# BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: Xây dựng hệ thống xử lý dữ liệu realtime sử dụng Confluent Platform.

- Mở đầu: Trình bày lý do chọn đề tài, mục đích, đối tượng và phạm vị nghiên cứu, ý nghĩa khoa học và thực tiễn của đề tài.

**1. Apache Kafka (OpenSource)**

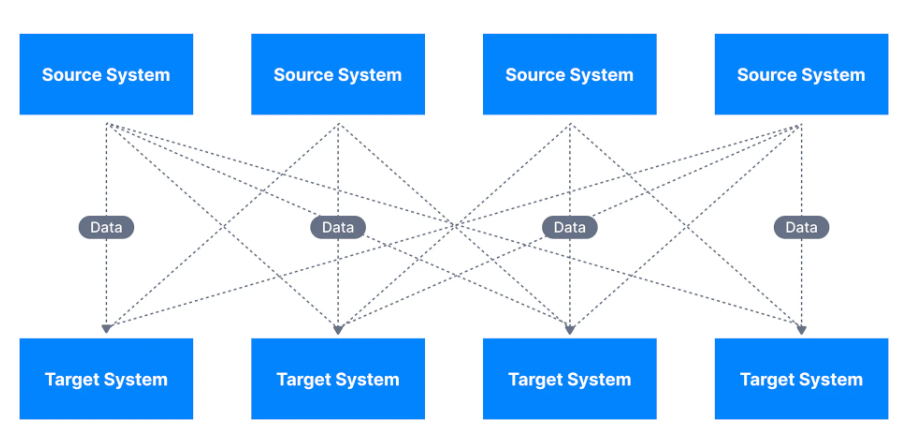
1.1 Giới thiệu tính năng của Apache Kafka

1.1.1 Giới thiệu Apache Kafka.

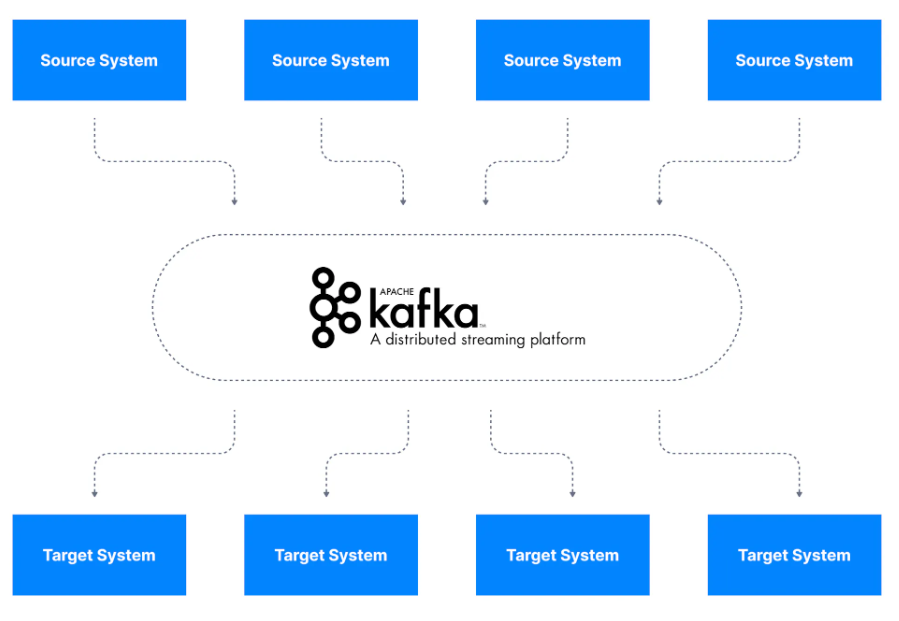
Khi nói đến truyền phát dữ liệu, Apache Kafka là một tiêu chuẩn thực tế, nó là một hệ thống phân tán mã nguồn mở bao gồm các máy chủ và máy khách. Apache Kafka được sử dụng chủ yếu để xây dựng các đường truyền dẫn dữ liệu theo thời gian thực.

Kafka được tạo ra bởi Linkedln để phục vụ các yêu cầu xử lý luồng nội bộ mà các hệ thống truyền phát tin nhắn truyền thống không thể đáp ứng. Phiên bản đầu tiên của nó được phát hành vào tháng 1 năm 2011. Kafka nhanh chóng trở nên phổ biến và kể từ đó trở thành một trong những dự án nổi tiếng nhất của Apache Foundation.

Một tổ chức điển hình có nhiều nguồn dữ liệu với các định dạng khách nhau. Tích hợp dữ liệu bao gồm việc kết hợp dữ liệu từ nhiều nguồn này thành một thể thống nhất về hoạt động kinh doanh của họ. Một vấn đề phức tạp đã phát sinh khi các luồng dữ liệu chồng chéo và khó quản lý, khó khăn trong việc đồng bộ các giao thức truyển tải dữ liệu (TCP – Transmission Control Protocol, HTTP – Hypertext Transfer Protocol, REST – Representational State Transfer, JDBC – Java Database Connectivity…) hay các định dạng dữ liệu (Binary, JSON, Avro…)



Apache Kafka ra đời giải quyết toàn bộ vấn đề nêu trên mà doanh nghiệp gặp phải, tích hợp dữ liệu, các Data Source- nguồn dữ liệu sẽ đẩy dữ liệu lên Apache Kafka và các Target System – hệ thống đích sẽ lấy dữ liệu từ Apache Kafka. Điều này tách riêng luồng dữ liệu nguồn và hệ thống đích, cho phép thực hiện giải pháp tích hợp dữ liệu đơn giản hóa.

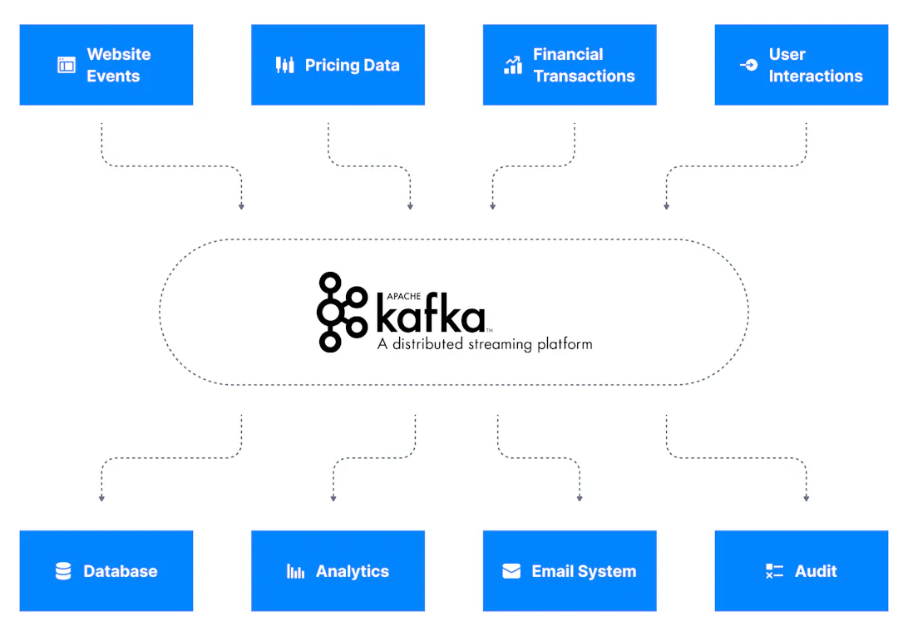


1.1.2 Data stream – Luồng dữ liệu trong Apache Kafka.

Data stream – Luồng dữ liệu là một chuỗi dữ liệu có khả năng không bị giới hạn, việc truyền phát dữ liệu có thể truy cập được ngay khi nó được tạo ra.

Apache Kafka được sử dụng để lưu trữ các luồng dữ liệu này, nó được lưu trữ trong Topic – chủ đề , sau đó cho phép cá hệ thống thực hiện xử lý luồng – một hành động thực hiện các phép tính, truy vẫn trên nguồn dữ liệu có khả năng vô tận và không ngừng mở rộng. Khi luồng được xử lý và lưu trữ trong Apache Kafka, nó có thể được chuyển sang hệ thống khác.

Mỗi ứng dụng trong một tổ chức, nơi chứa dữ liệu được tạo đều là một ứng dụng tạo luồng dữ liệu tiềm năng. Dữ liệu được tạo như một phần của luồng dữ liệu , thông lượng dữ liệu tới các luồng dữ liệu là khác nhau: một số luồng sẽ nhận được hang chục nghìn bản ghi mỗi giây và một số luồng sẽ nhận được một hoặc hai bản ghi mỗi giờ.



1.1.3 Một số tính ứng dụng của Apache Kafka trong thực tế:

* Hệ thống nhắn tin
* Theo dõi hoạt động
* Thu thập số liệu từ nhiều ví trí khác nhau, ví dụ: Thiết bị IOT
* Phân tích nhật ký ứng dụng
* Loại bỏ sự phụ thuộc của hệ thống
* Tích hợp các công nghệ Dữ liệu lớn như Spark, Flink, Storm, Hadoop

1.2 Giới thiệu về các thành phần cốt lõi trong Apache Kafka.



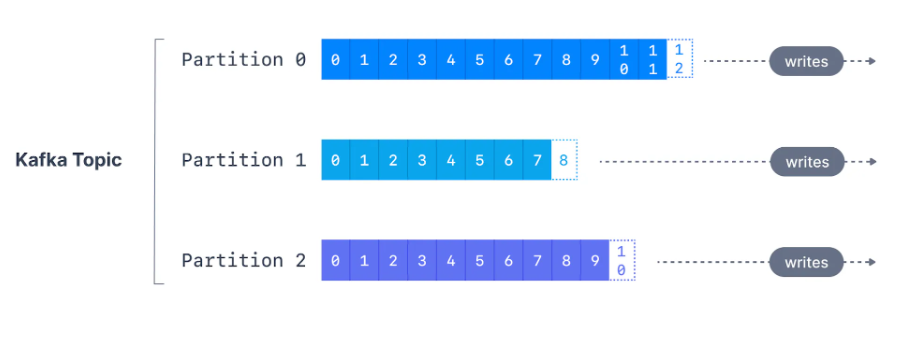
Kiến trúc cơ bản của Apache Kafka.

1.2.1 Kafka Topic – Chủ đề

Kafka topic là vùng chứa dữ liệu trong Apache Kafka. Kafka Topic gần giống với các bảng SQL tuy nhiên Kafka Topic không thể truy vấn được. Thay vào đó Apache Kafka tạo ra Producer – Các nhà sản xuất, Consumer – Các nhà tiêu thụ, Producer gửi dữ liệu vào Kafka Topic và Consumer đọc dữ liệu từ Kafka topic theo thứ tự.

Dữ liệu trong các chủ đề được lưu trữ ở dạng Key – Value ở định dạng nhị phân. Dữ liệu trong các Kafka Topic sẽ bị xóa sau một tuần (mặc định) còn được gọi là Message retention period, giá trị này có thể được cấu hình lại tùy theo nhu cầu của người sử dụng. Cơ chế xóa dữ liệu cũ này đảm bảo cụm Kafka không hết dung lượng ổ đĩa bằng cách tái chế các chủ đề theo thời gian.

1.2.1.1 Kafka Partition – Phân Vùng.



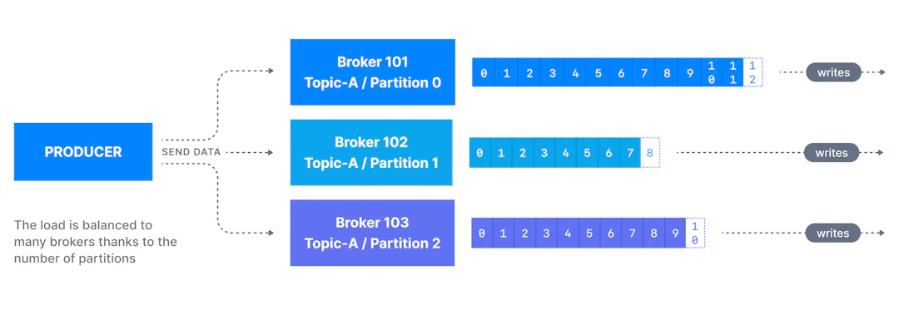
Kafka Topic sẽ chia thành một số Kafka Partition, 1 Kafka Topic có thể có nhiều Kafka Partition, số Partition của một chủ đề được cấu hình tại thời điểm tạo Kafka Topic, các Partition được đánh số từ 0 đến n-1 (n là số phân vùng cấu hình). Các message sẽ được thêm vào cuối mỗi phân vùng.

1.2.1.2 Kafka Offsets

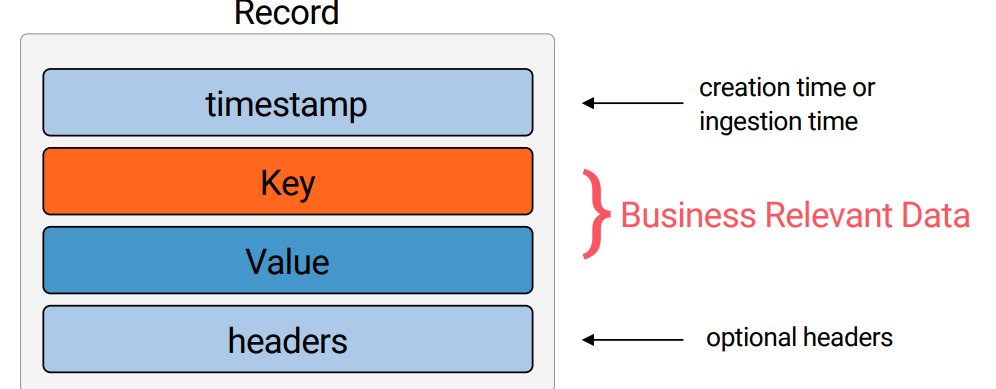
Mỗi giá trị số nguyên trên Parition được gọi là Offsets. Kafka Offsets là vị trí Message trong một Partition và nó là duy nhất. Có nghĩa là Offset 3 trong Partition 0 không biểu thị cùng dữ liệu với Offset 3 trong Partition 1. Mặc dù Message trong Kafka Topic sẽ được xóa theo thời gian nhưng Kafka Offsets sẽ không được sử dụng lại, chúng sẽ liên tục tăng lên theo 1 trình tự không bao giờ kết thúc.

1.2.2 Kafka Producer – Nhà sản xuất.

Kafka Topic sẽ được tạo trong Kafka, các ứng dụng sẽ gửi dữ liệu vào một Kafka Topic đó được gọi là Kafka Producer.



1.2.2.1 Message

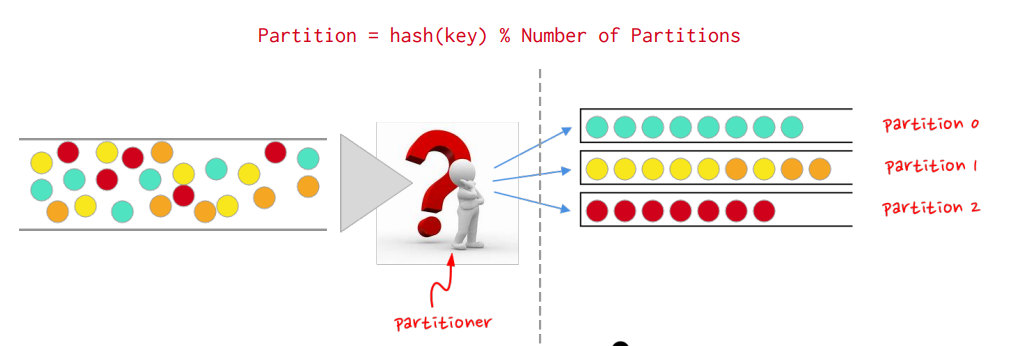


Message hay Record hay Event chỉ là cách gọi để chỉ dữ liệu được produce vào Kafka.

Cấu trúc của một Message bao gồm:

* Timestamp: Thời gian Message được thêm vào Partition.
* Key: Xác định Message sẽ được phân vào Partition nào.
* Value: Là phần tử dữ liệu mà doanh nghiệp muốn xử lý
* Header: Tiêu đề Message, có thể có.

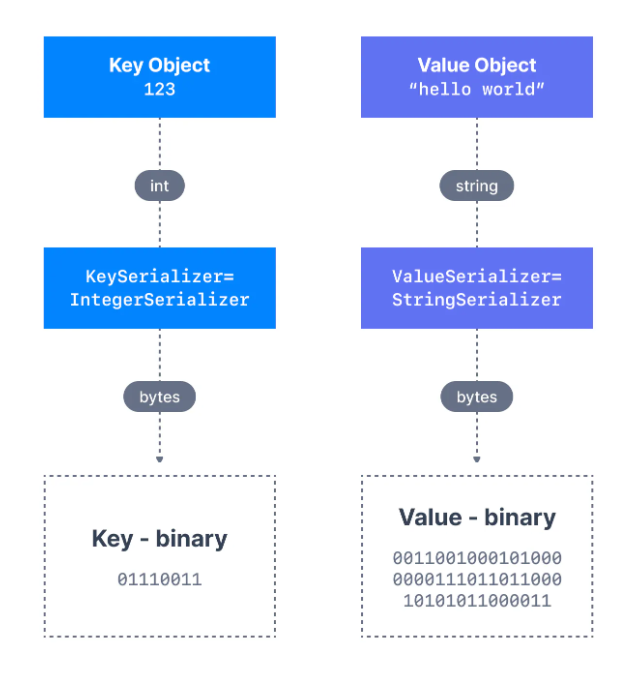
Kafka Producer gửi Message tới 1 Topic, Message sẽ được phân phối đến các Partition theo một cơ chế được gọi là Hash(key)



Trường hợp (key = null) không xác định được phân vùng, Messages sẽ được phân bổ đều vào các Partition trong một chủ đề. Điều này có nghĩa là các Message được gửi theo kiểu vòng tròn (Partition P0 rồi P1 rồi P2,… sau đó quay lại P0..)

1.2.2.2 Kafka Message Serializers

Ở đây thì 1 vấn đề đặt ra làm thế nào để Kafka có thể xử lý được nhiều định dạng của dữ liệu như định dạng số (number), định dạng kí tự (string), … Quá trình chuyển đổi đó được gọi là Message Serializtion.



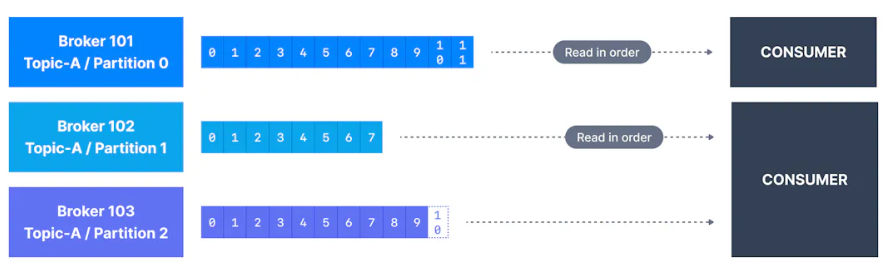
Như hình …, Đầu vào của key – value là định dạng Integer và String, Apache Kafka sử dụng IntegerSerializer để chuyển đổi key thành mảng byte và tương tự với value thì sử dụng StringSerializer để chuyển nó thành mảng byte.

1.2.3 Kafka Consumers

Khi một Topic được tạo trong Kafka và dữ liệu đã được đẩy vào Topic, các ứng dụng móc vào các luồng và sử dụng dử liệu. Các ứng dụng lấy dữ liệu từ một hoặc nhiều Topic được gọi là Kafka Consumer.

Các ứng dụng tích hợp trong thư viện Kafka Client để đọc được Kafka Topic vì nó được hỗ trợ hầu hết các ngôn ngữ lập trình phổ biến hiện nay như Python, Java, Go …

Kafka Consumer có thể đọc từ 1 đến nhiều Partition cùng một lúc và dữ liệu được đọc theo thứ tự của mỗi phân vùng.



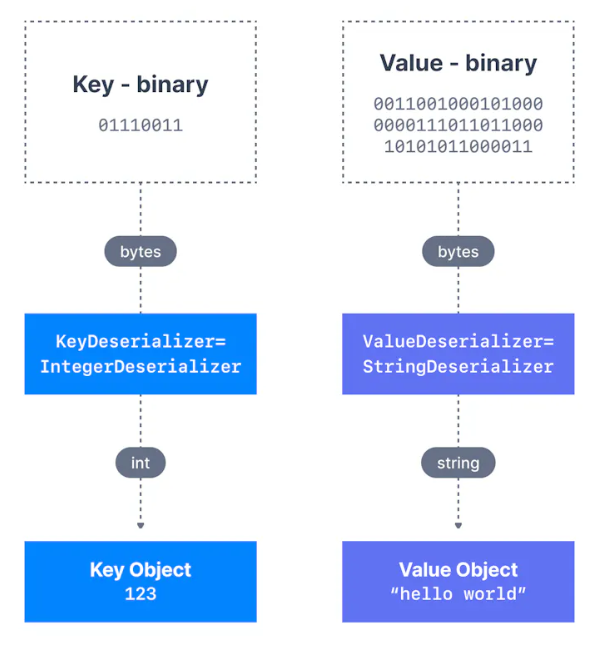
Một Consumer luôn đọc dữ liệu từ Offset theo thứ tự từ thấp đến cao, không thể đọc ngược lại vì dó cách triển khai của Apache Kafka và Client.

Nếu một Consumer sử dụng dữ liệu từ nhiều Partition, thứ tự đọc Message không được đảm bảo trên nhiều Partition vì chúng được consume đồng thời, nhưng thứ tự đọc trên từng Partition riêng lẻ vẫn được đảm bảo.

Cơ chế consume dữ liệu là “pull model”, có nghĩa là Kafka Consumer sẽ phải gửi request data tới Kafka Broker để có được dữ liệu đó thay vì yêu cầu các Kafka Broker liên tục đẩy dữ liệu đến Consumer. Việc triển khai này giúp Consumer có thể kiểm soát được tốc độ Consume các Topic.

1.2.3.1 Deserializer – giải tuần tự hóa.

Dữ liệu được đẩy vào Kafka được Serializer – tuần tự hóa. Điều này có nghĩa là dữ liệu mà Kafka Consumer nhận được cần được giải tuần tự hóa một cách chính xác để có thể sử dụng trong ứng dụng.



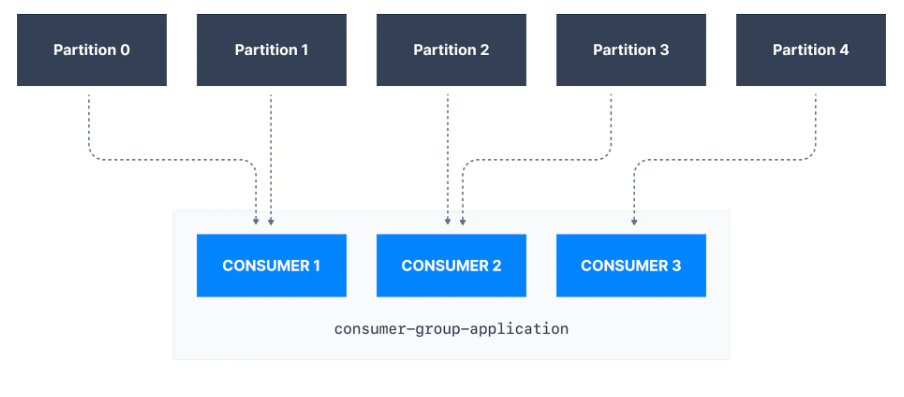
Như hình …, nếu Producer tuần tự hóa một String sử dụng StringSerializer, thì Consumer phải dùng StringDeserializer. Nếu Producer tuần tự hóa một Integer sử dụng IntegerSerializer thì Consumer phải dùng giải tuần tự hóa sử dụng IntegerDeserializer.

Định dạng tuần tự hóa và giải tuần tự hóa của một Topic không được thay đổi trong vòng đời của Topic. Nếu muốn chuyển đổi định dạng dữ liệu chủ đề (ví dụ từ JSON sang Avro) thì các tốt nhất là tạo một chủ đề mới và di chuyển các ứng dụng vào consume Topic mới đó. Vì việc giải tuần tự không chính xác có thể gây ra sự cố hoặc dữ liệu không nhất quán để cung cấp cho các ứng dụng xử lý.

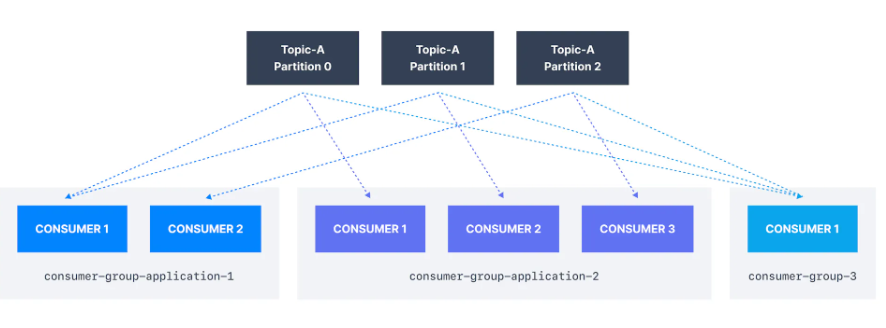
1.2.3.2 Kafka Consumer Groups

Các Consumer là một phần của ứng dụng và chúng có chung “logical job” có thể được nhóm lại với nhau gọi là Kafka Consumer Group.

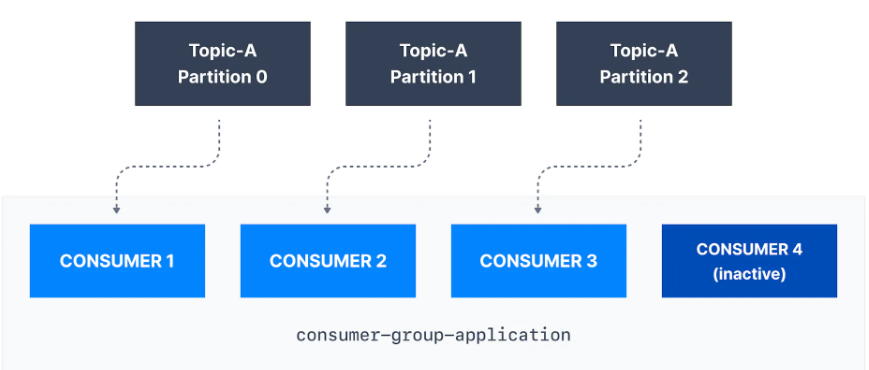
Một Topic thường có nhiều Partition, các Partition là một đươn vị song song dành cho Kafka Consumer. Lợi ích của việc tận dụng Kafka Consumer Group là các Consumer trong nhóm sẽ phối hợp để phân chia công việc đọc khỏi các Partition khác.



Mỗi ứng dụng (có thể có nhiều Kafka Consumers) đọc dữ liệu từ Kafka Topic phải chỉ định group.id để có thể quản lý được consumer consume dữ liệu.



Trong một Kafka Consumer Group, số Consumer nhiều hơn số Partition của Topic đang consume đến, thì phần hơn Consumer đó sẽ được vào hang chờ.



1.2.3.3 Kafka Comsumer Offsets.

Kafka Broker sử dụng topic có tên “\_\_consumer\_offsets” để theo dõi những thông điệp xử lý Message thành công từ được Consumer Group

Mỗi Message trong Topic đều có Partition ID và Offset ID đính kèm. Khi Consumer consume dữ liệu thành công sẽ luôn gửi thông báo tới \_\_consumer\_offset để Consumer Group có thể kiểm tra.

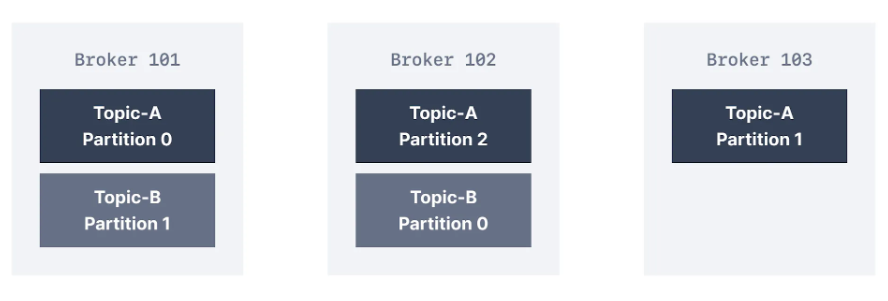


Trong trường hợp 1 Consumer mới được thêm vào Consumer Group, việc cần phải có \_\_consumer\_offset sẽ được tận dụng thông báo cho Consumer mới biết tiếp tục consume dữ liệu từ đâu.

1.2.4 Kafka Broker

Kafka Broker chứa các Kafka Topic và Kafka Partition lưu trữ dữ liệu trong một thư mục trên disk của máy chủ mà họ chạy.

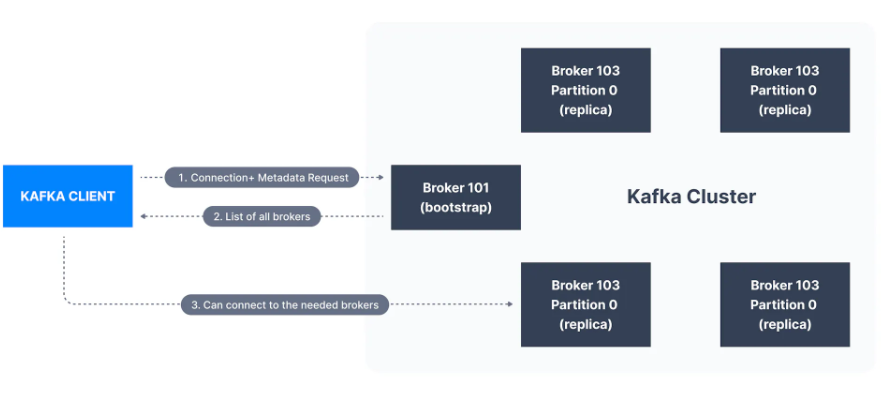
Để đạt được thông lượng cao và khả năng mở rộng theo Topic, Kafka Topic sẽ phân bổ Partition đồng đều mỗi Kafka Broker để đạt được cân bằng tải và khả năng mở rộng.



Trong hình …, có 2 Kafka Topic được minh họa. Topic A có 3 Partition được phân bổ đồng đều giữa 3 Kafka Broker trong cụm, ngoài ra có thể có nhiều hoặc ít Partition của 1 Topic trong 1 Kafka Broker. Topic B có 2 Partition, trong trường hợp này, Kafka Broker 103 không chứa Partition nào của Topic B.

Để có thể đảm bảo tính sẵn sàng cao – high vailability, và tính mở rộng – scalability, Kafka sẽ tổ chức các Kafka Broker thành 1 cụm gọi là Kafka Cluster.

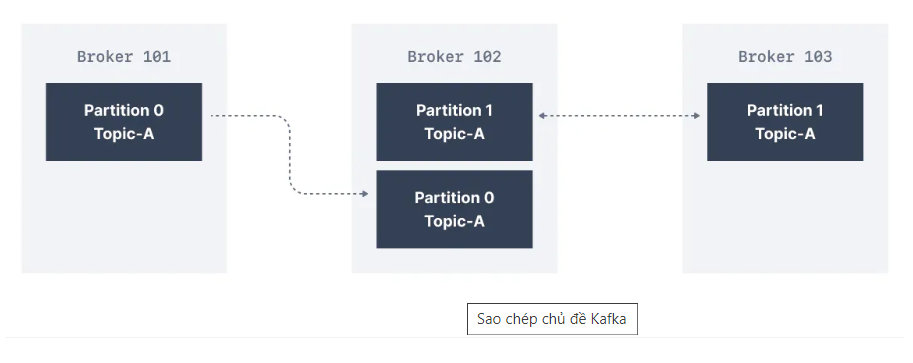
Khi Kafka Client muốn kết nối vào Kafka Cluster cần phải gửi request đến bất kì Kafka Broker nào trong cụm để lấy metadata – thông tin các Kafka Broker trong cụm. Để có thể chủ động kết nối gửi và nhận dữ liệu và tìm chính xác Broker nào chứa Topic, Partition liên quan.



1.2.5 Kafka Topic 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 Replication sao chjjeps dữ liệu giúp ngăn ngừa mất dữ liệu bằng cách ghi cùng một dữ liệu cho nhiều Broker.



Trong hình … Cụm này gồm 3 Kafka Broker, Khi một Message được ghi vào Partition P0 của Topic A trong Broker 101, nó sẽ được sao chép sang Broker 102. Tương tự Partiton 1 của Topic A cùng được sao chép sang Broker 102.

Nhờ có sự sao chép này khi Broker 101 hay Broker 103 bị lỗi thì vẫn dữ liệu vẫn được đảm bảo trên Broker 102.

1.2.6 Vai trò Zookeeper với Kafka.

Làm thế nào để Kafka Client theo dõi tất cả các Kafka Broker nếu có nhiều hơn một. Zookeepr được sử dụng cho mục đích này:

- Zookeeper được sử dụng để quản lý metadata chứa thông tin của Kafka.

- Zookeeper theo dõi những Broker trong Kafka Cluster.

- Zookeeper được Kafka Broker sử dụng để xác định Leader Broker, Leader Replicate của Topic.

- Zookeeper thông báo đến Kafka trong trường hợp có thay đổi(ví dụ : có Topic mới, Kafka Broker không hoạt động, …)

2. Confluent Kafka

Hệ thống bao gồm các thành phần:

**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 subcribe giữa Confluent Platform sang các hệ thống khác.

**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.

**Confluent Kafka Connect**: Là framework để streaming dữ liệu từ hệ thống Confluent Kafka với các hệ thống khác.

**Confluent ksqlDB**: là một công cụ giúp nhà phát triển có thể tạo các ứng dụng bằng cách sử dụng câu lệnh SQL đơn giả, 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 transform dữ liệu.

**Confluent Schema Registry**: tiến trình cung cấp cơ sở dữ liệu cho các schema được sử dụng trong cụm Confluent Kafka và xử lý việc phân phối, đồng bộ hóa các schema cho Producer và các Consumer bằng cách lưu trữ một bản sao của Schema trong bộ nhớ đệm cục bộ.

**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, connector, pipeline streaming, license,…

**Producer**: Các ứng dụng nói chung, publish Message và đẩy vào các server Confluent Kafka Broker

**Consumer**: Các ứng dụng nói chung, consume Message từ các server Confluent Kafka Broker.

3. So sánh Apache Kafka và Confluent Kafka

Là một nền tảng quản lý publish-subscribe phổ biến, Apache Kafka được sử dụng rộng rãi trong BigData, trong khi Confluent – một nền tảng truyền dữ liệu dựa trên Apache Kafka, cung cấp các công nghệ bổ sung khác. Mặc dù cả 2 nền tảng đều bao gồm các tính năng thiết yếu hỗ trợ hệ thống dữ liệu, nhưng khung của Apache Kafka chủ yếu được sử dụng trong hoạt động dữ liệu còn Confluent Kafka được sử dụng trong xử lý dữ liệu.

|  |  |  |
| --- | --- | --- |
| So sánh | Apache Kafka | Confluent Kafka |
| Use Case | Xử lý dữ liệu thời gian thực, theo dõi hoạt động (website, application), Messaging, tổng hợp log | Tương tự Apache Kafka, nâng cao các tính năng cho doanh nghiệp: sao chép đa trung tâm dữ liệu, quản trị dữ liệu và bảo mật cao. |
| Phổ biến | Được sử dụng rộng rãi bởi các nhà phải triển và tổ chức thuộc mọi quy mô | Phổ biến trong các doanh nghiệp lớn yêu cầu các tính năng và dịch vụ hỗ trợ tiên tiến. |
| Công nghệ | Open Source Distributed Event Streaming Platform | Được xây dựng dựa trên Apache Kafka và cung cấp nền tảng toàn diện hơn để Event Streaming với nhiều tính năng và công cụ bổ sung. |
| Hiệu suất | Thông lượng cao  (throughput), độ trễ thấp, khả năng chịu lỗi. | Tương tự Apache Kafka nhưng có thêm các công cụ giám sát và tối ưu hiệu suất bổ sung: sao chép trung tâm dữ liệu, cơ chế bộ nhớ đệm nâng cao. |
| Chi phí | Mã nguồn mở và miễn phí | Phiên bản Comminity miễn phí, Bản EnterPrise trả phí với các tính năng và dịch vụ hỗ trợ bổ sung. |
| Đặc điểm | Các tính năng cốt lõi: Pub/sub message, xử lý luồng và lưu trữ có khả năng chịu lỗi | Thêm các tính năng như Schema Registry, Connector, KsqlDB, multi-datacenter replication, Control center. |
| Giấy phép | Giấp phép Apache 2.0 | Giấy phép Confluent Comminity (dựa trên Apache 2.0) và giấy phép Confluent Enterprise. |
| Độ thân thiện, dễ sử dụng | Yêu cầu kiến thức chuyên môn để thiết lập, triển khai và định cấu hình nhưng có API tương đối đơn giản dành cho developer | Cung cấp các công cụ và dịch vụ để đơn giản hóa việc triển khai, cấu hình và quản lý. |
| Dịch vụ và Hỗ trợ | Có cộng đồng hỗ trợ nhưng không có hỗ trợ chính thức từ dự án Apache Kafka | Cung cấp dịch vụ hỗ trợ và đào tạo, tư vẫn doanh nghiệp |
| Cộng đồng | Cộng đồng mã nguồn mở lớn và tích cực | Cộng đồng nhỏ nhưng tập trung vào các trường hợp và tính năng sử dụng của doanh nghiệp |
| Khả năng tích hợp | Tích hợp nhiều nguồn dữ liệu, khung xử lý khác nhau | Tương tự Apache Kafka nhưng tích hợp bổ sung các công cụ và dịch vụ độc quyền của Confluent: Schema Registry, Control Center, REST Proxy, Linking Kafka Cluster. |
| Bảo mật | Bảo mật mã hóa SSL/TLS (Single Sercurity Layout/ Transport Layer Sercurity), xác thực( Authentication), ủy quyền (Authorization) | Tương tự Apache Kafka, thêm mã hóa dữ liệu, Audits Log, Kiểm soát truy cập với vai trò ( RBAC -Role-based Access Control) |
| Giám sát | Thông qua JMX – Java management extension và thao tác dòng lệnh. | Thông qua Confluent Control Center. |
| Triển khai | Có thể triển khai ở Local hoặc trên Cloud | Tương tự Apache Kafka, có thêm lựa chọn triển khai trên Confluent Cloud. |
| Hệ sinh thái | Hệ sinh thái lớn và đang phát triển các công cụ và dịch vụ bên thứ ba | Tương tự Apache Kafka nhưng bổ sung các công cụ và dịch vụ từ Confluent |
| Trình kết nối | Cung cấp thư viện gồm các trình kết nối để tích hợp với nhiều nguồn dữ liệu và phần lưu trữ khác nhau. | Tương tự Apache Kafka, bổ sung các Connector do Confluent phát triển. |

Ưu và nhược điểm của Confluent :

Ưu điểm:

* Trình kết nối có sẵn: Khoảng 100 trình kết nối được dựng sẵn từ Confluent Kafka giúp việc tích hợp dữ liệu với nhiều hệ thống khác nhau, chẳng hạn như cơ sở dữ liệu, dịch vụ Cloud, tiện ích IoT trở nên vô cùng đơn giản.
* Các tính năng nâng cao: Confluent Kafka cung cấp Schema Registry, Control Center, Rest Proxy.
* Dịch vụ được quản lý: Confluent Cloud là dịch vụ được quản lý cung cấp trên cơ sở trả tiền theo nhu cầu sử dụng
* Tính sẵn sàng cao: Khả năng khôi phục, chịu lỗi cao, tính sẵn sàng cao.

Nhược điểm:

* Hạn chế: Khóa nhà cung cấp, độ phức tạp tăng lên và phụ thuộc vào lộ trình của Confluent.
* Chi phí: Khách hang doanh nghiệp của Confluent Kafka phải trả phí để sử dụng nhiều tính năng, dịch vụ hỗ trợ.
* Độ phức tạp: các công cụ và tính năng phức tạp của Confluent Kafka có thể khó triển khai, định cấu hình và bảo trì phức tạp
* Phụ thuộc vào các công nghệ: Phụ thuộc vào công nghệ Zookeeper để hoạt động, khiến triển khai trở nên phức tạp.

Ưu điểm điểm của Apache Kafka:

* Thông lượng cao: Yêu cầu truyển dữ liệu lớn theo thời gian thực, Kafka có thể xử lý hang triệu Message mỗi giây với độ trễ thấp.
* Khả năng mở rộng: Khả năng mở rộng theo chiều ngang của Kafka giúp người dùng có thể dễ dàng thêm hoặc xóa các nút khỏi cụm để phù hợp với nhu cầu sử dụng.
* Chịu lỗi: Kafka được xây dựng để có khả năng chịu lỗi, nó vẫn có thể hoạt động ngay cả khi một số nút ngừng hoạt động hoặc một số thống báo bị thiếu.
* Tính linh hoạt: Kafka tương thích với nhiều nguồn dữ liệu và có nhiều ứng dụng.

Nhược điểm của Kafka:

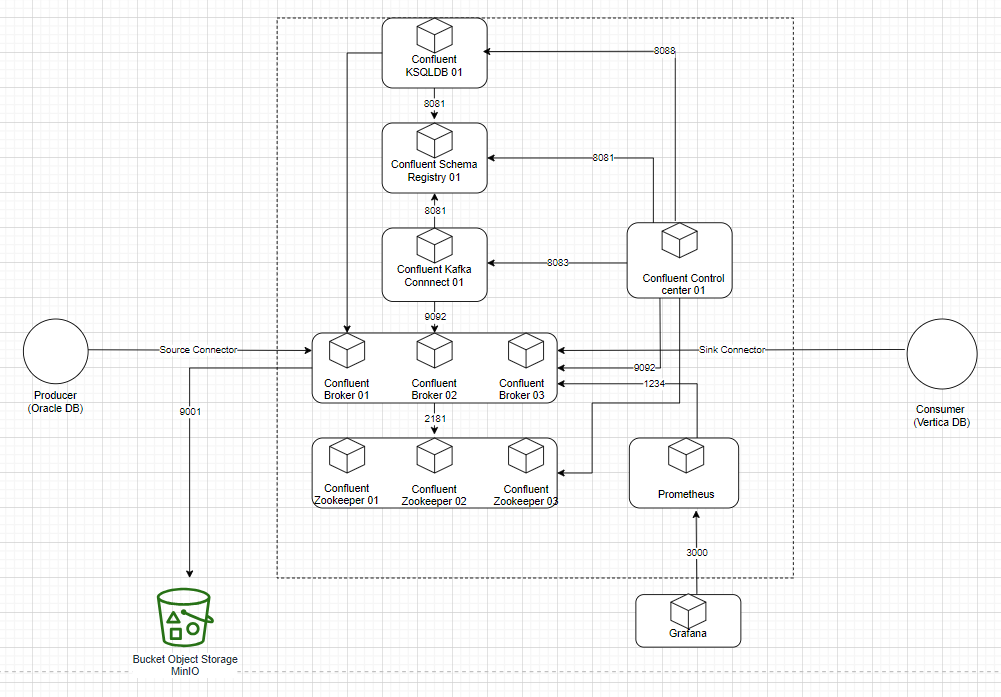
* Độ phức tạp: Kafka khó thiết lập và cài đặt, quản lý cần có kiến thức sâu rộng.
* Chi phí: Mặc dù là mã nguồn mở và được sử dụng miễn phí nhưng nhiều công nghệ và tính năng bổ sung được cung cấp bởi Confluent thì chỉ có thể thông qua giấy có tính phí.
* Công nghệ bổ sung: Người dùng có thể cần sử dụng công nghệ bổ sung chẳng hạn như Schema Registry, Rest Proxy để tận dụng tối ưu Kafka điều này làm phức tạp hệ thống
* Bảo trì: Kafka yêu cầu bảo trì liên tục để duy trì tính khả dụng và hiệu suất tốt, giống như bất kì hệ thống phân tán nào. Đối với các công ty thiếu nguồn lực công nghệ mạnh thì việc này có thể rất tốn kém.

**Các tính năng chỉ có khi sử dụng Confluent Kafka (Mang lại giá trị cho doanh nghiệp):**

1. Cải thiện tốc độ, hiệu năng của hệ thống.
2. Quản lý/ Giám sát tập trung trang thái từng cấu phần của cụm trên Control Center.
3. Dữ liệu được đồng bộ DC-DR (DC: Data Center, DR: Desister Recovery)
4. Thao tác/ Kiểm soát dữ liệu thuận tiện, dễ dàng với tính năng KsqlDB, Schema Registry.
5. Giảm thời gian rebalance khi khắc phục sự cố, khôi phục dữ liệu.
6. Có thêm lựa chọn Kafka chủ động lấy dữ liệu từ các hệ thống khác (Vì dữ liệu được đẩy 1 chiều t ừ các hệ thống lên Kafka lưu trữ, chưa có chiều Kafka chủ động get/lấy dữ liệu).

**TỔNG QUAN HỆ THỐNG CONFLUENT KAFKA:**

1. **Mô hình tổng quan hệ thống:**



1. **Mô tả hệ thống:**
2. **Cách thức hoạt động:**

* Application Producer/Consumer sẽ Produce/Consume dữ liệu từ Confluent Kafka Broker.
* Cụm Confluent Platform được cấu hình lưu trữ dữ liệu dưới Bucket (Object Storage).
* Service Confluent Kafka Connect cung cấp các Connector giúp streaming dữ liệu từ Confluent Platform sang các hệ thống khác.
* Service Confluent KsqlDB, người dùng tạo các Stream/Table để tạ các luồng Streaming Transform dữ liệu từ các Topic tùy theo yêu cầu nghiệp vụ và sink xuống hệ thống đích.
* Trong quá trình hoạt động, quản trị viên có thể quản lý hoặc giám sát cụm Confluent Phatform bằng service Confluent Control Center (WebUI). Bên cạnh đó, trên cụm Confluent Platform, quản trị viên có thể giám sát hoạt động của cụm bằng Grafana (Được tích hợp vào trong hệ thống Confluent Kafka)

1. **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 |

1. **Thông tin kết nối:**

Đường dẫn kết nối tới UI Confluent Control Center:

http://10.10.12.241:9021

**QUY TRÌNH TRIỂN KHAI CÀI ĐẶT:**

1. **Các gói cài đặt:**

|  |
| --- |
| 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 |

1. **Chi tiết đường dẫn các service:**

|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Cấu phần | Đường dẫn cài đặt |
| Config | Đường dẫn lưu file cấu hình | Confluent Zookeeper | /etc/kafka/zookeeper.properties |
| Confluent Server | /etc/kafka/server.properties |
| Confluent Schema Registry | /etc/schema-registry/schema-registry.properties |
| Confluent Kafka Connect | /etc/kafka/connect-distributed.properties |
| Confluent ksqlDB | /etc/ksqldb/ksql-server.properties |
| Confluent Control Center | /etc/confluent-control-center/control-center-production.properties |
| LOG | Đường dẫn lưu file log | Confluent Zookeeper | /data/confluent/apps/zookeeper/logs |
| Confluent Server | /data/confluent/apps/kafka/logs |
| Confluent Schema Registry | /data/confluent/apps/schema/logs |
| Confluent Kafka Connect | /data/confluent/apps/connect/logs |
| Confluent ksqlDB | /data/confluent/apps/ksqldb/logs |
| Confluent Control Center | /data/confluent/apps/control-center/logs |
| Data | Đường dẫn lưu dữ liệu | Confluent Zookeeper | /data/confluent/apps/zookeeper/data |
| Confluent Server | /data/confluent/apps/kafka/data |
| Confluent Schema Registry |  |
| Confluent Kafka Connect | /data/confluent/apps/connect/connectors |
| Confluent ksqlDB | /data/confluent/apps/ksqldb/data |
| Confluent Control Center | /data/confluent/apps/control-center/data |

1. **Workflow của quá trình cài đặt hệ thống Confluent Platform**
2. Cài đặt packages trên từng server
3. Cấu hình Confluent Zookeeper
4. Khởi động Confluent Zookeeper
5. Cấu hình Confluent Server
6. Khởi động Confluent Server
7. Cấu hình Confluent Schema Registry
8. Khởi động Confluent Schema Registry
9. Cấu hình Confluent Kafka Connect
10. Khởi động Confluent Kafka Connect
11. Cấu hình Confluent KSQLDB
12. Khởi động Confluent KSQLDB
13. Cấu hình Confluent Control Center
14. Khởi động Confluent Control Center
15. **Các bước chi tiết trong quá trình cài đặt hệ thống Confluent Platform**
    1. Cài đặt package:
16. 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 |

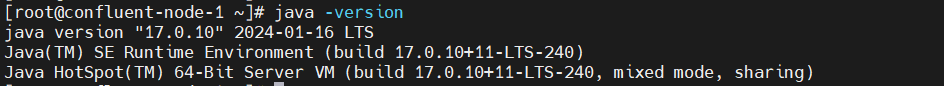
* 1. Kiểm tra java version:

Confluent recommend java version 17 cho Confluent Platform 7.6

1. Kiểm tra java version

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

1. Kết quả

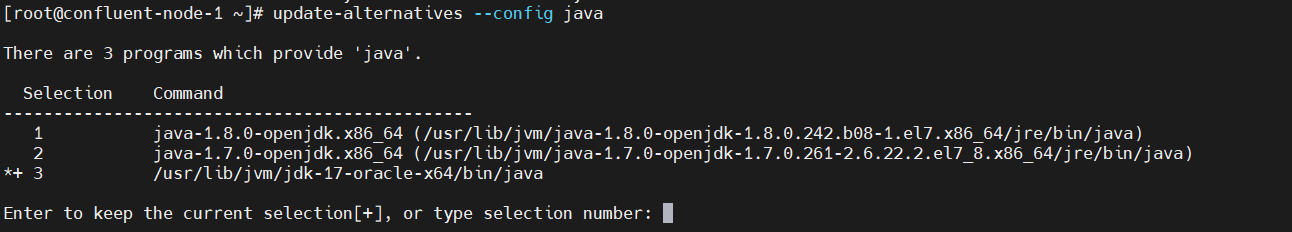


1. Nếu kết quả trả về là các phiên bản khác, cần phải thực hiện lệnh sau để thay đổi cấu hình java:

|  |
| --- |
| 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 |



Nhập số 3 để chọn version java.

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

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

Thêm đoạn mã sau vào file .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 |

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

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

Tạo group 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 file myid trong phân vùng chứa dữ liệu của Zookeeper, trong file chứa id của Zookeeper

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

1. Sử dụng câu lệnh cat để kiểm tra file myid

|  |
| --- |
| cat $ZOOKEEPER\_ROOT\_DIR/data/myid |



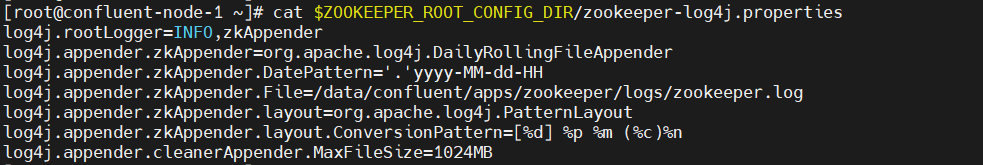
Bước 2: Cấu hình log4j cho Confluent Zookeeper

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 |



Bước 3: Cấu hình lại file 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 file 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 file 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 |

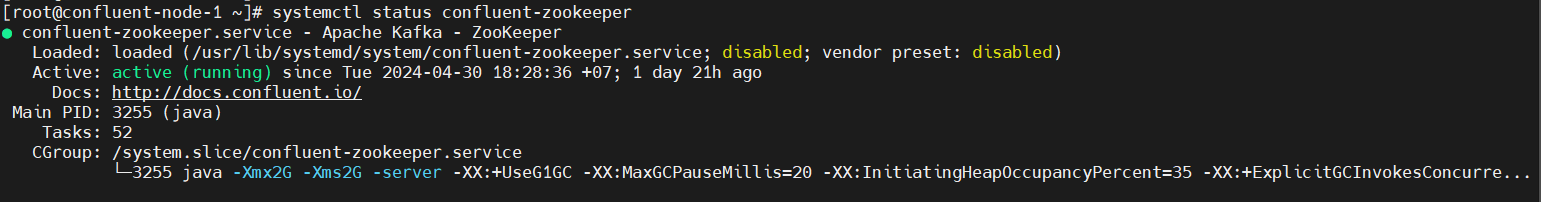
**Tương tự với confluent-node-2 và confluent-node-3.**

BƯỚC 5: Khởi chạy Confluent Zookeeper

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ thực hiện câu lệnh | 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 confluent zookeeper |  | tail -1000f /data/confluent/apps/zookeeper/logs/zookeeper.log |

Kiểm tra trạng thái của service:

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



1. **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

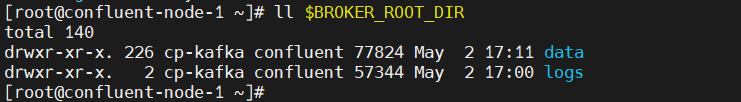
|  |
| --- |
| mkdir -p $BROKER\_ROOT\_DIR/{data,logs} |

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 |



Bước 2: Cấu hình lại file service cho Server:

1. Backup file confluent-server-service

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

1. Thêm cấu hình vào file 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 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 $BROKER\_ROOT\_CONFIG\_DIR/certs |

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

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

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

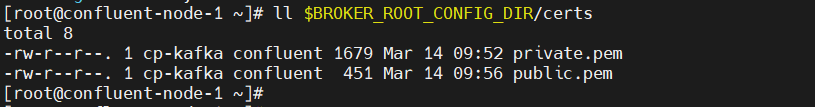
|  |
| --- |
| openssl rsa -in $BROKER\_ROOT\_CONFIG\_DIR/certs/private.pem -outform PEM -pubout -out $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem |

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 |



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 file cấu hình mới cho Server

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

1. Thêm cấu hình vào file 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-zFpGXqgNajg75aqNxj7WgPcqrrbJSWIirFgD1IKR2oXaXR2m8FWw  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  #MINIO\_S3\_API\_AWS  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  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 |

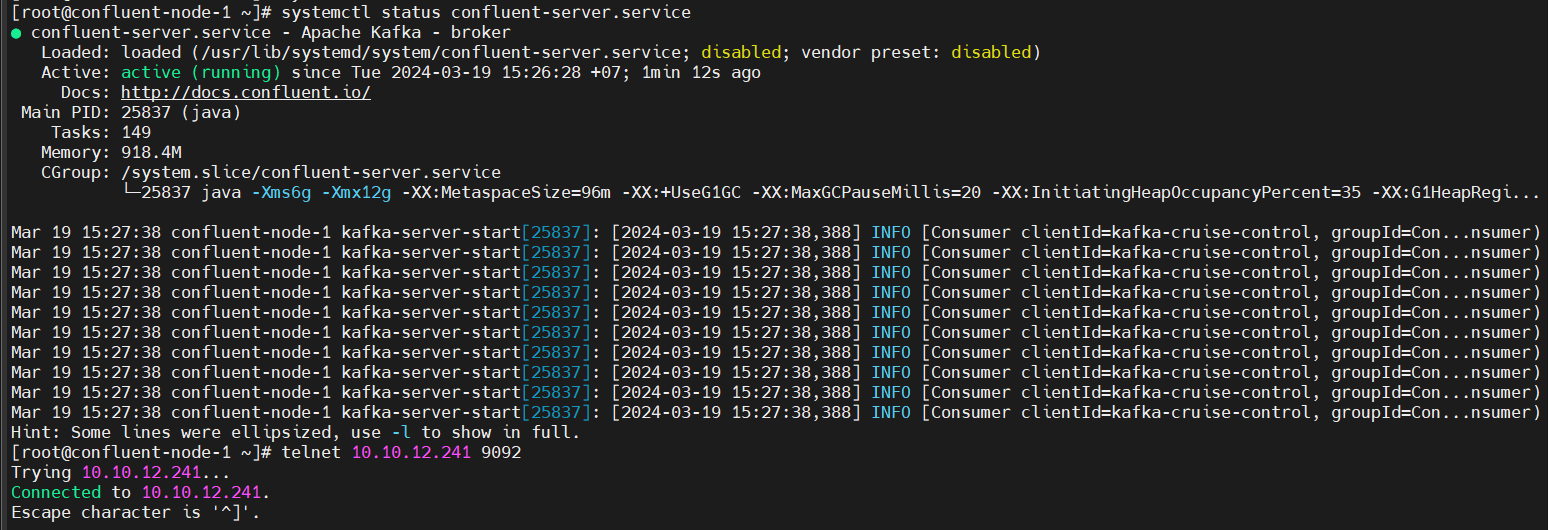
**Làm tương tự với confluent-node-2 và confluent-node-3.**

BƯỚC 5: Khởi chạy Confluent Server

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ thực hiện câu lệnh | 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 |



1. **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

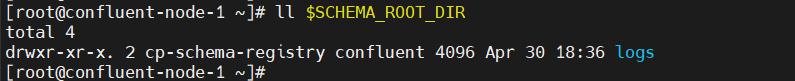
|  |
| --- |
| mkdir -p $SCHEMA\_ROOT\_DIR/logs |

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 |



Bước 2: Cấu hình lại file service cho :

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 file 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 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 $SCHEMA\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép file private key:

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

1. Sao chép file public key:

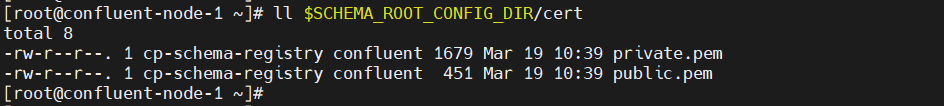
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $SCHEMA\_ROOT\_CONFIG\_DIR/certs |

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 |



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 file cấu hình mới cho Schema Registry

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

1. Thêm cấu hình vào file Schema Registry

Cấu hình tích hợp với AD/LDAP

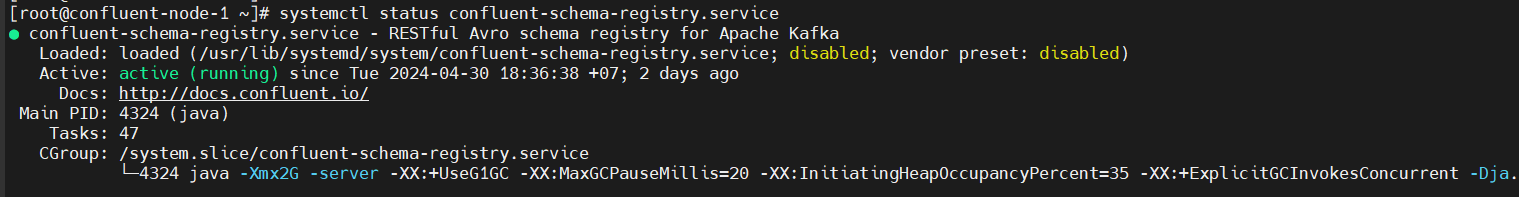
|  |
| --- |
| ####CONFIG LDAP####  listeners=http://0.0.0.0:8081  kafkastore.bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  kafkastore.topic=\_schemas  debug=false  kafkastore.security.protocol=SASL\_PLAINTEXT  kafkastore.sasl.mechanism=OAUTHBEARER  kafkastore.sasl.login.callback.handler.class=io.confluent.kafka.clients.plugins.auth.token.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 |

1. Khởi động Confluent Schema Registry Service:

|  |  |  |
| --- | --- | --- |
| Công việc | Server thực thi | 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 |



1. **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

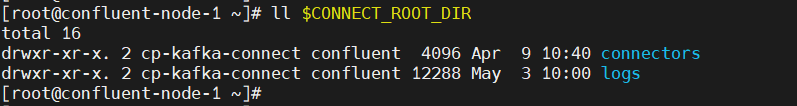
|  |
| --- |
| mkdir -p $BROKER\_ROOT\_DIR/{connector,logs} |

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 |



Bước 2: Cấu hình lại file 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 file 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 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 $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

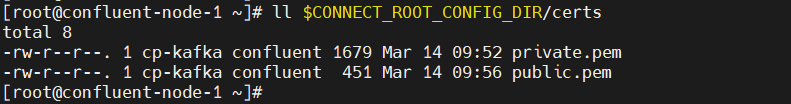
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $CONNECT\_ROOT\_CONFIG\_DIR/certs |

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 |



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 file cấu hình mới cho Confluent Kafka Connect

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

1. Thêm cấu hình vào file Confluent Kafka Connect

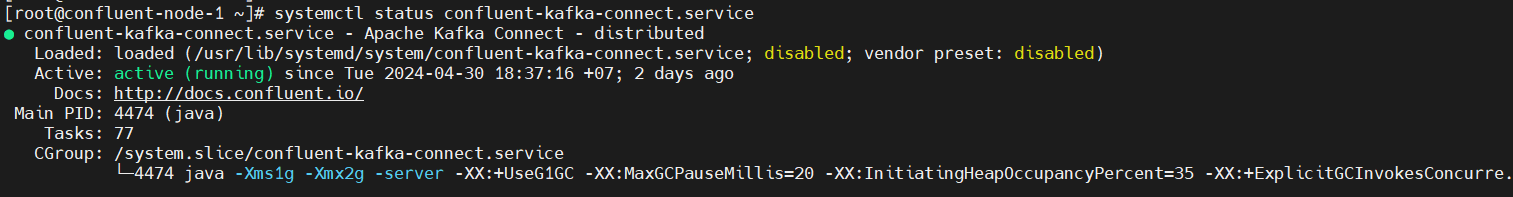
|  |
| --- |
| ###CONFIG RBAC####  key.converter.schemas.enable=true  value.converter.schemas.enable=true  offset.flush.interval.ms=10000  plugin.path=/usr/share/java,/usr/share/confluent-hub-components,/data/confluent/apps/connect/connectors  admin.bootstrap.servers=10.10.12.241:9091,10.10.12.242:9091,10.10.12.244:9091  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 |

Bước 5: Khởi động Confluent Kafka Connect

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ thực hiện | 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 |



1. **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

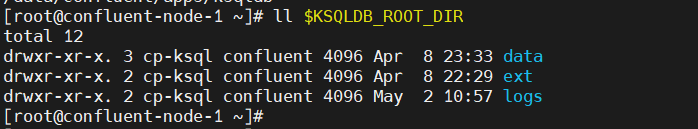
|  |
| --- |
| mkdir -p $KSQLDB\_ROOT\_DIR/{data,logs,ext} |

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 |



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

1. Backup file confluent-ksqldb-service

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

1. Thêm cấu hình vào file 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 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 $KSQLDB\_ROOT\_CONFIG\_DIR/certs |

1. Sao chép file private key

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

1. Sao chép file public key

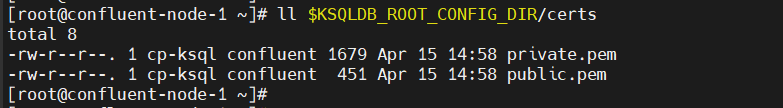
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $KSQLDB\_ROOT\_CONFIG\_DIR/certs |

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 |



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 file Server

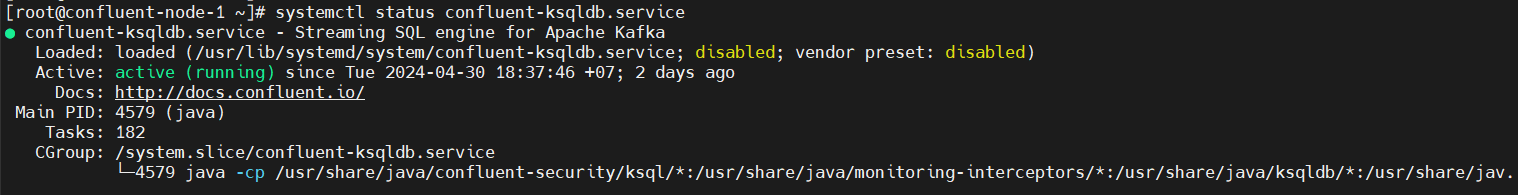
|  |
| --- |
| ###CONFIG RBAC####  listeners=http://0.0.0.0:8088  ksql.logging.processing.topic.auto.create=true  ksql.logging.processing.stream.auto.create=true  producer.interceptor.classes=io.confluent.monitoring.clients.interceptor.MonitoringProducerInterceptor  consumer.interceptor.classes=io.confluent.monitoring.clients.interceptor.MonitoringConsumerInterceptor  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 |

Bước 5: Khởi chạy Confluent KsqlDB service:

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ thực hiện câu lệnh | 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 |



1. **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

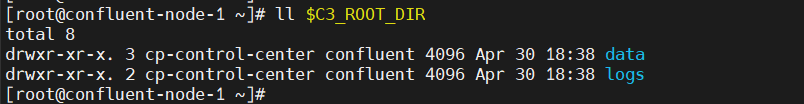
|  |
| --- |
| mkdir -p $C3\_ROOT\_DIR/{data,logs} |

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 |



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

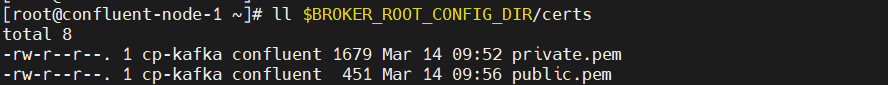
|  |
| --- |
| cp $BROKER\_ROOT\_CONFIG\_DIR/certs/public.pem $C3\_ROOT\_CONFIG\_DIR/certs |

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 |



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=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJjb250cm9sLWNlbnRlciIsIm5iNCI6IjE2OTU2MTM1ODEiLCJtb25pdG9yaW5nIjp0cnVlLCJsaWNlbnNlVHlwZSI6IkVudGVycHJpc2UiLCJpc3MiOiJDb25mbHVlbnQiLCJpYXQiOjE2OTM1NTE2MDAsImV4cCI6MTc5NTkzNTYwMCwiYXVkIjoiMDA2NFUwMDAwMHNxallQUUFZIn0=.Rjm0el5v7BKbC\_jcu\_lmuHQeoAhKQ3xP4MM1KSl8wo7859OogL954nO2BbNos3OwplXbvuu7\_IsSFyf8UBFK2zUtHV8elCfcIcAxu0uE-pMhYZoBuq\_OXePvrBfRs8TaJOzXq6CcdSrkekL-Ym7epXOO4YOWQzJXLU6GzVcOVDv8bTMEYtVHw-lE3nOYa7rLci9UiKKWex5u4hLdCJMvSNIFhOz5xm5WcGhhSv8mpZJapaVfcxJ5585\_xQ-w\_0QnG-mcLKDhNOkBfG97DId93KC4MIKQe1Cgs-zFpGXqgNajg75aqNxj7WgPcqrrbJSWIirFgD1IKR2oXaXR2m8FWw  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 |

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

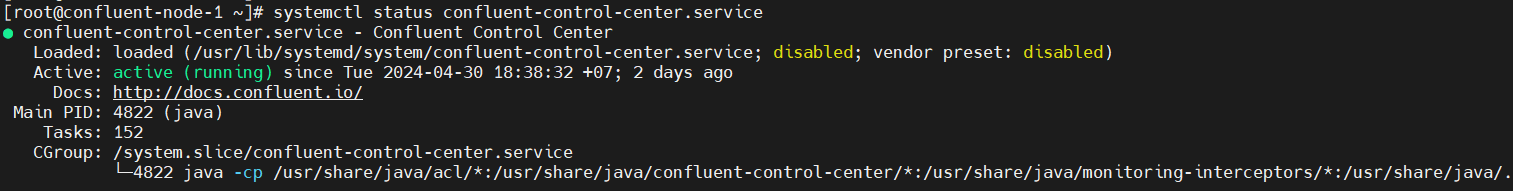
|  |
| --- |
| cluster.super.user.credentials=denodo  cluster.super.user.username=vertica\_4U  user.info=denodo:vertica\_4U  confluent.license=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJjb250cm9sLWNlbnRlciIsIm5iNCI6IjE2OTU2MTM1ODEiLCJtb25pdG9yaW5nIjp0cnVlLCJsaWNlbnNlVHlwZSI6IkVudGVycHJpc2UiLCJpc3MiOiJDb25mbHVlbnQiLCJpYXQiOjE2OTM1NTE2MDAsImV4cCI6MTc5NTkzNTYw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

Bước 5: Khởi chạy Confluent Control Center Service:

|  |  |  |
| --- | --- | --- |
| Công việc | Máy chủ thực hiện | 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 |



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

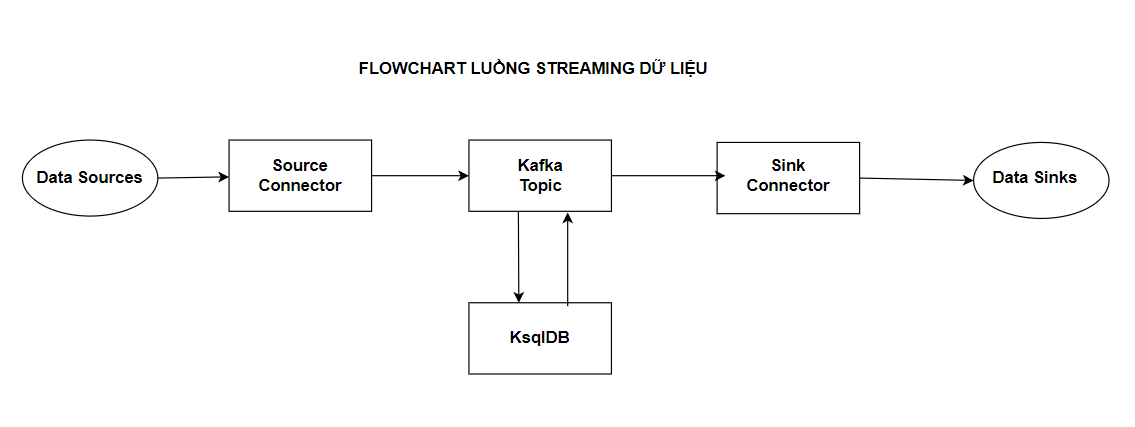
|  |
| --- |
| confluent login --url http://10.10.12.241:8091 |

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 |

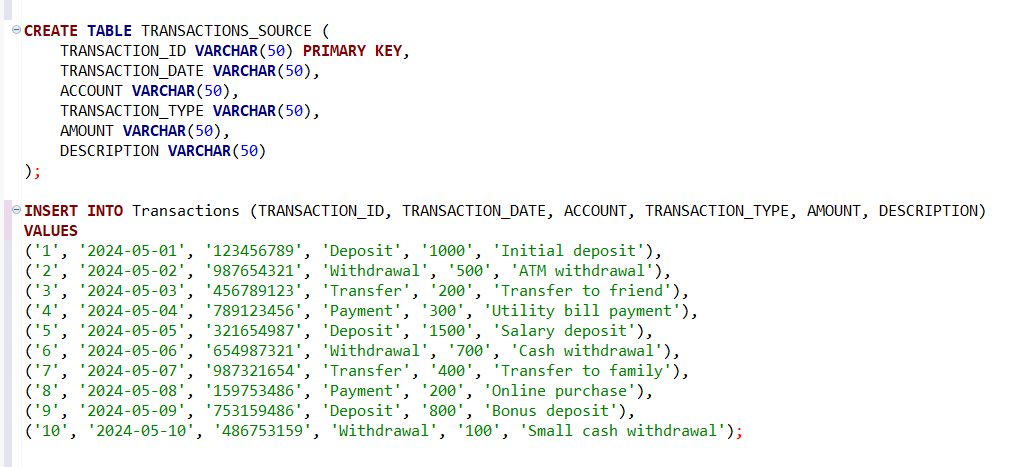
|  |
| --- |
| confluent iam rbac role-binding create --principal User:denodo --role SystemAdmin --kafka-cluster $kafka\_cluster --connect-cluster default\_ |

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



CÁC BƯỚC THỰC HIỆN

1. Chuẩn bị dữ liệu ở nguồn source:

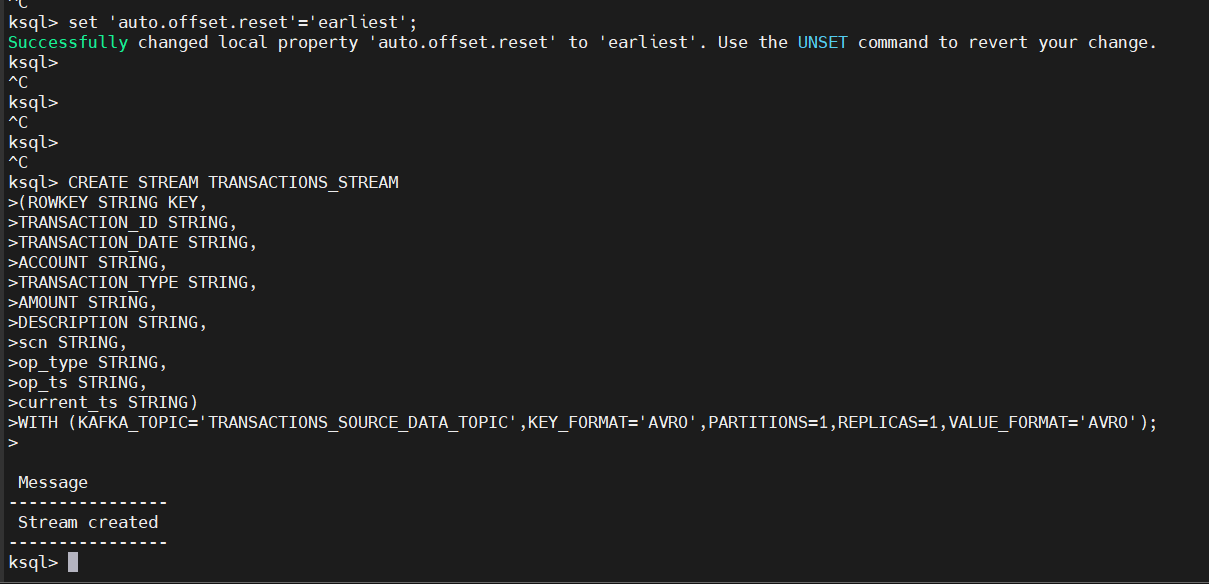


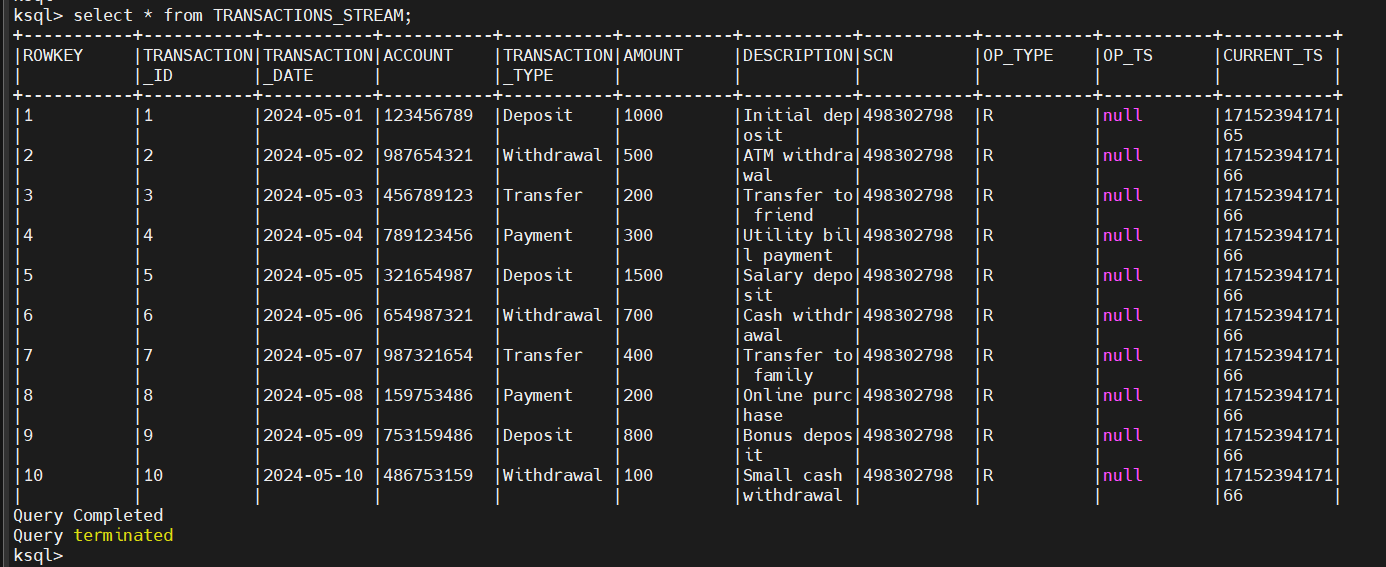
1. Cấu hình Source Connector

|  |
| --- |
| {  "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"  }  } |

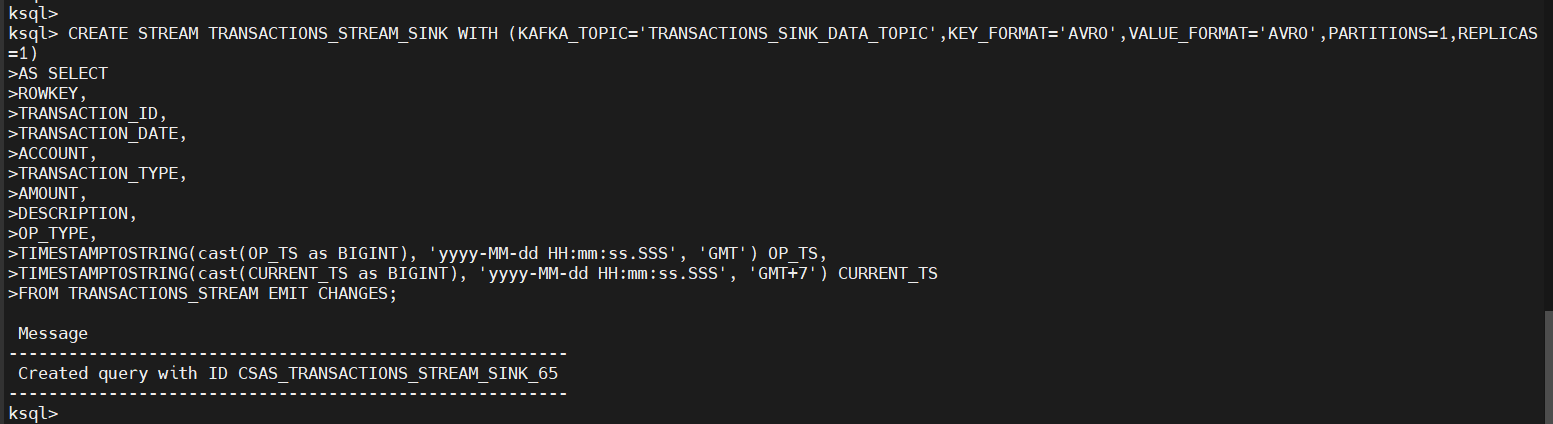
1. Tạo luồng Stream sử dụng KQSLDB service:

Đẩy dữ liệu từ topic TRANSACTIONS\_SOURCE\_DATA\_TOPIC vào luồng Stream TRANSACTIONS\_STREAM:





Tạo thêm luồng TRANSACTIONS\_STREAM\_SINK từ TRANSACTIONS\_STREAM để biến đối dữ liệu của cột OP\_TS, CURRENT\_TS từ int64 sang timestamp người có thể đọc được và đẩy dữ liệu vào topic mới TRANSACTIONS\_SINK\_DATA\_TOPIC.



1. Cấu hình sink connnector

|  |
| --- |
| {  "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",  "topics": "TRANSACTIONS\_SINK\_DATA\_TOPIC",  "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"  }  } |

Sink Connector chọc vào topic TRANSACTIONS\_SINK\_DATA\_