**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA HỆ THỐNG THÔNG TIN**





**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN CƠ SỞ DỮ LIỆU PHÂN TÁN**

**Đề tài: ELASTICSEARCH**

GVHD: **Nguyễn Thị Kim Yến**

Nhóm sinh viên thực hiện:

1. Nguyễn Quang Thái MSSV: 19522189
2. Trần Nhật Nam MSSV: 19521872
3. Nguyễn Hải Đăng MSSV: 19521315

🙡🙢 Tp. Hồ Chí Minh, 24/11/2021 🙠🙣

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

*……., ngày……...tháng……năm 2021*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên****)***

**MỤC LỤC**

[**Chương 1:**](#_1fob9te) [**Giới thiệu**](#_3znysh7) [**6**](#_1fob9te)

[**1.1**](#_3znysh7)  **Tổng quan** 6

[1.1.1](#_2et92p0) Khái niệm 6

1.1.2 Lịch sử hình thành 6

1.1.3 Một số khái niệm quan trọng 6

1.1.4 Mô hình lưu trữ 6

1.1.5 Cơ chế phân tán 7

[1.1.6](#_tyjcwt) Tóm tắt 7

[**1.2**](#_3dy6vkm) **Một số đặc trưng của ElasticSearch** 10

[1.2.1 Khả năng mở rộng](#_1t3h5sf) 10

[1.2.2 Tốc độ](#_4d34og8) 10

[1.2.3 Dễ dàng sử dụng API](#_2s8eyo1) 10

[1.2.4 Đa ngôn ngữ](#_17dp8vu) 10

[1.2.5](#_3rdcrjn) Theo định hướng tài liệu (Document-Oriented) 10

1.2.6 Tự động hoàn thành (autocompletion) 10

1.2.7 Schema-Free 10

[**Chương 2: Hướng dẫn cài đặt 11**](#_1y810tw)

**2**[**.1**](#_4i7ojhp) **Quy trình cài đặt Elasticsearch và công cụ Kibana** 11

2[.1.1 Q](#_2xcytpi)uy trình cài đặt Elasticsearch 11

2[.1.2](#_1ci93xb) Quy trình cài đặt Kibana 14

[**Chương 3: Ứng dụng/ Kết quả thực nghiệm/So sánh-Đánh giá 30**](#_3whwml4)

3[.1](#_2bn6wsx)  Kết quả thực nghiệm phân tán dữ liệu 30

3[.2](#_qsh70q)  So sánh – Đánh giá 38

3[.3](#_3as4poj)  Ứng dụng 45

[**Chương 4: Kết luận 51**](#_1pxezwc)

4[.1](#_49x2ik5)  Ưu điểm 51

4[.2](#_2p2csry)  Khuyết điểm 51

4[.3](#_147n2zr)  Hướng phát triển 52

[**TÀI LIỆU THAM KHẢO 53**](#_23ckvvd)

-----Hết-----

**BẢNG PHÂN CÔNG, ĐÁNH GIÁ THÀNH VIÊN:**

| Nguyễn Quang Thái | 19522189 |  |
| --- | --- | --- |
| Trần Nhật Nam | 19521872 |  |
| Nguyễn Hải Đăng | 19521315 |  |

# Chương 1: Giới thiệu

## Tổng quan



*Hình 1.1 Logo của Elasticsearch*

- Trong thế giới công nghệ ngày nay, một lượng lớn dữ liệu định xấp xỉ 2,5 nghìn tỷ byte được tạo ra mỗi ngày. Dữ liệu này chủ yếu đến từ các nguồn khác nhau: từ các trang social media sites, trang video sharing, cho đến các tổ chức quy mô trung bình đến khổng lồ. Toàn bộ các dữ liệu này được gọi là data ocean, hoặc gọi tên theo cách chuyên môn hơn, đó chính là Big Data. Phần lớn khi các dữ liệu này đứng riêng một mình mà không hề có sự thống kê hay phân tích nào, chúng sẽ trở nên vô nghĩa, đồng thời không có chút giá trị nghiên cứu nào cả. Để làm cho các dữ liệu này trở nên ý nghĩa, bạn cần có các công cụ phục vụ cho việc phân tích. Có rất nhiều công cụ phân tích sẵn có trên thị trường giúp bạn có thể khám phá, ghi lại, truy cập, phân tích và xử lý các dữ liệu phi cấu trúc này. Trong số tất cả có thể nhắc tới Elasticsearch là một trong những công cụ nổi bật nhất.

## Khái niệm

**- Elasticsearch** là một công cụ tìm kiếm dựa trên phần mềm **Lucene**. Nó cung cấp một bộ máy tìm kiếm dạng phân tán, có đầy đủ công cụ với một giao diện web **HTTP** có hỗ trợ dữ liệu **JSON Elasticsearch** được sử dụng bởi nhiều công ty lớn như **Facebook**, **GitHub**, **Wikipedia**,...

## Lịch sử hình thành

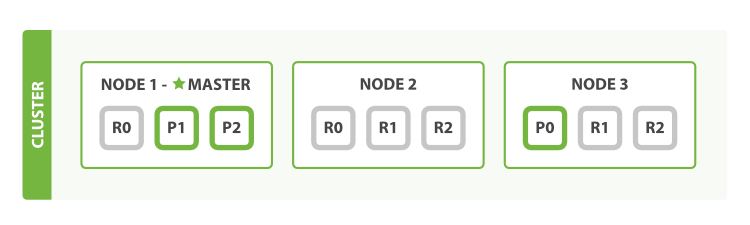
**- Shay Banon** tạo ra phần mềm tiền thân của **Elasticsearch**, được gọi là **Compass**, trong năm **2004**. Trong khi suy nghĩ về phiên bản thứ ba của **Compass**, ông nhận ra rằng cần thiết phải viết lại phần lớn **Compass** để tạo ra một giải pháp tìm kiếm có khả năng mở rộng. Vì vậy, ông đã tạo ra một giải pháp được xây dựng trên nguyên lý phân tán và sử dụng giao thức phổ biến là **HTTP**, thích hợp cho các ngôn ngữ lập trình khác. **Shay Banon** phát hành phiên bản đầu tiên của **Elasticsearch** vào tháng hai, năm **2010**. Nó có thể tích hợp được với các ứng dụng sử dụng các ngôn ngữ lập trình phổ biến như **Java**, **PHP**, **JavaScript**, **.NET**, **Ruby**, **Python**,...

**Năm 2012,** công ty **Elasticsearch BV** được thành lập để cung cấp dịch vụ thương mại và sản phẩm xung quanh phần mềm **Elasticsearch** và các phần mềm liên quan. Trong tháng 6 năm **2014**, công ty thông báo tăng **70 triệu USD** trong một vòng tài trợ Loạt C, chỉ 18 tháng sau khi thành lập công ty. Vòng tài trợ đã được dẫn dắt bởi **New Enterprise Associates (NEA)**. Các nhà tài trợ bao gồm **Benchmark Capital** và **Index Ventures**. Vòng này mang đến tổng số kinh phí hơn **104 triệu đô**. Vào tháng năm **2015**, công ty **Elasticsearch** đổi tên thành **Elastic**.

## Một số khái niệm quan trọng

* **Node**: một server tham gia tạo thành **cluster**, nó có vai trò lưu dữ liệu, đánh chỉ mục và cung cấp khả năng tìm kiếm. Một node cũng cần xác định một cái tên, mặc định một tên duy nhất (**UUID**) được sinh ra cho nó khi nó chạy trừ khi bạn gán cho nó một cái tên cụ thể. Mặc định mỗi node cần thiết lập nối vào một **cluster** đã đặt tên (mặc định là **elasticsearch**), có nghĩa là khi **node** hoạt động nó sẽ tìm đến **cluster** với tên chỉ định để nối vào. Nếu như bạn chạy một **server** (**node**) mà mạng của bạn chưa có một **cluster** nào với tên chỉ ra, thì node này sẽ tự động là một **cluster** với tên mặc định **elasticsearch**.
* **Cluster**: tập hợp của nhiều **Node**- **Elasticsearch** kết nối với nhau để lưu giữ dữ liệu và cung cấp chức năng đánh chỉ mục, tìm kiếm dữ liệu trên các **server** đó. Một **cluster** xác định bởi tên duy nhất (mặc định tên là **elasticsearch**). Dù bạn chỉ tạo ra hệ thống với **1 server** (**node**) thì vẫn có một **cluster**, sau này có thể nối nhiều **server phân tán** vào **cluster** để mở rộng khả năng của hệ thống mà ở đó một **node** (**server**) có thể có một chức năng riêng như (**master node**, **data**, **client**...).
* **Index:** Tập hợp các tài liệu (**document**), những tài liệu này có một số tên thuộc tính tính chất chung. Thường mỗi **index** là một loại dữ liệu nào đó của bạn ví dụ như **index** chứa các sản phẩm, **index** chứa các đơn hàng, **index** chứa các bài viết ... Mỗi **index** được đặt một cái tên (nhớ là chữ thường), tên này dùng để thi hành các chức năng như đánh chỉ mục, tìm kiếm, cập nhật ... cho các dữ liệu (**document**) trong nó. Mỗi **node** có thể tạo nhiều **index**.
* **Document:** Đơn vị thông tin lưu trong **index** (giống như dòng lưu trong bảng), đơn vị thông tin này biểu diễn bằng **JSON** (**JavaScript Object Notation**), qua đó nó lưu loại dữ liệu của bạn như sản phẩm, bài viết, đơn hàng ... Một **index** lưu bao nhiêu **Document** cũng được.
* **Shard:** Để giải quyết các vấn đề hiệu năng khi lưu trữ dữ liệu lớn bị giới hạn bởi phần cứng. **Elasticsearch** cung cấp khả năng chia một **index** ra thành từng mảnh nhỏ hơn - mỗi mảnh nhỏ đó gọi là **shard**. Khi tạo ra **index**, bạn có thể chia nó ra thành bao nhiêu **shard** tùy bạn. Một **shard** đó vẫn có đầy đủ chức năng như **index** nhưng độc lập với **index**, vào có thể lưu ở các **node** khác nhau. **Shard** nó giúp giải quyết vấn đề hiệu năng tốc độ, tìm kiếm song song trên nhiều **node** phân tán. Ngoài ra bạn cũng có khái niệm **replica** đó là một bản **backup**, **copy** của **shard** để **Elasticsearch** có thể phục hồi nếu một **shard** nào đó bị chết.
* **Primary Shard:** Nếu hình dung quan hệ **master-slave** như **MySQL** thì **primary shard** là **master**. Dữ liệu được lưu tại **1** **primary shard**, được đánh **index** ở đây trước khi chuyển đến **replica shard**. Mặc định của **Elasticsearch** là **5 primary shard** cho một **index** (**1 index** trong **Elasticsearch** tương đương với **1** **database** trong **MySQL**). Một khi đã khởi tạo **index** thì bạn không thể thay đổi số lượng **primary shard** cho nó.
* **Replica Shard:** Một **primary shard** có thể không hoặc có **replica shard**. Mặc định **Elasticsearch** là 1 **replica shard** trên một **primary shard**. Vai trò của **replica shard** cũng giống như **slave** trong **MySQL**, đảm bảo khi **primary shard** có sự cố thì dữ liệu vẫn toàn vẹn và thay thế được **primary shard**, đồng thời tăng tốc độ đọc vì **replica shard** có thể là nhiều hơn 1. Trong quá trình hoạt động, bạn được thay thế số lượng **replica shard** một cách thoải mái (không như **primary shard**).

Dưới đây là một mô hình đơn giản cho kiến trúc: **cluster-node-shard** của **Elasticsearch**.



Nhìn vào hình trên chúng ta có thể thấy, dữ liệu được lưu trữ ở **cluster** với 3 nodes trong đó node 1 là **master**. Có **3 primary shards**, 2 trong số đó được đặt ở **Node 1**, còn lại ở **Node 3**. **Mỗi primary shard** có **2 replica shard** (ví dụ **primary shard P0** ở **Node 3** thì có **replica shard R0** ở **Node 1** và một shard nữa ở **Node 2**). Việc sắp đặt vị trí **primary shard** là ngẫu nhiên, còn các **replica shard** luôn được đảm bảo là nó không nằm cùng node với **primary shard** (Tại sao? Tưởng tượng node ngưng hoạt động mà toàn bộ shard ở node đó thì sẽ mất hết sạch dữ liệu). Thêm nữa là không bắt buộc **primary shard** đều nằm ở **node master**, vì việc phân tán các **primary shard** giúp phân tán công đoạn ghi dữ liệu, giúp giảm tải cho một **node**. Việc lựa chọn node cho các thao tác đọc được thực hiện bởi thuật toán **Round-robin**.

## Mô hình lưu trữ

**Elasticsearch** sử dụng **index** để lưu trữ và xử lý dữ liệu, nó tương tự một cơ sở dữ liệu thông thường. Dữ liệu sẽ được lưu trữ thành các **document** dưới dạng **JSON**. Mỗi một **index** có nhiều **type** (tương tự như bảng trong cơ sở dữ liệu), điều này sẽ giúp chúng ta phân tách các loại dữ liệu khác nhau khi lưu vào cùng một **index**.

Mọi tài liệu được lưu vào trong cùng một **type** của một **index** sẽ có cấu trúc các trường và kiểu dữ liệu giống hệt nhau (giống như mỗi bảng có một schema trong các hệ cơ sở dữ liệu quan hệ)

Ngoài ra, **Elasticsearch** cònsử dụng **Lucene** của **Apache** - một thử viên full text search viết bằng **Java**. Về bản chất, nó sử dụng một cấu trúc dữ liệu gọi là **inverted index (chỉ mục đảo ngược)** để có thể thực hiện tìm kiếm với hiệu suất cao.

Tài liệu chính đơn vị cơ sở để quản lý dữ liệu trong **Elasticsearch** và **inverted index** được tạo ra bằng việc **tokenize** các khái niệm trong tài liệu. Bằng việc sử dụng kỹ thuật **inverted index**, một bảng chỉ mục các khái niệm và danh sách tài liệu liên quan đến khái niệm đó sẽ được tạo ra.

Nó khá tương đồng với chỉ mục của một cuốn sách, nơi hiển thị một danh sách các khái niệm được sử dụng cùng với số trang mà nó xuất hiện. Trong **Elasticsearch**, khi nói rằng một tài liệu được **index**, thì chúng ta hiểu rằng, **inverted index** của tài liệu đó đã được tạo ra.

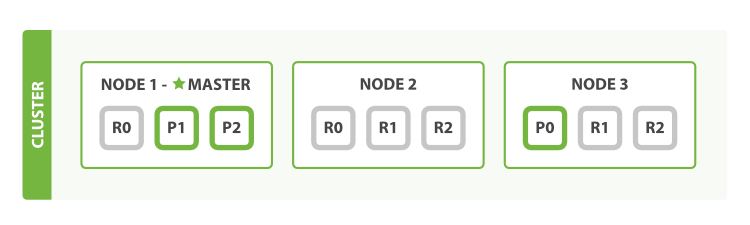
Ví dụ, khi chúng ta cần tìm từ khoá "**Việt Nam**", Elasticsearch sẽ thực hiện tìm kiếm trên **inverted index** (rất nhanh do các từ được sắp xếp), tìm được từ khoá "**Việt Nam**", sau đó trả về **ID** của các tài liệu tương ứng. Để có thể tìm kiếm nâng cao hơn (ví dụ, tìm với từ khoá "**Viet Nam**"), thì quá trình phân tích và **index** tài liệu là rất quan trọng.

Ngoài ra, quá trình tìm kiếm thực tế còn phức tạp hơn nữa, khi mà **Elasticsearch** còn phải đánh giá mức độ tương qua giữa những từ trong **index** với từ khóa cần tìm.

## Cơ chế phân tán

* **Lưu trữ dữ liệu**

Giả sử chúng ta đã có một sơ đồ cluster như hình dưới đây, với vị trí **primary shard**, **replica shard** đã cố định. Vậy với một bản ghi (**document**) mới, chúng ta sẽ lưu nó ở **shard** nào ?

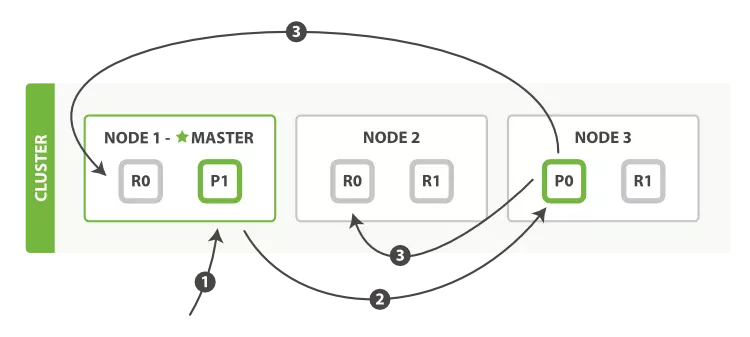


Elasticsearch sử dụng công thức sau để chọn shard cho việc lưu trữ:

**shard = hash(routing) % number\_of\_primary\_shards**

**hash** là một hàm tính toán cố định của **Elasticsearch**, **routing** là 1 đoạn text duy nhất theo **document**, thường là **\_id** của **document** đó. **number\_of\_primary\_shards** là số lượng **primary shard** của **cluster**. Giá trị shard này là đi kèm với **document**, nó được dùng để xác định **shard** nào sẽ lưu **document** và cũng dùng để cho **routing**, vì bạn sẽ tìm **document** theo **id** của nó. Đó là lý do chúng ta không thể thay đổi số lượng **primary shard**, nếu không công thức trên sẽ không cho kết quả giống như lúc ban đầu.

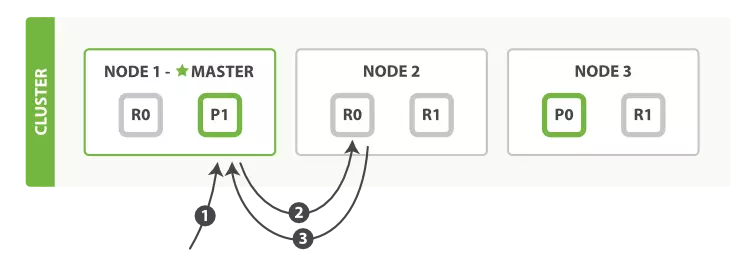
* *Quá trình lưu dữ liệu*



Hình trên đây mô phỏng quá trình lưu trữ, xóa dữ liệu của Elasticsearch. Có thể chia thành 3 bước.

1. **Request** được gửi đến **node master (Node 1)**. Tại đây thực hiện tính toán với công thức ở trên để tìm ra **primary shard** của **document** sẽ là **0**.
2. Sau khi xác định được **primary shard** là **0**, **request** sẽ được gửi đến **node 3**, nơi chứa **P0**.
3. **Node 3** thực hiện **request** và xử lý dữ liệu. Sau khi thành công, nó gửi tiếp **request** đến các **replica shard** ở **Node 1** và **Node 2** để đảm bảo dữ liệu thống nhất giữa các node.

* *Quá trình lấy dữ liệu*



Hình trên mô phỏng quá trình lấy dữ liệu theo **id** của **document**. Có thể chia thành 3 bước.

1. **Request** được gửi đến **node master (Node 1)**. Tại đây xác định **primary shard** cho **document** sẽ là **0**.
2. Do tất cả node đều lưu dữ liệu, nên **master node** sẽ chọn ra 1 node và lấy dữ liệu ở shard số 0. Việc chọn này giúp giảm tập trung vào một node. Thuật toán **Round-robin** được sử dụng để các shard được chọn khác nhau ở mỗi **request**. Trong trường hợp này **Node 2** được chọn.
3. **Replica 0** ở **Node 2** trả về kết quả cho **master node**.

* **Tìm kiếm dữ liệu**

## *Công đoạn truy vấn*

* *Elastic thực hiện các công việc sau cho công đoạn truy vấn:*

1. **Node** nhận **request** (**node trung gian**) sẽ gửi **broadcast request** đó đến tất cả các **node** khác. Tại mỗi **node** này sẽ chỉ định **shard** thực hiện công việc tìm kiếm theo yêu cầu. **Shard** có thể là **primary** hoặc **replica shard**.
2. Mỗi **shard** sẽ thực hiện công việc tìm kiếm, trả về **id** và **score** của **document**. Trong đó **score** là giá trị dùng để sắp xếp. Danh sách gồm **id**, **score** này là một danh sách đã được sắp xếp, mang tính chất cục bộ.
3. **Node trung gian** sau khi nhận kết quả trả về từ các **node** khác sẽ thực hiện công việc sắp xếp toàn cục tất cả các **document**, dựa theo **id** và **score**. Cuối cùng trả ra kết quả về phía **client**.

* *Công đoạn lấy dữ liệu*
* Sau công đoạn truy vấn, **node trung gian** đã có được danh sách **id** của bản ghi sắp xếp theo **score**. Tiếp theo, nó sẽ thực hiện lấy nội dung bản ghi dựa theo **id** đó. Công đoạn này về cơ bản là giống với **Quá trình lấy dữ liệu** đã được nêu ra ở trên.

## Tóm tắt

* Elasticsearch là một công cụ tìm kiếm - Search Engine.
* Elasticsearch được thiết kế và tạo ra bởi Lucene Apache.
* Elasticsearch thực chất hoạt động như một máy chủ - Web Server với khả năng tìm kiếm cực kỳ nhanh chóng - Near Real Time thông qua giao thức Restful.
* Elasticsearch có khả năng thống kê và phân tích dữ liệu cực kỳ hiệu quả.
* Elasticsearch chạy trên một máy chủ riêng và đồng thời nó giao tiếp thông qua Restful. Điều này làm cho Elasticsearch không bị phụ thuộc vào client viết bằng gì hay hệ thống hiện tại của bạn viết bằng gì. Điều này khiến cho Elasticsearch dễ dàng tích hợp vào hệ thống của bạn, và bạn chỉ cần thực hiện gửi Request http lên là Elasticsearch sẽ trả về cho một kết quả phù hợp nhất.
* Elasticsearch là một hệ thống phân tán và có khả năng mở rộng, khi lắp thêm node cho Elasticsearch là nó sẽ tự động mở rộng.
* Elasticsearch là một mã nguồn mở - Open Source được phát triển bằng Java

# Các đặc trưng của ElasticSearch

* + 1. **Khả năng mở rộng**

Rất dễ mở rộng và đáng tin cậy. Đây là một tính năng rất quan trọng giúp đơn giản hóa các kiến trúc phức tạp và tiết kiệm thời gian trong quá trình thực hiện các dự án.

* + 1. **Tốc độ**

Sử dụng distributed inverted indices để tìm các kết quả phù hợp nhất cho full-text searches giúp việc tìm kiếm từ các tập dữ liệu rất lớn trở nên vô cùng nhanh chóng.

* + 1. **Dễ dàng sử dụng API**

Cung cấp các API RESTful đơn giản và sử dụng các schema-free JSON documents giúp việc indexing, searching, và querying thật dễ dàng.

* + 1. **Đa ngôn ngữ**

Một trong những tính năng khác biệt nhất mà **Elasticsearch** sở hữu đó chính là đa ngôn ngữ. Nó hỗ trợ nhiều tài liệu được viết bằng các ngôn ngữ khác nhau như tiếng Ả Rập, tiếng Braxin, tiếng Trung, tiếng Anh, tiếng Pháp, tiếng Hindi, tiếng Hàn,...

* + 1. **Theo định hướng tài liệu (Document-Oriented)**

Lưu trữ các thực thể phức tạp trong thế giới thực dưới dạng các structured JSON documents và lập chỉ mục tất cả các trường theo mặc định giúp cho dữ liệu luôn có thể tìm kiếm được. Vì không có hàng và cột dữ liệu, bạn có thể thực hiện full-text search phức tạp một rất cách dễ dàng.

* + 1. **Tự động hoàn thành**

Cung cấp chức năng tự động hoàn thành. Bằng cách dự đoán từ bằng cách sử dụng rất ít ký tự, **autocompletion** sẽ tăng tốc độ tương tác giữa con người và máy tính.

* + 1. **Schema-Free**

**Elasticsearch** là schema-free vì nó chấp nhận các tài liệu **JSON**. Nó cố gắng phát hiện cấu trúc dữ liệu, lập chỉ mục dữ liệu, làm cho dữ liệu có thể tìm kiếm được.

# Chương 2: Hướng dẫn cài đặt

## 2.1 Quy trình cài đặt Elasticsearch và công cụ Kibana

### **2.1.1 Quy trình cài đặt Elasticsearch**



*Hình 3.1.1.1 Quy trình cài đặt.*

Bước 1 - Cài đặt phiên bản Java mới nhất hoặc nếu bạn đã cài đặt Java rồi kiểm tra phiên bản của nó bằng cách sử dụng lệnh java–version trong cmd.

Lưu ý: Phiên bản Java từ 7 trở lên.

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Hình 3.1.1.2.*

Bước 2 - Truy cập https://www.elastic.co/fr/downloads/elasticsearch.

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

*Hình 3.1.1.3*

Bước 3 - Nhấp vào Download để nhận tệp zip.

Bước 4 - Khi tệp được tải xuống, giải nén nội dung.

Bước 5 - Đến elasticsearch-7.13.1> bin.

Ảnh có chứa văn bản, ảnh chụp màn hình, đen, màn hình

Mô tả được tạo tự động

*Hình 3.1.1.4*

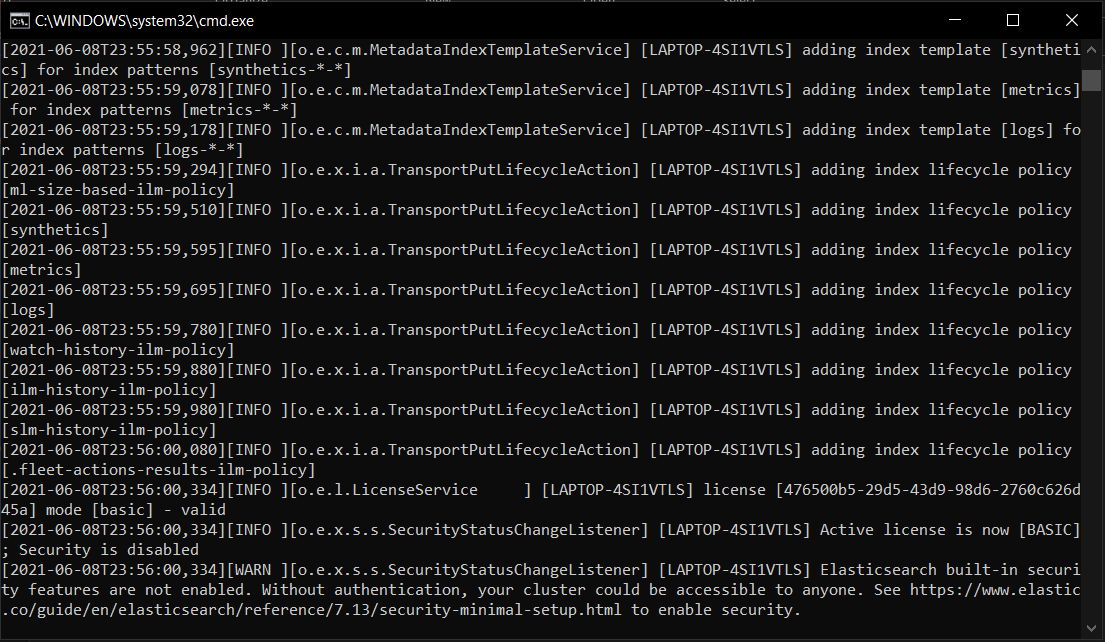
Bước 6 - Bên trong thư mục bin, tìm tập tin elasticsearch và click đúp để khởi động máy chủ Elasticsearch.

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Hình 3.1.1.5*

Bước 7 - Đợi máy chủ khởi động.



*Hình 3.1.1.6*

Bước 8 - Mặc định Elasticsearch lắng nghe các yêu cầu HTTP gửi đến ở cổng 9200, có nghĩa bạn sẽ tương tác với nó ở địa chỉ <http://localhost:9200>. Mở trình duyệt và gõ localhost: 9200 để kiểm tra xem máy chủ có đang chạy hay không.

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Hình 3.1.1.7*

Bước 9 - Nếu bạn thấy thông báo được hiển thị trên trên trình duyệt, có nghĩa là các cài đặt đang thành công.

**2.1.2 Quy trình cài đặt Kibana**

Kibana là một nền tảng phân tích hiển thị dữ liệu từ Elasticsearch một cách trực quan dễ sử dụng, Kibana cũng là một công cụ mã nguồn mở miễn phí, cho tất cả mọi người sử dụng. Kibana cung cấp các tính năng cho người dùng quản lý như biểu đồ cột, biểu đồ đường, biểu đồ tròn, biểu đồ nhiệt và nhiều loại chart khác nữa. [5]



Bước 1 - Truy cập <https://www.elastic.co/fr/downloads/kibana>.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Bước 2 - Nhấp vào Downloads để nhận tệp zip.

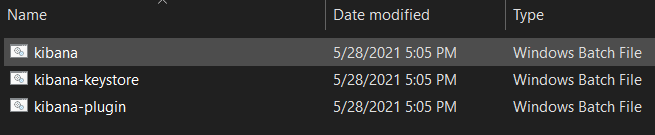
Bước 3 - Khi tệp được tải xuống, giải nén nội dung.

Bước 4 - Đến kibana-7.13.1-windows-x86\_64> bin.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Bước 5 - Bên trong thư mục bin, tìm tập tin kibana và click đúp để khởi động máy chủ.



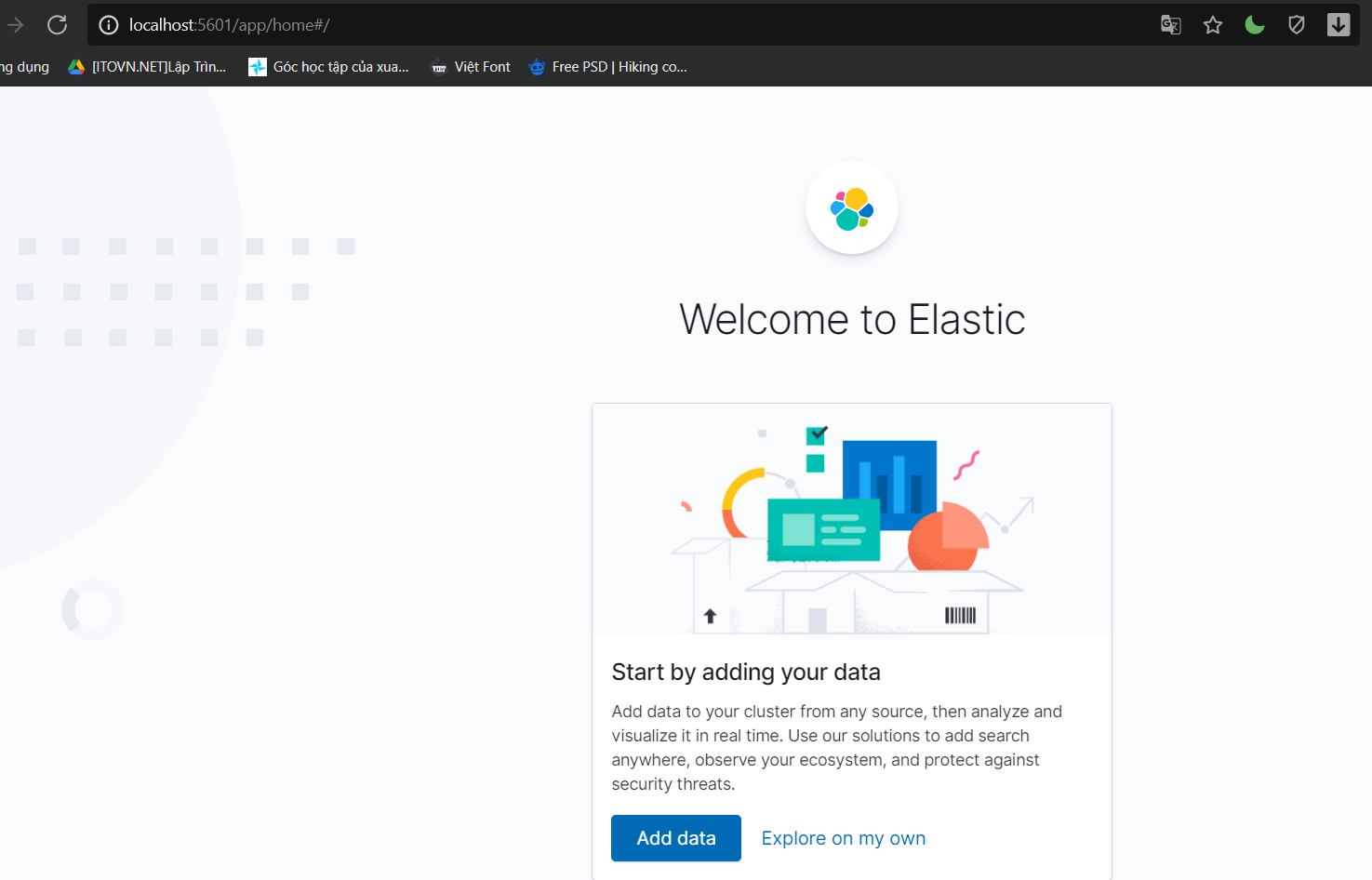
Bước 6 - Đợi máy chủ khởi động.

Ảnh có chứa văn bản, ngoài trời, ảnh chụp màn hình, tấm

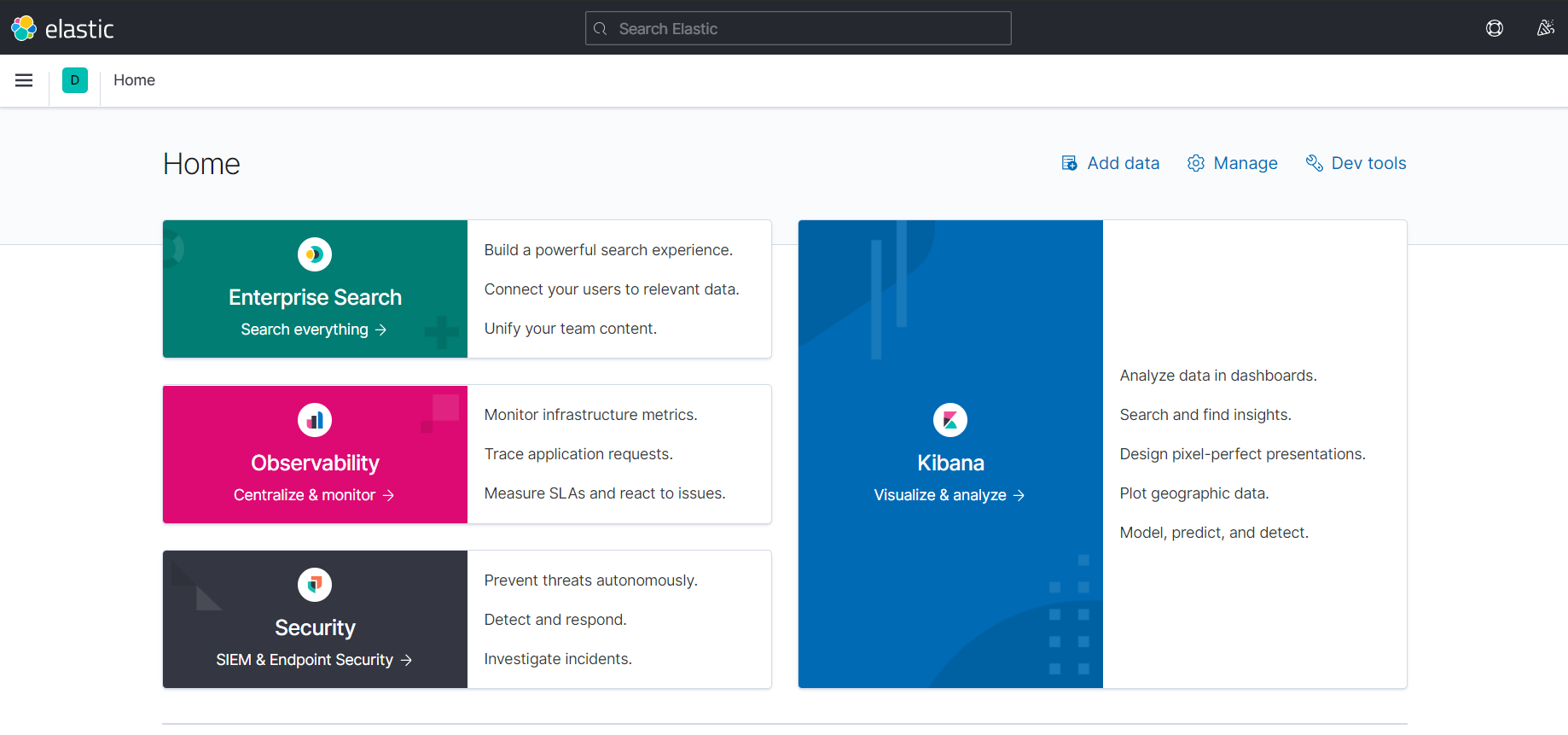
Mô tả được tạo tự động

Bước 7 - Mở trình duyệt và gõ localhost:5601.

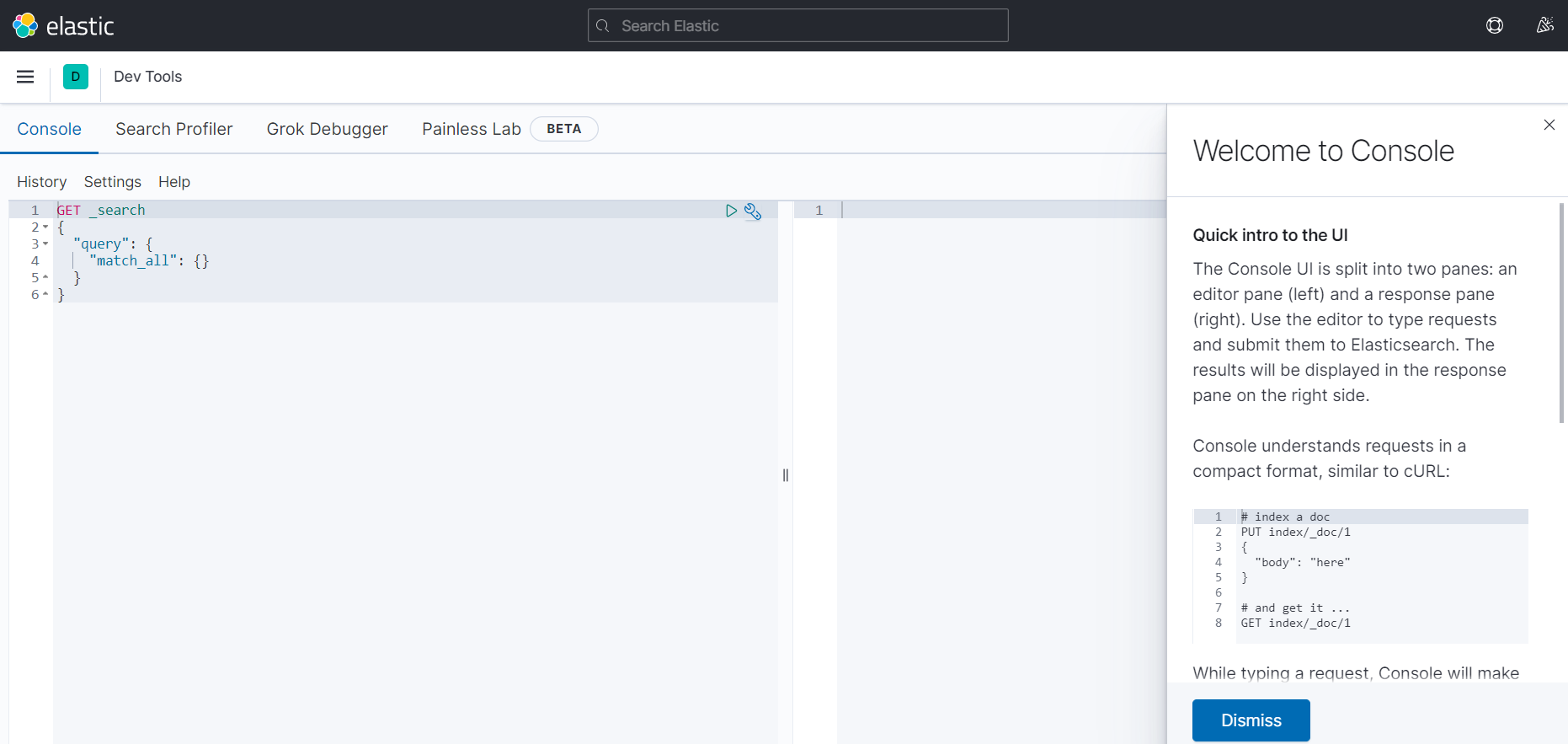
Bước 8 - Nếu bạn thấy thông báo được hiển thị trên trên trình duyệt, có nghĩa là các cài đặt đang thành công.



Chọn Dev tools ở phía trên phải của màn hình để truy cập vào công cụ Console, tại đó bạn gõ các HTTP Request và gửi đến Elastic một cách trực quan (kể cả kết quả trả về).



Giao diện của Dev tools.



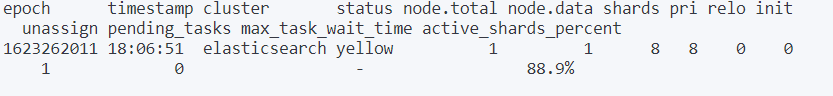
**2.2 Sử dụng Elasticsearch cơ bản**

**Kiểm tra Elasticsearch**

Trong quá trình Elasticsearch vận hành, cần giám sát tình trạng của nó, để kiểm tra thực hiện HTTP/REST đến \_cat/health?v



Kết quả:



Kết quả cho thấy tên của cluster là elasticsearch và trạng thái là green, có ba trạng thái:

- Green Mọi thứ OK, cluster có thể thực hiện đầy đủ chức năng

- Yellow Dữ liệu ổn, nhưng có các bản backup replicas

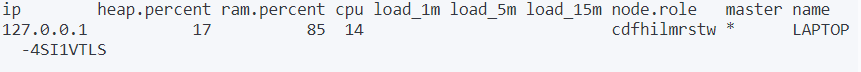
- Red dữ liệu có vấn đề

Tương tự, **kiểm tra danh sách các node** (server) thực hiện GET đến /\_cat/nodes?v

Vd:



Kết quả:



Ở đây trả về có một node (server) trong cluster, node này có tên là LAPTOP.

**Khởi tạo một Index**

Như đã đề cập, Index là một tập hợp những dữ liệu có chung một số tên thuộc tính, tính chất – nó giống như một table trong SQL. Để tạo một index, ta chỉ việc PUT đến địa chỉ /indexname?pretty

Vd:



Kết quả:

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Liệt kê các Index**

Muốn liệt kê các Index thì GET đến /\_cat/indices?v

Vd:



Kết quả:

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Xóa một index**

Muốn xóa đi một index có tên customer, ta DELETE đến địa chỉ /indexname?pretty

Vd:



Kết quả:



**Lưu tài liệu vào Index / Thay thế tài liệu**

Như đã biết các đơn vị dữ liệu là các Document, những Document này được biểu diễn bằng JSON, ví dụ đây là một dữ liệu:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Mỗi dữ liệu lưu vào Index sẽ có một id gán cho nó, giờ ta sẽ lưu dữ liệu trên vào Index customer và gán cho nó id là 1, để làm điều đó ta thực hiện phương thức PUT đến /indexname/\_doc/id?pretty đính kèm dữ liệu JSON.

Vd:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Sau khi chạy lệnh PUT, kết quả ES trả về là:

Ảnh có chứa văn bản

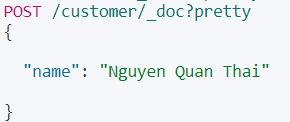
Mô tả được tạo tự động

Cho biết lưu thành công dữ liệu vào index customer, với id là 1

Nếu thực hiện PUT theo cú pháp trên, nếu id tài liệu đã tồn tại trước đó, nó sẽ được thay thế bởi dữ liệu mới => CẬP NHẬT

Nếu thực hiện PUT và không chỉ ra id thì ES sẽ tự sinh ID cho tài liệu mới. Bạn có thể dùng POST để tạo mới cho trường hợp này POST /indexname/\_doc/?pretty

Vd:



Kết quả:

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Đọc tài liệu từ Index**

Nếu đã biết id của tài liệu trong Index customer, việc đọc nội dung của nó thực hiện bằng GET đến địa chỉ /indexname/\_doc/id?pretty

Vd:



Kết quả:

Ảnh có chứa văn bản

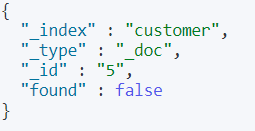
Mô tả được tạo tự động

Dữ liệu trả về là nội dung của tài liệu ở khóa \_source đi kèm các thông tin như index, id …

Nếu tài liệu không tồn tại, ví dụ GET với id = 5, thì nó sẽ thông báo không tìm thấy “found” : false:



Kết quả:



**Xóa tài liệu khỏi Index**

Để xóa tài liệu với id chỉ việc thực hiện DELETE đến địa chỉ /indexname/\_doc/id?pretty

Vd:



Kết quả:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Thậm chí nếu muốn xóa cả index có tên là customer cũng thực hiện DELETE đến /customer?pretty

Vd:



Kết quả:

Ảnh có chứa văn bản

Mô tả được tạo tự động

**2.3 Truy vấn tìm kiếm cơ bản trong Elasticsearch**

Thực hiện các query truy vấn tìm kiếm đơn giản đến Elasticsearch, kết hợp các điều kiện tìm kiếm.

**2.3.1 Nạp dữ liệu mẫu JSON**

Để thực hành ES cung cấp một tập dữ liệu mẫu, gồm 1000 bản ghi lưu trữ thông tin tài khoản bank, ta sẽ nạp dữ liệu này vào Elasticsearch để thực hành. Mỗi dữ liệu Document có cấu trúc dạng:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Trước tiên tải về file JSON chứa 1000 bản ghi dữ liệu này tại: Tải về JSON Data Bank [6], lưu nó thành file accounts.json, rồi gõ lệnh sau để nạp nó vào index bank.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Giờ bạn đang có Index tên bank chứa 1000 Document với cấu trúc mỗi Document như trên.

Có thể thử kiểm tra một Document, ví dụ kiểm tra id = 3



Kết quả:

Ảnh có chứa văn bản, bản đồ

Mô tả được tạo tự động

**2.3.2 Truy vấn đơn giản đến Elasticsearch**

Để thực hiện tìm kiếm đến dữ liệu trong index có tên bank thực hiện GET đến /indexname/\_search, và gắn kèm thông tin cần tìm kiếm dưới dạng JSON.

Ví dụ sau tìm tất cả các Document có trong index bank:



Kết quả:

Ảnh có chứa bản đồ

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

…

Chú ý những phần dữ liệu trả về:

took – thời gian (mili giây) hoàn thành truy vấn.

timed\_out - cho biết vượt thời gian (thời gian hoàn thành vượt ngưỡng bị dừng) hay không.

\_shards - cho biết tìm trên bao nhiêu shard (phân đoạn index).

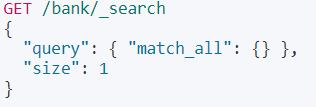
hits - kết quả tìm kiếm.

hits.total - tổng số kết quả tìm được.

hits.hits - mảng dữ liệu thực sự trả về (mặc định 10 document đầu tiên).

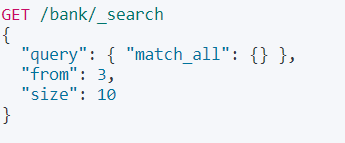
Trong các câu truy vấn search, ta có thể đưa vào nhiều loại tham số để có được kết quả thích hợp, ví dụ cho thêm vào tham số size để lấy số kết quả trả về (mặc định nếu không chỉ ra nó sẽ trả về tối đa 10).

Vd:



Ta cũng có thể chỉ ra dữ liệu trả về bắt đầu từ phần tử nào với tham số from

Vd:



sắp xếp kết quả trả về

Ví dụ các dữ liệu sắp xếp giảm dần theo trường dữ liệu balance

Ảnh có chứa văn bản

Mô tả được tạo tự động

**2.3.3 Thực hiện tìm kiếm với Elasticsearch**

Phần này sẽ thực hiện các truy vấn tìm kiếm với tham số phức tạp hơn.

**Tùy chọn trường dữ liệu trả về**

Mặc định Document trả về từ các truy vấn tìm kiếm đúng như cấu trúc nó lưu vào Index, nếu muốn giới hạn lấy những trường dữ liệu cụ thể nào đó thôi thì ta dùng tham số \_source với mảng liệt kê các trường dữ liệu muốn lấy. Ví dụ chỉ lấy các dữ liệu account\_number và balance của bank:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Kết quả:

Ảnh có chứa văn bản

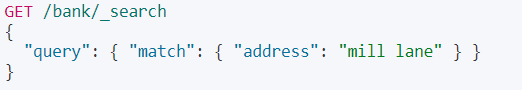
Mô tả được tạo tự động

**Tìm kiếm dữ liệu phù hợp**

Ở các ví dụ trước, sử dụng match\_all ở query sẽ trả về toàn bộ Document có trong Index. Bây giờ sẽ sử dụng đến match, giả sử tìm các tài khoản bank có account\_number bằng 20, sẽ viết truy vấn như sau:

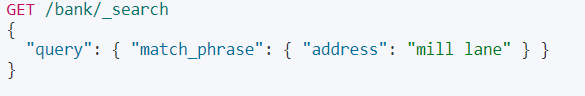


Tương tự, nếu muốn tìm các tài khoản mà địa chỉ address có chứa mill HOẶC lane:



Nếu muốn trả về cả cụm từ mill lane thì cần đến match\_phrase

Vd:



**Sử dụng điều kiện logic bool trong truy vấn tìm kiếm**

Sử dụng điều kiện must kết quả trả về khi tất cả các truy vấn là đúng, ví dụ sau tìm tất cả các địa chỉ có chứa từ mill VÀ lane

Vd:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Sử dụng điều kiện should nó tương tự như phép logic HOẶC.

Vd:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Sử dụng điều kiện must\_not tương tự như phép logic PHỦ ĐỊNH

Ảnh có chứa văn bản

Mô tả được tạo tự động

Kết hợp nhiều điều kiện must, must\_not, should vào truy vấn.

Vd:

Ảnh có chứa văn bản

Mô tả được tạo tự động

**2.3.4 Lọc kết quả Elasticsearch**

Để lọc dùng đến filter, ví dụ sử dụng loại range để lọc lấy lấy dữ liệu balance trong khoảng nào đó.

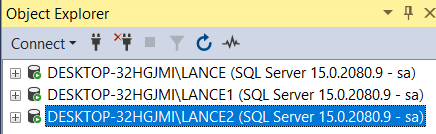
Vd:

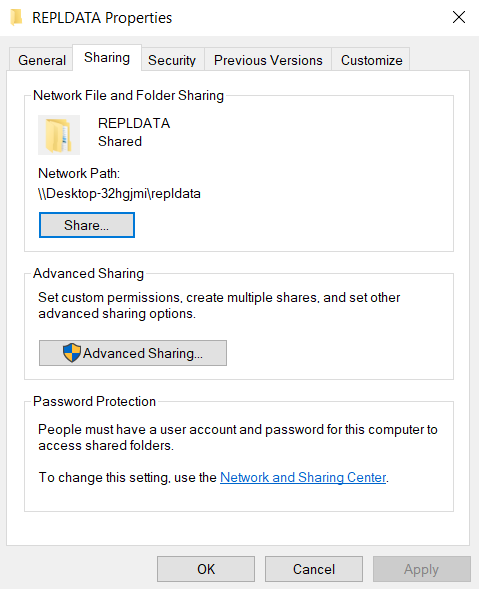


Trên là các tìm kiếm cơ bản, ngoài ra còn nhiều loại cú pháp tạo query tìm kiếm khác như wildcard, regexp, term, fuzzy, query\_string ...

# Chương 3: Ứng dụng/ Kết quả thực nghiệm/So sánh-Đánh giá

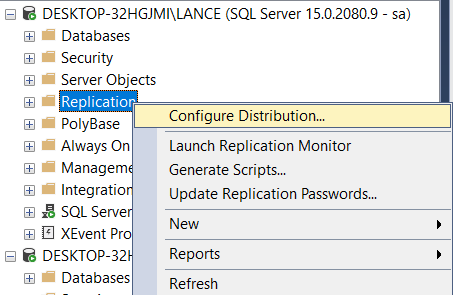
## 3.1 Kết quả thực nghiệm phân tán dữ liệu

Tạo 1 server gốc (LANCE) và 2 server con (LANCE1, LANCE2) để phân tán dữ liệu từ server chính.

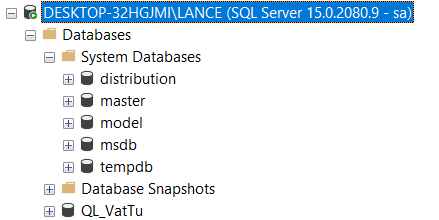
Tiếp theo, ta tạo 1 folder (REPLDATA) để chứa các dữ liệu trong quá trình update dữ liệu từ các phân mảnh về cơ sở dữ liệu gốc và ngược lại. Sau đó, cho folder này là 1 snapshot folder, thực chất là 1 shared folder cho phép các user được quyền read/write.

Trước khi phân tán được dữ liệu, ta phải cấu hình nó.

* Bước 1: Configure distribution

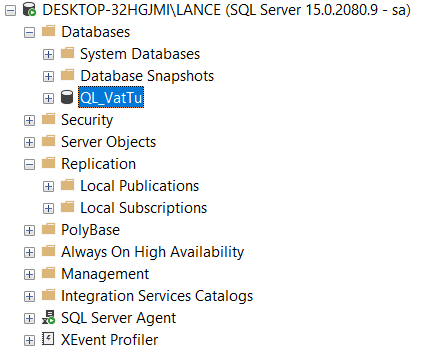


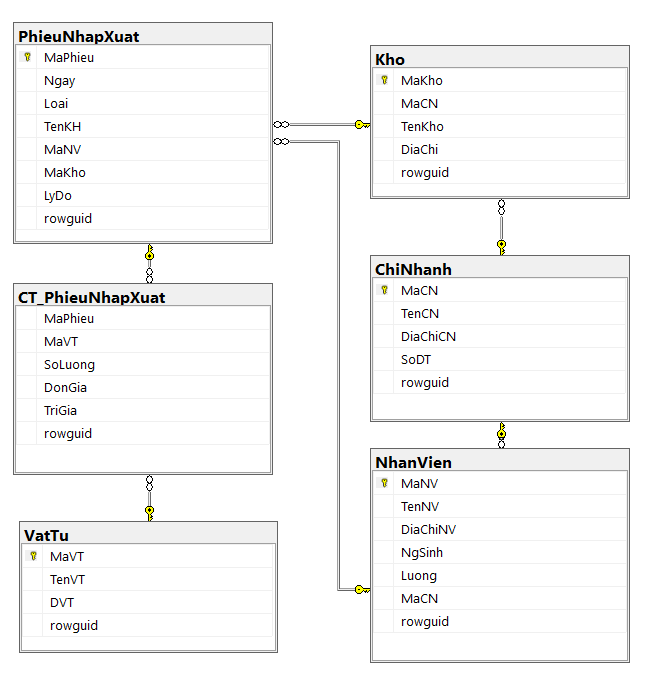
Sau khi cấu hình xong, trong System Databases sẽ hiển thị mục distribution



* Bước 2: Create publication (chứa lược đồ phân mảnh)

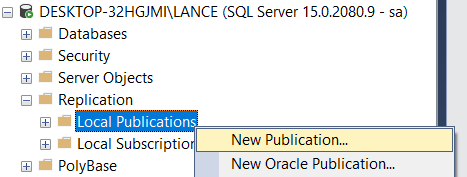
Trước khi tạo publication, ta phải có 1 database. Ở đây, em dùng database QL\_VatTu (quản lí vật tư).



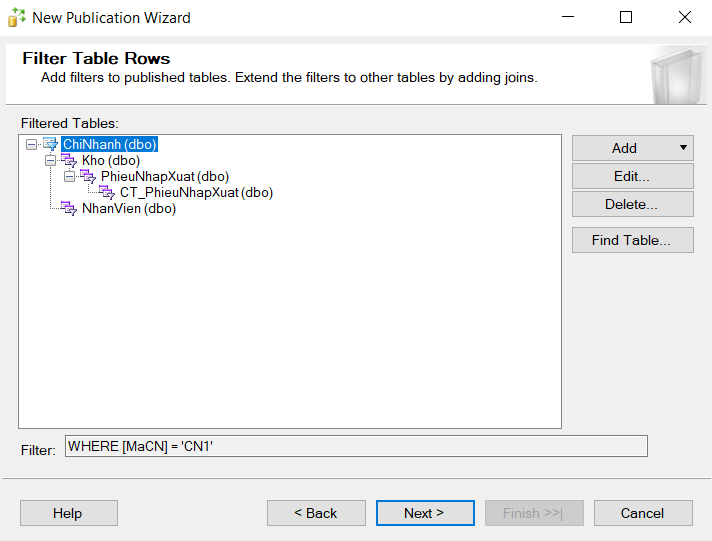


*Database Diagrams của database QL\_VatTu*

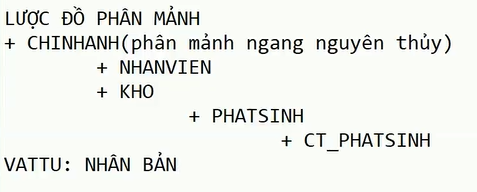
Sau đó, ta tạo publications để phân tán dữ liệu QL\_VatTu.



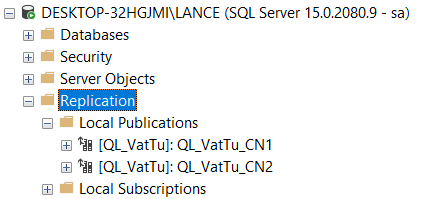
Ta tạo lược đồ phân mảnh.



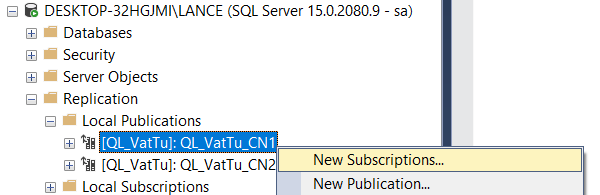
Ta sẽ phân tán theo mã chi nhánh (chi nhánh 1: ‘CN1’, chi nhánh 2: ‘CN2’)



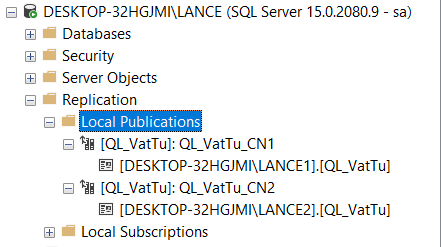
Sau khi tạo xong ta sẽ được 1 publication cho chi nhánh 1. Làm tương tự ta sẽ được 1 pulication cho chi nhánh 2



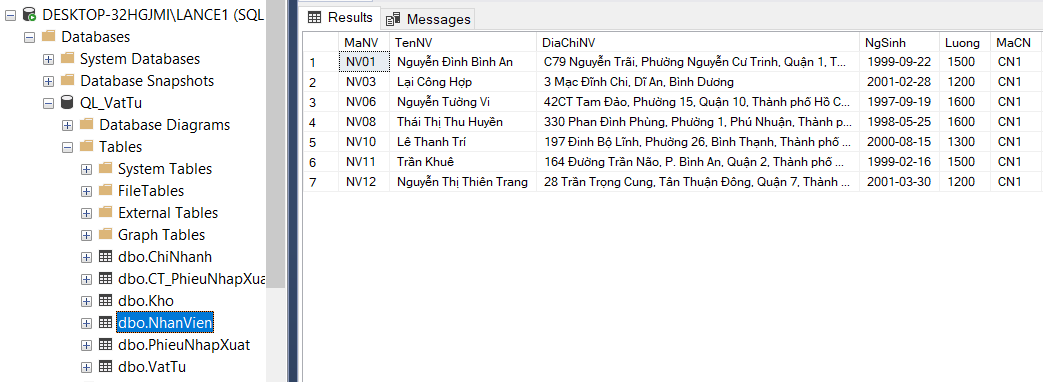
* Bước 3: Create Subscription (chỉ định server nào sẽ chứa publication đó)



Sau khi tạo xong ta sẽ được 2 Subscriptions cho 2 Publications tương ứng



Làm xong bước tất cả các bước trên, dữ liệu từ database QL\_VatTu sẽ được phân tán sang các server con (LANCE1, LANCE2) theo mã chi nhánh (MaCN) tương ứng.



Ví dụ: Bảng nhân viên ở database QL\_VatTu của server LANCE1 sẽ chỉ chứa những nhân viên có mã chi nhánh là CN1.

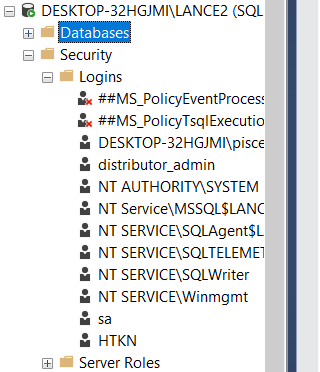
Sau khi cấu hình xong, ta tạo Linked Server để có thể liên kết server này đến server khác.

Việc này tạo ra nhằm mục đích: khi đứng ở server này có thể truy xuất đến server khác mà không phải thông qua server gốc để giảm tải cho server gốc.

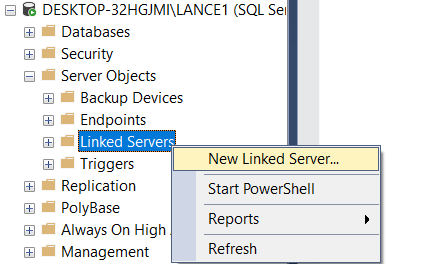
Để tạo Linked Server: cụ thể là từ server LANCE1 đến server LANCE2

* Bước 1: Tạo remote login (HTKN) bên server LANCE2

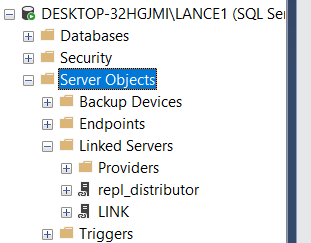
Sau khi tạo xong, ta sẽ được 1 login tên HTKN



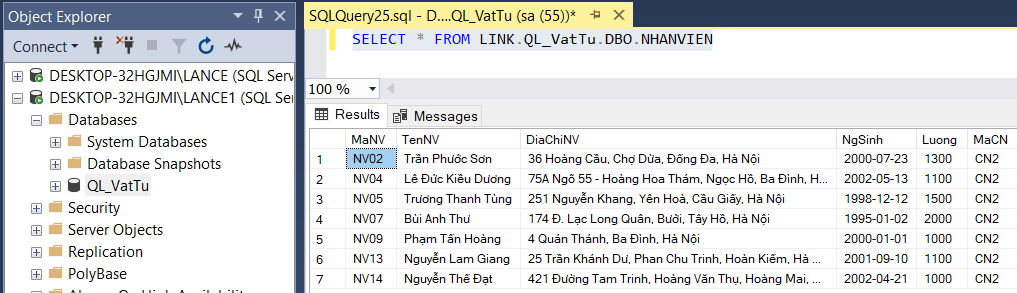
* Bước 2: Tạo Linked Server từ server LANCE1 đến server LANCE2



Sau khi tạo 1 Linked Server tên LINK xong, ta sẽ được như hình dưới.



Sau khi tạo xong Linked Server ta có thể dễ dàng truy xuất dữ liệu của server LANCE2 từ server LANCE1.

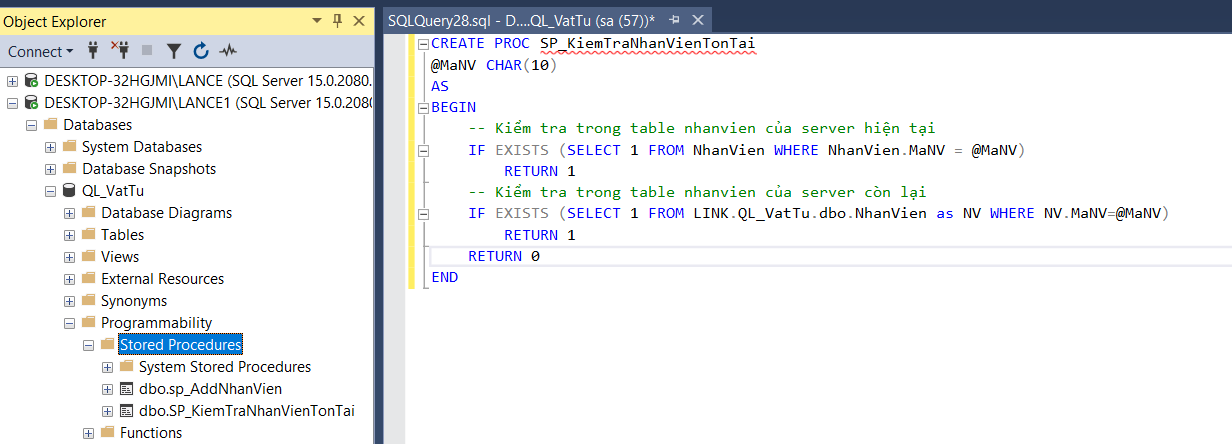


Truy xuất toàn bộ nhân viên của server LANCE2 từ server LANCE1 thông qua ‘LINK’.

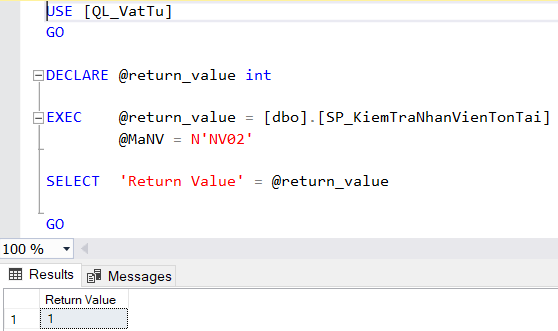
Để làm rõ hơn chức năng của của Linked Server, ta sẽ đến ví dụ: muốn thêm nhân viên ở server này thì trước hết ta phải kiểm tra thử MaNV (mã nhân viên) đã tồn tại ở server này lẫn các server phân mảnh khác chưa.

* Tạo thủ tục (procedure) để kiểm tra mã nhân viên.

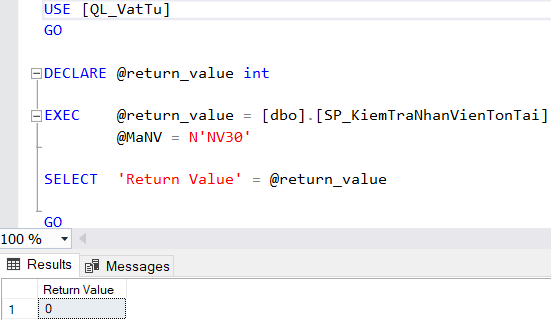
Ý tưởng: Kiểm tra mã nhân viên, nếu tồn tại thì trả về 1. Nếu chưa tồn tại thì trả về 0



Chạy thử: ‘NV02’ thuộc chi nhánh 2 (server LANCE2) => trả về 1



Chạy thử: ‘NV30’ chưa thuộc server nào => trả về 0

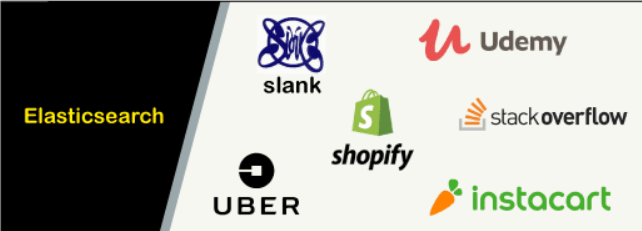


## 3.2 So sánh – Đánh giá

##### **3.2.1 So sánh Elasticsearch với Splunk**



* **Khách hàng**

Một số công ty như Uber, Stack Overflow, Udemy, Shopify, Instacart và Slank, v.v. sử dụng **ElasticStack** để lưu trữ, phân tích, tìm kiếm và trực quan hóa dữ liệu của họ. Nơi Elasticsearch, Logstash và Kibana thực hiện vai trò của họ rất sắc nét.

Mặt khác, **Splunk** cũng được sử dụng bởi một số công ty như Starbucks, Craftbase, Intuit, SendGrid, Yelp, Rent the Runway và Blend, v.v. [7]

* **Công cụ tích hợp**

**Elasticsearch** tích hợp với Logstash và Kibana để hoạt động giống như Splunk. Ngoài ra, nó cũng có thể tích hợp với nhiều công cụ khác, chẳng hạn như - Datadog, Couchbase, Amazon Elasticsearch Services và Contentful, v.v. Bằng cách tích hợp với Logstash và Kibana, nó hoạt động giống như Splunk.



**Splunk** cũng có thể tích hợp với một số công cụ khác như OverOps, Google Anthos, PagerDuty, Amazon Guard Duty và Wazuh, v.v. [7]

* **Chi phí**

**Elasticsearch là** một công **cụ mã nguồn mở,** có nghĩa là nó có sẵn miễn phí.

**Splunk**, nó không **phải là một công cụ mã nguồn** mở. Vì vậy, nó không có sẵn miễn phí trên thị trường. Splunk đi kèm với 15 ngày dùng thử miễn phí. Ta phải trả tiền cho nó 75 $/tháng nếu muốn dùng tiếp. [7]

* **Phát hành**

**Elasticsearch** được phát hành vào năm 2010 bởi Shay Banon. Nó đến khi mọi người muốn một số công cụ khác để so sánh hiệu suất của Splunk.

**Splunk** được phát hành vào năm 2003 bởi Michael Baum, Erik Swan và Rob Das. Họ đã đưa ra giải pháp thu thập logo dữ liệu và nhật ký dữ liệu máy để xác nhận tính toàn vẹn của hệ thống. [7]

* **Di chuyển dữ liệu và quản lý người dùng**

Về cơ bản, nó phụ thuộc vào loại định dạng dữ liệu được hỗ trợ bởi các công cụ này. Elasticsearch và Splunk đều có các đối tượng gửi dữ liệu duy trì đường dẫn của các tệp và tải lên.

**Elasticsearch** không có trình hướng dẫn và tính năng được tải sẵn. Trong Elasticsearch, chúng ta phải xác định ánh xạ của từng loại trường với giá trị của nó.

**Splunk đi**kèm với các trình hướng dẫn và tính năng được tải sẵn, dễ dàng và đáng tin cậy để sử dụng. Những tính năng này giúp người quản lý quản lý tài nguyên. Cùng với đó, nó cũng có các tính năng tích hợp và được cấu hình sẵn cho phép người dùng ánh xạ dữ liệu vào các thực thể với các giá trị tương ứng của chúng.

**Kết luận:**  Splunk là một cách tiếp cận dễ dàng và đáng tin cậy trong trường hợp di chuyển dữ liệu và quản lý người dùng, trong khi Elasticsearch đang nhận được các tính năng này rất nhanh. [7]

* **Trực quan hóa giao diện người dùng và bảng điều khiển**

Hầu hết thời gian, người dùng muốn làm việc với giao diện người dùng đồ họa dễ sử dụng và hiểu. Họ làm quen với nó trong thời gian rất ít hơn. **Splunk đã cải** thiện giao diện người dùng của mình với một bảng điều khiển mới và các điều khiển của nó. Nó có các tính năng thú vị của việc xuất bảng điều khiển của họ sang PDF

Mặt khác, **Elasticsearch không có** giao diện người dùng riêng. Người dùng cần cài đặt Kibana để thực hiện tác vụ. Kibana có các chủ đề nền thú vị mà Splunk không có. Bảng điều khiển của nó chứa các điều khiển khác nhau để hành động. Do đó, bảng điều khiển ở Kibana tốt hơn Splunk. [7]

##### **3.2.2 So sánh Elasticsearch với Cassandra**



| **Tham số so sánh** | **Elasticsearch** | **Cassandra** |
| --- | --- | --- |
| **Giấy phép** | Elasticsearch được Facebook phát triển để lưu trữ dữ liệu phi cấu trúc ở dạng tài liệu. | Cassandra được phát triển bởi Apache Open Source Projects. |
| **Tốc độ** | Tìm kiếm chỉ mục hiệu quả giúp Elasticsearch nhanh hơn. | Cassandra nhanh hơn cho các truy vấn của kịch bản nhỏ. |
| **Tích hợp với** | **Công cụ Elasticsearch** tích hợp với Logstash và Kibana cho một ngăn xếp ELK hoàn chỉnh. Ngoài ra, nó cũng có thể tích hợp với nhiều công cụ khác, chẳng hạn như - Datadog, Couchbase, Amazon Elasticsearch Services và Contentful, v.v. | **Apache Cassandra cũng** có thể tích hợp với một số công cụ khác như Datadog, Presto, Buddy, Kong và Redash, v.v. |
| **Chi phí** | Đây là một công cụ mã nguồn mở. Vì vậy người dùng không cần phải trả bất kì khoản phí nào. | Giống như Elasticsearch, Apache Cassandra cũng là một công cụ miễn phí. |
| **Phụ Thuộc** | Elasticsearch phụ thuộc vào hiệu quả của việc triển khai thuật toán. | Cassandra phụ thuộc vào việc thực thi. |
| **Hỗ trợ ngôn ngữ** | Một số ngôn ngữ như Java, .NET, Perl, Groovy, PHP và Python được Elasticsearch hỗ trợ. | Cassandra cũng cung cấp hỗ trợ cho nhiều ngôn ngữ, chẳng hạn như Erlang, Ruby, Scala, Go, Java và Python, v.v. Hầu hết, nó hỗ trợ các ngôn ngữ lập trình hướng đối tượng. |
| **Dễ sử dụng** | Vì Elasticsearch dựa trên REST API, vì vậy thật dễ dàng để viết truy vấn và giao dịch. | Trong Apache Cassandra, thật dễ dàng để viết các truy vấn và kịch bản. |
| **Hiệu năng** | Elasticsearch cung cấp tính khả dụng cao và nó cũng có khả năng tìm kiếm dựa trên chỉ mục nhanh, giúp cải thiện hiệu suất của nó. | Apache Cassandra cung cấp hiệu suất truyến tính. |
| **Khả năng mở rộng** | Cung cấp khả năng mở rộng cao và thời gian chạy truy vấn nhanh hơn. | Khả năng mở rộng cao. |
| **Các công ty được sử dụng** | Một số công ty như Uber, Stack Overflow, Udemy, Shopify, Instacart và Slank, v.v sử dụng ElasticStack để lưu trữ, phân tích, tìm kiếm và trực quan hóa dữ liệu của họ. Nơi Elasticsearch, Logstash và Kibana thực hiện vai trò của họ rất sắc nét. | Cassandra cũng được sử dụng bởi một số công ty như Spotify, Facebook, Netflix, công nghệ Uber, v.v. |

Bảng so sánh dựa trên một số thông số [8]

##### **3.2.3 So sánh Elasticsearch với MongoDB**

****

* **Điểm tương đồng**
* Lưu trữ dữ liệu trong tài liệu JSON mà không có lược đồ.
* Hỗ trợ lập chỉ mục dữ liệu.
* Được sử dụng cho một tập dữ liệu khổng lồ - thứ yêu cầu xử lý.
* Hỗ trợ dữ liệu phân mảnh (phân vùng dựa trên các chỉ mục để quản lý lưu trữ).
* Được phát triển bởi các cộng đồng mã nguồn mở và do đó có tài liệu chi tiết.
* Hỗ trợ tính đồng thời. Nhiều phiên bản của cùng một dữ liệu có thể được tạo và các thao tác khác nhau có thể được thực hiện trên dữ liệu mà các phiên bản nắm giữ. [9]
* **Điểm khác biệt**

| **Elasticsearch** | **MongoDB** |
| --- | --- |
| Là một cơ sở dữ liệu NoSQL được viết bằng Java. | Là một cơ sở dữ liệu NoSQL hướng tài liệu được viết bằng C++. |
| Có giao diện REST API (hỗ trợ HTTP) | Không có giao diện REST API |
| Sử dụng mô hình không gian Vector, truy xuất thông tin, v.v… để phù hợp với kết quả có khả năng nhất. | Dữ liệu được truy xuất bằng cách so sánh các tài liệu. |
| Việc chuyển đổi nhị phân là không thể với tài liệu JSON. | Có thể chuyển đổi JSON thành BSON (Phiên bản nhị phân của JSON). |
| Lucene Indexer lập chỉ mục. | Người dùng nên lập chỉ mục. |
| Nó đòi hỏi rất nhiều sự chú ý từ các lập trình viên để tạo ra ứng dụng tốt nhất. | Đây là một cơ sở dữ liệu thân thiệt với người dùng, đòi hỏi ít sự chú ý hơn từ các lập trình viên. |
| Là một lựa chọn tốt để thực hiện tìm kiếm toàn văn bản. | Nó cho phép ta thực hiện các thao tác CRUD mà không cần hỗ trợ toàn văn bản. |
| Hỗ trợ ít ngôn ngữ (Java, Javascript, Perl, .NET, PHP, Python, Groovy, Ruby) | Hỗ trợ hầu hết tất cả các ngôn ngữ hiện hành. |
| Nó không hỗ trợ các hoạt động của Mapreduce trên dữ liệu khổng lồ nhưng chạy bộ lọc tìm kiếm văn bản trên dữ liệu khổng lồ. | Cho phép các hoạt động của Mapreduce trên dữ liệu để quản lý dữ liệu quy mô lớn. |
| Có tính năng TTL tích hợp | Không có tính năng TTL |
| Có khả năng mất dữ liệu trong quá trình ghi dữ liệu. | Thao tác ghi đáng tin cậy và chính xác. |
| Truy xuất dữ liệu đáng tin cậy do giao diện web. | Vì không có giao diện web tích hợp, các ứng dụng của bên thứ ba có thể được sử dụng để truy xuất dữ liệu không phải lúc nào cũng an toàn. Do đó, nó không hoàn toàn đáng tin cậy. |
| Được sử dụng chủ yếu trong các ứng dụng liên quan đến Internet of Things (IoT), di động, điện toán đám mây, v.v. | Được sử dụng chủ yếu trong các ứng dụng liên quan đến tìm kiếm, bảo mật, số liệu, bản đồ, ghi nhật ký, v.v. |
| Elasticsearch chỉ có một vài ứng dụng của bên thứ ba. Ví dụ: Đám mây đàn hồi Dremio. | MongoDB, do được sử dụng rộng rãi, có nhiều ứng dụng của bên thứ ba. Ví dụ: Studio 3T, DHawk, CData, ScaleGrid, v.v. |

##### **3.2.4 Đánh giá**

## 3.3 Ứng dụng

Để có ý tưởng về cách Elasticsearch ứng dụng trong hệ thống lớn, hãy cùng xem xét 3 ví dụ điển hình sau:

1. ElasticSearch là phần back-end chính cho trang web:

Ta có một trang web cho phép người dùng viết bài đăng lên blog, nhưng ta cũng muốn có khả năng tìm kiếm các bài viết. Ta có thể sử dụng ElasticSearch để lưu trữ toàn bộ dữ liệu có liên quan đến bài đăng và phục vụ cho các câu truy vấn.

1. Thêm ElasticSearch vào một hệ thống hiện có:

Ta có thể đang đọc bài viết này bởi vì ta đang có sẵn hệ thống lưu trữ dữ liệu và ta muốn thêm vào chức năng tìm kiếm.

1. Sử dụng ElasticSearch như là phần back-end của những giải pháp đã được xây dựng sẵn:

Bởi vì Elasticsearch là mã nguồn mở và cung cấp giao thức HTTP, cho nên có một hệ sinh thái lớn hỗ trợ nó.

Ví dụ: ElasticSearch phổ biến về dữ liệu log tập trung, cung cấp các công cụ có sẵn để viết và đọc ElasticSearch, ngoài ra còn có phần cấu hình các công cụ theo cách mà ta muốn, vì thế ta không cần phải phát triển gì thêm.

Chúng ta hay xem xét kỹ hơn từng kịch bản: [10]

##### **3.3.1 Elasticsearch là phần back-end chính cho trang web**

Theo cách truyền thống, các công cụ tìm kiếm được triển khai trên nơi lưu trữ dữ liệu kiên cố để cung cấp khả năng tìm kiếm nhanh và có liên quan. Bởi vì các công cụ tìm kiếm trong lịch sử không có một nơi lưu trữ bền vững hay tính năng cần thiết như thống kê.

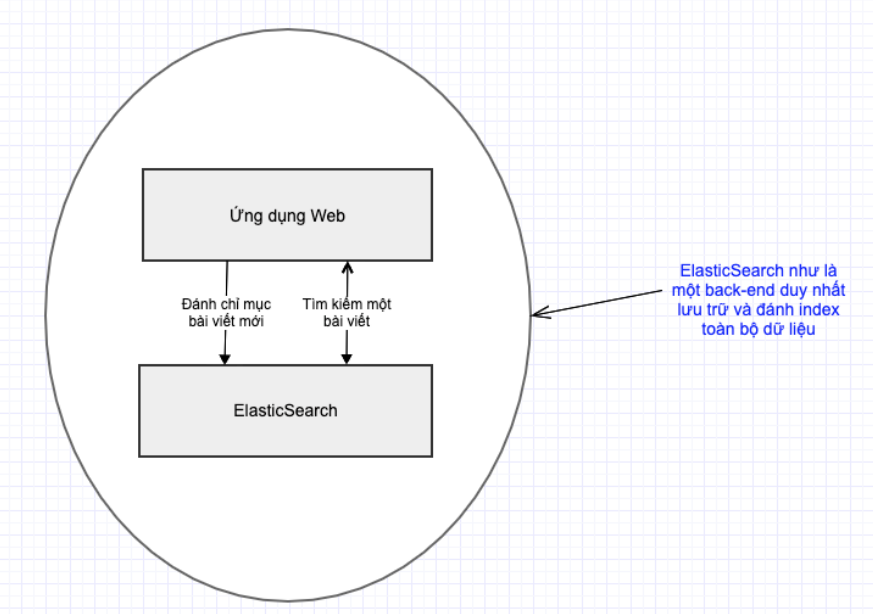
Elasticsearch là một trong những công cụ tìm kiếm hiện đại cung cấp khả năng: lưu trữ bền vững, thống kê, và nhiều tính năng khác mà ta mong đợi từ một nơi lưu trữ dữ liệu.

Nếu ta bắt đầu một dự án mới, ta nên xem xét và sử dụng Elasticsearch như là nơi lưu trữ dữ liệu duy nhất để giữ cho thiết kế của ta đơn giản nhất có thể. Điều này có thể không phù hợp với tất cả các trường hợp.

Ví dụ: Khi cần phải cập nhập dữ liệu thường xuyên, thì ta phải sử dụng ElasticSearch ở phía trên của cơ sở dữ liệu.

Lưu ý: Giống như NoSQL, ElasticSearch KHÔNG hỗ trợ phiên giao dịch (transaction).

Chúng ta sẽ quay trở lại ví dụ về blog: Ta có thể lưu trữ những bài viết mới trong ElasticSearch. Tương tự, ta có thể sử dụng ElasticSearch để lấy dữ liệu, tìm kiếm, và thống kê toàn bộ dữ liệu.



Điều gì sẽ xảy ra nếu server mất kết nối? Ta có thể được chấp nhận lỗi (fault tolerance) bằng cách sao chép dữ liệu tới server khác.

Nhiều tính năng khác làm cho ElasticSearch là một nơi lưu trữ dữ liệu NoSQL hấp dẫn.

Nó không thể tuyệt vời cho mọi thứ, nhưng có thể nên cân nhắc thử, nếu thêm một nơi lưu trữ dữ liệu vào tổng thể thiết kế có đáng hay không?[10]

##### **3.3.2 Thêm Elasticsearch vào một hệ thống sẵn có**

ElasticSearch không phải lúc nào cũng cung cấp những chức năng mà ta cần từ một nơi lưu trữ dữ liệu. Một số trường hợp ta sử dụng ElasticSearch ở một nơi lưu trữ dữ liệu khác.

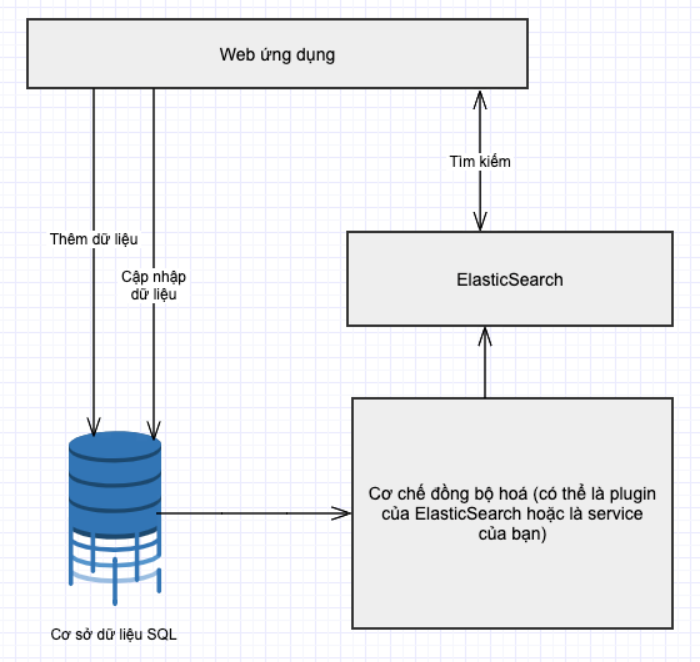
Ví dụ: Hỗ trợ phiên giao dịch (transaction support) và mối quan hệ phức tạp giữa dữ liệu là những tính năng mà ElasticSearch không hỗ trợ, ít nhất là trong phiên bản 1.

Nếu ta muốn sử dụng những tính năng này, hãy cân nhắc vấn đề sử dụng ElasticSearch ở một nơi lưu trữ dữ liệu khác.

Hoặc là ta đã có sẵn hệ thống phức tạp, nhưng muốn thêm chức năng tìm kiếm. Sẽ thật nguy hiểm nếu như ta thiết kế lại toàn bộ hệ thống cho mục đích sử dụng duy nhất. Cách tiếp cận an toàn hơn là thêm Elasticsearch vào hệ thống và làm cho nó tương tác với các thành phần có sẵn.

Bằng cách nào đi nữa. Nếu có 2 nơi lưu trữ dữ liệu, thì ta phải tìm cách cho chúng đồng bộ hoá dữ liệu.

Phụ thuộc vào kho lưu trữ dữ liệu chính là gì và cách trình bày dữ liệu, ta có thể triển khai plugin Elasticsearch để giúp chúng đồng bộ hoá, như được mô tả ở hình dưới.



Giả sử như ta có cửa hàng bán lẻ online với thông tin sản phẩm được lưu trữ trong cơ sở dữ liệu SQL. Ta cần tìm kiếm dữ liệu nhanh chóng và có liên quan, ta cài đặt ElasticSearch.

Để đánh chỉ mục dữ liệu, ta cần triển khai một cơ chế đồng bộ hoá, có thể là plugin của ElasticSearch hoặc một service do chính ta xây dựng. Cơ chế đồng bộ hoá này có thể kéo tất cả các dữ liệu phù hợp cho từng sản phẩm và lập chỉ mục trong ElasticSearch, nơi mà sản phẩm được lưu dưới dạng tài liệu (document).

Khi người dùng nhập tiêu chí tìm kiếm trên trang web, thì web ứng dụng thực hiện truy vấn ElasticSearch cho tiêu chí đó. ElasticSearch trả về sản phầm phù hợp với tiêu chí, và được sắp xếp theo cách ta muốn.

Cách sắp xếp có thể dựa trên điểm liên quan, cho biết từ tìm kiếm xuất hiện bao nhiều lần trong mỗi sản phẩm, hoặc bất kỳ thứ gì được lưu trữ trong sản phẩm như là: cách sản phẩm được thêm gần đây, xếp hạng trung bình, hoặc thậm chí là sự kết hợp của những thứ đó.

Việc chèn hoặc cập nhập thông tin vẫn có thể thực hiện trên cơ sở dữ liệu chính, do đó ta chỉ sử dụng ElasticSearch để xử lý chức năng tìm kiếm.

Nó phụ thuộc vào cơ chế đồng bộ hoá để giúp cho ElasticSearch luôn cập nhập những thay đổi mới nhất.

Khi ta muốn tích hợp ElasticSearch với những thành phần khác, có thể sử dụng những công cụ có sẵn đã tích hợp chức năng cần thiết.

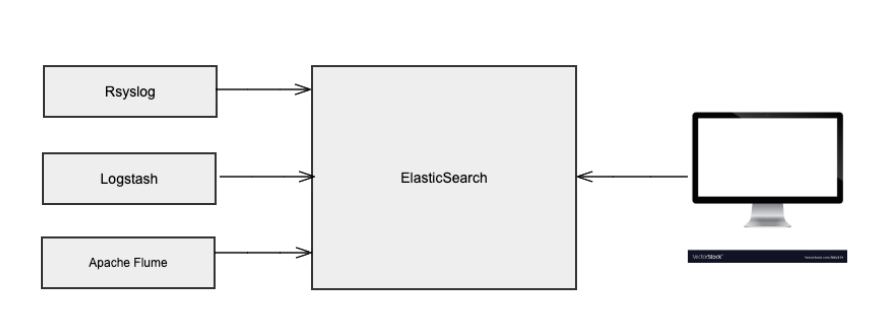
Như chúng ta sẽ khám khá trong phần tiếp theo, có một cộng đồng lớn, xây dựng công cụ (tools) cho ElasticSearch, và đôi khi ta không cần xây dựng bất cứ thứ gì. [10]

##### **3.3.3 Tích hợp Elasticsearch với các công cụ có sẵn**

Trong một số trường hợp thực tế, ta không phải viết bất kỳ dòng code nào cho ElasticSearch. Nhiều công cụ có sẵn hoạt động với ElasticSearch, vì vậy ta không cần phải viết từ đầu.

Ví dụ: Ta muốn triển khai framework lưu log với quy mô lớn bao gồm những chức năng: lưu trữ, tìm kiếm, và phần tích số lượng lớn các sự kiện.

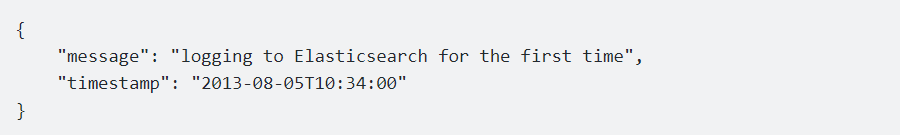
Như hình minh hoạ dưới đây, để xử lý log và xuất ra kết quả trong ElasticSearch, ta có thể sử dụng công cụ log như Rsyslog hoặc Logstash hoặc Apache Flume. Để tìm kiếm và phân tích log trên giao diện trực quan, ta có thể sử dụng Kibana.



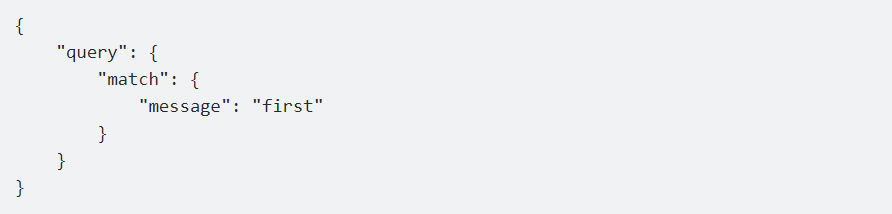
Thực tế ElasticSearch là mã nguồn mở theo giấy phép của Apache 2. Đó là lí do tại sao có nhiều công cụ hỗ trợ ElasticSearch.

Mặc dù ElasticSearch được viết bằng Java, nhưng có nhiêu hơn 1 API cho ta làm việc với nó. ElasticSearch cũng hỗ trợ REST API giúp ứng dụng nào cũng có thể truy cập mà không cần phân biệt ngôn ngữ lập trình của ta là gì. Hơn nữa, các yêu cầu và trả lời REST thường là định dạng JSON (JavaScript Object Notation).

Ví dụ: Khi ta đánh chỉ mục cho sự kiện log, thì chúng sẽ được hiển thị như sau:



Yêu cầu tìm kiếm sự kiện log với giá trị là "first" sẽ trông như sau:



Truyền dữ liệu và chạy câu truy vấn thông qua đối tượng JSON qua HTTP giúp cho ta dễ mở rộng - từ hệ thống log Rsyslog cho tới kết nối framework như là Apache ManifoldCF để tương tác với ElasticSearch.

Nếu ta xây dựng 1 ứng dụng mới từ đầu hoặc ta mới thêm chức năng tìm kiếm và hệ thống sẵn có, REST API là một trong những tính năng làm cho ElasticSearch trở nên hấp dẫn. [10]

# Chương 4: Kết luận

## 4.1 Ưu điểm

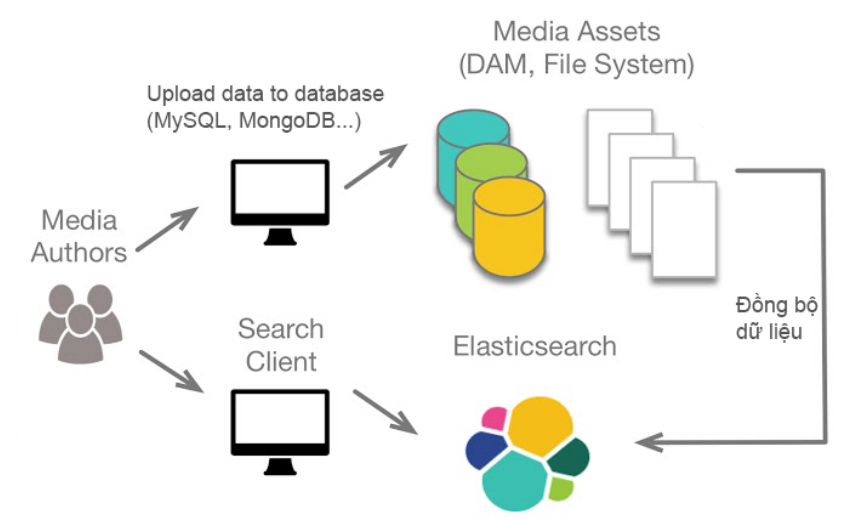
* Elasticsearch tương thích để chạy trên mọi nền tảng vì nó được phát triển bằng Java.
* Nó là một công cụ tìm kiếm thời gian thực, có nghĩa là chỉ một giây trước khi tài liệu được thêm vào là có thể tìm kiếm được trong công cụ này.
* Elasticsearch đưa ra khái niệm về cổng, cho phép tạo các bản sao lưu đầy đủ một cách dễ dàng.
* Nó được định hướng tài liệu phân tán giúp dễ dàng mở rộng quy mô trong tổ chức lớn. Nhà phát triển có thể dễ dàng tích hợp nó vào bất kỳ tổ chức lớn nào bằng cách mở rộng quy mô.
* Có thể dễ dàng xử lý nhiều hợp đồng thuê nhà trong Elasticsearch so với Apache Solr.
* Tài liệu của nó có sẵn bằng một số ngôn ngữ. Do đó, mọi người từ các vùng khác nhau có thể sử dụng nó bằng ngôn ngữ của họ.
* Elasticsearch là một mã nguồn mở. Vì vậy, không cần phải trả bất kỳ chi phí bản quyền nào để tải xuống.
* Nó hỗ trợ tất cả các loại tài liệu, chỉ ngoại trừ những loại không hỗ trợ kết xuất văn bản. [11]

## 4.2 Khuyết điểm

* Đôi khi, vấn đề về tình huống não chia đôi xảy ra trong Elasticsearch.
* Không giống như Apache Solr, Elasticsearch không hỗ trợ đa ngôn ngữ để xử lý dữ liệu yêu cầu và phản hồi.
* Elasticsearch không phải là một kho lưu trữ dữ liệu tốt như các tùy chọn khác như MongoDB, Hadoop, v.v. Nó hoạt động tốt cho các trường hợp sử dụng nhỏ, nhưng trong trường hợp phát trực tuyến dữ liệu của TB mỗi ngày, nó sẽ bị nghẹt hoặc mất dữ liệu.
* Nó là một công cụ tìm kiếm lưu trữ dữ liệu linh hoạt và mạnh mẽ, nhưng hơi khó học. Đặc biệt là về cách sử dụng tìm kiếm doanh nghiệp, nó không đơn giản như tìm kiếm ngoài hộp. [11]

## 4.3 Hướng phát triển

* Vì Elasticsearch không phải là một kho lữu trữ dữ liệu tốt như đã đề cập ở phần nhược điểm. Nên để phát huy tối ưu ta sẽ kết hợp cả MongoDB và Elasticsearch. MongoDB sẽ được dùng để làm database lưu dữ liệu còn Elasticsearch sẽ được đồng bộ dữ liệu với MongoDB. Khi thực hiện tìm kiếm ta sẽ tìm kiếm trên Elasticsearch. [12]



# 

# TÀI LIỆU THAM KHẢO

[1]. Elasticsearch là gì? Cách sử dụng Elasticsearch.

https://bizflycloud.vn/tin-tuc/news-20181015092614974.html, accessed: 08/06/2021.

[2]. Giới thiệu và cài đặt Elasticsearch và Kibana bằng Docker.https://xuanthulab.net/gioi-thieu-va-cai-dat-elasticsearch-va-kibana-bang-docker.html, accessed: 10/06/2021.

[3]. Giới thiệu và cài đặt Elasticsearch và Kibana bằng Docker. <https://xuanthulab.net/gioi-thieu-va-cai-dat-elasticsearch-va-kibana-bang-docker.html>, accessed: 03/06/2021.

[4]. Elasticsearch là gì? Những thông tin hữu ích không thể bỏ lỡ. <https://timviec365.vn/blog/elasticsearch-la-gi-new13166.html>, accessed: 03/06/2021.

[5]. Tìm hiểu về Kibana và sử dụng một cách hiệu quả.

https://levanphu.info/kibana-la-gi-tim-hieu-ve-kibana-va-su-dung-mot-cach-hieu-qua, accessed: 09/06/2021.

[6]. Json data bank

https://download.elastic.co/demos/kibana/gettingstarted/accounts.zip, accessed: 10/06/2021.

[7] Elasticsearch vs Splunk

https://www.javatpoint.com/elasticsearch-vs-splunk, truy cập ngày 5/6/2021.

[8] Elasticsearch vs Cassandra

https://www.javatpoint.com/elasticsearch-vs-cassandra, truy cập ngày 5/6/2021.

[9] MongoDB vs Elasticsearch

https://www.educba.com/mongodb-vs-elasticsearch/, truy cập ngày 5/6/2021.

[10] Khám phá ứng dụng điển hình của Elasticsearch

https://viblo.asia/p/kham-pha-ung-dung-dien-hinh-cua-elasticsearch-Ljy5V7joKra,

truy cập ngày 7/6/2021.

[11] Advantages and disadvantage of Elasticsearch

https://www.javatpoint.com/advantages-and-disadvantages-of-elasticsearch,

truy cập ngày 7/6/2021.

[12] Ưu nhược điểm của Elasticsearch, So sánh Elasticsearch với MongoDB

https://stackjava.com/mongodb/uu-nhuoc-diem-cua-elasticsearch-so-sanh-elasticsearch-voi-mongodb.html, truy cập ngày 7/6/2021.