạy MinIO

1. Khởi confluent server service

|  |
| --- |
| systemctl restart confluent server |

1. Khởi chạy minIO

|  |
| --- |
| minio server /data/minio --console-address :9001 |

1. Truy cập domain : 10.10.12.241:9001

## 3.4 Tích hợp Prometheus và Grafana

### 3.4.1 Triển khai Prometheus

Bước 1: Cài gói JMX để trích xuất dữ liệu của Kafka

1. Tạo thư mục chứa tệp cài đặt

|  |
| --- |
| mkdir /data/prometheus/ |

1. Tải tệp cài đặt Prometheus:

|  |
| --- |
| wget https://repo1.maven.org/maven2/io/prometheus/jmx/jmx\_prometheus\_javaagent/0.20.0/jmx\_prometheus\_javaagent-0.20.0.jar |

1. Cấu hình trong file kafka\_broker.yml:
2. Thêm cấu hình vào tệp kafka-server-start:

|  |
| --- |
| sed -i '/EXTRA\_ARGS=${EXTRA\_ARGS-'\''-name kafkaServer -loggc'\''}/a\EXTRA\_ARGS=-javaagent:/data/prometheus/jmx\_prometheus\_javaagent-0.20.0.jar=1234:/data/prometheus/kafka\_broker.yml' /bin/kafka-server-start |

1. Khởi động lại server:

|  |
| --- |
| systemctl daemon-reload  systemctl restart confluent-server |

(Sau khi khởi động lại server thành công, kiểm tra metric trên domain 10.10.12.241:1234)

Làm tương tự với 2 node còn lại, kiểm tra metric trên 2 domain: 10.10.12.242:1234, 10.10.12.244:1234.

Bước 2: Cài đặt prometheus:

1. Tải và cài đặt Prometheus:

|  |
| --- |
| wget <https://github.com/prometheus/prometheus/releases/download/v2.45.3/prometheus-2.45.3.linux-amd64.tar.gz> |

|  |
| --- |
| tar xzf prometheus-2.45.3.linux-amd64.tar.gz |

1. Thêm cấu hình sau vào tệp Prometheus.yml

|  |
| --- |
| static\_configs:  - targets: ["10.10.12.241:9090"]  - job\_name: "kafka"  static\_configs:  - targets: ["10.10.12.241:1234"]  - targets: ["10.10.12.242:1234"]  - targets: ["10.10.12.244:1234"]  label:  env: "dev" |

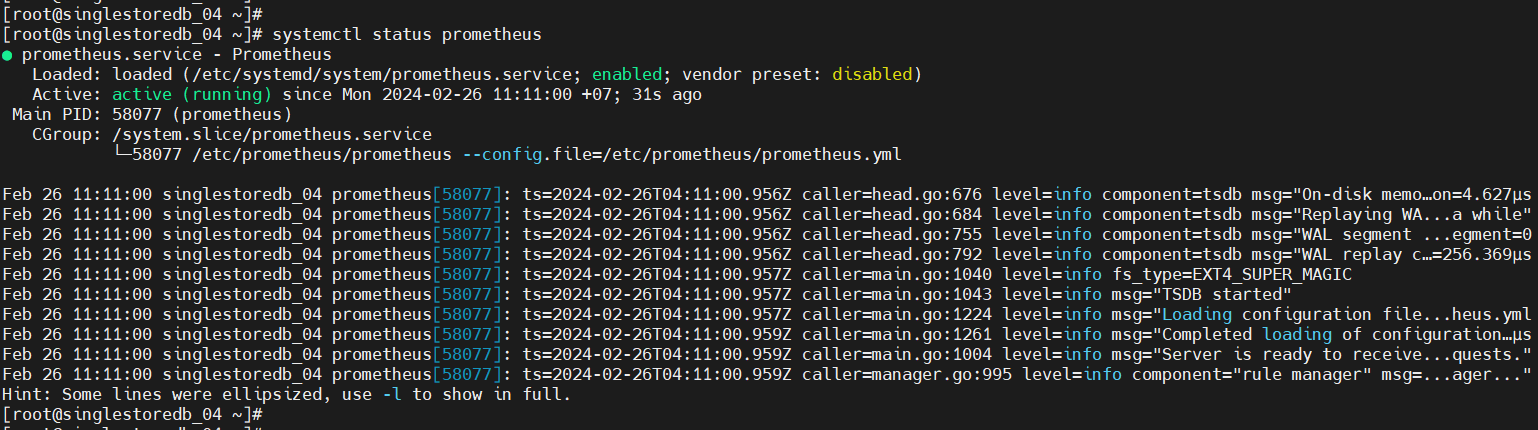
1. Tạo tệp Prometheus.service và thêm cấu hình.

|  |
| --- |
| vi /etc/systemd/system/prometheus.service |

|  |
| --- |
| [Unit]  Description=Prometheus  Wants=network-online.target  After=network-online.target  [Service]  ExecStart=/opt/prometheus/prometheus-2.45.3.linux-amd64/prometheus --config.file=/opt/prometheus/prometheus-2.45.3.linux-amd64/prometheus.yml  Restart=always  [Install]  WantedBy=multi-user.target |

1. Khởi động Prometheus:

|  |
| --- |
| systemctl daemon-reload  systemctl restart prometheus  systemctl enable prometheus  systemctl status prometheus |



Hình 3. 20 Trạng thái bật thành công của service Prometheus

Truy cập Prometheus qua domain: 10.10.12.241:9090.

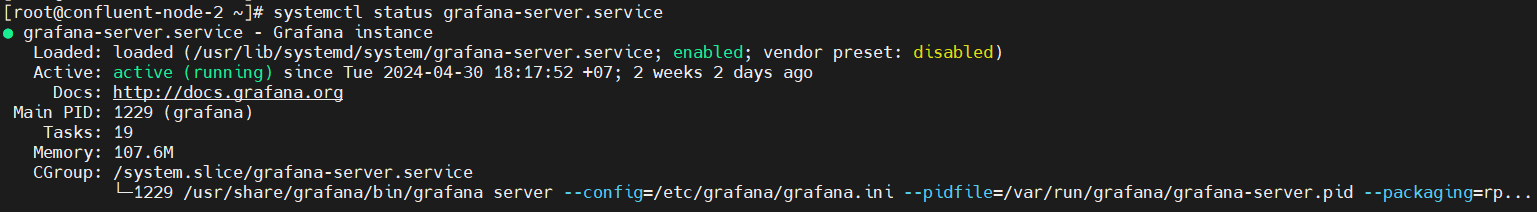
### 3.4.2 Triển khai Grafana:

1. Tải và cài đặt Grafana

|  |
| --- |
| yum install -y <https://dl.grafana.com/enterprise/release/grafana-enterprise-10.3.3-1.x86_64.rpm> |

1. Khởi chạy Grafana:

|  |
| --- |
| systemctl daemon-reload  systemctl restart grafana-server.service  systemctl enable grafana-server  systemctl status grafana-server.service |

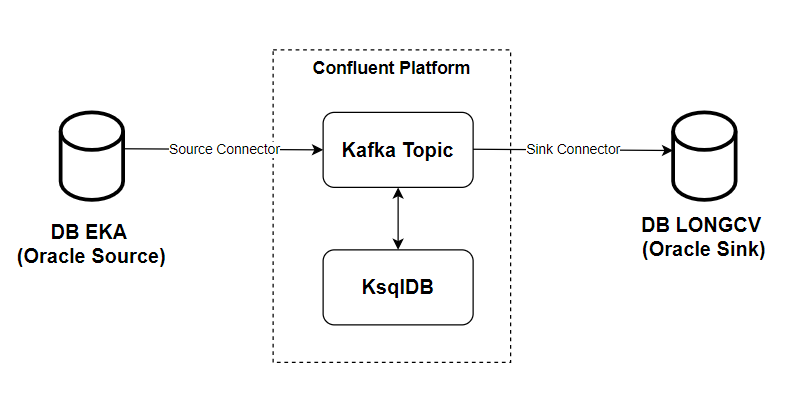


Hình 3. 21 Trạng thái bật thành công của service Grafana

Truy cập Grafana qua domain: 10.10.12.242:3000.

**3.5 Tạo luồng Streaming và xử lý dữ liệu**

### 3.5.1 Mô hình luồng streaming



Hình 3. 22 Mô hình luồng stream

### 3.5.2 Mô tả cách hoạt động của luồng

1. Confluent kết nối tới cơ sở dữ liệu nguồn bằng Source Connector OracleCdcSourceConnector. Cơ sở dữ liệu nguồn produce Message tới Kafka Topic.

2. Dữ liệu được đưa vào Confluent KsqlDB để xử lý.

3. Confluent KsqlDB trả dữ liệu đã xử lý vào 1 Kafka Topic mới.

4. Sink Connector JdbcSinkConnector kết nối với cơ sở dữ liệu đích, cơ sở dữ liệu đích consume Message từ Kafka Topic đến đích.

### 3.5.3 Giải thích cơ chế hoạt động của Confluent Connector

#### 3.5.3.1 Source Connector OracleCdcSourceConnector

OracleCdcSourceConnector là một Source Connector trong Confluent Platform được sử dụng để kết nối và đồng bộ dữ liệu từ cơ sở dữ liệu Oracle nguồn tới Kafka Topic.

Đây là một CDC - Change Data Capture cho phép theo dõi phát hiện các thay đổi dữ liệu trong cơ sở dữ liệu Oracle nguồn. LogMiner – Công cụ của Oracle, cho phép truy cập redo-log nơi mà các thay đổi dữ liệu được ghi lại. Connector sẽ đọc redo-log, xác định các thay đổi dữ liệu để tiếp tục xử lý.

Connector chuyển đổi mỗi thay đổi dữ liệu thành Message tương ứng rồi gửi vào Kafka Topic được chỉ định. Các thay đổi dữ liệu bao gồm việc chèn (insert), cập nhật (update), xóa (delete).

#### 3.5.3.2 Sink Connector JdbcSinkConnector

JdbcSinkConnector là một Sink Connector trong Confluent Platform được sử dụng để kết nối và đồng bộ dữ liệu từ Kafka Topic tới cơ sở dữ liệu Oracle đích thông qua JDBC (Java Database Connectivity).

Connector chuyển đổi các Message trong Kafka Topic từ định dạng Kafka thành các bản ghi phù hợp với cấu trúc cơ sở dữ liệu đích. JDBC cung cấp phương thức cho phép thực hiện hoạt động chèn (insert), cập nhật (update) dữ liệu vào bảng cơ sở dữ liệu đích.

### 3.5.4 Các bước thực hiện tạo luồng Streaming và xử lý dữ liệu.

Trong thực tế, đối với một hệ thống phức tạp, dữ liệu đầu vào của hệ thống này sẽ là dữ liệu đầu ra của hệ thống khác ví dụ như:

* Sinh viên cập nhật thông tin trên website của nhà trường. Thông tin đó sẽ là đầu vào của cơ sở dữ liệu. các application lấy thông tin từ cơ sở dữ liệu để phục vụ các hoạt động khác.
* Yêu giao dịch cổ phiếu của từ một sàn giao dịch. Sàn giao dịch gửi thông tin giao dịch cho hệ thống tài chính xác minh và xử lý. Hệ thống tài chính trả kết quả lại cho các sàn giao dịch.

Tuy nhiên, để đơn giản hóa quá trình xây dựng đồ án, dữ liệu đầu vào của hệ thống Confluent Platform là dữ liệu được đinh nghĩa sẵn.

Định dạng dữ liệu đầu vào: Dạng chuỗi kí tự VARCHAR.

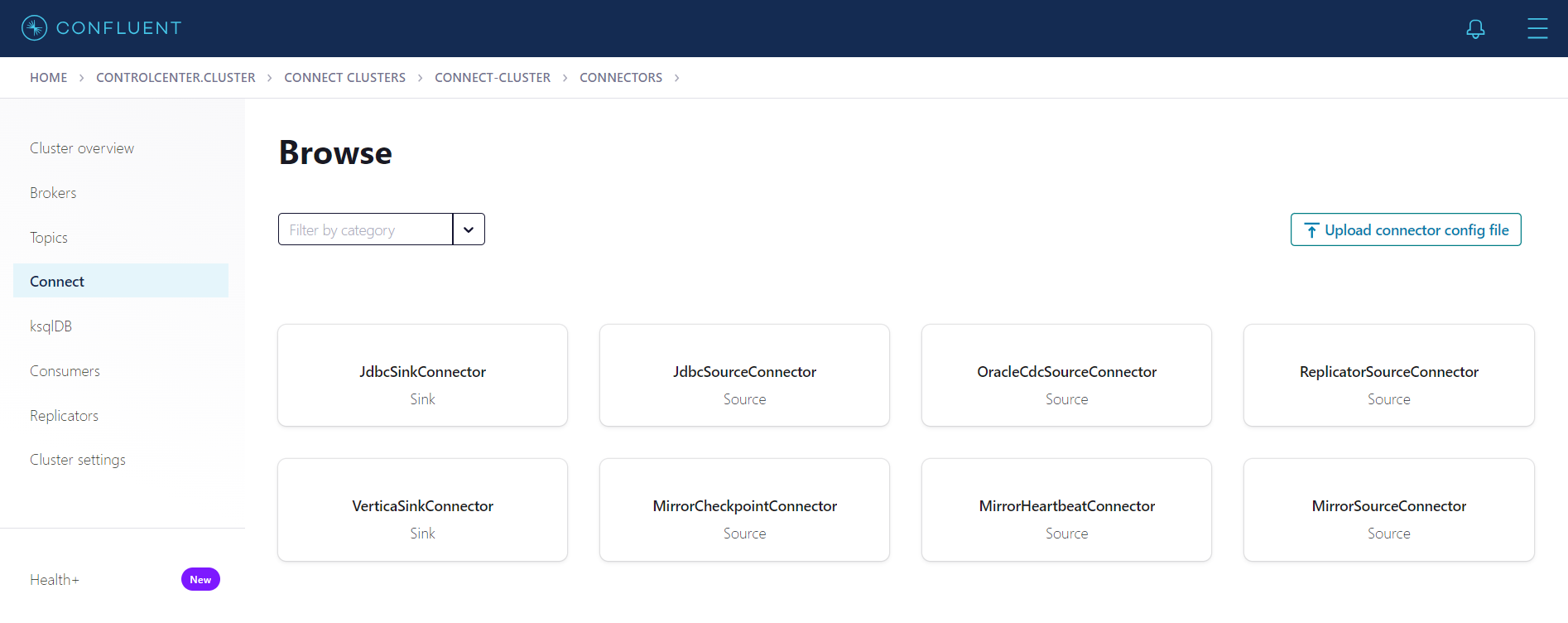
Định dạng dữ liệu đầu ra: Dạng chuỗi kí tự VARCHAR.

Bước 1: Gửi dữ liệu từ cơ sở dữ liệu nguồn EKA (Oracle) vào Kafka Topic

1. Chuẩn bị dữ liệu ở nguồn:

|  |
| --- |
| CREATE TABLE TRANSACTIONS\_SOURCE (  TRANSACTION\_ID VARCHAR(50) PRIMARY KEY,  TRANSACTION\_DATE VARCHAR(50),  ACCOUNT VARCHAR(50),  TRANSACTION\_TYPE VARCHAR(50),  AMOUNT VARCHAR(50),  DESCRIPTION VARCHAR(50)  );  INSERT INTO Transactions (TRANSACTION\_ID, TRANSACTION\_DATE, ACCOUNT, TRANSACTION\_TYPE, AMOUNT, DESCRIPTION)  VALUES  ('1', '2024-05-01', '123456789', 'Deposit', '1000', 'Initial deposit'),  ('2', '2024-05-02', '987654321', 'Withdrawal', '500', 'ATM withdrawal'),  ('3', '2024-05-03', '456789123', 'Transfer', '200', 'Transfer to friend'),  ('4', '2024-05-04', '789123456', 'Payment', '300', 'Utility bill payment'),  ('5', '2024-05-05', '321654987', 'Deposit', '1500', 'Salary deposit'),  ('6', '2024-05-06', '654987321', 'Withdrawal', '700', 'Cash withdrawal'),  ('7', '2024-05-07', '987321654', 'Transfer', '400', 'Transfer to family'),  ('8', '2024-05-08', '159753486', 'Payment', '200', 'Online purchase'),  ('9', '2024-05-09', '753159486', 'Deposit', '800', 'Bonus deposit'),  ('10', '2024-05-10', '486753159', 'Withdrawal', '100', 'Small cash withdrawal'); |

1. Tạo Source Connector OracleCdcSourceConnector trên giao diện Control Center:



Hình 3. 23 Giao diện tạo source connector

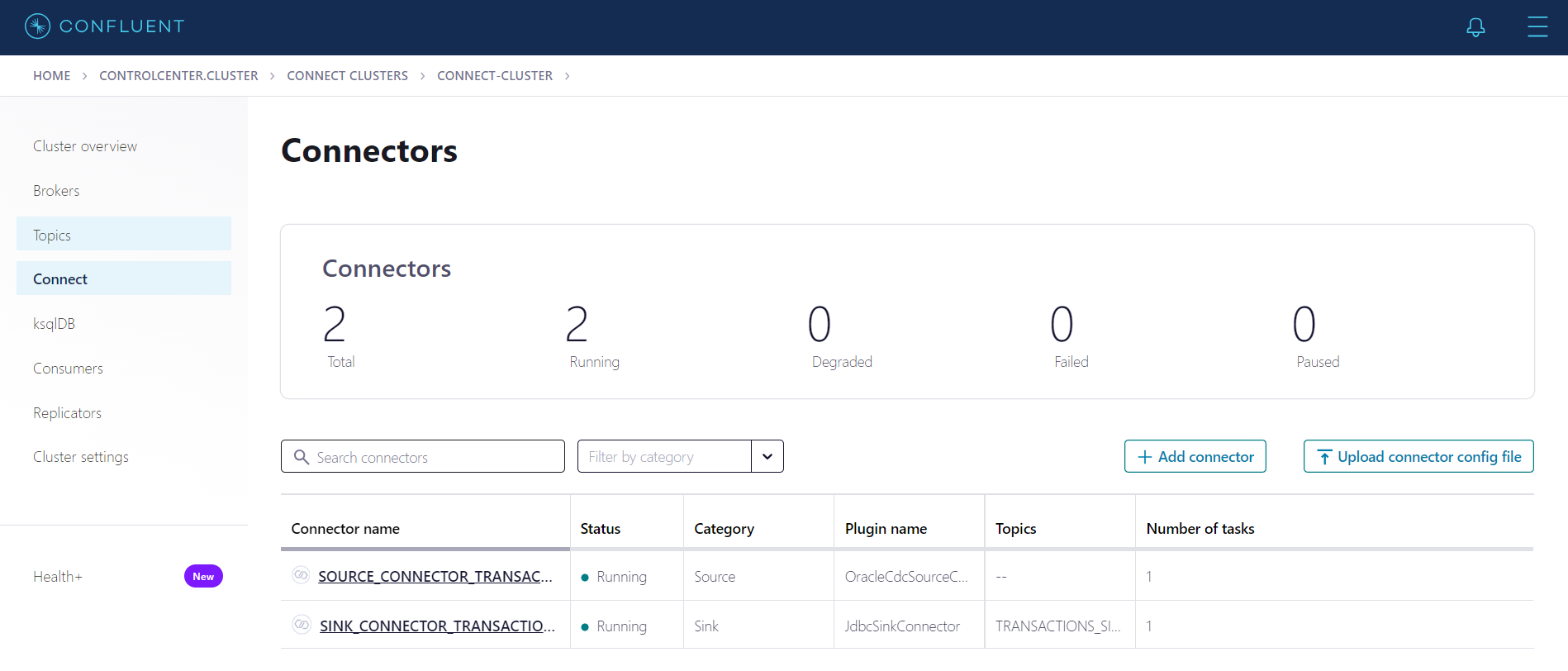
1. Cấu hình Source Connector OracleCdcSourceConnector

|  |
| --- |
| {  "name": "SOURCE\_CONNECTOR\_TRANSACTIONS",  "config": {  "connector.class": "io.confluent.connect.oracle.cdc.OracleCdcSourceConnector",  "tasks.max": "1",  "key.converter": "io.confluent.connect.avro.AvroConverter",  "value.converter": "io.confluent.connect.avro.AvroConverter",  "topic.creation.groups": "redo",  "oracle.server": "10.10.11.50",  "oracle.port": "1521",  "oracle.sid": "eka",  "oracle.username": "huynb",  "oracle.password": "oracle\_4U",  "start.from": "snapshot",  "redo.log.topic.name": "redo-log-TEST",  "redo.log.corruption.topic": "redo-corruption-TEST",  "redo.log.consumer.bootstrap.servers": "10.10.12.241:9092, 10.10.12.242:9092, 10.10.12.244:9092",  "numeric.mapping": "best\_fit\_or\_decimal",  "table.inclusion.regex": "eka.EKA.TRANSACTIONS\_SOURCE",  "table.exclusion.regex": "",  "table.topic.name.template": "TRANSACTIONS\_SOURCE\_DATA\_TOPIC",  "connection.pool.max.size": "20",  "confluent.topic.bootstrap.servers": "10.10.12.241:9092",  "confluent.topic.replication.factor": "3",  "principal.service.name": "denodo",  "principal.service.password": "vertica\_4U",  "topic.creation.default.partitions": "1",  "topic.creation.redo.partitions": "1",  "topic.creation.redo.include": "redo-include-TEST",  "topic.creation.default.replication.factor": "1",  "topic.creation.redo.retention.ms": "1209600000",  "topic.creation.redo.replication.factor": "1",  "topic.creation.default.cleanup.policy": "delete",  "topic.creation.redo.cleanup.policy": "delete",  "value.converter.schema.registry.url": "http://10.10.12.241:8081",  "key.converter.schema.registry.url": "http://10.10.12.241:8081"  }  } |

**Giải thích cấu hình:**

* Tasks.max: số tác vụ tối đa sử dụng cho Connector.
* Connector.class: tên của class cho Connector.
* Key.converter: chuyển đổi định dạng Key của Message được gửi vào Kafka.
* Value.converter: chuyển đổi định dạng Value của Message được ghi vào kafka
* Oracle.server: Địa chỉ ip của máy chủ Oracle.
* Oracle.port: cổng giao tiếp giữa connector và Oracle
* Oracle.sid: id để truy cập hệ thống Oracle.
* Oracle.username: tên đăng nhập vào server Oracle
* Oracle.password: mật khẩu đăng nhập vào server Oracle
* Redo.log.topic.name: tên Topic lưu nhật ký hành động của Oracle.
* Redo.log.consumer.bootstrap.server: máy chủ thực hiện consume Topic redo log
* Table.inclusion.regex: Regex kết nối tới Table của cơ sở dữ liệu.
* Table.topic.name.template: Tên topic nhận dữ liệu được capture từ hoạt động thay đổi của Table.
* Value.converter.schema.registry.url: Chỉ định địa chỉ url của Confluent Schema Registry.
* Key.convert.schema.registry.url: Chỉ định địa chỉ url của Confluent Schema Registry.

1. Kiểm tra trạng thái connector:

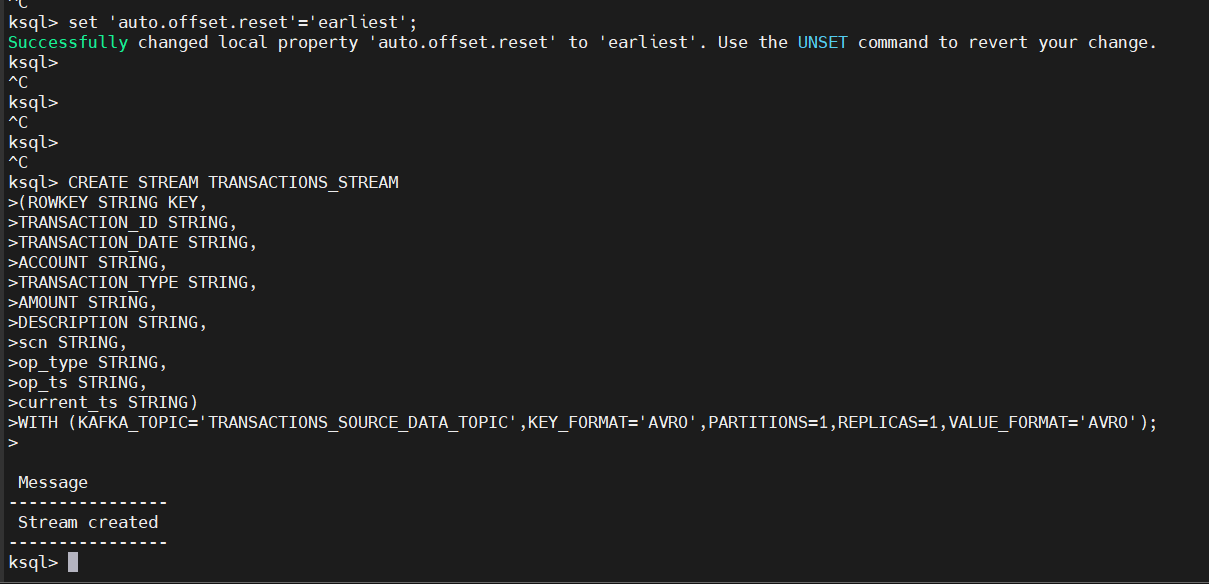


Hình 3. 24 Giao diện trạng thái của connector trên Control Center

Nếu trạng thái “Running” thì đã tạo connector thành công.

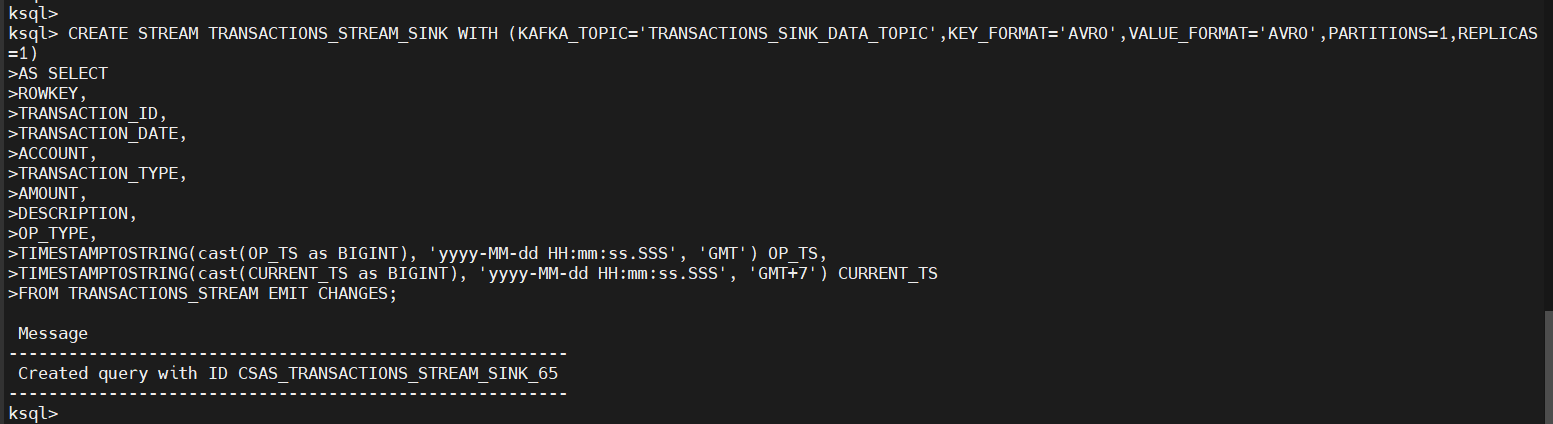
Bước 2: Sử dụng Confluent KsqlDB xử lý dữ liệu:

1. Kéo dữ liệu từ topic TRANSACTIONS\_SOURCE\_DATA\_TOPIC vào luồng Stream TRANSACTIONS\_STREAM. (Tùy chỉnh chọn những trường cần thiết)



Hình 3. 25 Create stream TRANSACTIONS\_STREAM

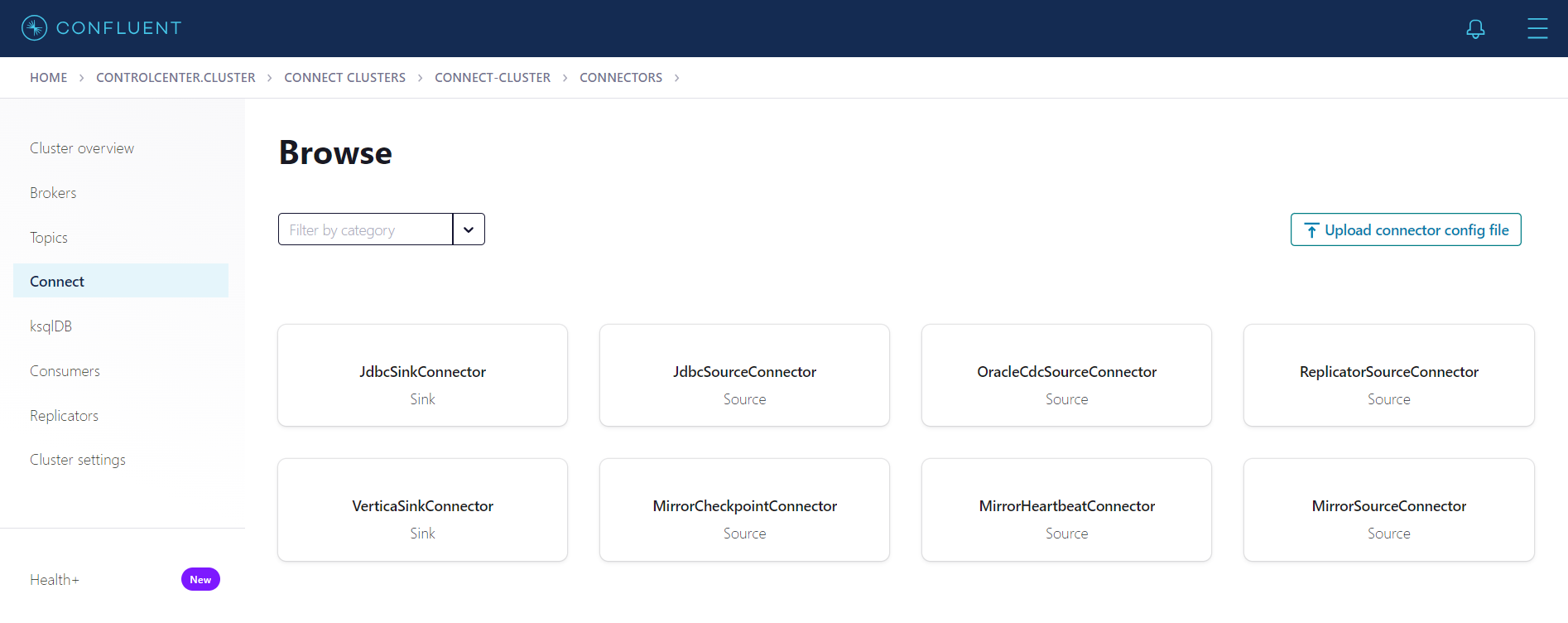
Tạo thêm luồng TRANSACTIONS\_STREAM\_SINK từ TRANSACTIONS\_STREAM để xử lý dữ liệu, kết quả đẩy vào topic mới TRANSACTIONS\_SINK\_DATA\_TOPIC.



Hình 3. 26 Create stream TRANSACTIONS\_STREAM\_SINK

Bước 3: Đẩy dữ liệu đến đích:

1. Tạo Sink Connector JdbcSinkConnector trên giao diện Control Center



Hình 3. 27 Giao diện tạo connector trên Control Center

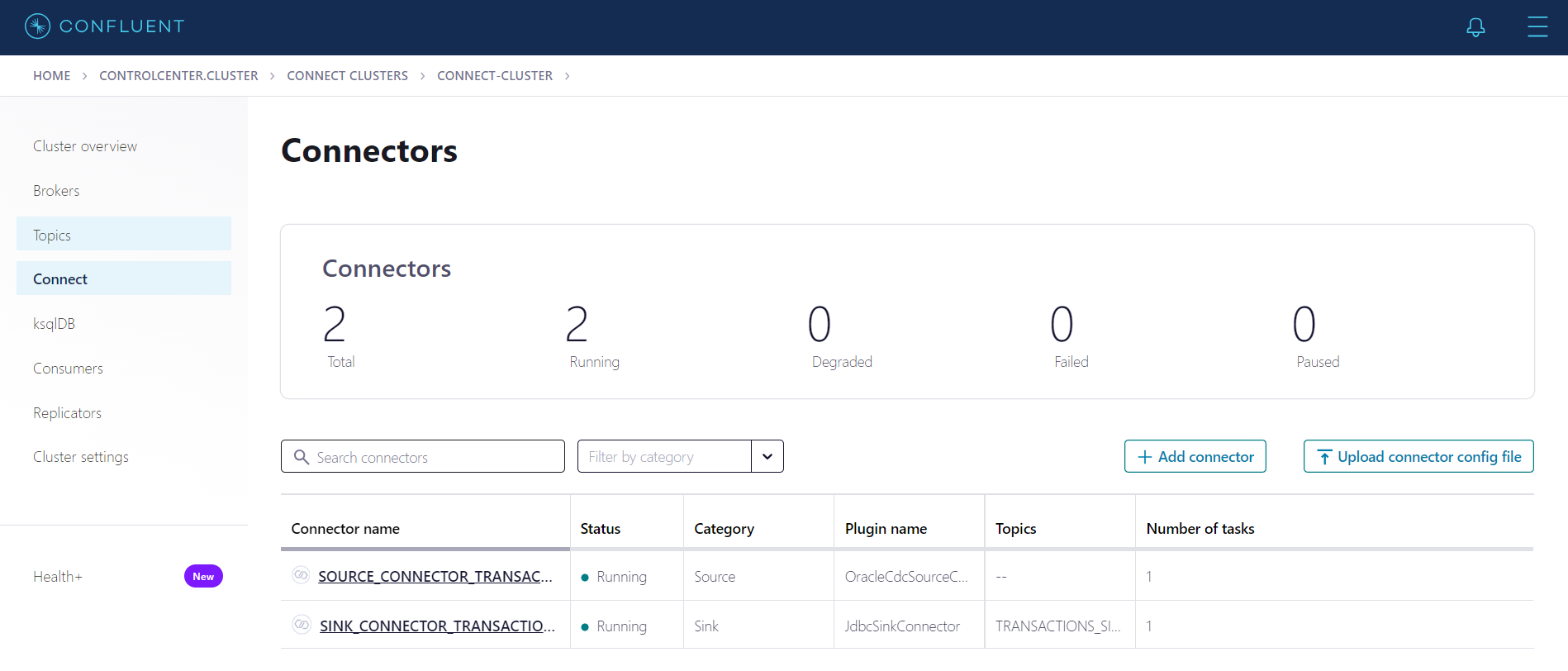
1. Cấu hình Sink Connector

|  |
| --- |
| {  "name": "SINK\_CONNECTOR\_TRANSACTIONS",  "config": {  "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",  "tasks.max": "1",  "key.converter": "io.confluent.connect.avro.AvroConverter",  "value.converter": "io.confluent.connect.avro.AvroConverter",  "transforms": "masking",  "topics": "TRANSACTIONS\_SINK\_DATA\_TOPIC",  "transforms.masking.type": "org.apache.kafka.connect.transforms.MaskField$Value",  "transforms.masking.fields": "ACCOUNT",  "transforms.masking.replacement": "\*\*\*\*\*\*\*\*\*\*",  "connection.url": "jdbc:oracle:thin:@10.10.11.50:1521/eka",  "connection.user": "huynb",  "connection.password": "oracle\_4U",  "dialect.name": "OracleDatabaseDialect",  "insert.mode": "upsert",  "table.name.format": "LONGCV.TRANSACTIONS\_SINK",  "pk.mode": "record\_value",  "pk.fields": "TRANSACTION\_ID",  "auto.create": "true",  "trim.sensitive.log": "false",  "principal.service.name": "denodo",  "principal.service.password": "vertica\_4U",  "value.converter.schema.registry.url": "http://10.10.12.241:8081",  "key.converter.schemas.enable": "true",  "value.converter.schemas.enable": "true",  "key.converter.schema.registry.url": "http://10.10.12.241:8081"  }  } |

**Giải thích cấu hình:**

* Key.converter: Chuyển đổi định dạng Key của Message giữa kafka Connect và Kafka Broker.
* Value.converter: chuyển đổi định dạng Value của Message giữa Kafka Connect và Kafka Broker.
* Topic: tên topic subscribe để consume dữ liệu.
* Transforms: Tên kiểu biến đổi
* Transforms.masking.type: Kiểu biến đổi
* Transforms.masking.fields: Trường biến đổi
* Transforms.masking.replacement: Biến đổi thay thế.
* Insert.mode: chế độ chèn được thực hiện.
* Table.name.format: định dạng tên bảng ánh xạ dưới cơ sở dữ liệu đích.
* Pk.mode: chế độ khóa chính của bản ghi
* Pk.fields: trường khóa chỉnh được chỉ định.
* Auto.create: tự động tạo bảng dưới cơ sở dữ liệu dựa vào tên bảng ánh xạ nếu Table chưa tồn tại.
* Key.converter.schemas.enable: cho phép đính kèm schema vào Key Message khi ghi vào Kafka
* Value.converter.schemas.enable: cho phép đính kèm schema vào Value Message khi ghi vào Kakfa.

1. Kiểm tra trạng thái connector



Hình 3. 28 Giao diện trạng thái của connector trên Control Center

Nếu trạng thái “Running” đã tạo connector thành công.

Hoàn thành quy trình tạo luồng.

## 3.6 Confluent Platform xử lý dữ liệu:

**Quá trình xử lý dữ liệu:**

Bước 1:Khi Producer gửi Message vào Kafka Broker:

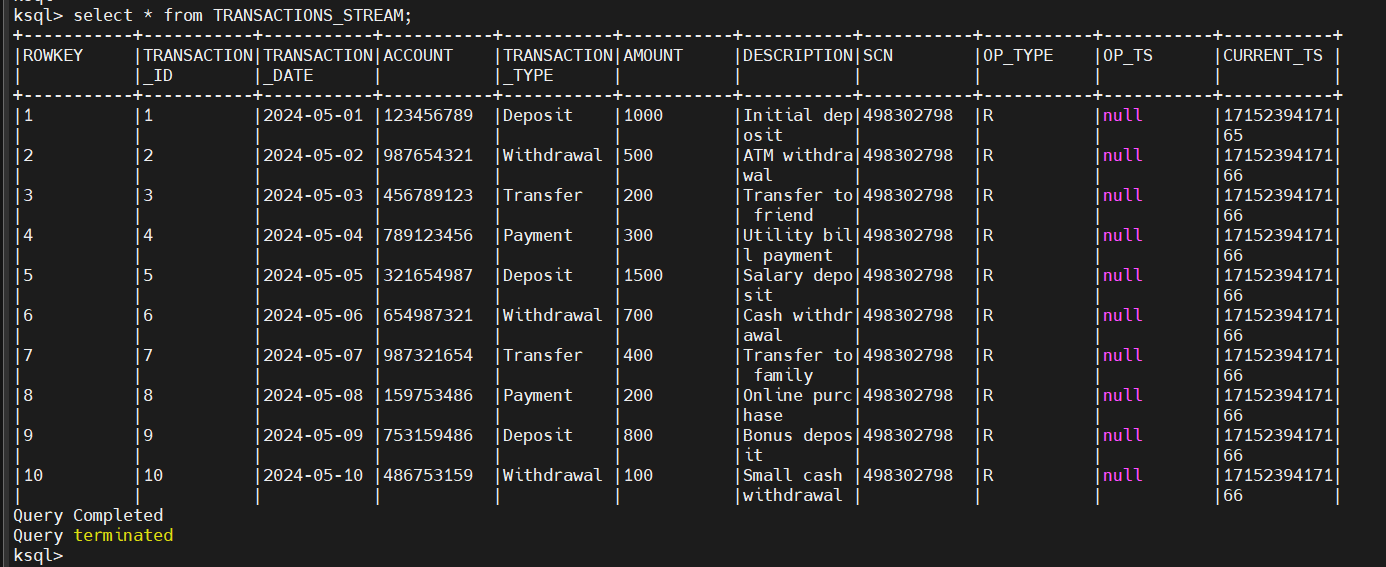
Kafka Connector sẽ Serializer – tuần tự hóa, xử lý định dạng dữ liệu từ Chuối kí tự Varchar sang Avro (Theo cấu hình) để Confluent có thể hỗ trợ. Thêm thông, timestamp,header,key và các trường giá trị value:

(Mẫu 1 message trong topics: TRANSACTIONS\_SOURCE\_DATA\_TOPIC)

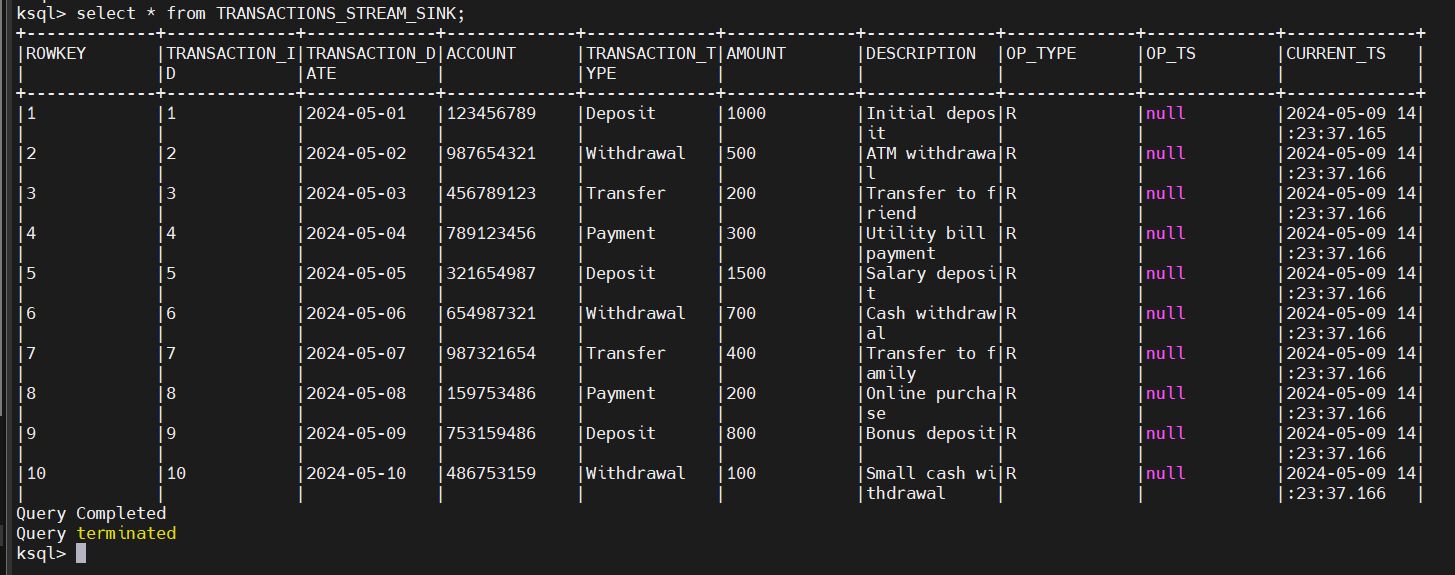
|  |
| --- |
| [  {  "topic": "TRANSACTIONS\_SOURCE\_DATA\_TOPIC",  "partition": 0,  "offset": 31,  "timestamp": 1715871986566,  "timestampType": "CREATE\_TIME",  "headers": [],  "key": "\u0000\u0000\u0000\u0000\u0005\f874239",  "value": {  "TRANSACTION\_ID": "874239",  "TRANSACTION\_DATE": { "string": "2024-05-15" },  "ACCOUNT": { "string": "X216353123" },  "TRANSACTION\_TYPE": { "string": "Payment" },  "AMOUNT": { "string": "100000" },  "DESCRIPTION": { "string": "Transfer to account C987654321" },  "table": { "string": "eka.EKA.TRANSACTIONS\_SOURCE" },  "scn": { "string": "500865185" },  "op\_type": { "string": "U" },  "op\_ts": { "string": "1715897179000" },  "current\_ts": { "string": "1715871981574" },  "row\_id": { "string": "AAAtasAAHAAAALHAAF" },  "username": { "string": "HUYNB" }  }  }  ] |

#### Bước 2: KsqlDB xử lý dữ liệu:

* Loại bỏ các trường không sử dụng trong value như : table, scn, row\_id, username.
* Biến đổi các trường OP\_TS, CURRENT\_TS từ 64int sang timestamp.
* Trả dữ liệu ra 1 Topic mới: TRANSACTIONS\_SINK\_DATA\_TOPIC



Hình 3. 29 Dữ liệu trong stream TRANSACTIONS\_STREAM



Hình 3. 30 Dữ liệu trong stream TRANSACTIONS\_SINK\_STREAM

(Mẫu 1 Message trong Topic TRANSACTIONS\_SINK\_DATA\_TOPIC sau khi được xử lý)

|  |
| --- |
| [  {  "topic": "TRANSACTIONS\_SINK\_DATA\_TOPIC",  "partition": 0,  "offset": 31,  "timestamp": 1715871986566,  "timestampType": "CREATE\_TIME",  "headers": [],  "key": "\u0000\u0000\u0000\u0000\u0005\f874239",  "value": {  "TRANSACTION\_ID": { "string": "874239" },  "TRANSACTION\_DATE": { "string": "2024-05-15" },  "ACCOUNT": { "string": "X216353123" },  "TRANSACTION\_TYPE": { "string": "Payment" },  "AMOUNT": { "string": "100000" },  "DESCRIPTION": { "string": "Transfer to account C987654321" },  "OP\_TYPE": { "string": "U" },  "OP\_TS": { "string": "2024-05-16 22:06:19.000" },  "CURRENT\_TS": { "string": "2024-05-16 22:06:21.574" }  }  }  ] |

Bước 3: Consumer lấy dữ liệu từ Topic

Kafka Connector JDBC sẽ Deserializer – giải tuần tự hóa Key và value của message từ Avro sang dạng cấu trúc dữ liệu của cơ sở dữ liệu để có thể chèn, update dữ liệu.

Kafka Connector masking (ẩn) trường dữ liệu ACCOUNT ( Theo cấu hình) dưới dạng \*\*\*.

**3.7 Tổng kết chương 3**

Trong chương 3, em đã trình bày triển khai cài đặt hệ thống Confluent Platform. Giải thích cấu hình của các thành phần trong hệ thống. Quy trình triển khai cài đặt tích hợp MinIO, Prometheus, Grafana và quy trình tạo luồng streaming dữ liệu, quy trình xử lý dữ liệu.

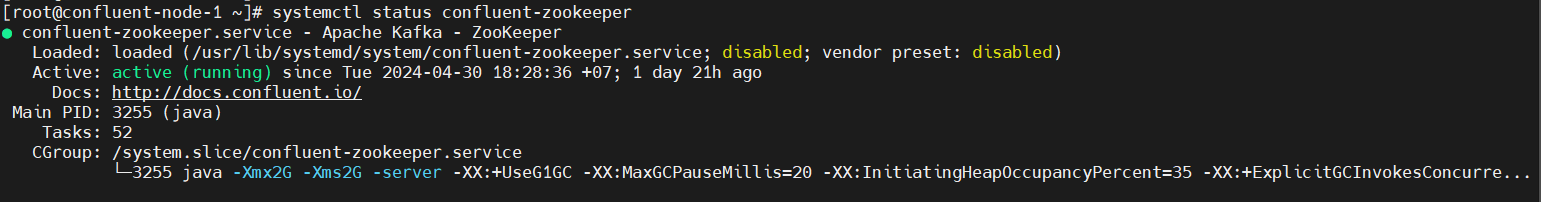
# CHƯƠNG 4: KẾT QUẢ VÀ KIỂM THỬ

## 4.1 Kết quả đạt được

### 4.1.1 Đối với hệ thống Confluent Platform

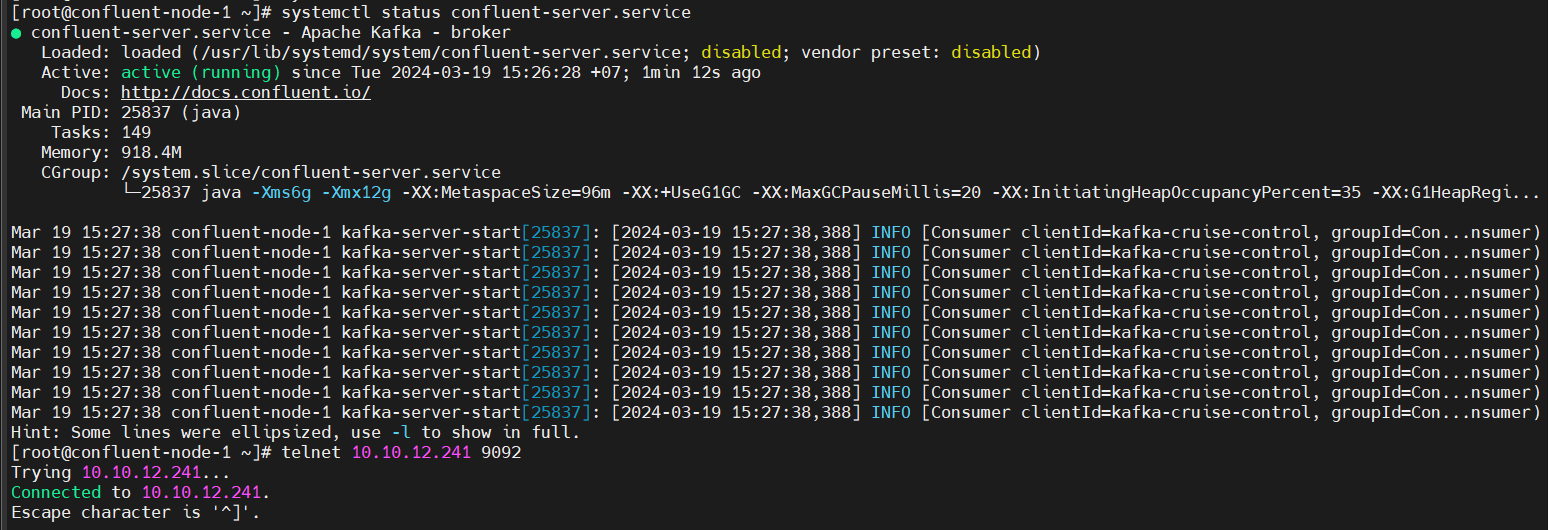
Cài đặt thành công 6 service của hệ thống:

1. Confluent Zookeeper:



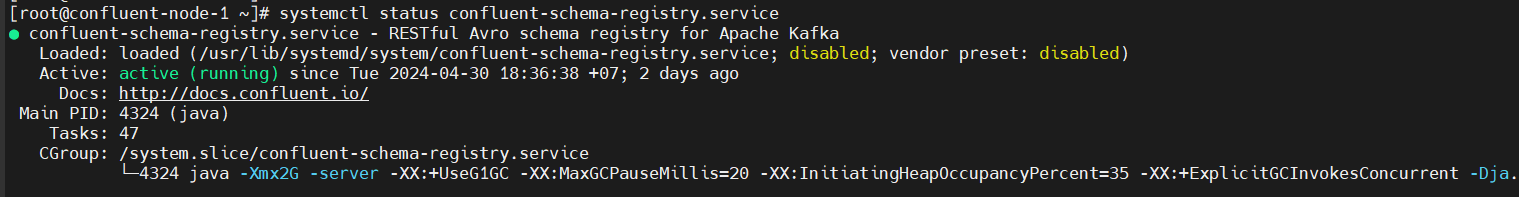
Hình 4. 1 Trạng thái service Zookeeper

2. Confluent Server:



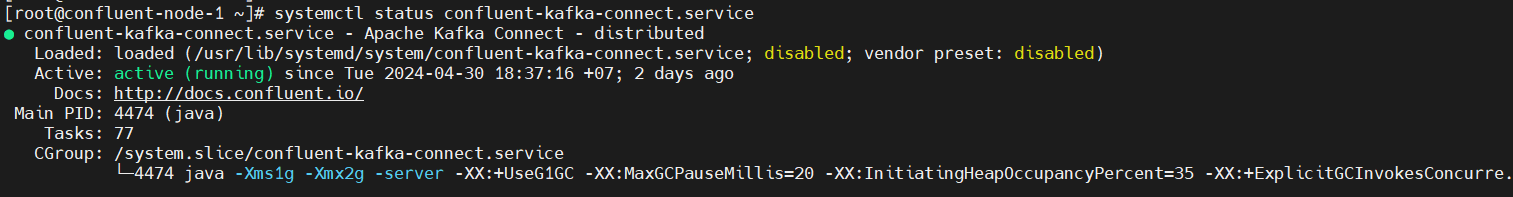
Hình 4. 2 Trạng thái service Server

3. Confluent Schema Registry:



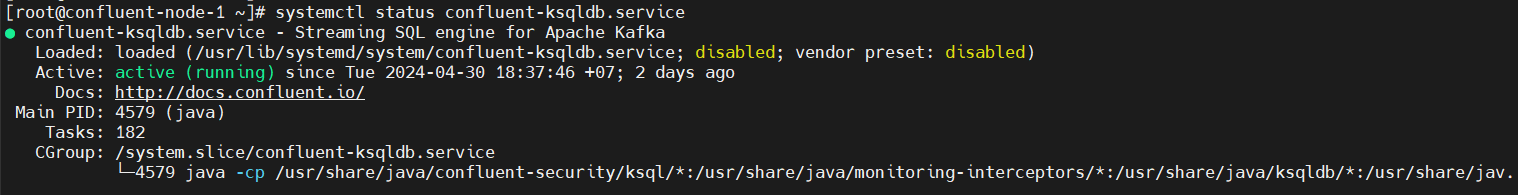
Hình 4. 3 Trạng thái service Schema Registry

4. Confluent Kafka Connect:



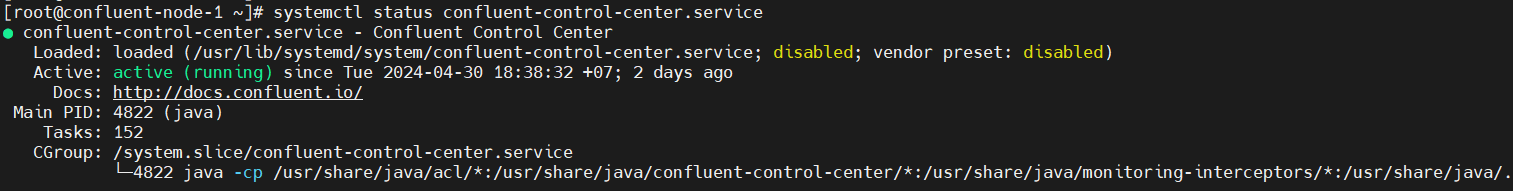
Hình 4. 4 Trạng thái service Kafka Connect

5. Confluent KsqlDB:



Hình 4. 5 Trạng thái service KsqlDB

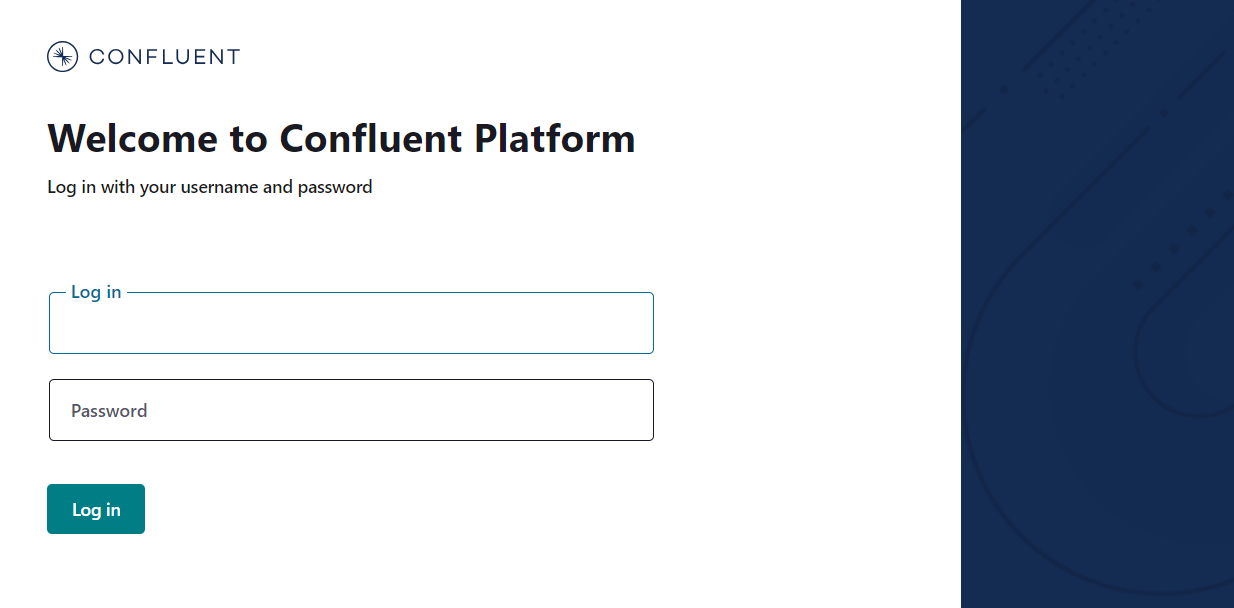
6. Confluent Control Center:



Hình 4. 6 Trạng thái service Control Center

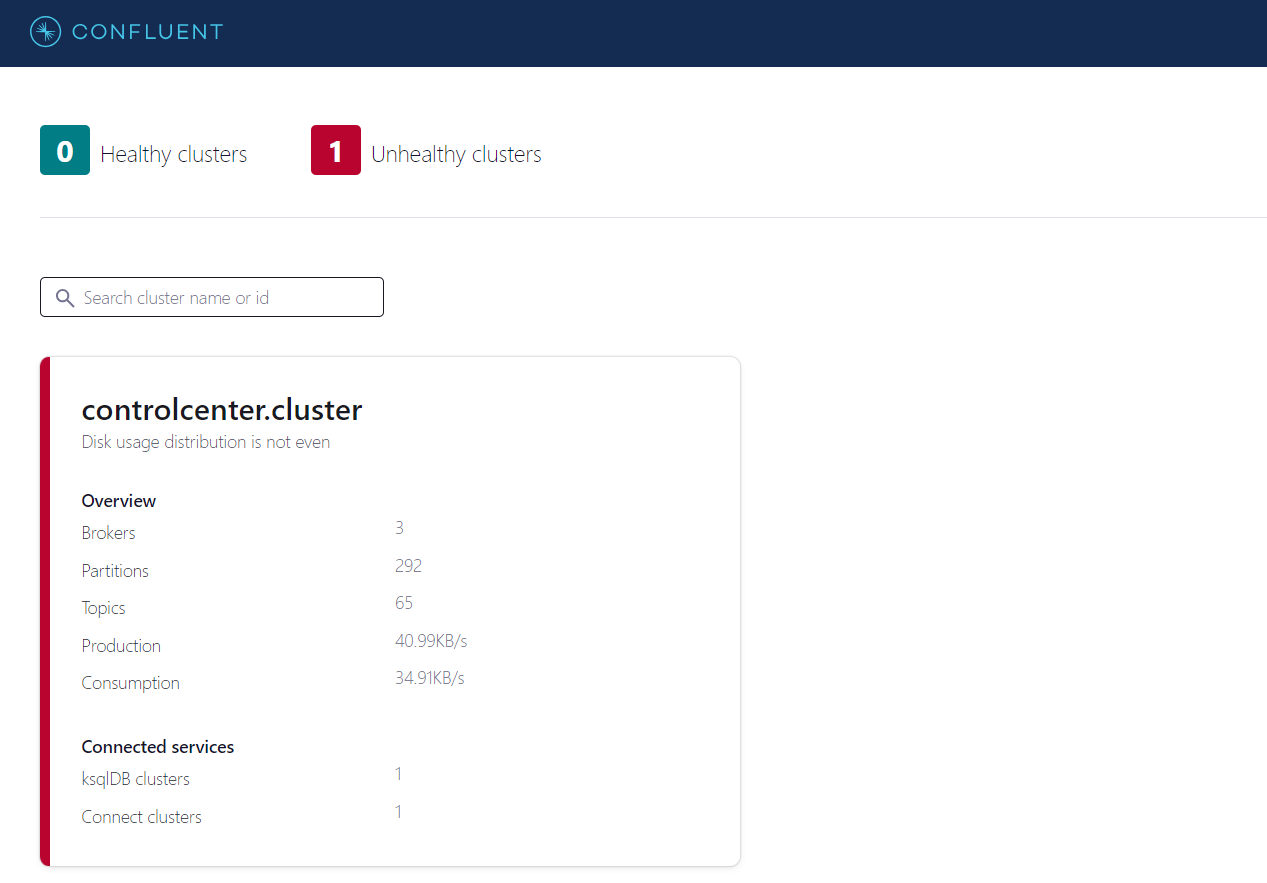
### 4.1.2 Giao diện website của Confluent Control Center:

1. Giao diện đăng nhập của Confluent Control Center:



Hình 4. 7 Giao diện đăng nhập Control Center

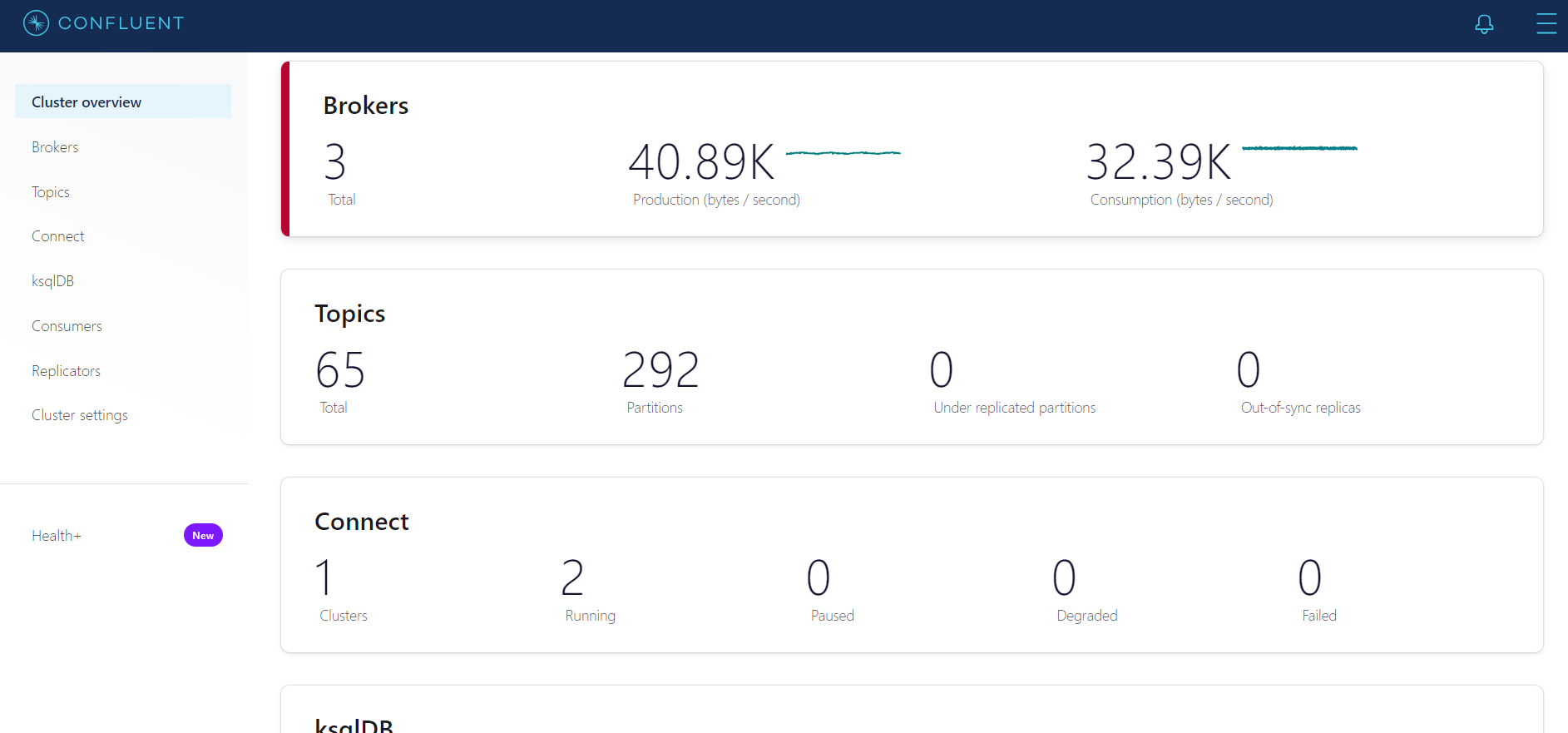
2. Giao diện tính năng Health+:



Hình 4. 8 Giao diện tính năng Health+

Tính năng Health+ lấy dữ liệu của hê thống và thông báo nhanh trạng thái của cụm.

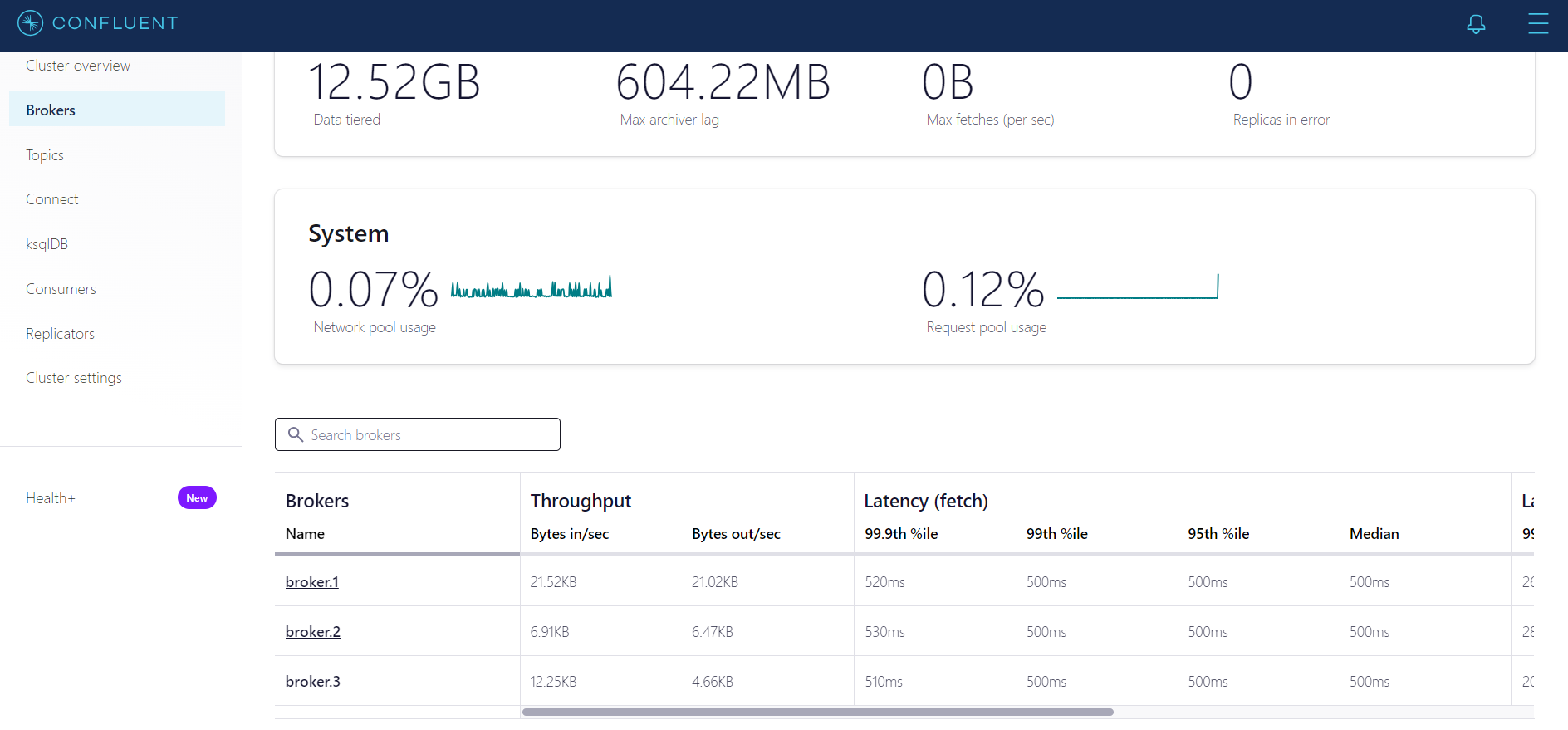
3. Giao diện overview của Cluster:



Hình 4. 9 Giao diện Overview của Cluster

Cho thấy được thông tin tổng số Broker, tổng dữ liệu produce/consume, tổng Topics, Partition, Connector đang chạy,…

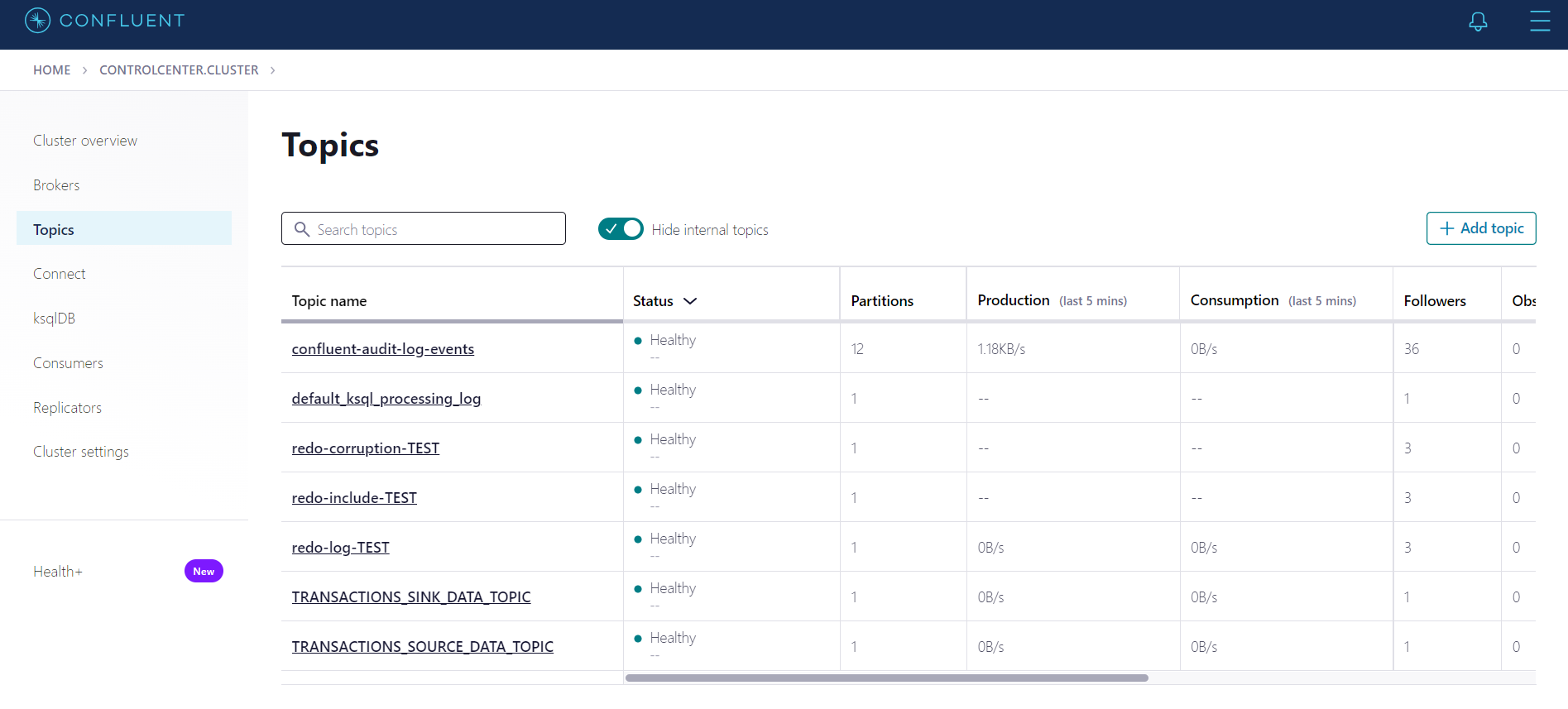
4. Giao điện overview của Broker:



Hình 4. 10 Giao diện overrview Broker

Cho phép theo dõi các thông số liên quan đến các Broker.

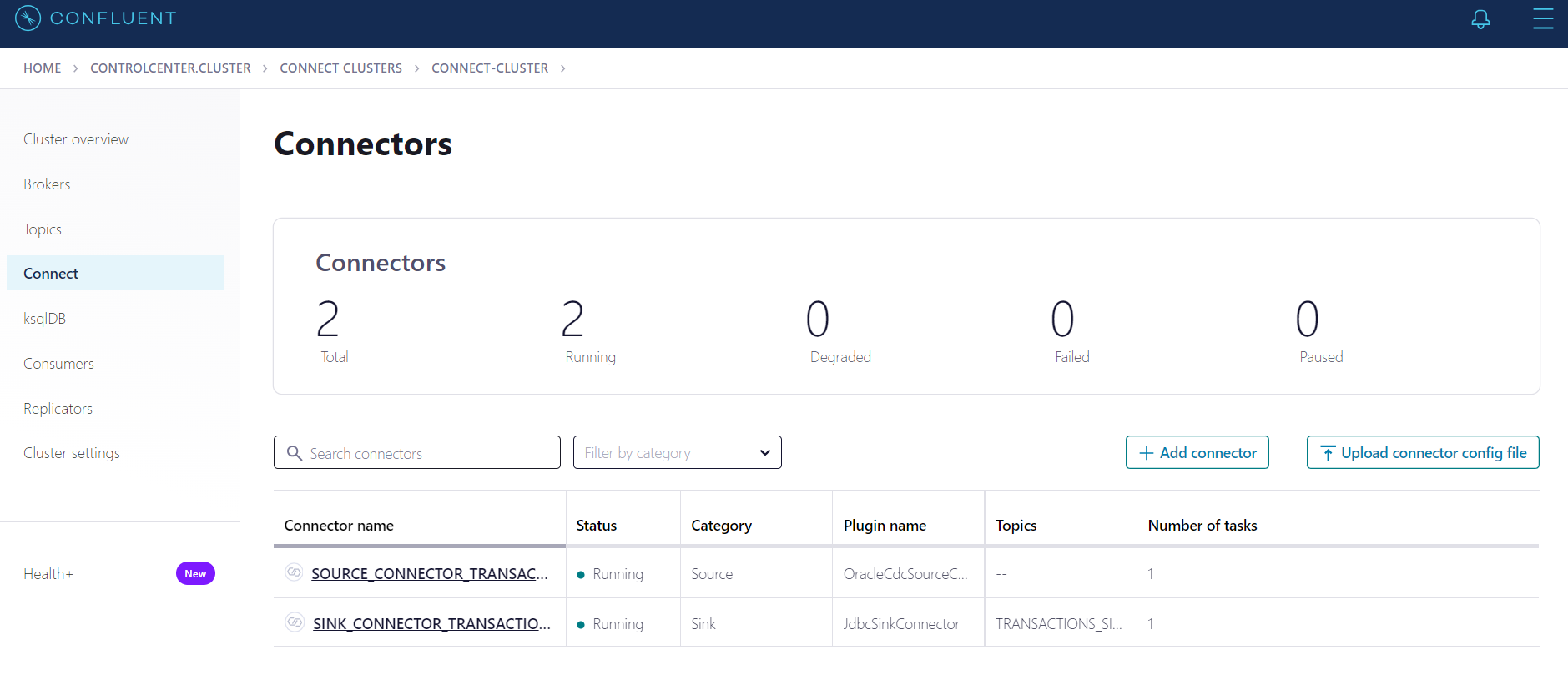
5. Giao diện Topics:



Hình 4. 11 Giao diện Topics

Chức năng thêm sửa xóa topic. Thông tin trạng thái, partitions,…

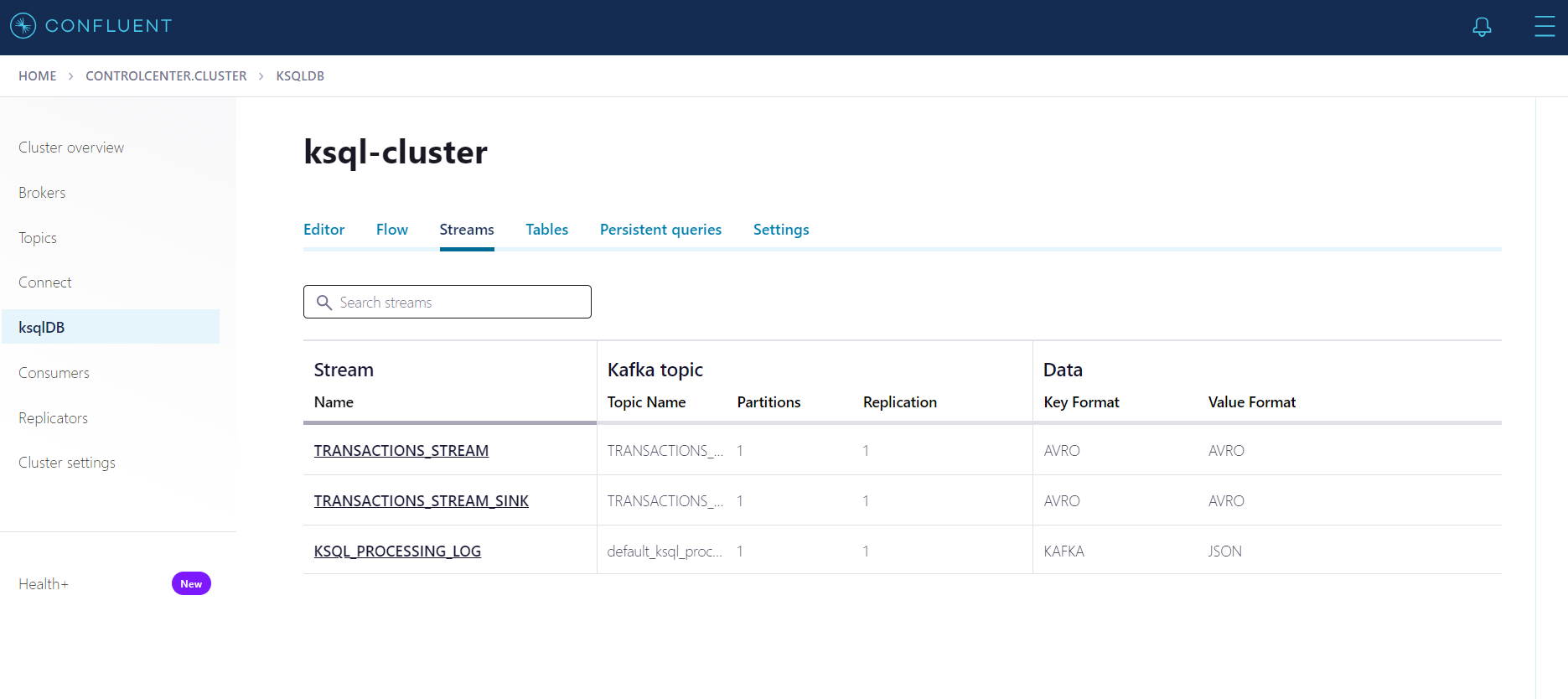
6. Giao diện Kafka Connect:



Hình 4. 12 Giao diện Kafka Connect

Chức năng thêm, sửa, xóa các connector. Giám sát trạng thái của connector.

7. Giao diện KsqlDB:

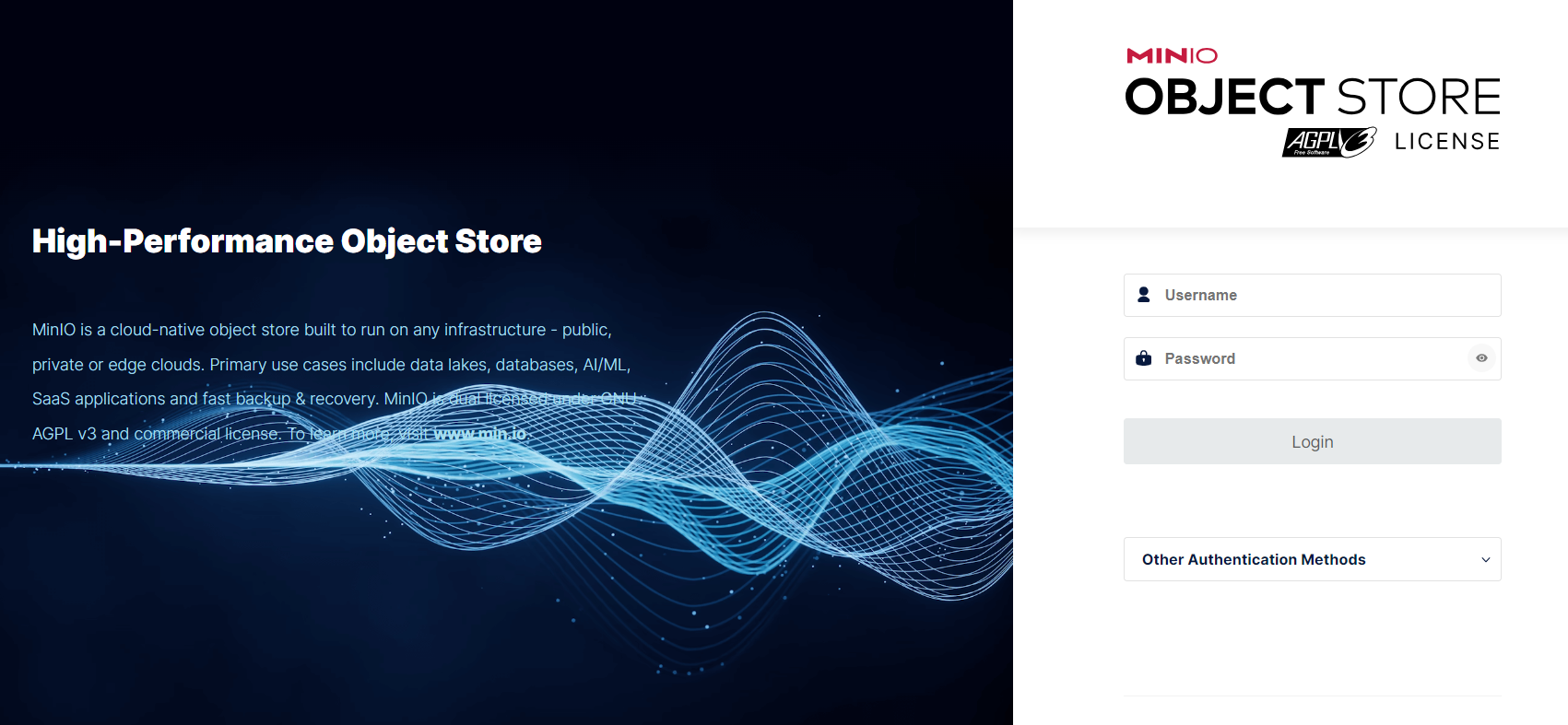


Hình 4. 13 Giao diện KsqlDB

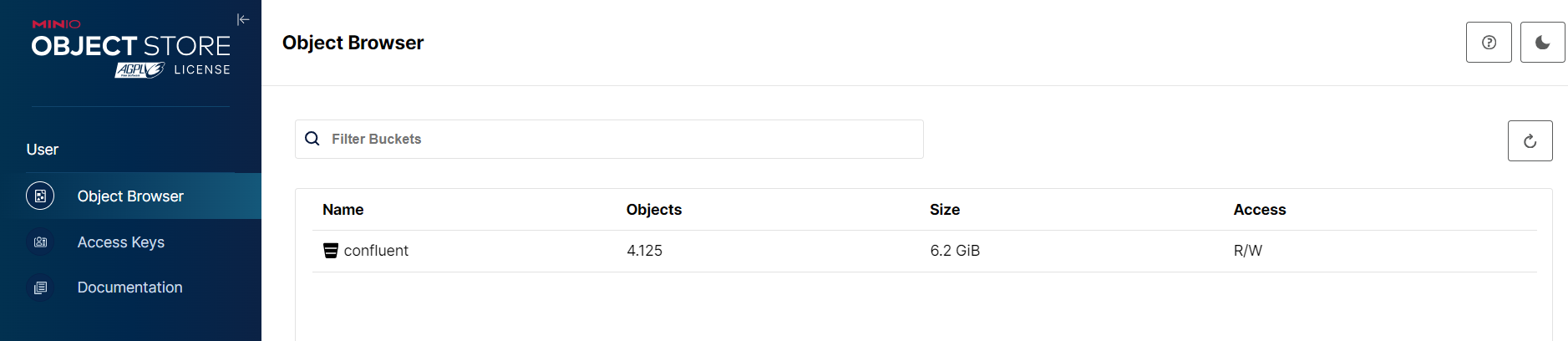
Chức năng thêm, sửa, xóa các stream. Giám sát trạng thái của luồng.

### 4.1.3 Công nghệ đã tích hợp thành công:

1. MinIO

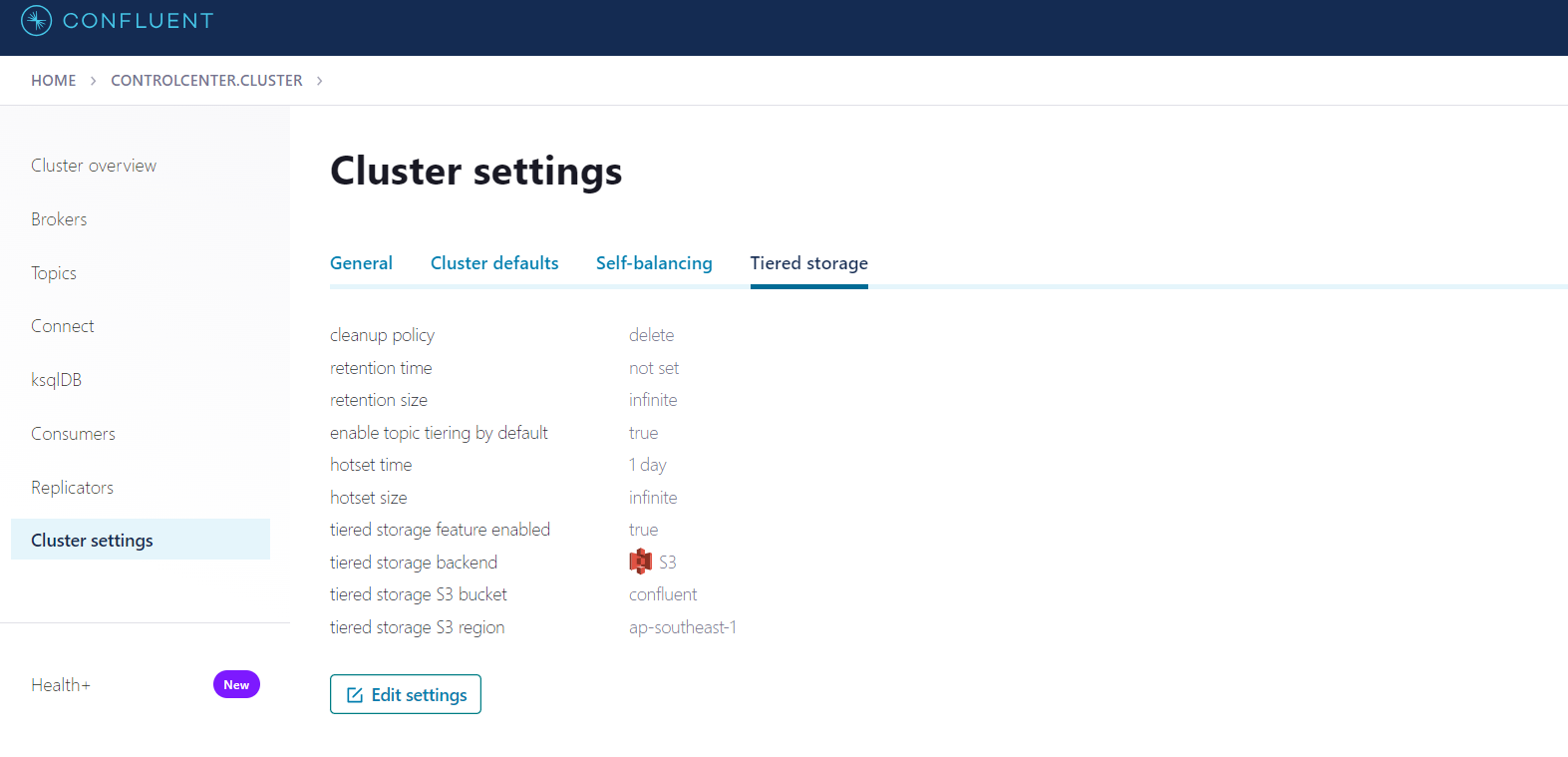


Hình 4. 14 Giao diện đăng nhập MinIO



Hình 4. 15 Giao diện chính MinIO

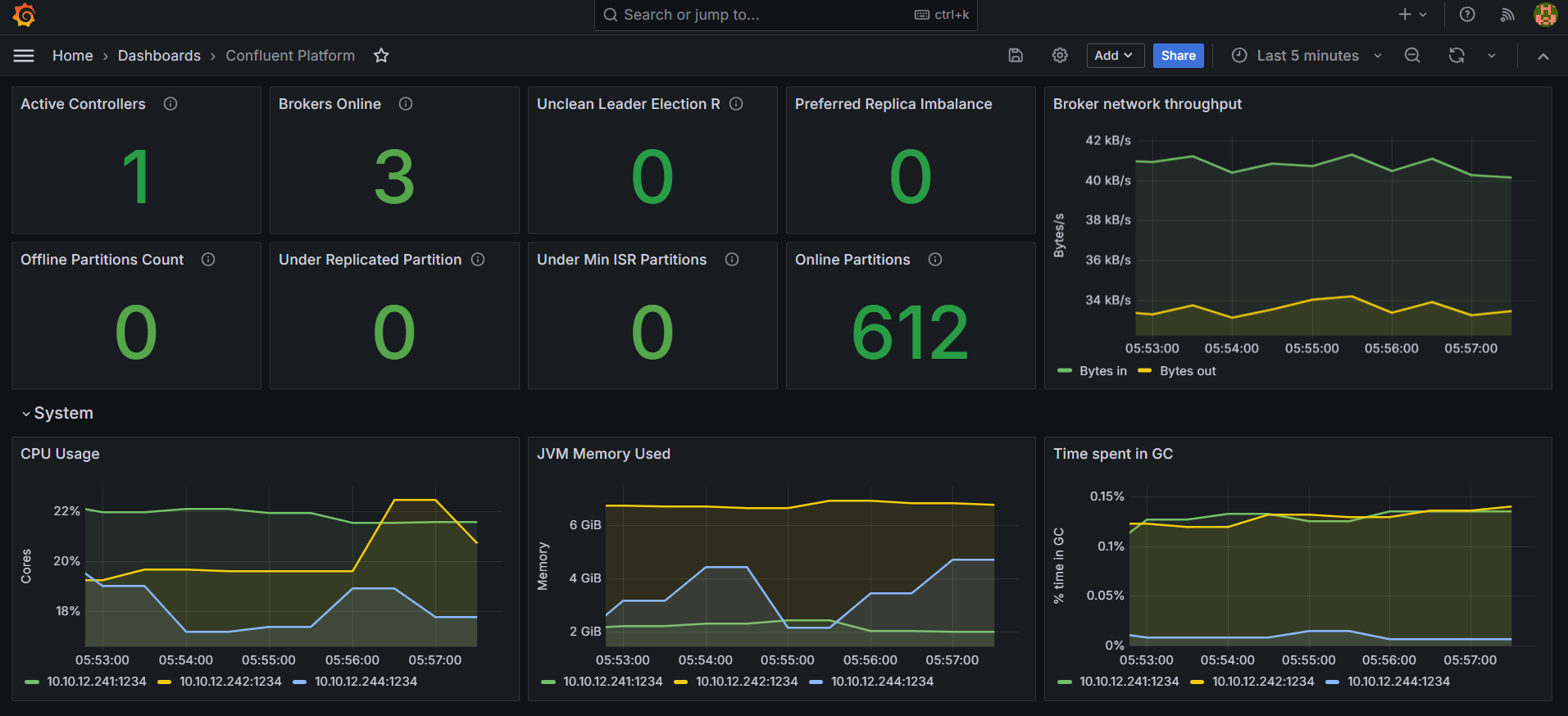
MinIO đã lưu trữ dữ liệu của Hệ thống confluent.



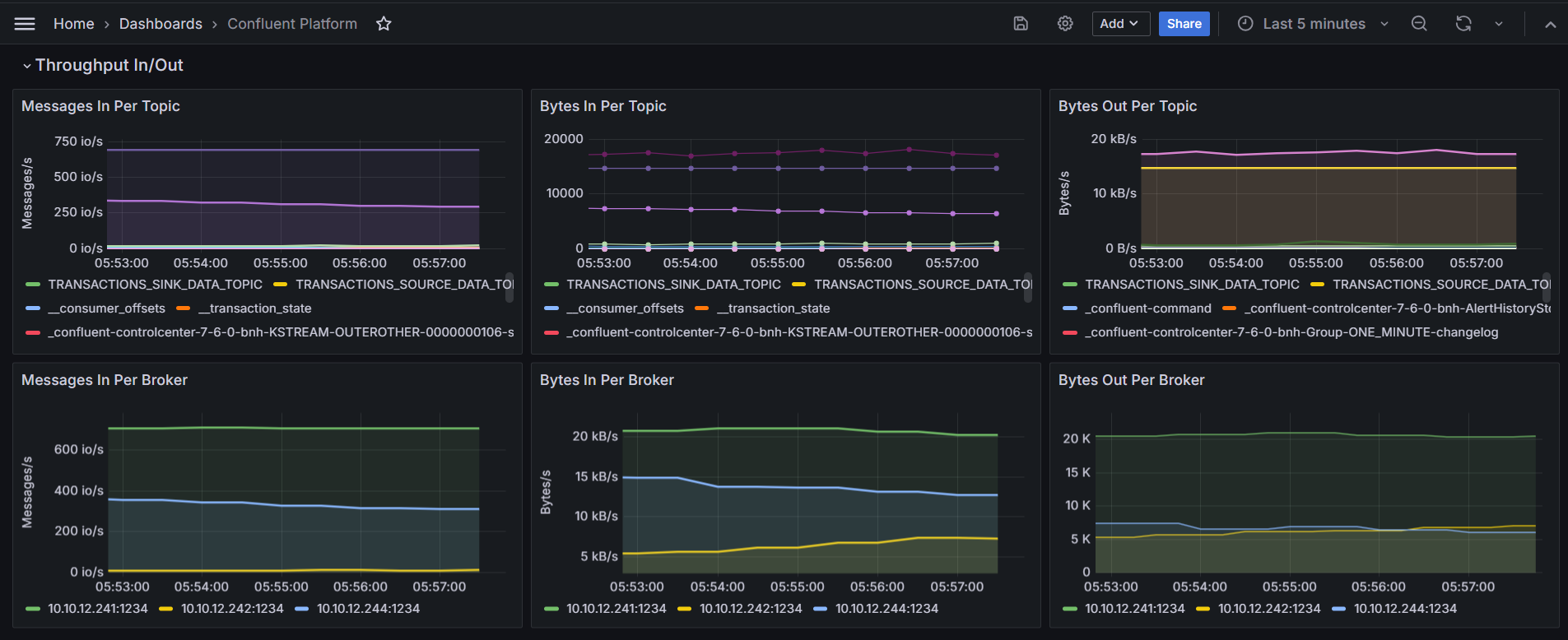
Hình 4. 16 Giao diện Control Center đã tích hợp MinIO

1. Grafana

Mô hình trực quan dễ quan sát, quản lý.



Hình 4. 17 Giao diện Grafana



Hình 4. 18 Giao diện Grafana

## 4.2 Kiểm thử hệ thống:

Bảng 4. 1 Bảng kiểm thử hệ thống

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Trường hợp | Các bước thực hiện | Kết quả thực hiện | Đánh giá |
| 1 | Kiểm tra tính HA của Zookeeper | 1. Kiểm tra đảm bảo 3 nodes Zookeeper vẫn hoạt động bình thường.  2. Stop 1 node Zookeeper bất kì  3. Kiểm tra log Zookeeper  4.Restart lại node vừa tắt. | - Cụm Zookeeper vẫn hoạt động khi stop service 1 node Zookeeper.  -Dữ liệu được đồng bộ vào node Zookeeper sau khi restart. | Đạt |
| 2 | Kiểm tra tính HA của Broker | 1. Kiểm tra đảm bảo 3 nodes Broker vẫn hoạt động bình thường.  2. Stop 1 node Broker bất kì  3. Produce 5 Message vào topic test.  4. Kiểm tra file log broker.  5. Kiểm tra tổng message trên topic | - Cụm Broker vẫn hoạt động bình thường khi stop 1 node Broker.  - Message vẫn được produce vào topic test đầy đủ. | Đạt |
| 3 | Kiểm tra khả năng mở rộng | 1. Thêm 1 node Broker vào cụm  2. Kiểm tra tình trạng hệ thống  3. Kiểm tra các thông số trên grafana: Broker Online, throughtput in/out,… | - Sau khi thêm 1 node vào cụm, hệ thống vẫn hoạt động bình thường.  - Các thông số xử lý Message In Per Topic, Bytes In/Out per Topic đều tăng.  - Dữ liệu toàn vẹn. | Đạt |

## 4.3 Kiểm thử luồng streaming và xử lý dữ liệu

### 4.3.1 Tiêu chí

Kiểm thử luồng streaming từ nguồn database EKA (Oracle) đến đích là database LONGCV (Oracle). Tiêu chí như sau:

1. Tiêu chí streaming:

* Khi Insert, Update, Delete dữ liệu ở nguồn EKA (Oracle) thì đồng bộ real-time sang đích LONGCV (Oracle). (Độ trễ không đáng kể).
* Kiểu dữ liệu đầu vào: chuỗi ký tự (Varchar).
* Kiểu dữ liệu đầu ra: chuỗi ký tự (Varchar).
* Đảm bảo tính toàn vẹn và chính xác của dữ liệu.

2. Tiêu chí dữ liệu ở đích:

* Masking (ẩn) trường dữ liệu ACCOUNT dưới dạng \*\*.
* Thêm trường OP\_TYPE: trạng thái của bản ghi.
* Thêm trường OP\_TS: đánh dấu thời gian commit dưới nguồn EKA.
* Thêm trường CURRENT\_TS: đánh dấu thời gian commit vào Kafka.

### 4.3.2 Câu lệnh truy vấn ở cơ sở dữ liệu nguồn EKA (Oracle)

1. Insert:

|  |
| --- |
| INSERT INTO TRANSACTIONS\_SOURCE (TRANSACTION\_ID, TRANSACTION\_DATE, ACCOUNT, TRANSACTION\_TYPE, AMOUNT, DESCRIPTION)  VALUES ('675781', '2024-05-15', 'HG12315412', 'Deposit', '80000', 'Initial deposit');  INSERT INTO TRANSACTIONS\_SOURCE (TRANSACTION\_ID, TRANSACTION\_DATE, ACCOUNT, TRANSACTION\_TYPE, AMOUNT, DESCRIPTION)  VALUES ('412354', '2024-05-15', 'QWE1251333', 'Withdrawal', '6500', 'Cash withdrawal');  INSERT INTO TRANSACTIONS\_SOURCE (TRANSACTION\_ID, TRANSACTION\_DATE, ACCOUNT, TRANSACTION\_TYPE, AMOUNT, DESCRIPTION)  VALUES ('297655', '2024-05-15', 'IKKO998882', 'Transfer', '5000', 'Transfer to account XLS654321'); |

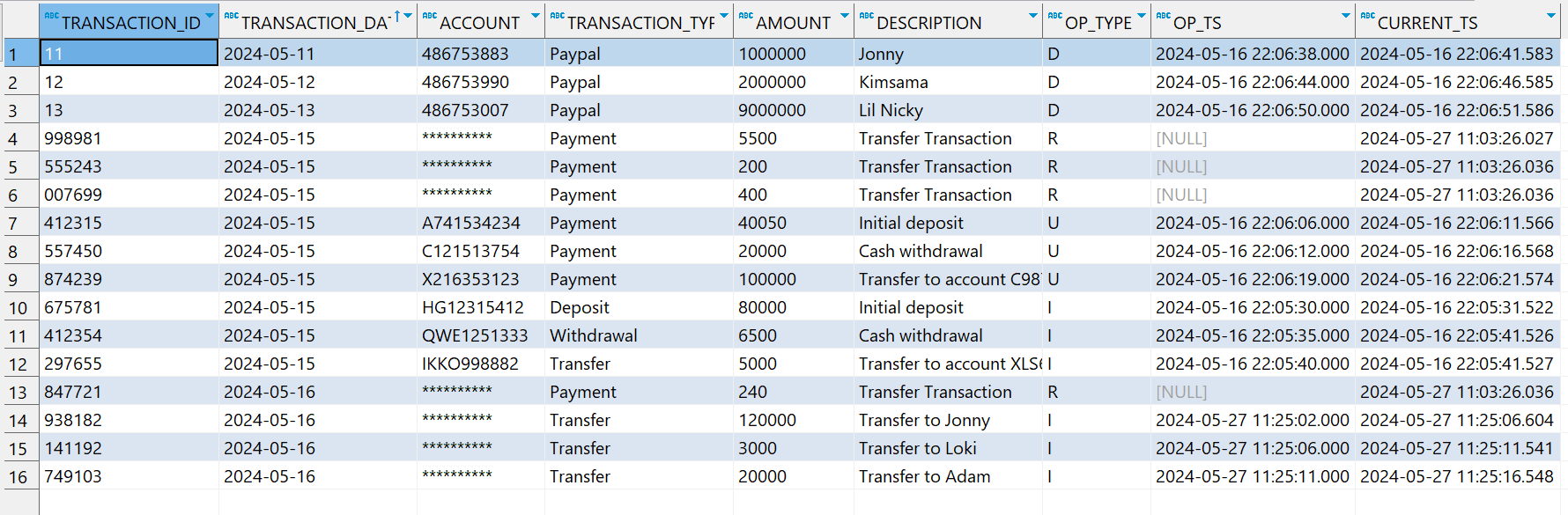
1. Update

|  |
| --- |
| UPDATE TRANSACTIONS\_SOURCE  SET TRANSACTION\_TYPE = 'Payment'  WHERE TRANSACTION\_ID = '412315';  UPDATE TRANSACTIONS\_SOURCE  SET TRANSACTION\_TYPE = 'Payment'  WHERE TRANSACTION\_ID = '557450';  UPDATE TRANSACTIONS\_SOURCE  SET TRANSACTION\_TYPE = 'Payment'  WHERE TRANSACTION\_ID = '874239'; |

1. Delete

|  |
| --- |
| DELETE FROM TRANSACTIONS\_SOURCE WHERE TRANSACTION\_ID='11';  DELETE FROM TRANSACTIONS\_SOURCE WHERE TRANSACTION\_ID='12';  DELETE FROM TRANSACTIONS\_SOURCE WHERE TRANSACTION\_ID='13'; |

**4.3.3** **Kết quả dữ liệu ở cơ sở dữ liệu đích LONGCV (Oracle)**



Hình 4. 19 Kết quả dữ liệu ở LONGCV

Như vậy hệ thống Confluent Platform đã streaming, xử lý dữ liệu thành công. Đạt tất cả các tiêu chí.

#### 4.3.3.1 Bảng tính toán độ trễ thời gian streaming

Bảng 4. 2 Bảng tính toán độ trễ thời gian streaming

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction\_ID | Query | OP\_TS | Current\_TS | Time difference (OP\_TS-Current\_TS)(giây) |
| 675781 | Insert | 2024-05-16 22:05:30.000 | 2024-05-16 22:05:31.522 | 1.522 |
| 412354 | Insert | 2024-05-16 22:05:35.000 | 2024-05-16 22:05:41.526 | 6.526 |
| 297655 | Insert | 2024-05-16 22:05:40.000 | 2024-05-16 22:05:41.527 | 1.527 |
| 412315 | Update | 2024-05-16 22:06:06.000 | 2024-05-16 22:06:11.566 | 5.566 |
| 557450 | Update | 2024-05-16 22:06:12.000 | 2024-05-16 22:06:16.568 | 4.568 |
| 874239 | Update | 2024-05-16 22:06:19.000 | 2024-05-16 22:06:21.574 | 2.574 |
| 11 | Delete | 2024-05-16 22:06:38.000 | 2024-05-16 22:06:41.583 | 3.583 |
| 12 | Delete | 2024-05-16 22:06:44.000 | 2024-05-16 22:06:46.585 | 2.585 |
| 13 | Delete | 2024-05-16 22:06:50.000 | 2024-05-16 22:06:51.586 | 1.586 |

Chú thích: - OP\_TS: Thời gian thực hiện query dưới database

* Current\_TS: Thời gian dữ liệu được đẩy vào Kafka.

Từ bảng số liệu thu thập được ta có:

1. Thời gian chênh lệch trung bình khi INSERT dữ liệu là: (1.522 + 6.526 + 1.527)/3 = 3.191 (giây)
2. Thời gian chênh lệch trung bình khi UPDATE dữ liệu là: (5.566+ 4.568 + 2.574)/3 = 4.236 (giây)
3. Thời gian chênh lệch trung bình khi DELETE dữ liệu là: (3.583+ 2.585+ 1.586)/3 = 2.584 (giây)

Ta có thể thấy thời gian chênh lệch khi thực hiện các câu truy vấn dưới database nguồn và cập nhật dưới database đích là rất nhỏ.

#### 4.3.3.2 Nguyên nhân chính giải thích tại sao hệ thống streaming Confluent Kafka chỉ đạt được gần thời gian thực (near real-time)

Giải thích vì:

* Độ trễ mạng: Mạng có thể gây ra độ trễ trong việc truyền tải dữ liệu từ hệ thống nguồn tới đích.
* Xử lý và tính toán: Các bước xử lý như trích xuất, biến đổi, phân tích dữ liệu cần thời gian để hoàn thành.
* Đồng bộ hóa và độ chính xác.
* Cấu hình hệ thống và tinh chỉnh chưa tối ưu.
* Tài nguyên cung cấp chưa đáp ứng tốt.

#### 4.3.3.3 Tính ứng dụng của streaming, xử lý dữ liệu real-time

Thực tế, trong quá trình em triển khai Confluent Platform cho các ngân hàng và tổ chức tài chính. Họ thường sử Confluent Platform để xử lý giao dịch thanh toán thời gian thực, sự kiện liên quan đến tài khoản, chuyển tiền, giao dịch chứng khoán, giám sát gian lận và theo dõi các hoạt động tài chính.

Bài toán áp dụng vào giao dịch biến động số dư:

Khi một giao dịch được thực hiện, việc streaming, xử lý dữ liệu real-time giữa cơ sở dữ liệu nguồn và cơ sở dữ liệu đích giúp cập nhật tài khoản người dùng ngay lập tức. Bên ứng dụng sẽ lấy dữ liệu từ cơ sở dữ liệu đích gửi tin nhắn SMS thông báo về thuê bao của khách hàng. Giúp khách hàng biết giao dịch đã thành công và số dư đã được cập nhật.

Thời gian khách hàng nhận được tin nhắn SMS cũng sẽ bị trễ từ 3-5 giây.

(Vì một số lí do bảo mật nên em không thể có hình ảnh chứng minh ở đây). Điều đó chứng tỏ luồng streaming được xây dựng trong đồ án là đạt tiêu chí streaming.

## 4.4 Tổng kết chương 4

Trong chương 4, em đã trình bày kết quả đạt được sau khi hoàn thành dự án: Triển khai toàn bộ dịch vụ của hệ thống Confluent, tích hợp được các công nghệ có trong mô hình hệ thống. Kiểm thử hệ thống, kiểm thử luồng streaming.

# KẾT LUẬN

Đề tài “Xây dựng hệ thống xử lý dữ liệu real-time sử dụng Confluent Platform” . Qua quá trình thực hiện đề tài, em đã tìm hiểu, tích lũy và vận dụng được những kiến thức công nghệ được học từ trường, từ nơi làm việc và đã hoàn thành được những phần sau đây.

Kết quả đạt được:

* Phân tích, thiết kế, triển khai cài đặt hệ thống xử lý dữ liệu real-time Confluent Platform.
* Tạo luồng streaming, xử lý dữ liệu real-time giữa hai cơ sở dữ liệu.
* Tích hợp công nghệ MinIO, Grafana.

Hướng nghiên cứu trong tương lai:

* Nghiên cứu triển khai tính năng Cluster Linking liên kết giữa 2 cụm Confluent DC (Data Center) và DR (Disaster Recovery).
* Nghiên cứu mô hình Active-Active Multi Site nâng cao tính sẵn sàng, liên tục của hệ thống.
* Nghiên cứu tinh chỉnh tối ưu lại cấu hình cụm Confluent Platform để tối ưu độ trễ khi streaming.

Em muốn bày tỏ lòng biết ơn chân thành nhất đến quý thầy cô trong khoa Công nghệ thông tin vì đã dành tâm huyết giảng dạy và truyền đạt kiến thức quý giá và cần thiết trong những năm học tại trường, đóng góp quan trọng cho sự hoàn thiện của đề tài đồ án tốt nghiệp này. Đặc biệt, tôi muốn gửi lời cảm ơn đặc biệt tới cô TS. Phạm Văn Hà đã hướng dẫn và hỗ trợ tận tâm, giúp em hoàn thành báo cáo đồ án tốt nghiệp cùng sản phẩm đi kèm một cách hoàn thiện nhất.

Do thiếu kinh nghiệm thực tế cũng như thời gian hạn chế, ứng dụng của em vẫn còn nhiều hạn chế. Vì vậy, em mong rằng các quý thầy cô có thể đóng góp ý kiến để giúp tôi hoàn thiện ứng dụng tốt hơn.

# TÀI LIỆU THAM KHẢO

[1]. <https://docs.confluent.io/platform/current/platform.html>

[2]. <https://min.io/docs/minio/linux/index.html>

[3]. <https://prometheus.io/docs/introduction/overview/>

[4]. <https://grafana.com/docs/grafana/latest/>