**MỤC LỤC**

[**MỤC LỤC** 1](#_Toc43534121)

[**I.** **Elasticsearch** 2](#_Toc43534122)

[**1.** **Khái niệm** 2](#_Toc43534123)

[**2.** **Cài đặt** 2](#_Toc43534124)

[**2.1** **Chạy Elasticsearch** 2](#_Toc43534125)

[**2.2** **Chạy Kibana** 5](#_Toc43534126)

[**3.** **Thao tác** 6](#_Toc43534127)

[**3.1** **Index cho data** 6](#_Toc43534128)

[**3.2** **Search** 7](#_Toc43534129)

[**II.** **Elasticsearch và Spring** 11](#_Toc43534130)

[**1.** **Tạo ứng dụng** 11](#_Toc43534131)

[**2.** **Thao tác với Elasticsearch** 12](#_Toc43534132)

[**2.1 Tạo data** 12](#_Toc43534133)

[**2.2 Get data** 13](#_Toc43534134)

[**2.3 Update document** 14](#_Toc43534135)

[**2.4 Upsert data** 15](#_Toc43534136)

[**2.5 Delete document** 16](#_Toc43534137)

[**2.6 Search document** 17](#_Toc43534138)

[**III.** **Logstash** 20](#_Toc43534139)

[**1.** **Khái niệm** 20](#_Toc43534140)

[**2.** **Cài đặt** 20](#_Toc43534141)

[**3.** **Thao tác với MySQL** 20](#_Toc43534142)

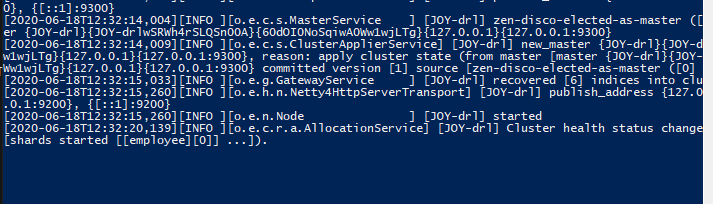
[**Migrate data từ MySQL vào ES** 23](#_Toc43534143)

[**IV.** **Tổng kết** 27](#_Toc43534144)

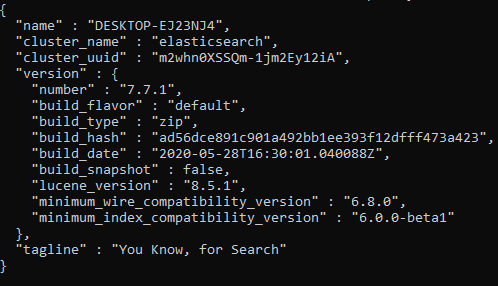
[**V.** **Tài liệu tham khảo** 27](#_Toc43534145)

1. **Elasticsearch**
   1. **Khái niệm**

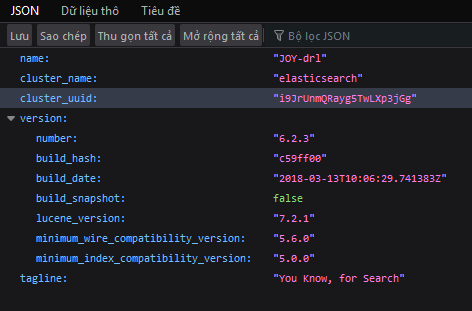
* Elasticsearch là một Search Engine chạy trên nền Lucene Apache, hoạt động như một web service với khả năng tìm kiếm dữ liệu nhanh chóng thông qua Restful.
* Hoạt động như một web service độc lập.
* Elasticsearch là một hệ thống phân tán có khả năng mở rộng theo chiều ngang (horizontal scalability).
  + **Ưu điểm:**
  + Do hoạt động như một web service và giao tiếp thông qua Restful nên ES không phụ thuộc vào Client hoặc hệ thống hiện tại viết bằng ngôn ngữ gì, giúp việc tích hợp vào hệ thống dễ dàng hơn.
  + Khả năng tìm kiếm dữ liệu nhanh, dù giá trị tìm kiếm không chính xác (sai chính tả, sai cú pháp) thì ES vẫn có thể tìm được kết quả gần đúng nhật.
  + Có khả năng phân tích và thống kê dữ liệu.
  + Sử dụng Query DSL (Domain Specific Language), cung cấp việc truy vấn phức tạo bằng JSON.
* Tham khảo thêm về Query DSL tại:
  + <https://viblo.asia/p/query-dsl-trong-elasticsearch-Eb85oJq2l2G>
  + <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/query-dsl.html>
  + **Nhược điểm:**
  + ES không có transactional do chủ yếu tập trung vào tìm kiếm và phân tích dữ liệu nên các request CRUD (tạo, cập nhật, xóa, …) có thể xảy ra lỗi dẫn đến sai về logic. Vì vậy ES thường được dung làm database search song song với database chính dung để lưu trữ dữ liệu.
  + Do được dùng làm database phụ nên ES yêu cầu phải migrate data từ database chính nên sẽ không thích hợp với những hệ thống cập nhật dữ liệu thường xuyên do sẽ mất nhiều thời gian để đánh index.
  1. **Cài đặt**
* Tải Elasticsearch: https://www.elastic.co/downloads/past-releases/elasticsearch-6-2-3
* Tải kibana: <https://www.elastic.co/fr/downloads/past-releases/kibana-6-2-3>
* Tải jdk 1.8.
  1. **Chạy Elasticsearch**
* Chạy elasticsearch.bat tại elasticsearch\bin\elasticsearch.bat



* Khi chạy elasticsearch sẽ đọc file cấu hình elasticsearch.yml tại: **config\elasticsearch.yml**
* Elasticsearch chạy trên port: **9200**.
* Kiểm tra bằng cách chạy lệnh curl -X GET "localhost:9200/?pretty"



* Hoặc vào đường dẫn <http://localhost:9200>



* **Có thể chạy Elasticsearch dưới dạng service chạy nền hoặc tự động start khi ứng dụng khởi động bằng cách chạy lệnh:**
* .\elasticsearch-service.bat install|remove|start|stop|manager
* **install**: cài đặt Elasticserach như một service.
* **remove**: xóa Elasticsearch nếu đã install và dừng nếu đang chạy.
* **start**: chạy Elasticsearch service nếu nó đã cài đặt.
* **stop**: dừng Elasticsearch service.
* **manager**: mở giao diện service.
* Khi chạy elasticsearch thì mặc định đã chạy một **Cluster** với một **Node** đơn lẻ. Có thể chạy nhiều elasticsearch, lúc này sẽ hình thành một cụm **cluster** với các **Node** giao tiếp với nhau.
* **Lưu ý:** Trong trường hợp chạy nhiều elasticsearch có cùng một cấu hình **cluster.name** có thể gây ra lỗi cho các **Node**.

#### **Master Node:**

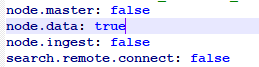
* Thông thường trong một hệ thống sẽ có một **Node** đóng vai trò là **Master Node** để quản lý việc tạo, xóa **index**, sắp xếp các shard, xóa **Node** trong **Cluster**. Trong trường hợp có sự thay đổi về trạng thái của cluster thì **Master Node** sẽ thông báo đến các node khác trong cluster.
* ***Lưu ý***: trong một hệ thống **chỉ có một** **Master Node** hoạt động tại một thời điểm cụ thể.
* Cấu hình **Master Node:**



* Mặc định khi chạy elasticseach thì **Node** đó vừa là **Master Node** vừa là **Data Node**.

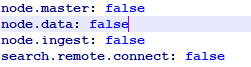
#### **Data Node:**

* Data Node là node giữ vai trò lưu trữ dự liệu và thực hiện các hoạt động tạo, cập nhật, tìm kiếm, đọc dữ liệu. Trong một cụm **Cluster** có thể có nhiều Data Node, khi một Data Node gặp sự cố thì cụm cluster vẫn có thể hoạt động.
* Cấu hình **Data Node**:



#### **Client Node:**

* Client Node đóng vai trò điều hướng các request đến Master Node hoặc Data Node tùy theo chức năng.
* Client Node không lưu trữ bất kỳ dữ liệu nào và cũng không thể trở thành Master Node.
* Cấu hình **Client Node**:

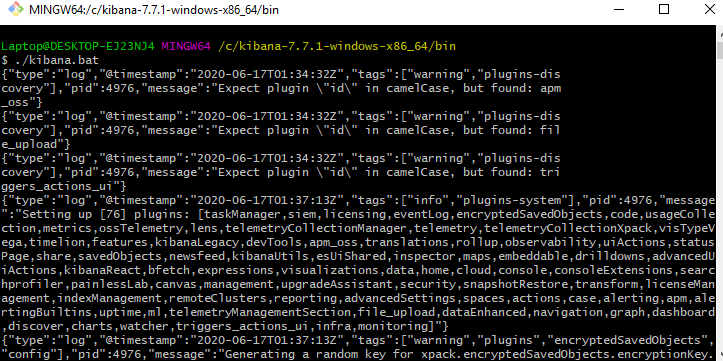


#### **Ingest Node:**

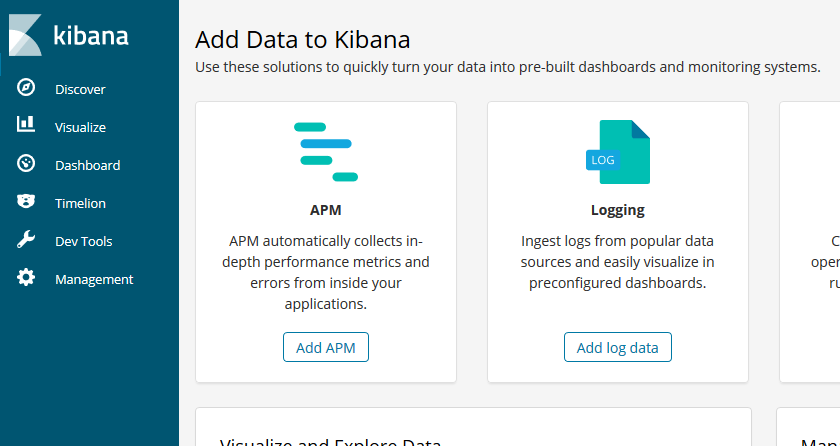
* Ingest Node hỗ trợ quá trình xử lý các document trước khi quá trình index băt đầu.
* Cấu hình **Ingest Node**:



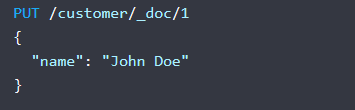
* 1. **Chạy Kibana**
* Chạy file kibana.bat trong /bin.



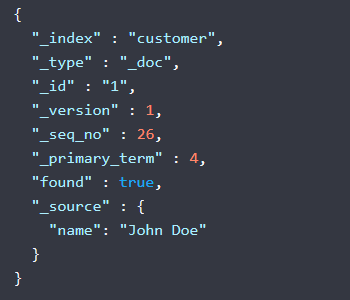
* Trong file kibana.yml trong folder config, trỏ đến instance của elasticsearch, mặc định là: elasticsearch.hosts: [<http://localhost:9200>]
* Truy cập vào **localhost:5601.**

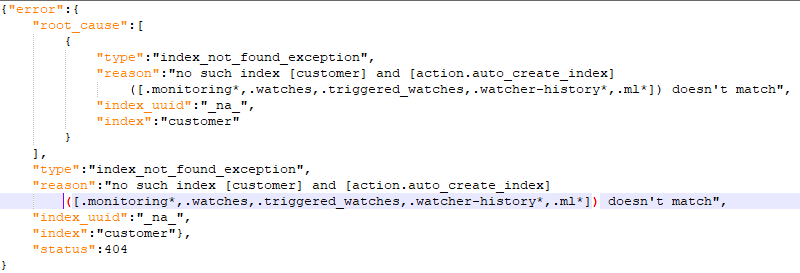


* 1. **Thao tác**
  2. **Index cho data**
* Đánh index cho data là hành động đưa data dạng JSON vào index của Elasticsearch.
* Cú pháp *PUT /[index]/[type]/[id]*:
  + **index**: tên của index.
  + **type**: loại document.
  + **id**: id của index



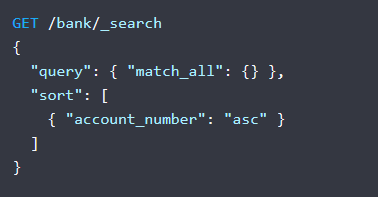
* Request này sẽ tạo ra một index **‘customer’** nếu index này chưa tồn tại, với **id = 1**.
* Chạy **curl -X GET "localhost:9200/customer/\_doc/1?pretty"** để kiểm tra.



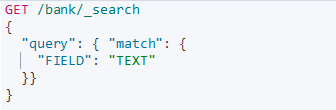
* **Lưu ý:** Trong trường hợp xảy ra lỗi
* 
* Lỗi này xảy ra do thuộc tính **action.auto\_create\_index** chưa được **enable**.
* Vào file **elasticsearch.yml** và thêm dòng:
* **action.auto\_create\_index: .monitoring\*,.watches,.triggered\_watches,.watcher-history\*,.ml\***
  1. **Search**
* Truy cập: <https://raw.githubusercontent.com/elastic/elasticsearch/master/docs/src/test/resources/accounts.json> và copy nội dung.
* Trong dev tool của kibana chạy:



* Lấy danh sách bank:

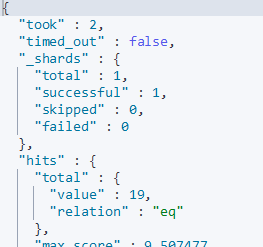


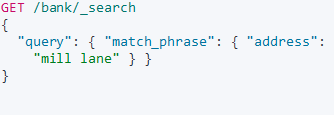
* Request trên trả về danh tất cả document của index ‘**bank’** và sắp xếp tăng dần theo **‘account\_number’**.
* **took**: thời gian Elasticsearch dùng để thực hiện truy vấn (milisecond).
* **max\_score**: số điểm tối đa cho mức độ chính xác của kết quả.
* **time\_out**: request có timeout hay không.
* **hits.total.value**: bao nhiêu kết quả phù hợp với điều kiện search.
* **hits.sort**: vị trí của document trong danh sách kết quả.
* **hits.\_score**: số điểm đại diện cho mức độ chính xác của kết quả, điểm càng cao thì kết quả càng chính xác.
* Để tìm kiếm theo giá trị của thuộc tính:

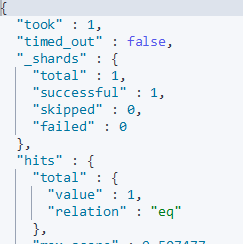


* **match**: sẽ tìm kết quả giống với giá trị mong muốn.
* **FIELD**: tên thuộc tính của document muốn tìm kiếm.
* **TEXT**: giá trị mong muốn của thuộc tính.

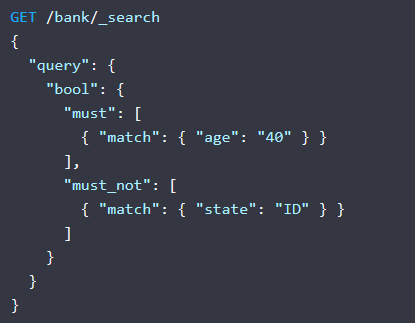




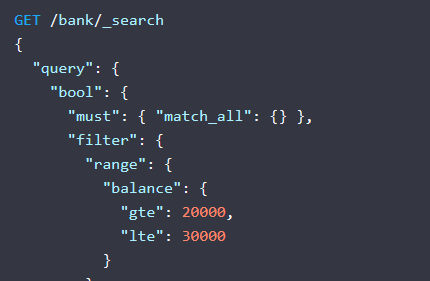
* hits.total.value = 19: có 19 kết quả gần giống với giá trị search.
* Trong trường hợp giá trị mong muốn là một câu thì sử dụng **match\_phase** thay cho **match**.
* 



* hits.total.value = 1: chỉ có 1 kết quả đúng với giá trị search.
* Để xây dựng câu truy vấn phức tạp hơn, có thể sử dụng **bool** để kết hợp nhiều câu truy vấn.



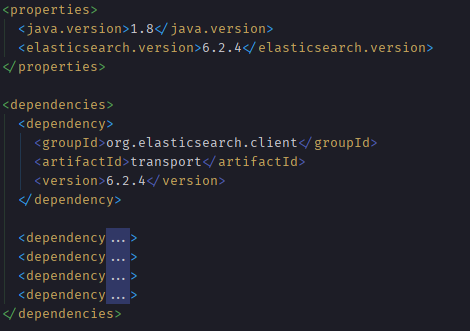
* Request trên sẽ tra về kết quả các tài khỏan có khách hàng 40 tuổi và nhưng không bao gồm những người sống ở Idaho (ID).



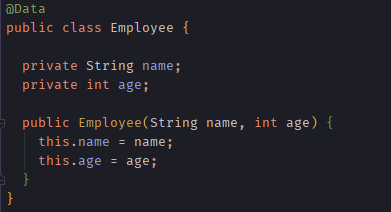
* Request trên trả về tất cả các tài khoản có balance trong khoảng từ 2000 – 30000.
* **must**, **should, must not** và **filter** được gọi là các mệnh đề truy vấn (query clause).
* **must**: điều kiện truy vấn phải có trong các kết quả và sẽ ảnh hưởng đến điểm số.
* **should**: điều kiện try vấn nên có trong kết quả search.
* **must\_not**: điều kiện truy vấn bắt buộc không được có trong kết quả. **must\_not** được tính như một filter.
* **filter**: bắt buộc phải có trong kết quả truy vấn, tuy nhiên khác với **must, filter** không ảnh hưởng đến điểm số của kết quả.
* Tham khảo thêm tại: <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/docs.html>

1. **Elasticsearch và Spring**
2. **Tạo ứng dụng**

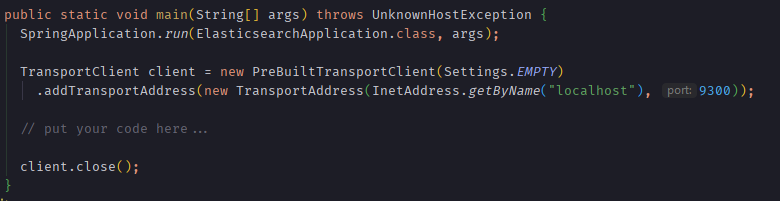
* Tạo ứng dụng Spring.
* Thêm dependency và elasticsearch version:



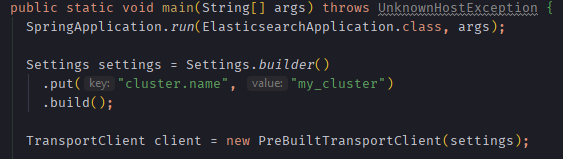
* Tạo class **Employee.java**:



* Thiết lập kết nối đến Elasticsearch:



* Cluster name mặc định là **’elasticsearch’**. Trong trường hợp cluster.name không phải **’elasticsearch’** thì sẽ thay thế bằng:

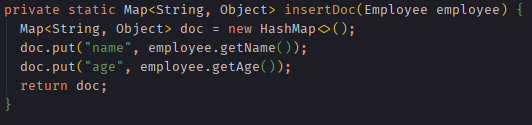


* **my\_cluster**: tên cluster do người dùng định nghĩa trong **elasticsearch.yml.**

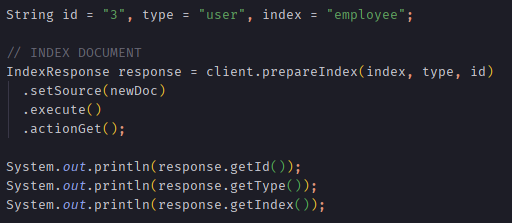
## **Thao tác với Elasticsearch**

### **2.1 Tạo data**

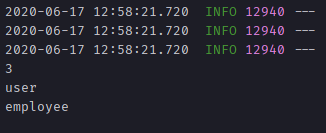




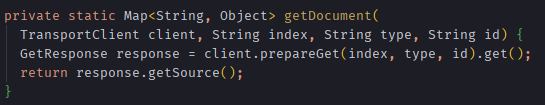
* Tạo **index** cho document, trong trường hợp này là đối tượng **newDoc:**

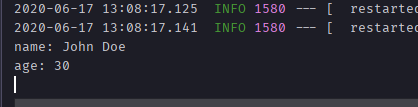


* Chạy thử:

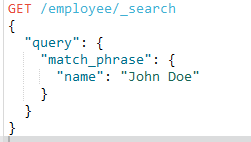


### **2.2 Get data**



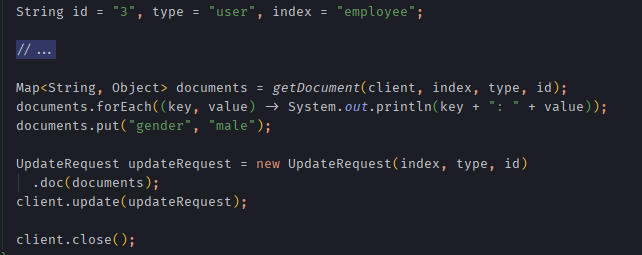


* Trên giao diện Dev tool của Kibana chạy :

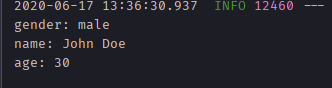




### **2.3 Update document**

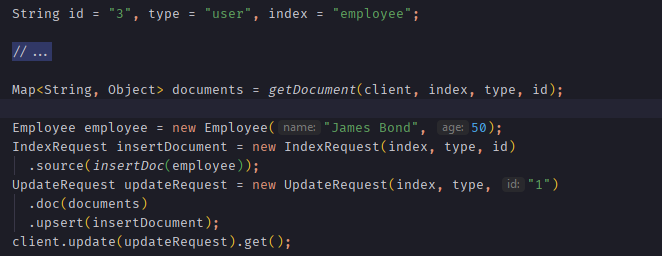


* Thêm giá trị gender vào documents.
* UpdateRequest sẽ kết hợp document mới tìm được với document có sẵn, document mới có giá trị “**gender**” còn document cũ không có nên nó sẽ kết hợp cả 2.
* Chạy lại phương thức getDocument sẽ có kết quả:

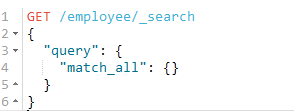


### **2.4 Upsert data**

* **Upsert** được dùng trong trường hợp câp nhật data nhưng không tìm thấy thì data mới sẽ được thêm vào.

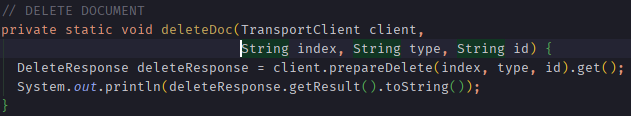


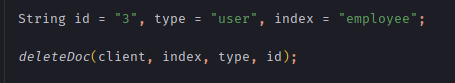
* Tạo đối tượng Employee mới.
* IndexRequest là giá trị sẽ được insert vào.
* UpdateRequest sẽ tìm document với index = employee, type = user và id = 1. Tuy nhiên chỉ có document có id = 3 nên nội dung bên trong **upsert()** sẽ được chạy.
* Sử dụng Kibana để kiểm tra:



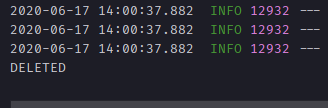


### **2.5 Delete document**

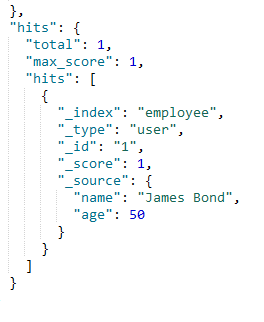




* Hàm **deleteDoc()** sẽ xóa document có id = 3.

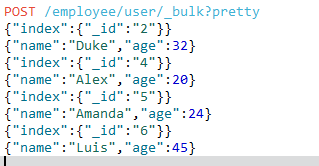


* Dùng kibana để kiểm tra:

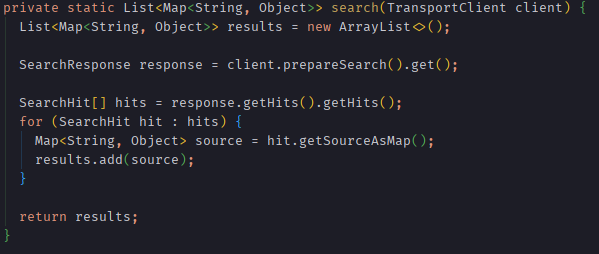


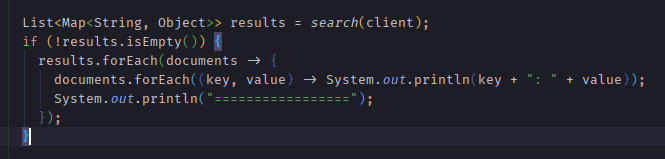
### **2.6 Search document**

* Dùng kibana để add thêm data:

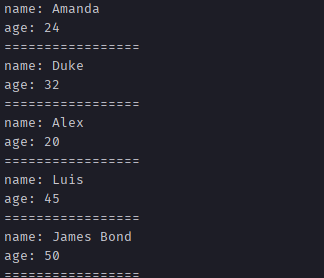


* Lấy danh sách tất cả document:





* Kết quả:



* Chạy bằng Kibana sẽ có kết quả:



* Tham khảo thêm tại: <https://www.elastic.co/guide/en/elasticsearch/client/java-api/6.2/index.html>
* Source code demo: <https://github.com/BinhAnVn96/Elasticsearch-java.git>

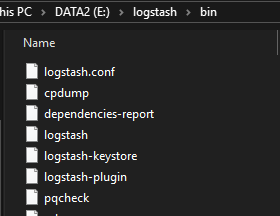
1. **Logstash**
2. **Khái niệm**

* **Logstash** là một công cụ mã nguồn mở dùng để thu thập dữ liệu (theo thời gian thực) từ nhiều nguồn khác nhau, loại bỏ và đồng hóa dữ liệu với nhau.
* Do ES dùng làm công cụ tìm kiếm và phân tích dữ liệu trong hệ thống nên **Logstash** được tích hợp dùng làm công cụ thu thập data từ database và migrate vào ES.
* **Logstash** có 3 thành phần:
  + **INPUT:** đây là đầu vào dùng để thu thập data từ syslog, database và nhiều nguồn khác do người dùng cấu hình.
  + **FILTER:** sau khi dữ liệu thu thập vào Server Logstash thì filter cung cấp một lượng lớn bộ lọc để người dùng có thể lấy những data cần thiết.
  + **OUTPUT:** đầu ra của dữ liệu, Logstash hỗ trợ nhiều các đích đến như HTTP, TCP/UDP, email, … Ta có thể tích hợp Logstash vào các công cụ hổ trợ tính toán, biểu đồ,…
* **Tham khảo thêm về cơ chế hoạt động của Logstash tại:**

<https://blog.cloud365.vn/logging/ELK-part-6-Tim-hieu-ve-logstash/>

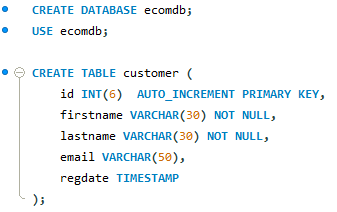
1. **Cài đặt**

* Cài đặt MySQL.
* Tải Logstash: <https://www.elastic.co/downloads/logstash>
* Tạo file **logstash.conf** tại thư mục /logstash/bin. Hiện tại chỉ cần tạo một file rỗng là được.



1. **Thao tác với MySQL**

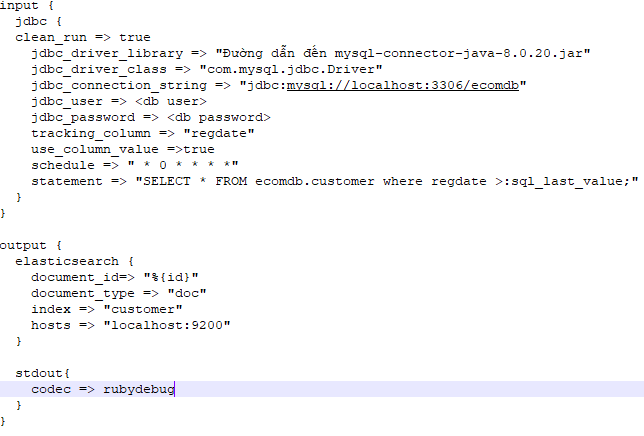
* Tạo database ecomdb và table customer:



* Insert data vào bảng customer:



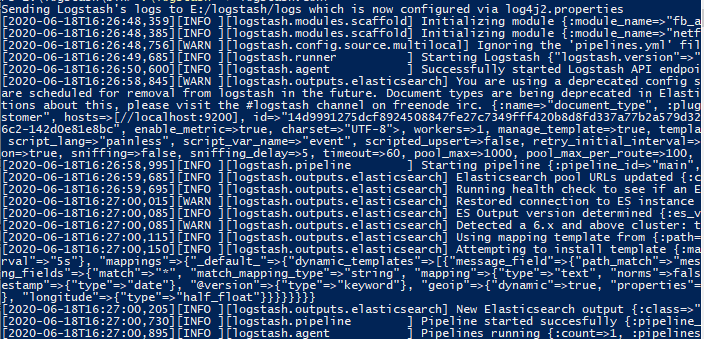
* Mở file **logstash.conf** và thêm cấu hình để kết nối đến MySQL.

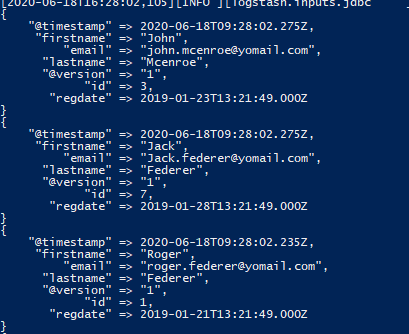


* **jdbc\_driver\_library**: đường dẫn đến MySQL Connector J, thông thường khi cài đặt MySQL sẽ có kèm Connector J, trường hợp không có thì tải tại: <https://dev.mysql.com/downloads/connector/j/5.1.html>
* **jdbc\_connection\_string**: thiết lập kết nối đến MySQL.
* **jdbc\_user:** user của database, thường là “root”.
* **jdbc\_password**: mật khẩu db.
* **tracking\_column**: giá trị mà LS sẽ dùng để track dữ liệu, dựa vào trường này LS sẽ thu thập data mới nhất. Giá trị track nên sử dụng giá trị thời gian như ngày cập nhật của dữ liệu để dễ dàng lấy được giá trị mới nhất.
* **use\_column\_value**:
  + **true**: sẽ lấy giá trị của track\_column để thay thế cho **:sql\_last\_value** ở statement.
  + **false: :sql\_last\_value** sẽ là giá trị ở lần chạy cuối cùng của statement.
* **Schedule:** là thời gian chạy statement, sử dụng Cron Schedule. Tìm hiều thêm về Cron: <https://www.freeformatter.com/cron-expression-generator-quartz.html>
* **statement:** sẽ được chạy để LS thu thập kết quả. Trong trường hợp này LS sẽ thu thập tất cả data từ bảng customer.
* **index:** tên của index khi migrate từ database, trường hợp này là customer.
* **document\_id:** là id của document cho index customer. **%{id}** giá trị id của customer sẽ được gán cho id của index.
  + Sử dụng id của đối tượng làm id của index sẽ tránh được việc data bị duplicate khi thực hiện migrate dữ liệu từ MySQL vào ES.
* **hosts**: đầu ra của dữ liệu, ở đây là ES.
* **Lưu ý:** nếu sử dụng thuộc tính trong logstash.conf sẽ khác nếu sử dụng phiên bản 7.0 trở lên. Tham khảo thêm tại: <https://www.elastic.co/guide/en/logstash/7.x/config-examples.html>

### **Migrate data từ MySQL vào ES**

* Vào nơi chứa file logstash.conf và chạy lệnh **.\logstash -f logstash.conf**.



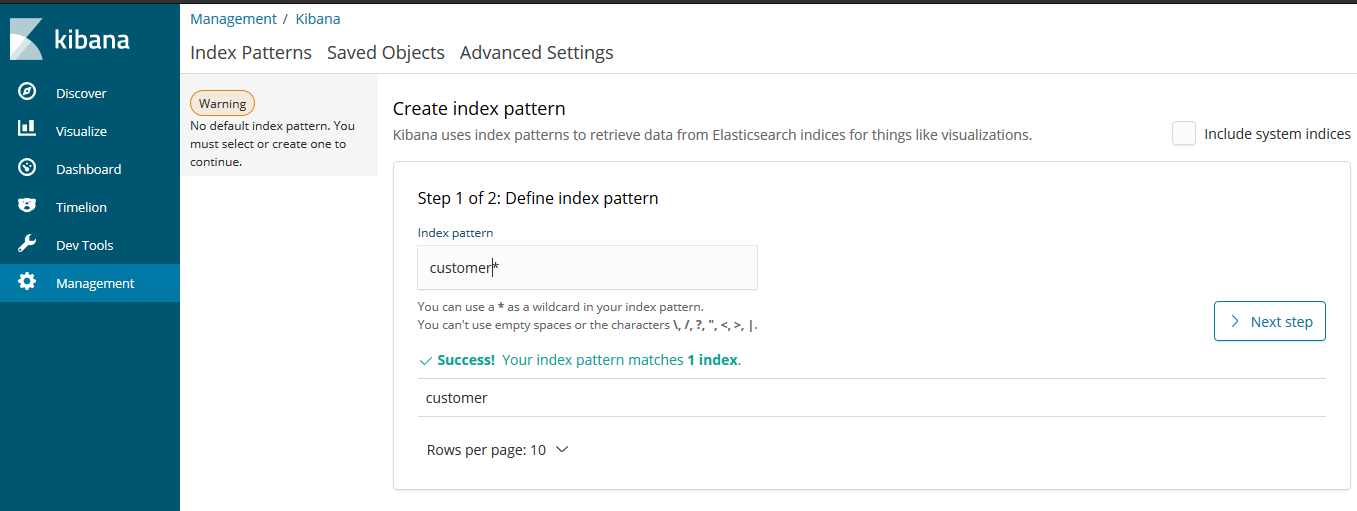


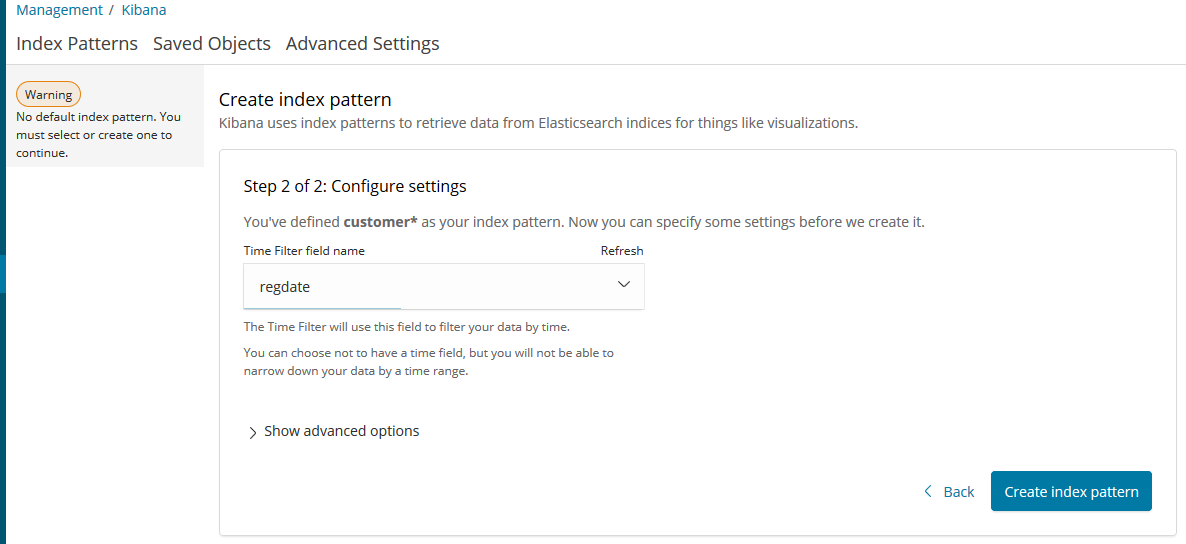
* Sử dụng Dev-tool kibana để kiểm tra





* Vào Management của Kibana, tạo một index pattern

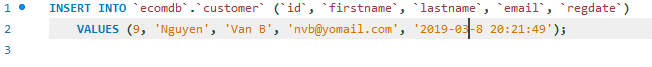




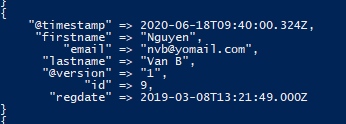
* Vào tab Discover và chọn index vừa tạo sẽ thấy được data đã import vào ES.



* Insert thêm data vào bảng customer.



* Tùy thuộc vào thời gian của schedule trong config mà LS sẽ migrate data từ database vào ES.



* **Lưu ý:** Để data trong ES luôn được cập nhật nên cấu hình thời gian chạy của schedule ngắn, có thể vài giây cập nhật 1 lần hoặc chạy mỗi giây để cập nhật liên tục. tùy thuộc vào hệ thống cũng như mục đích sử dụng.
* Nếu data từ database đã bị xóa thì LS không nhận biết được và không thể xóa được data trên ES. Trong trường hợp đó trong database nên có một cột đánh dấu (vd: isDeleted) sau đó mới thực hiện migrate vào ES để có thể dựa vào giá trị này để tìm kiếm và thống kê dữ liệu.

1. **Tổng kết**

* Elasticsearch là một engine hỗ trợ tìm kiếm dữ liệu nhanh chóng, hoạt động như một webservice nên ES có thể dễ dàng tích hợp vào bết kỳ hệ thống nào.
* Bên cạnh việc tìm kiếm, ES còn được sử dụng trong các hệ thống phân tích dữ liệu nhờ khả năng tìm kiếm và thống kê dữ liệu nhanh chóng.
* ES là một dạng database NoSQL và được thiết kế với nhiều Node chạy cùng lúc, từ đó tránh được Single Point of Failure.
* Tuy nhiên bên cạnh những ưu điểm thì ES cũng có những hạn chế như không có transaction, từ đó dẫn đến khả năng có thể xảy ra lỗi trong quá trình tạo, cập nhật dữ liệu.
* Để khắc phục điểm yếu này, ES sẽ được sử dụng như một search engine chạy song song với database chính, kết nối với nhau thông qua Logstash.

1. **Tài liệu tham khảo**

* <https://viblo.asia/p/elasticsearch-la-gi-1Je5E8RmlnL#_elasticsearch-la-gi--1>
* <https://viblo.asia/p/query-dsl-trong-elasticsearch-Eb85oJq2l2G>
* <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/docs.html>
* <https://www.elastic.co/guide/en/logstash/6.2/introduction.html>
* <https://blog.cloud365.vn/logging/ELK-part-6-Tim-hieu-ve-logstash/>
* <https://www.javainuse.com/spring/springboot-microservice-elk>
* <https://qbox.io/blog/migrating-mysql-data-into-elasticsearch-using-logstash>
* <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/query-dsl.html>