**KHẢO SÁT TÍNH NĂNG CÔNG CỤ FHIR-GATEWAY**

# Tổng quan về công cụ FHIR – Gateway

FHIR – Gateway là một lớp dịch vụ được tích hợp nhiều tính năng làm định danh, cầu nối, liên kết, chia sẻ dữ liệu, nằm giữa các Client FHIR REST (Representational State Transfer) và các Resource theo dạng xử lý đường ống (Pipeline).

## Tổng quan

Diagram

Description automatically generated

Hệ thống FHIR – Gateway được chia làm 4 khối chính với tính năng và vai trò khác nhau góp phần quản lý, truy xuất, dẫn truyền dữ liệu từ khối đầu tiên cho tới khối cuối cùng. 4 khối đó bao gồm:

* **Khối dữ liệu đầu vào**: Bao gồm các luồng dữ liệu từ nhiều nguồn khác nhau cùng có nhu cầu thực hiện kết nối với hệ thống FHIR.
* **Khối FHIR – Gateway**: Đóng vai trò tiếp nhận các luồng dữ liệu từ đầu vào từ đó tiến hành phân luồng, quản trị, giám sát, tìm kiếm, truy xuất, tương tác dữ liệu đó với kho dữ liệu.
* **Khối kho lưu trữ dữ liệu**: Bao gồm các kho dữ liệu dùng để lưu trữ, bảo mật các luồng dữ liệu đi qua hệ thống bao gồm (HAPI FHIR, PostgreSQP, Kafka, Minio v.v…).
* **Khối chuẩn hoá dữ liệu**: Khối này có chức năng chuẩn hoá cấu trúc các dữ liệu có sẵn nhằm đáp ứng các tổ chức quốc tế kiểm định chất lượng dịch vụ y tế (i2b2, OMOP v.v…).

## Thiết lập cấu hình

|  |  |  |
| --- | --- | --- |
| Biến môi trường | Mô tả | Mặc định |
| SPRING\_DATASOURCE\_URL | URL JDBC của Postgres DB để lưu trữ các tài nguyên FHIR đã nhận, cần được đặt thành một biến trống nếu không có PSQL DB nào được kết nối | jdbc:postgresql://fhir-db:5432/fhir |
| SPRING\_DATASOURCE\_USERNAME | Tên người dùng của PostgresSQL DB | postgres |
| SPRING\_DATASOURCE\_PASSWORD | Mật khẩu cho PostgresSQL DB | postgres |
| SERVICES\_PSEUDONYMIZER\_ENABLED | Liệu có nên bật biệt hiệu hay không | true |
| SERVICES\_PSEUDONYMIZER\_URL | URL của dịch vụ FHIR Pseudonymizer | <http://fhir-pseudonymizer:8080/fhir> |
| SERVICES\_LOINC\_CONVERSIONS\_URL | URL của dịch vụ chuyển đổi LOINC | <http://loinc-converter:8080/conversions> |
| SERVICES\_FHIRSERVER\_ENABLED | Có gửi các tài nguyên đã nhận đến máy chủ FHIR hạ lưu hay không | false |
| SERVICES\_FHIRSERVER\_URL | URL của máy chủ FHIR để gửi dữ liệu đến | <http://fhir-server:8080/fhir> |
| SERVICES\_KAFKA\_ENABLED | Cho phép đọc các tài nguyên FHIR từ và ghi chúng trở lại một cụm Kafka | false |
| SERVICES\_KAFKA\_GENERATE\_OUTPUT\_TOPIC\_MATCH\_EXPRESSION | Cho phép tạo động tên của chủ đề đầu ra Kafka dựa trên chủ đề đầu vào. Được sử dụng để đặt một biểu thức chính quy được áp dụng cho chủ đề đầu vào và kết quả khớp đầu tiên được thay thế bằng giá trị của SERVICES\_KAFKA\_GENERATE\_OUTPUT\_TOPIC\_REPLACE\_WITH. Bạn có thể đặt giá trị này thành để thêm tiền tố vào chủ đề đầu ra. | "" |

Để định cấu hình triển khai cho hệ thống, bạn có thể thay đổi các biến môi trường sau để phù hợp với các nhu cầu và điều kiện của hệ thống:

## Tương tác với FHIR – Gateway

Hệ thống FHIR - Gateway không phải là một máy chủ FHIR chính thức và chỉ hỗ trợ một tập hợp con các tương tác máy chủ RESTful. Hỗ trợ các trường dữ liệu POST/PUT/DELETE

* POST/PUT:

Gateway chỉ hỗ trợ các tài nguyên tồn tại được HTTP POSTed dưới dạng Gói FHIR sử dụng ngữ nghĩa update-as-create (cập nhật giống như tạo mới).

Lần lượt test tất cả các Resource thường được sử dụng trong chuẩn HL7-FHIR (Observation, Patient, Organization, Encounter, Location, Practitioner, Condition, Composition v.v…) bằng giao thức API gọi thẳng trực tiếp đến server fhir-gateway tại địa chỉ [*http://fhir-server:8080/fhir*](http://fhir-server:8080/fhir) thông qua phần mềm Postman.

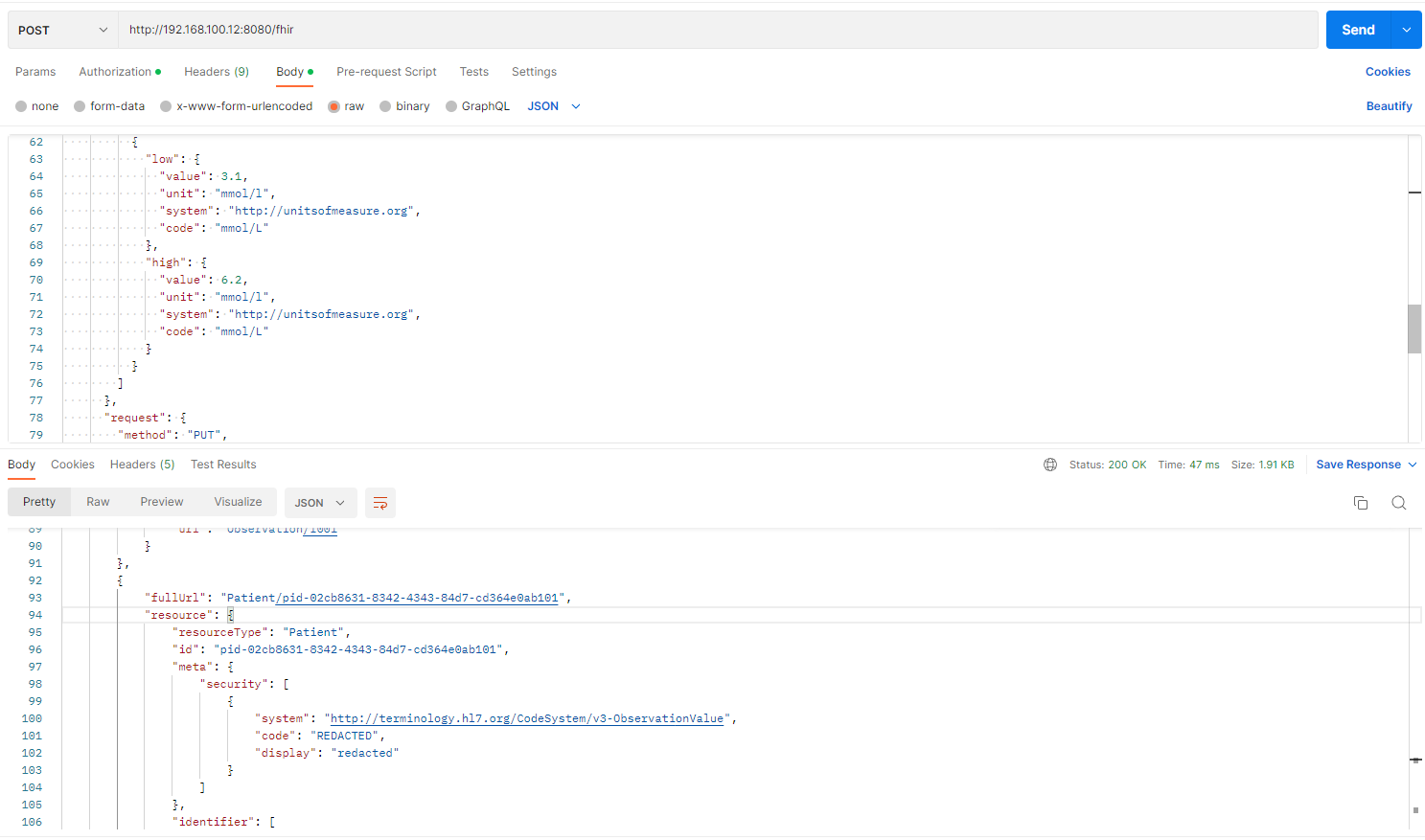
* **Observation**

**Text

Description automatically generated**

* **Patient**
* **Organization**
* **Encounter**
* **Location**
* **Practitioner**
* **Condition**
* **Composition**

Kết quả trả về khi thực hiện POST/PUT API thư mục bundle.json thông qua phần mềm Postman.



* DELETE

Các gói FHIR chứa các yêu cầu DELETE cũng được xử lý và sẽ dẫn đến việc xóa tài nguyên được chỉ định trong URL yêu cầu. Lưu ý rằng các tài nguyên được đánh dấu là is\_deleted trong Cơ sở dữ liệu PostgreSQL của Gateway thay vì bị xóa vật lý.

Lưu ý rằng không hỗ trợ tạo cũng như xóa có điều kiện. Trong khi điều này hoạt động:

{

"request": {

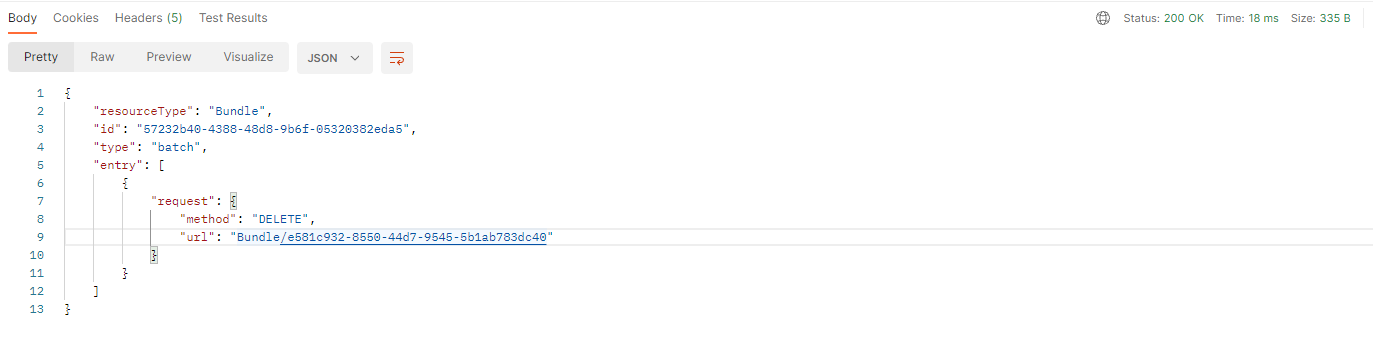
"method": "DELETE",

"url": "Patient/234"

}

}

Kết quả trả về khi thực hiện DELETE API thông qua phần mềm Postman.



# Hệ thống Apache Kafka

## Tổng quan về hệ thống Apache Kafka

Apache Kafka nhìn chung là một giao thức dùng để thực hiện phát các sự kiện trực tiếp (Event Streaming Flatform).

Kafka kết hợp ba khả năng chính để bạn có thể triển khai các trường hợp sử dụng của mình để phát trực tuyến sự kiện (Event Streaming) từ đầu đến cuối (end-to-end) bằng một giải pháp đã được thử nghiệm trong quá trình duy nhất:

* Để xuất bản (ghi) và đăng ký (đọc) các luồng sự kiện, bao gồm nhập / xuất liên tục dữ liệu của bạn từ các hệ thống khác.
* Để lưu trữ các luồng sự kiện lâu dài và đáng tin cậy bao lâu bạn muốn.
* Để xử lý các luồng sự kiện khi chúng xảy ra hoặc hồi cứu.

Và tất cả chức năng này được cung cấp theo cách phân tán, có khả năng mở rộng cao, đàn hồi, chịu lỗi và an toàn. Kafka có thể được triển khai trên phần cứng, máy ảo và vùng chứa, tại chỗ cũng như trên đám mây. Bạn có thể chọn giữa việc tự quản lý môi trường Kafka của mình và sử dụng các dịch vụ được quản lý hoàn toàn do nhiều nhà cung cấp cung cấp.

## Sự kiện trực tuyến

Phát trực tiếp sự kiện (Event Streaming) là tương đương kỹ thuật số của hệ thống thần kinh trung ương của cơ thể con người. Đó là nền tảng công nghệ cho thế giới 'luôn hoạt động', nơi các doanh nghiệp ngày càng được xác định bằng phần mềm và tự động hóa, và nơi người sử dụng phần mềm nhiều phần mềm hơn.

Nói về mặt kỹ thuật, phát trực tuyến sự kiện là hoạt động thu thập dữ liệu trong thời gian thực từ các nguồn sự kiện như cơ sở dữ liệu, cảm biến, thiết bị di động, dịch vụ đám mây và ứng dụng phần mềm dưới dạng luồng sự kiện; lưu trữ các luồng sự kiện này một cách lâu dài để truy xuất sau này; thao tác, xử lý và phản ứng với các luồng sự kiện trong thời gian thực cũng như hồi cứu; và định tuyến các luồng sự kiện tới các công nghệ đích khác nhau nếu cần. Do đó, phát trực tuyến sự kiện đảm bảo một luồng liên tục và diễn giải dữ liệu để thông tin phù hợp ở đúng nơi, vào đúng thời điểm.

Phát trực tuyến sự kiện (Event Streaming) được áp dụng cho nhiều trường hợp sử dụng khác nhau trên nhiều ngành nghề và tổ chức. Những ví dụ thực tiễn của nó bao gồm:

* Để xử lý các khoản thanh toán và giao dịch tài chính trong thời gian thực, chẳng hạn như tại các sàn giao dịch chứng khoán, ngân hàng và bảo hiểm.
* Để theo dõi và giám sát ô tô, xe tải, đội xe và lô hàng trong thời gian thực, chẳng hạn như trong lĩnh vực hậu cần và công nghiệp ô tô.
* Để liên tục nắm bắt và phân tích dữ liệu cảm biến từ các thiết bị IoT hoặc các thiết bị khác, chẳng hạn như trong các nhà máy và công viên gió.
* Để thu thập và phản ứng ngay lập tức với các tương tác và đơn đặt hàng của khách hàng, chẳng hạn như trong ngành bán lẻ, khách sạn và du lịch cũng như các ứng dụng di động.
* Theo dõi bệnh nhân trong quá trình chăm sóc tại bệnh viện và dự đoán những thay đổi của tình trạng bệnh để đảm bảo điều trị kịp thời trong trường hợp khẩn cấp.
* Để kết nối, lưu trữ và cung cấp dữ liệu có sẵn do các bộ phận khác nhau của công ty tạo ra.
* Để làm nền tảng cho nền tảng dữ liệu, kiến trúc hướng sự kiện và dịch vụ vi mô.

## Cách thức hoạt động của Apache Kafka

Kafka là một hệ thống phân tán bao gồm các máy chủ và máy khách giao tiếp thông qua giao thức mạng TCP hiệu suất cao. Nó có thể được triển khai trên phần cứng kim loại thuần, máy ảo và vùng chứa trong môi trường tại chỗ cũng như trên đám mây.

Máy chủ (Server): Kafka được chạy dưới dạng một cụm gồm một hoặc nhiều máy chủ có thể mở rộng nhiều trung tâm dữ liệu hoặc vùng đám mây. Một số máy chủ này tạo thành lớp lưu trữ, được gọi là các nhà môi giới. Các máy chủ khác chạy Kafka Connect để liên tục nhập và xuất dữ liệu dưới dạng các luồng sự kiện để tích hợp Kafka với các hệ thống hiện có của bạn như cơ sở dữ liệu quan hệ cũng như các cụm Kafka khác. Để cho phép bạn triển khai các trường hợp sử dụng quan trọng đối với nhiệm vụ, cụm Kafka có khả năng mở rộng cao và khả năng chịu lỗi: nếu bất kỳ máy chủ nào của nó bị lỗi, các máy chủ khác sẽ tiếp quản công việc của họ để đảm bảo hoạt động liên tục mà không bị mất dữ liệu.

Máy khách (Client): Cho phép viết các ứng dụng phân tán và dịch vụ có chức năng đọc, ghi và xử lý các luồng sự kiện song song, trên quy mô lớn và theo cách có khả năng chịu lỗi ngay cả trong trường hợp có sự cố mạng hoặc lỗi máy. Kafka đóng gói một số ứng dụng khách như vậy, được tăng cường bởi hàng chục khách hàng do cộng đồng Kafka cung cấp: các ứng dụng khách có sẵn cho Java và Scala bao gồm thư viện Kafka Streams cấp cao hơn, cho Go, Python, C / C ++ và nhiều chương trình khác ngôn ngữ cũng như API REST.

## Các khái niệm và thuật ngữ chính

Một **sự kiện (event)** ghi lại thực tế là "điều gì đó đã xảy ra" trên thế giới hoặc trong doanh nghiệp của bạn. Nó còn được gọi là bản ghi hoặc tin nhắn trong tài liệu. Khi bạn đọc hoặc ghi dữ liệu vào Kafka, bạn thực hiện việc này dưới dạng sự kiện. Về mặt khái niệm, một sự kiện có khóa, giá trị, dấu thời gian và tiêu đề siêu dữ liệu tùy chọn. Đây là một sự kiện ví dụ:

Sự kiện: "Alice"

Nội dung sự kiện: "Đã thanh toán 200 đô la cho Bob"

Dấu thời gian của sự kiện: "Ngày 25 tháng 6 năm 2020 lúc 2:06 chiều"

**Producers** là những ứng dụng khách công bố (viết) các sự kiện cho Kafka và **consumers** là những người đăng ký (đọc và xử lý) các sự kiện này. Ở Kafka, **producers** và **consumers** hoàn toàn tách biệt và không thể hiểu nhau, đó là yếu tố thiết kế quan trọng để đạt được khả năng mở rộng cao như Kafka được biết đến. Ví dụ, **producers** không bao giờ cần phải đợi Consumers. Kafka cung cấp nhiều đảm bảo khác nhau như khả năng xử lý các sự kiện chính xác một lần.

Các sự kiện được sắp xếp và lưu trữ lâu dài trong các **topics**. Rất đơn giản, một chủ đề tương tự như một thư mục trong hệ thống tệp và các sự kiện là các tệp trong thư mục đó. Tên chủ đề mẫu có thể là "thanh toán". Các chủ đề ở Kafka luôn có nhiều Producers và nhiều Consumers: một chủ đề có thể không có, một hoặc nhiều Producers viết sự kiện cho chủ đề đó, cũng như không, một hoặc nhiều Consumers đăng ký các sự kiện này. Các sự kiện trong một chủ đề có thể được đọc thường xuyên nếu cần — không giống như các hệ thống nhắn tin truyền thống, các sự kiện không bị xóa sau khi sử dụng. Thay vào đó, bạn xác định thời gian Kafka sẽ lưu giữ các sự kiện của bạn thông qua cài đặt cấu hình theo chủ đề, sau đó các sự kiện cũ sẽ bị loại bỏ. Hiệu suất của Kafka không đổi về kích thước dữ liệu, vì vậy việc lưu trữ dữ liệu trong thời gian dài là hoàn toàn ổn.

Các chủ đề (topics) được phân vùng, có nghĩa là một chủ đề được trải rộng trên một số "nhóm" nằm trên các nhà môi giới Kafka khác nhau. Vị trí phân tán dữ liệu của bạn rất quan trọng đối với khả năng mở rộng vì nó cho phép các ứng dụng khách đọc và ghi dữ liệu từ / đến nhiều nhà môi giới cùng một lúc. Khi một sự kiện mới được xuất bản cho một chủ đề, nó thực sự được nối vào một trong các phân vùng của chủ đề. Các sự kiện có cùng một khóa sự kiện (ví dụ: khách hàng hoặc ID xe) được ghi vào cùng một phân vùng và Kafka đảm bảo rằng bất kỳ Consumers nào của một phân vùng chủ đề nhất định sẽ luôn đọc các sự kiện của phân vùng đó theo đúng thứ tự như khi chúng được viết.

Diagram

Description automatically generated

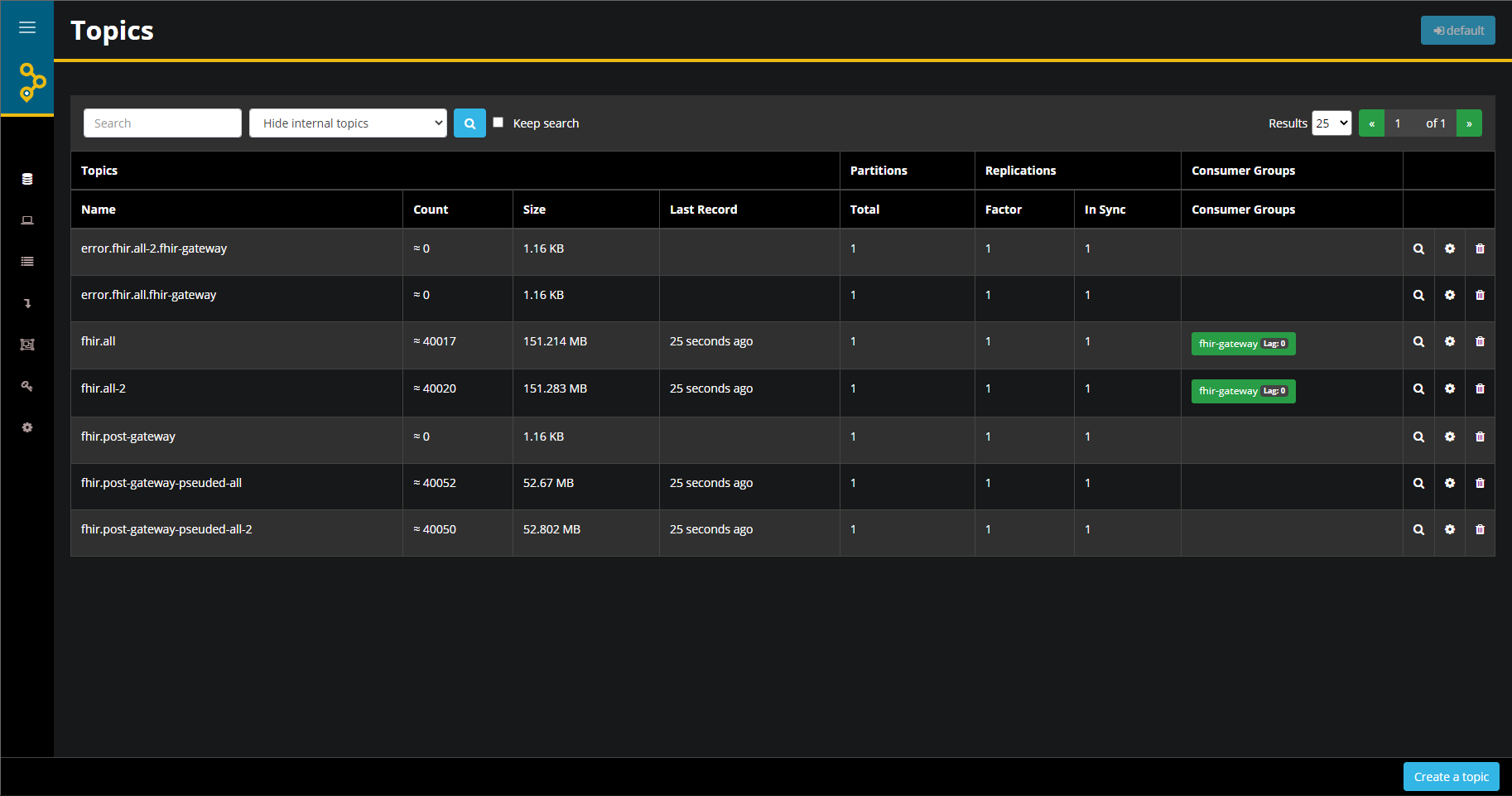
## API Kafka

Ngoài công cụ dòng lệnh cho các tác vụ quản lý và điều hành, Kafka có năm API cốt lõi cho Java và Scala:

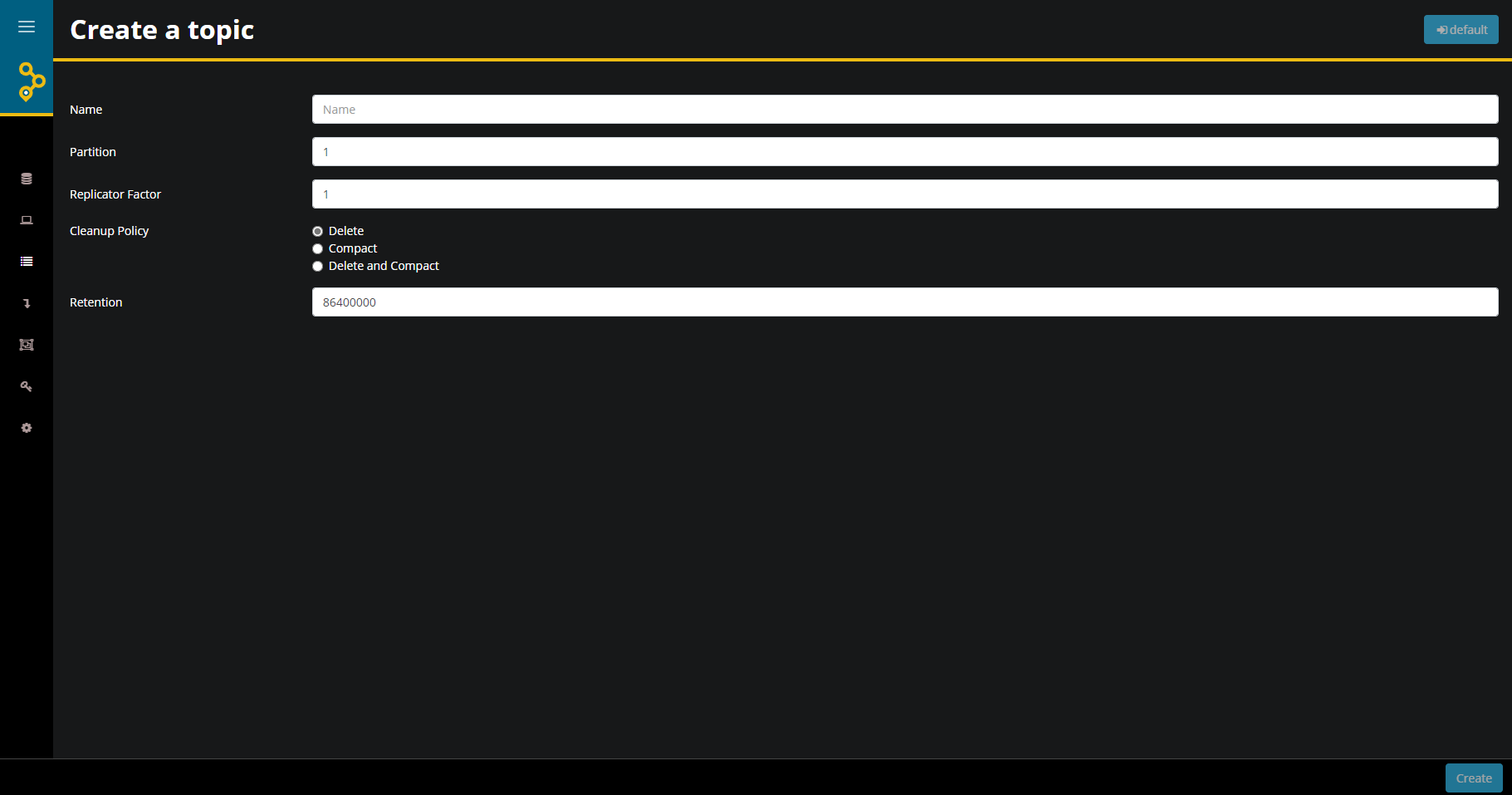
* Admin API để quản lý và kiểm tra các chủ đề, nhà môi giới và các đối tượng Kafka khác.
* Producer API để xuất bản (viết) một luồng sự kiện cho một hoặc nhiều chủ đề Kafka.
* Consumer API để đăng ký (đọc) một hoặc nhiều chủ đề và xử lý luồng sự kiện được tạo ra cho họ.
* Kafka Streams API để triển khai các ứng dụng xử lý luồng và dịch vụ vi mô. Nó cung cấp các chức năng cấp cao hơn để xử lý các luồng sự kiện, bao gồm các phép biến đổi, các hoạt động trạng thái như tổng hợp và kết hợp, tạo cửa sổ, xử lý dựa trên thời gian sự kiện và hơn thế nữa. Đầu vào được đọc từ một hoặc nhiều chủ đề để tạo đầu ra cho một hoặc nhiều chủ đề, chuyển đổi hiệu quả luồng đầu vào thành luồng đầu ra.
* Kafka Connect API để xây dựng và chạy các trình kết nối nhập / xuất dữ liệu có thể sử dụng lại sử dụng (đọc) hoặc tạo (ghi) các luồng sự kiện từ và đến các hệ thống và ứng dụng bên ngoài để chúng có thể tích hợp với Kafka. Ví dụ: một trình kết nối với cơ sở dữ liệu quan hệ như PostgreSQL có thể nắm bắt mọi thay đổi đối với một tập hợp các bảng.

## Trang quản lý và theo dõi dữ liệu AKHQ.io

AKHQ là một Kafka GUI (Graphical User Interface), hiểu nôm na là một giao diện đồ họa người dùng cho hệ thống Kafka. AKHQ.io cho phép các nhà phát triển hiểu rõ hơn về các cụm Kafka của họ, xem các chủ đề (topics), duyệt dữ liệu bên trong các chủ đề, cũng như xem các nhóm Consumers và độ trễ của họ. Công nghệ này cũng có thể quản lý sổ đăng ký lược đồ và cho phép người quản lý hệ thống thu thập các tạo tác trực quan hóa để hỗ trợ các gói.



Chúng ta có thể tạo thêm nhiều “topics” đúng với nhu cầu của Producers và Consumers.



# Hệ thống giám sát và khắc phục sự cố JAEGER

Jaeger là một phần mềm mã nguồn mở, theo dõi phân tán end-to-end, có chức năng chính là giám sát và khắc phục sự cố giao dịch trong các hệ thống phân tán phức tạp.

## Tổng quan về hệ thống Jaeger

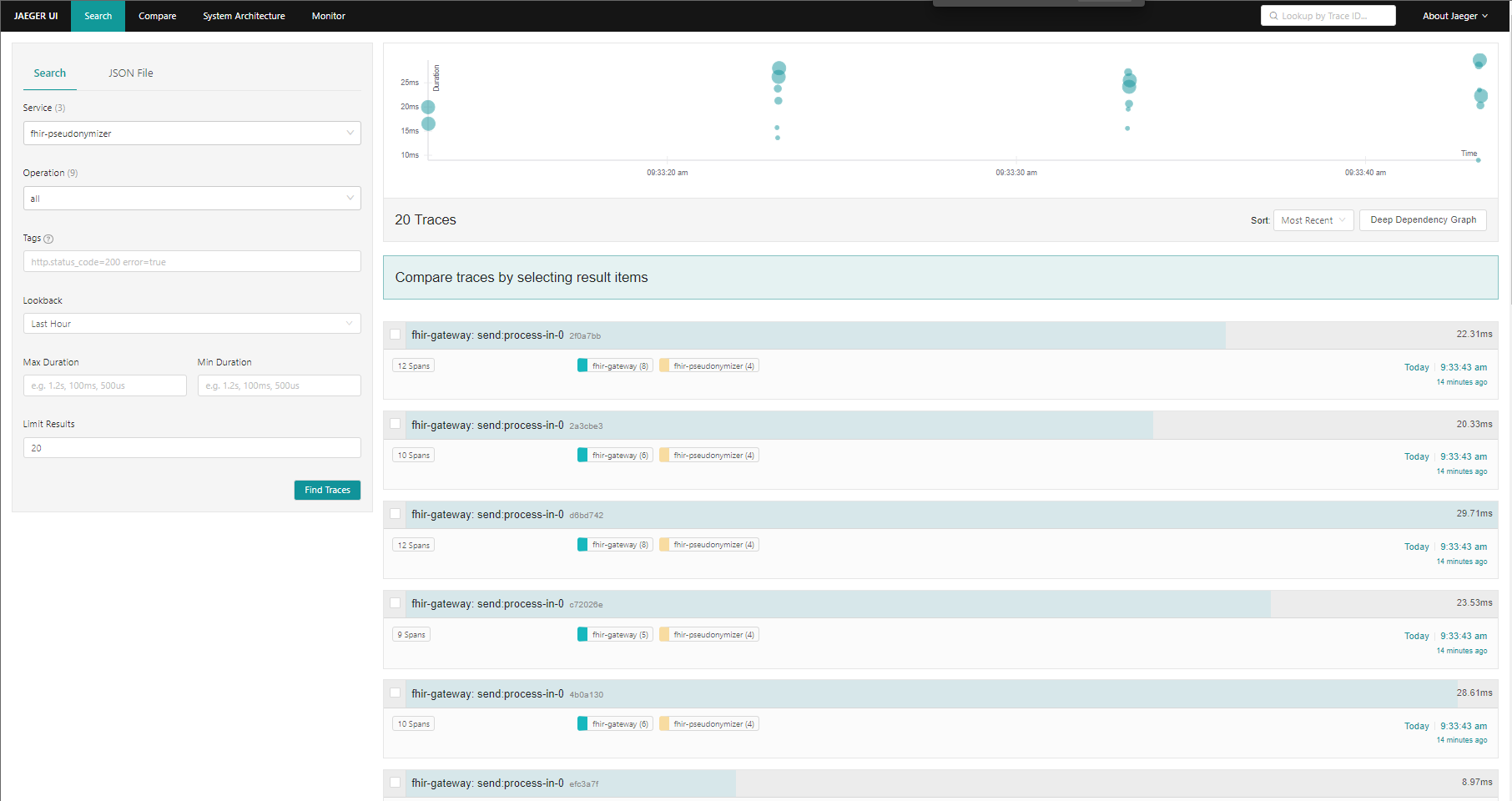
Jaeger, lấy cảm hứng từ Dapper và OpenZipkin, là một hệ thống theo dõi phân tán được phát hành dưới dạng mã nguồn mở bởi Uber Technologies. Nó được sử dụng để giám sát và khắc phục sự cố các hệ thống phân tán dựa trên dịch vụ vi mô, bao gồm:

* Truyền bối cảnh phân tán
* Giám sát giao dịch phân tán
* Phân tích nguyên nhân gốc rễ
* Phân tích sự phụ thuộc của dịch vụ
* Tối ưu hóa hiệu suất / độ trễ

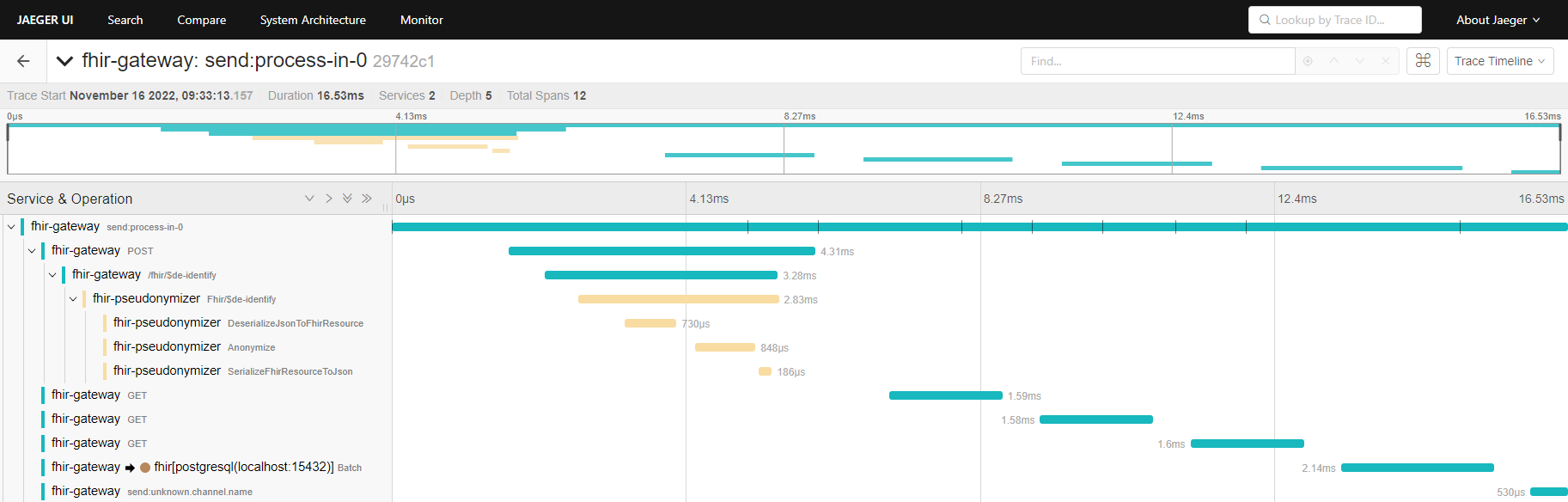
Jaeger có các tính năng nổi bật như sau:

* Mô hình dữ liệu lấy cảm hứng từ OpenTracing.
* Sử dụng lấy mẫu trả trước nhất quán với từng xác suất dịch vụ / điểm cuối riêng lẻ.
* Nhiều bộ nhớ phụ được tích hợp sẵn: Cassandra, Elasticsearch.
* Chương trình phụ trợ lưu trữ bên ngoài được cộng đồng hỗ trợ thông qua plugin gRPC: PostgreSQL với Promscale, ClickHouse.
* Đồ thị cấu trúc liên kết hệ thống.
* Lấy mẫu thích ứng.
* Quy trình xử lý dữ liệu sau thu thập (sắp ra mắt).
* Giám sát Hiệu suất Dịch vụ (Service Performance Monitoring – SPM).

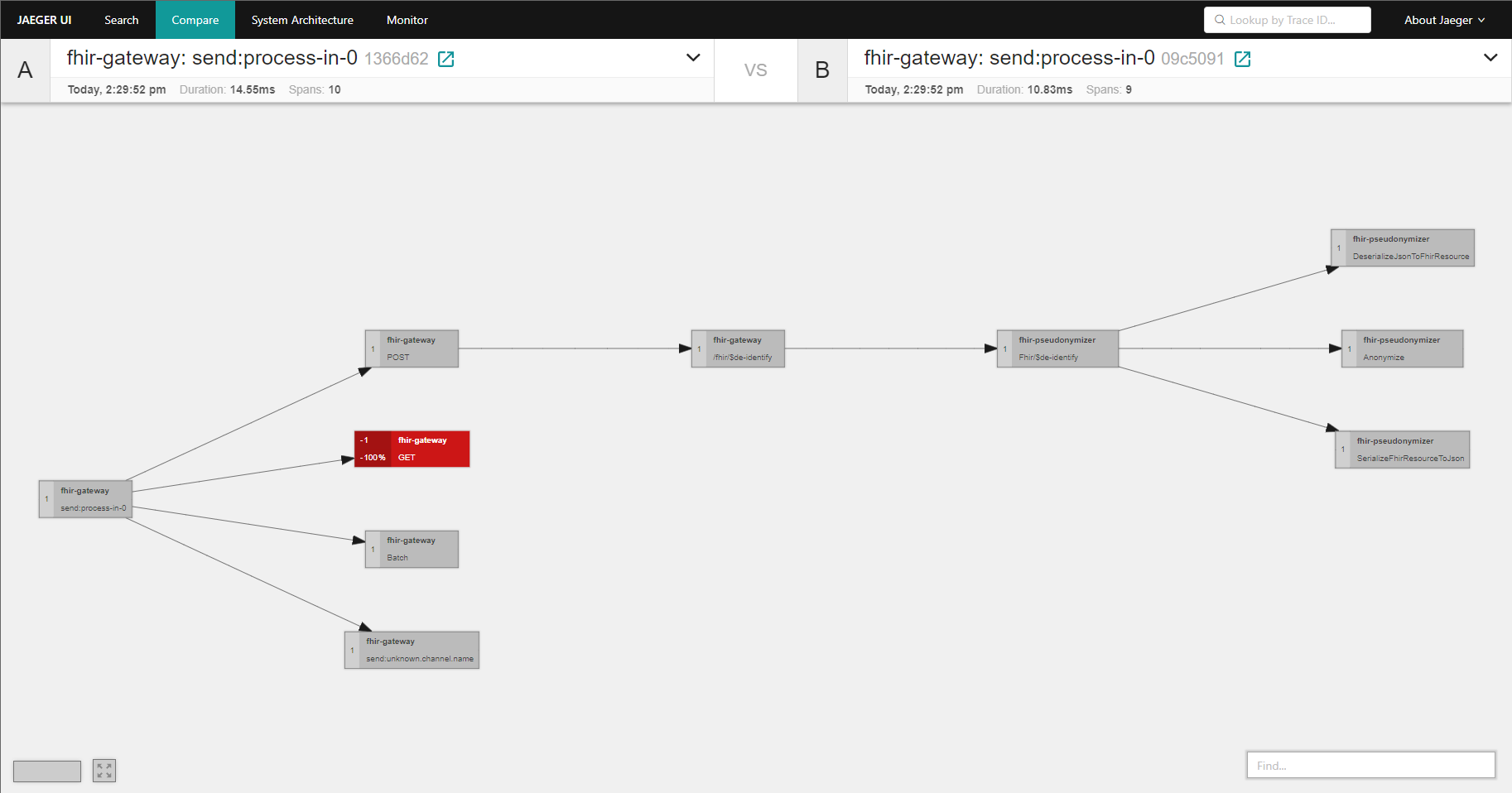
Chế độ xem dấu vết (Traces View)



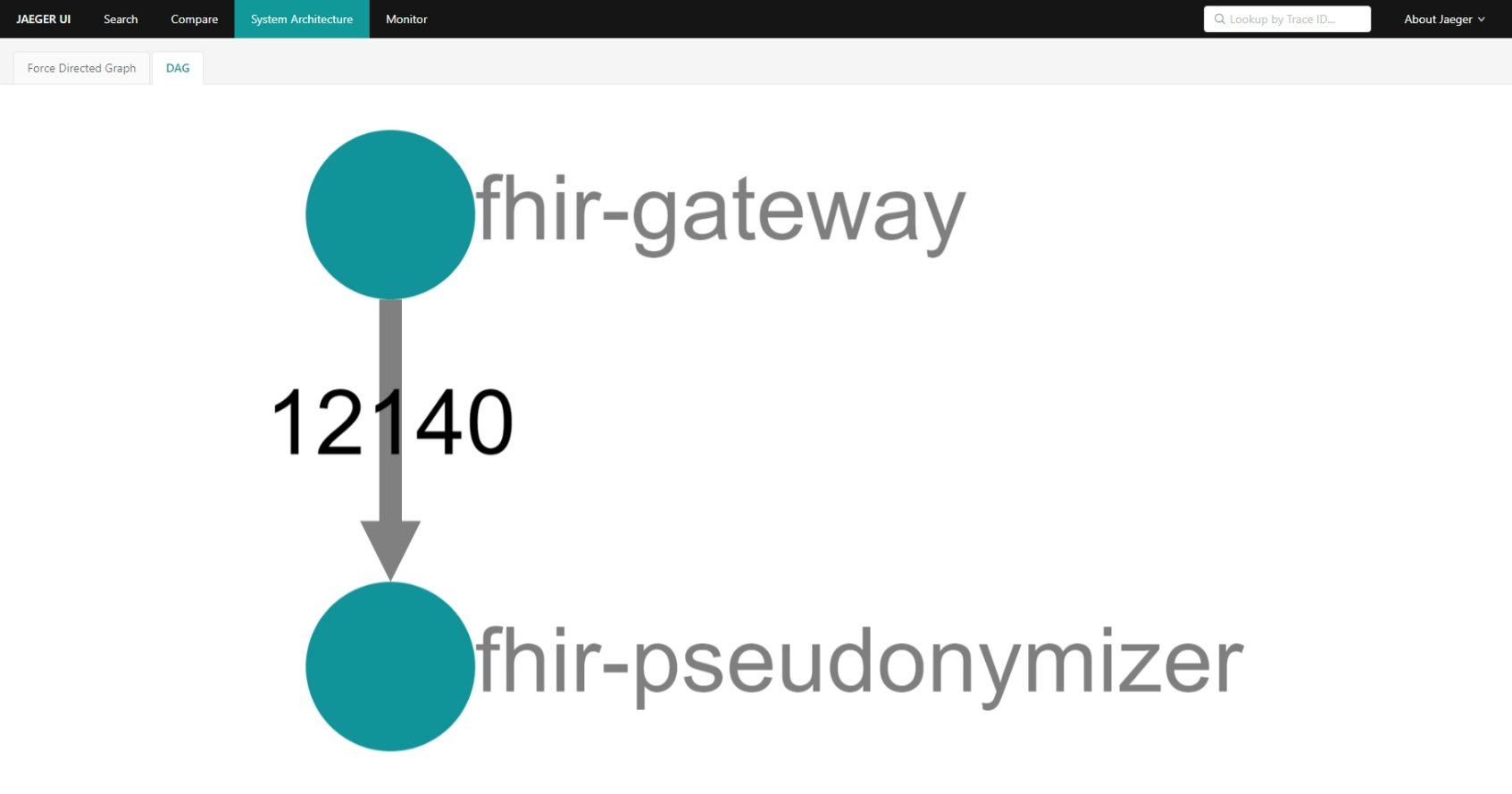
Chế độ xem dấu vết chi tiết (Trace Detail View)



Chế độ so sánh các dấu vết (Compare traces)



Chế độ hiển trị kiến trúc của hệ thống (System Architecture) dưới dạng đồ thị Acyclic có hướng (Directed Acyclic Graph – DAG)



## Cấu trúc của hệ thống Jaeger

## Cấu hình cho hệ thống Jaeger

# Dịch vụ ký danh FHIR Pseudonymizer

## Graphical user interface, text, application Description automatically generated