Elasticsearch

Elasticsearch là một công cụ tìm kiếm dựa trên nền tảng Apache 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 phát triển bằng Java và được phát hành dạng nguồn mở theo giấy phép Apache. (Theo wikipedia)

Tóm lại :

* Elasticsearch là một search engine.
* Elasticsearch được kế thừa từ Lucene Apache
* Elasticsearch thực chất hoặt động như 1 web server, có khả năng tìm kiếm nhanh chóng (near realtime) thông qua giao thức RESTful
* Elasticsearch có khả năng phân tích và thống kê dữ liệu
* Elasticsearch chạy trên server riêng và đồng thời giao tiếp thông qua RESTful do vậy nên nó không phụ thuộc vào client viết bằng gì hay hệ thống hiện tại của bạn viết bằng gì. Nên việc tích hợp nó vào hệ thống bạn là dễ dàng, bạn chỉ cần gửi request http lên là nó trả về kết quả.
* Elasticsearch là 1 hệ thống phân tán và có khả năng mở rộng tuyệt vời (horizontal scalability). Lắp thêm node cho nó là nó tự động auto mở rộng cho bạn.
* Elasticsearch là 1 open source được phát triển bằng Java

**Các khái niệm cần biết**

## 1, Document

Document là một JSON object với một số dữ liệu. Đây là basic information unit trong ES. Hiểu 1 cách cơ bản thì đây là đơn vị nhỏ nhất để lưu trữ dữ liệu trong Elasticsearch.

## 2, Index.

* Index có lẽ là 1 khái niệm quá quen thuộc đối với các anh em dùng Mysql rồi. Khi đọc đến đây có lẽ ae đã thừa hiểu chức năng của index là gì rồi. Tuy nhiên nếu các bạn nghĩ rằng index trong ES hoàn toàn giống trong Mysql thì các bạn nhầm rồi nhé !

Trong Elasticsearch , sử dụng một cấu trúc được gọi là inverted index . Nó được thiết kế để cho phép tìm kiếm full-text search. Cách thức của nó khá đơn giản, các văn bản được phân tách ra thành từng từ có nghĩa sau đó sẽ đk map xem thuộc văn bản nào. Khi search tùy thuộc vào loại search sẽ đưa ra kết quả cụ thể.

* nó tương đương với khái niệm một database

## 3, Shard

* Shard là đối tượng của Lucene , là tập con các documents của 1 Index. Một Index có thể được chia thành nhiều shard.
* Mỗi node bao gồm nhiều Shard . Chính vì thế Shard mà là đối tượng nhỏ nhất, hoạt động ở mức thấp nhất, đóng vai trò lưu trữ dữ liệu.
* Chúng ta gần như không bao giờ làm việc trực tiếp với các Shard vì Elasticsearch đã support toàn bộ việc giao tiếp cũng như tự động thay đổi các Shard khi cần thiết.
* Có 2 loại Shard là : primary shard và replica shard.

### 3.1 : Primary Shard

* Primary Shard là sẽ lưu trữ dữ liệu và đánh index . Sau khi đánh xong dữ liệu sẽ được vận chuyển tới các Replica Shard.
* Mặc định của Elasticsearch là mỗi index sẽ có 5 Primary shard và với mỗi Primary shard thì sẽ đi kèm với 1 Replica Shard.

### 3.2 : Replica Shard

* Replica Shard đúng như cái tên của nó, nó là nơi lưu trữ dữ liệu nhân bản của Primary Shard
* Replica Shard có vai trò đảm bảo tính toàn vẹn của dữ liệu khi Primary Shard xảy ra vấn đề.
* Ngoài ra Replica Shard có thể giúp tăng cường tốc độ tìm kiếm vì chúng ta có thể setup lượng Replica Shard nhiều hơn mặc định của ES

Để hiểu hơn về Primary Shard hay Replica Shard mọi người có thể vào [đây](https://viblo.asia/p/elasticsearch-distributed-search-ZnbRlr6lG2Xo#replica-shard-6) để tìm hiểu rõ hơn. Trong bài này anh Long đã viết khá chi tiết về các khái niệm cũng như cách thức hoạt động lưu trữ và tìm kiểm của ES nên mình xin phép không nói thêm gì về phần này.

## 4, Node

* Là trung tâm hoạt động của Elasticsearch. Là nơi lưu trữ dữ liễu ,tham gia thực hiện đánh index cúa cluster cũng như thực hiện các thao tác tìm kiếm
* Mỗi node được định danh bằng 1 **unique name**

## 5, Cluster

* Tập hợp các nodes hoạt động cùng với nhau, chia sẽ cùng thuộc tính cluster.name. Chính vì thế Cluster sẽ được xác định bằng 1 'unique name'. Việc định danh các cluster trùng tên sẽ gây nên lỗi cho các node vì vậy khi setup các bạn cần hết sức chú ý điểm này
* Mỗi cluster có một node chính (master), được lựa chọn một cách tự động và có thể thay thế nếu sự cố xảy ra. Một cluster có thể gồm 1 hoặc nhiều nodes. Các nodes có thể hoạt động trên cùng 1 server . Tuy nhiên trong thực tế , một cluster sẽ gồm nhiều nodes hoạt động trên các server khác nhau để đảm bảo nếu 1 server gặp sự cố thì server khác (node khác) có thể hoạt động đầy đủ chức năng so với khi có 2 servers. Các node có thể tìm thấy nhau để hoạt động trên cùng 1 cluster qua giao thức unicast.

**6 Type** : hay còn gọi là Mapping Type: là 1 tập các document cùng loại (tương đương với khái niệm collection trong MongoDB hay khái niệm table trong database SQL)

Chức năng chính của Cluster đó chính là quyết định xem shards nào được phân bổ cho node nào và khi nào thì di chuyển các Cluster để cân bằng lại Cluster

# Ưu nhược điểm của ES

## Ưu điểm

* Tìm kiếm dữ liệu rất nhanh chóng, mạnh mẽ dựa trên Apache Lucene ( near-realtime searching)
* Có khả năng phân tích dữ liệu (Analysis data)
* Khả năng mở rộng theo chiều ngang tuyệt “vòi”
* Hỗ trợ tìm kiếm mờ (fuzzy), tức là từ khóa tìm kiếm có thể bị sai lỗi chính tả hay không đúng cú pháp thì vẫn có khả năng elasticsearch trả về kết quả tốt.
* Hỗ trợ Structured Query DSL (Domain-Specific Language ), cung cấp việc đặc tả những câu truy vấn phức tạp một cách cụ thể và rõ ràng bằng JSON.Các bạn có thể tìm hiểu thêm tại [đây](https://viblo.asia/p/query-dsl-trong-elasticsearch-Eb85oJq2l2G)
* Hỗ trợ nhiều Elasticsearc client như Java, PhP, Javascript, Ruby, .NET, Python

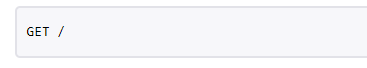
## Nhược điểm

* Elasticsearch được thiết kế cho mục đích search, do vậy với những nhiệm vụ khác ngoài search như CRUD thì elastic kém thế hơn so với những database khác như Mongodb, Mysql …. Do vậy người ta ít khi dùng elasticsearch làm database chính, mà thường kết hợp nó với 1 database khác.
* Trong elasticsearch không có khái niệm database transaction , tức là nó sẽ không đảm bảo được toàn vẹn dữ liệu trong các hoạt độngInsert, Update, Delete.Tức khi chúng ta thực hiện thay đổi nhiều bản ghi nếu xảy ra lỗi thì sẽ làm cho logic của mình bị sai hay dẫn tới mất mát dữ liệu. Đây cũng là 1 phần khiến elasticsearch không nên là database chính.
* Không thích hợp với những hệ thống thường xuyên cập nhật dữ liệu. Sẽ rất tốn kém cho việc đánh index dữ liệu.

## API trong Elasticsearch – Đọc/Ghi dữ liệu với Elasticsearch

Để định dạng dữ liệu trả về của mỗi API, các bạn thêm tham số ?format=yaml vào cuối mỗi API, định dạng trả về có thể là text, json, yaml…

**API Hiển thị thông tin version, cluster**



**API Hiển thị chi tiết thông tin node:**



Hoặc

GET /\_cat/master

## API liệt kê, hiển thị các Index trong Elasticsearch

GET /\_cat/indices

Them /\_cat/indices?format=json để hiển thị dữ liệu dang json

# API tạo Index trong Elasticsearch

Để tạo Index trong Elasticsearch ta dùng method PUT với url /

* PUT /{index\_name}

# API xóa Index trong Elasticsearch

Để tạo Index trong Elasticsearch ta dùng method DELETE với url /

* DELETE /index\_name

Ví dụ xóa index "user"

* DELETE http://localhost:9200/user

## Xóa nhiều index trong cùng 1 request

Method DELETE localhost:9200/person,player

## Xóa tất cả Index trong 1 request

DELETE /\_all

Thẻ \_mapping dùng để lấy danh sách mapping như có những index nào, index gồm những type nào, type gồm những fields nào…

**Hiển thị danh sách tất cả các type, mapping trong Elasticsearch**

* GET /\_all/\_mapping
* GET /\_mapping

**Hiển thị danh sách các type, mapping trong một index**

* GET /index\_name/\_mapping

**Hiển thị mapping, các fields trong 1 type**

* GET /index\_name/type\_name/\_mapping

Ví dụ: hiển thị các type, mapping trong index stackjava

* GET localhost:9200/stackjava/\_mapping

## API tạo document Elasticsearch.

Có 2 cách để tạo document trong Elasticsearch:

**Cách 1: Dùng API POST**

* POST /index\_name/type\_name/id
* {
* "field" : "value"
* }

Trong đó:

* index\_name và type\_name là tên của Index và tên của Type, nếu chưa tồn tại thì sẽ được tự động tạo
* id là id của document được tạo, nếu không truyền id thì nó sẽ tự động tạo ra 1 chuỗi để làm id.
* Trường hợp index, type và id truyền vào đã tồn tại thì nó sẽ tương đương với câu lệnh update.

**Cách 2: Dùng API PUT**

Dùng method PUT với tham số op\_type=create

* PUT /index\_name/type\_name/id?op\_type=create
* { ... }

Hoặc:

* PUT /index\_name/type\_name/id/create
* { ... }

Tuy nhiên với cách này, nếu bộ  index\_name, type\_name và id đã tồn tại thì nó sẽ báo lỗi

document\_already\_exists\_exception

## API get document Elasticsearch.

* GET /index\_name/type\_name/id

Ví dụ:

Lấy document có id = 1 trong type player của index stackjava

Tuy nhiên trong dữ liệu trả về có khá nhiều field không cần thiết như index name, version…  
Để lấy data của mỗi document ta dùng thêm thẻ \_source vào sau url

* GET /index\_name/type\_name/id/\_source

Trường hợp chỉ muốn lấy 1 số field trong document ta dùng tham số \_source=field1,field2

## API Search document Elasticsearch

Để search document trong Elasticsearch ta dùng method GET với path \_search

**Tìm tất cả document trong tất cả các index**

* GET /\_search

Hoặc

* GET /\_all/\_search

**Tìm tất cả document trong 1 index**

* GET /index\_name/\_search

**Tìm tất cả document trong 1 type**

* GET /index\_name/type\_name/\_search

## Search theo tham số

Để search dữ liệu theo tham số đầu vào ta dùng biến q=tham\_so

Ví dụ tìm tất cả các document có chứa từ man

* GET /\_search?q=man

Trường hợp muốn search theo field chỉ rõ thì ta dùng ?q=field\_name:query

Ví dụ tìm tất cả các document có chứa từ man trong field name

* GET /\_search?q=name:man

Trường hợp truy vấn theo nhiều điều kiện phức tạp:

Ví dụ tìm tất cả các document thỏa mãn các điều kiện sau:

* có từ man hoặc woman trong field name
* có từ marvel trong field studio

Câu truy vấn sẽ là:

* +name:(man woman) +studio:(marvel)

Tuy nhiên trong url sẽ không hiểu các ký tự như dấu cách, + ( )

Ta cần phải chuyển nó sang dạng URL encode:

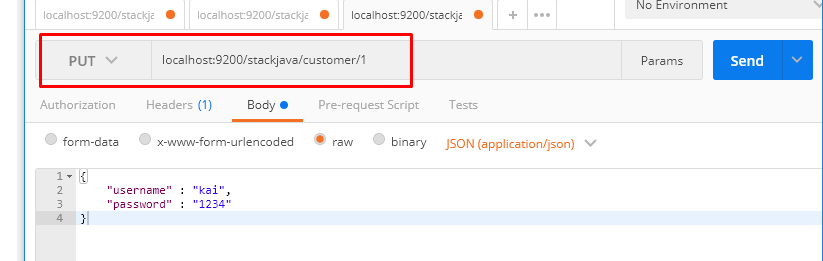
* %2Bname%3A%28man+woman%29+%2Bstudio%3A%28marvel%29

## API Update document Elasticsearch.

Có 2 cách để cập nhật, chỉnh sửa document trong Elasticsearch.

**Cách 1: Modifiy document bằng method PUT**

* PUT /index\_name/type\_name/id
* {
* "field": "data"
* }



**Cách 2: Update document bằng method POST**

* POST /index\_name/type\_name/id
* {
* "field": "data"
* }

Update document bằng method POST có tác dụng tương đương với method PUT

Tuy nhiên bản chất của việc update document bằng method POST khác với method PUT ở chỗ là method PUT sẽ replace / sửa lại dữ liệu trên document đã có còn method POST là xóa document cũ đi (nếu có tồn tại) và tạo 1 document mới.

## Xóa dữ liệu trong Elasticsearch

Có 2 cách xóa dữ liệu trong Elasticsearch là dùng method DELETE hoặc dùng method POST với thẻ \_delete\_by\_query

**Cách 1: Sử dụng DELETE API**

* DELETE /index\_name/type\_name/id

**Mongoose**

Model được định nghĩa thông qua giao diện Schema. Schema định nghĩa các trường được lưu trong mỗi document đi kèm với điều kiện xác thực và giá trị mặc định cho chúng.

Schema sau đó được "biên dịch" thành mô hình qua phương thức mongoose.model(). Một khi đã có mô hình thì bạn có thể dùng nó để tìm, thêm, sửa, và xoá các đối tượng cùng kiểu.