ELASTICSEARCH

Đặng Quang Hà - 01/10/2021

Mục lục

Phần 1: Giới thiệu tổng quan

- Elasticsearch là gì?
- Elasticsearch hoạt động như thế nào?
- Tại sao nên sử dụng Elasticsearch?
- Ưu nhược điểm của ES

Phần 1: Giới thiệu tổng quan

Elasticsearch là gì

- Elasticsearch là một search engine

- Elasticsearch thực chất hoạt 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

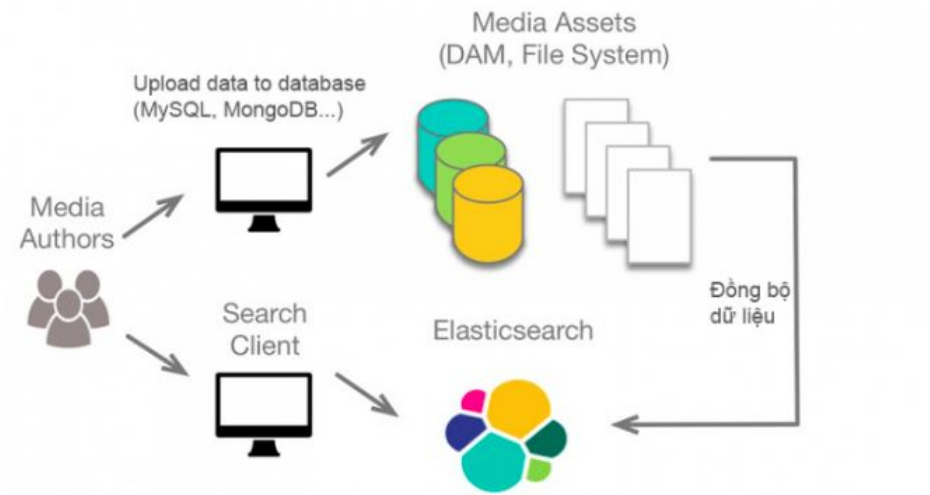
- Không phụ thuộc vào client viết bằng gì hay hệ thống hiện tại của bạn viết bằng gì, bạn chỉ cần gửi request http lên là nó trả về kết quả.

Phần 1: Giới thiệu tổng quan

Elasticsearch hoạt động như thế nào?

Các dữ liệu được người dùng tải lên sẽ lưu vào database sau đấy đồng bộ hóa sang Elasticsearch.

Khi người dùng tìm kiếm thì sẽ tìm kiếm trên Elasticsearch, tốc độ vừa nhanh, vừa giảm tải cho database.



Phần 1: Giới thiệu tổng quan

Tại sao nên sử dụng Elasticsearch?

- Tại sao phải dùng ES trong khi tìm kiếm văn bản có thể sử dụng câu lệnh LIKE SQL cũng được?

==> Nếu search bằng truy vấn LIKE “%one%” thì kết quả sẽ chỉ cần chứa “one” là ra. Ví dụ: “phone”, “zone”, “money”, “alone” … nói chung sẽ là 1 list kết quả không mong muốn.

- Còn search bằng ES thì gõ “one” sẽ chỉ có “one” được trả về mà thôi. Truy vấn LIKE không thể truy vấn từ có dấu. Ví dụ: từ khoá có dấu là “có”, nếu truy vấn LIKE chỉ gõ “co” thì sẽ không trả về được chính xác kết quả. Về Perfomance thì ES sẽ là tốt hơn, truy vấn LIKE sẽ tìm kiếm đơn thuần toàn văn bản không sử dụng index, nghĩa là tập dữ liệu càng lớn thì tìm kiếm càng lâu, trong khi ES lại “đánh index” cho các trường được chọn để tìm kiếm.

Phần 1: Giới thiệu tổng quan

Ưu nhược điểm của ES

ƯU ĐIỂM	NHƯỢC ĐIỂM
- Tìm kiếm dữ liệu cực nhanh. - Phân tích và thống kê dữ liệu search data theo vị trí địa lý. - Tìm kiếm full text. - Sử dụng RESTful nên không phụ thuộc vào client viết bằng gì. - Hệ thống phân tán và có khả năng mở rộng tuyệt vời, chỉ cần thêm node. - Là mã nguồn mở.	- Không dùng làm database chính. - Không phù hợp với dự án nhỏ. - Không phù hợp hệ thống thường xuyên cập nhật dữ liệu. - Không cung cấp bất kỳ tính năng nào cho việc xác thực và phân quyền, kém bảo mật.

Phần 2: Hướng dẫn cài đặt và làm quen

- Từ kho lưu trữ trên Linux hoặc MacOS (tar.gz)
- Với .zip trên Windows
- Với Debian Package
- Với RPM
- Với windows msi
- **Với docker**
- Trên macOS với Homebrew

Phần 2: Hướng dẫn cài đặt và làm quen

https://github.com/dangquangha/elasticsearch-training

sudo sysctl -w vm.max_map_count=262144

docker-compose up

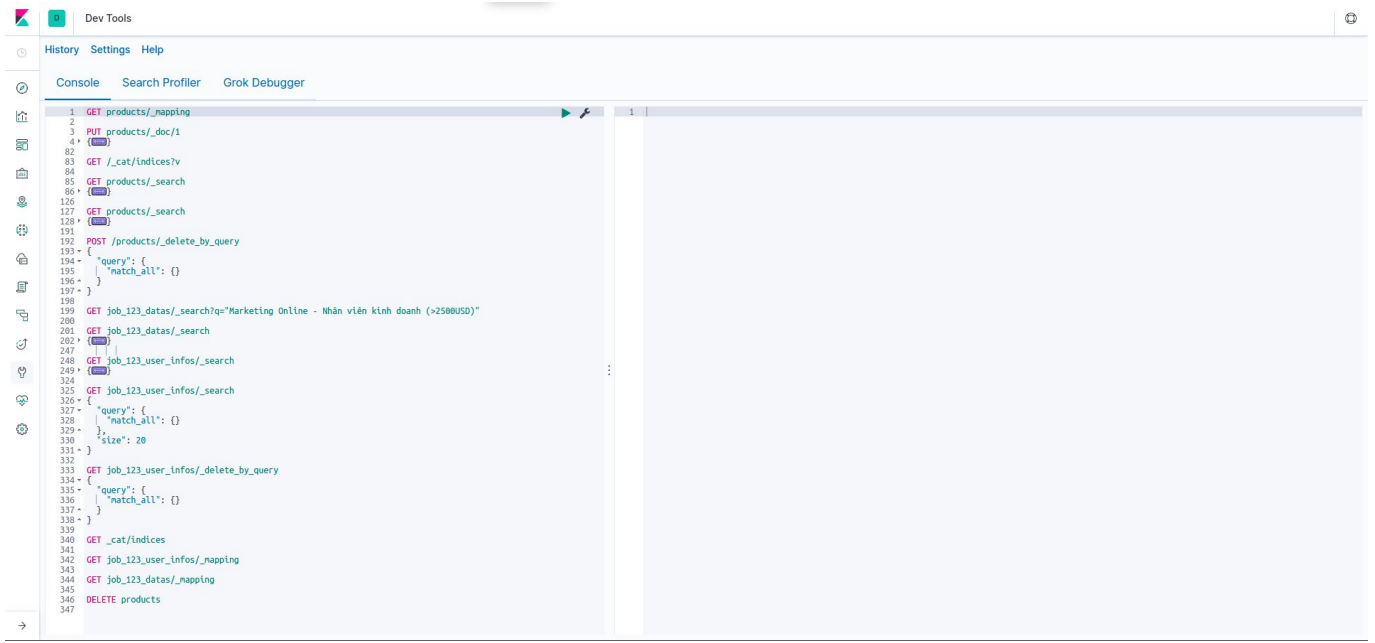
Truy cập kibana
http://localhost:5601/

docker-compose.yml

```yaml
version: '3'

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.0.1
    container_name: elasticsearch
    environment:
      - node.name=elasticsearch
      - cluster.name=datasearch
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - cluster.initial_master_nodes=elasticsearch
    ulimits:
      memlock:
        soft: -1
        hard: -1
    ports:
      - "9200:9200"
    volumes:
      - esdata:/usr/share/elasticsearch/data

  kibana:
    image: docker.elastic.co/kibana/kibana:7.0.1
    container_name: kibana
    ports:
      - "5601:5601"

volumes:
  esdata:
    driver_opts:
      device: /home/quangha/Desktop/ES/data
      o: bind
      type: none

```

Phần 2: Hướng dẫn cài đặt và làm quen

Kibana là một công cụ quản lý, giám sát tương tác với Elasticsearch một cách trực quan qua môi trường web



Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

Các khái niệm

- Documents
- Index
- Shard & replicas
- Node
- Cluster

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

Các khái niệm

1. Document
- Là một JSON object lưu 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
- Là một tập hợp các document, những document này có một số đặc điểm, 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 (phải là chữ thường), tên này dùng để thi hành các chức năng như đánh chỉ mục, tìm kiếm, cập nhật ... cho các document trong nó.

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

Các khái niệm

2. Index
Ví dụ: Khi đánh phụ lục cho quyển sách

```
Chapter 1:.....1
      Mục 1.....3
      Mục 2.....6
      ...

```

```
A: Coding(page139-140), Q&amp;amp;A(page125-167)
B: Testing(page197), Maintain(page123)

```

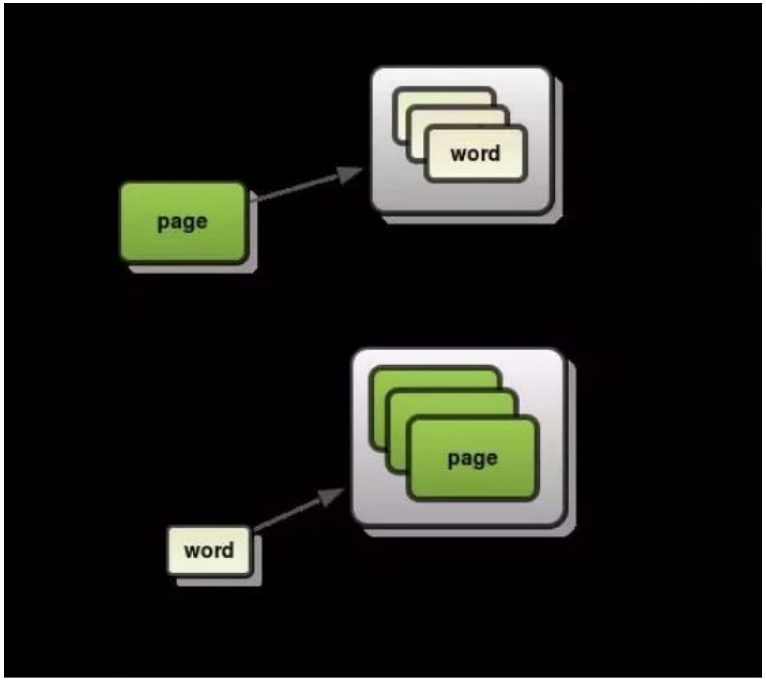
Forward Index: Đánh index theo nội dung, page: page -> words
Inverted Index: Đánh index theo keyword: words -> pages

=> Như vậy, với việc dùng inverted index với các keywords. Thay vì đọc từng page để tìm kiếm, ES sẽ tìm kiếm keyword trong index nên kết quả trả về sẽ rất nhanh.

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

Các khái niệm

2. Index



Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

Các khái niệm

3. Shard
- Để giải quyết các vấn đề hiệu năng khi lưu trữ dữ liệu lớn bị giới hạn bởi phần cứng ES 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, và 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 ES có thể phục hồi nếu một shard nào đó bị chết.

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

Các khái niệm

4. Node
- Đó là một server tham ra tạo thành cluster, nó có vai trò lưu dữ liệu, đánh chỉ mục và cung cấp khả năng tìm kiếm.
- Mỗi node được định danh bằng 1 unique name

5. Cluster
- Là các nodes - Elasticsearch kết nối với nhau để lưu giữ dữ liệu và cung cấp chức năng đánh chỉ mục, tìm kiếm dữ liệu trên các nodes đó.
- Một cluster xác định bởi tên duy nhất (mặc định tên là elasticsearch).
- Bạn cũng lưu ý, 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...).

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

Các khái niệm

Relational database	Elasticsearch
Database	Index
Table	Type
Record	Document
Schema	Mapping
Index	Everything is indexed
SQL	Query DSL
Select * From table	GET http://
Update table Set	PUT http://

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

1 số câu lệnh cơ bản

- Kiểm tra các index hiện có

```
GET _cat/indices?v
```

- Tạo 1 index mới

```
PUT index_name?pretty
```

- Xóa 1 index

```
DELETE index_name
```

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

1 số câu lệnh cơ bản

- Tạo 1 document trong index (hoặc thay thế)

```
PUT /student/_doc/1
{
  "name": "Đặng Quang Hà",
  "age": 21
}
```

=>

```
PUT /index_name/_doc/id
{
  "key": "value"
}
```

- Lấy ra dữ liệu của document

```
GET /student/_doc/1?pretty
```

=>

```
GET /index_name/_doc/id?pretty
```

- Update 1 trường của document

```
POST student/_update/1
{
  "doc": {
    "name": "Nguyễn Văn A"
  }
}
```

Phần 3: Các khái niệm và 1 số câu lệnh cơ bản

1 số câu lệnh cơ bản

- Thao tác với nhiều document cùng lúc

```
POST _bulk
{"index":{"_index":"student", "_id":"1"}}
{"name":"Đăng Quang Hà", "age": 21}
{"index":{"_index":"student", "_id":"2"}}
{"name":"Nguyễn Văn A", "age": 22}
{"index":{"_index":"student", "_id":"3"}}
{"name":"Trần Văn B", "age": 23}
```

```
POST _bulk
{"update": { "_index": "student", "_id": "1"}}
{"doc": {"name" : "Đăng Quang Hà 2"}}
{"update": { "_index": "student", "_id": "2"}}
{"doc": {"name" : "Đặng Quang Hà 3"}}
```

```
POST _bulk
{"delete":{"_index":"student", "_id":"1"}}
{"delete":{"_index":"student", "_id":"2"}}
{"delete":{"_index":"student", "_id":"3"}}
```

Bài tập vận dụng

Bài 1: Tạo index “users” chứa thông tin người dùng: tên, tuổi, quê quán, ngành nghề

Bài 2: Insert vào index “users" 3 documents theo 2 cách

- Insert từng document
- Insert nhiều documents cùng lúc

Bài 3: Thay thế document thứ 2 thành

- name: “Đặng Quang A”, age: “21”, home_town: “Hà Nội”, job: “IT"

Bài 4: Update thông tin của document thứ 2 thành

- name: “Nguyễn Văn A”, home_town: “Hải Phòng”

Bài 5: Xóa document thứ 2

Bài 6: Update documents thứ nhất và thứ 3 cùng lúc

- job: “student"

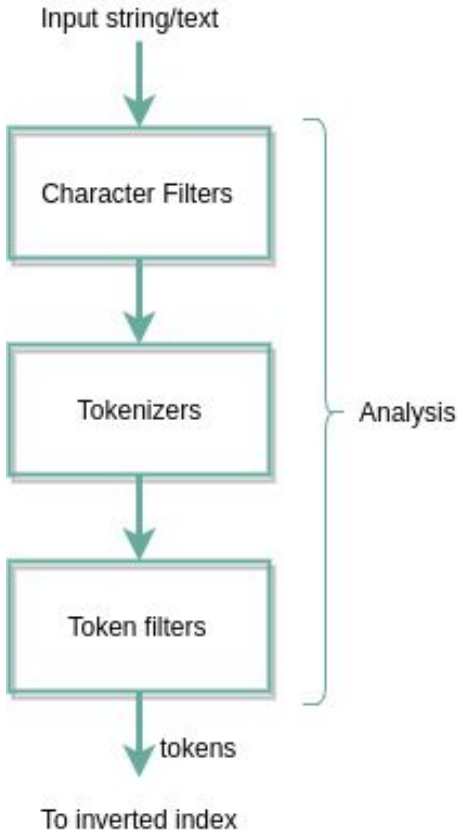
Phần 4: Text analysis

1. Khái niệm

- Analysis là một quá trình được tự động thực hiện để phân tích dữ liệu trước khi lưu vào inverted index trong Elasticsearch

- Gồm các bước sau:

- **Character filtering**: Chuyển các ký tự về dạng dữ liệu mình mong muốn
- **Breaking into tokens**: Phân tách thành các tokens độc lập sử dụng các tokenizers
- **Token filtering**: Biến đổi các token (thêm, bớt, xóa, sửa)
- **Token indexing**: Lưu các token đó vào inverted index



Phần 4: Text analysis

1.1 Character filtering

- Xử lý chuỗi đầu vào trước khi được tách từ, làm sạch chuỗi

VD: loại bỏ các thẻ html_tag hay chuyển ký hiệu & thành thành chữ "and"

"share your experience with NoSql & big data technologies"

=> "share your experience with NoSql and big data technologies"

1.2 Breaking into tokens

- Phân tách chuỗi thành các tokens độc lập sử dụng các tokenizers.

"share your experience with NoSql and big data technologies"

=> "share", "your", "experience", "with", "NoSql", "and", "big", "data", "technologies"

Phần 4: Text analysis

1.3 Token filtering

- Áp dụng bộ lọc từ, làm mượt thêm, convert từ sau khi chuỗi được phân tách thành các token

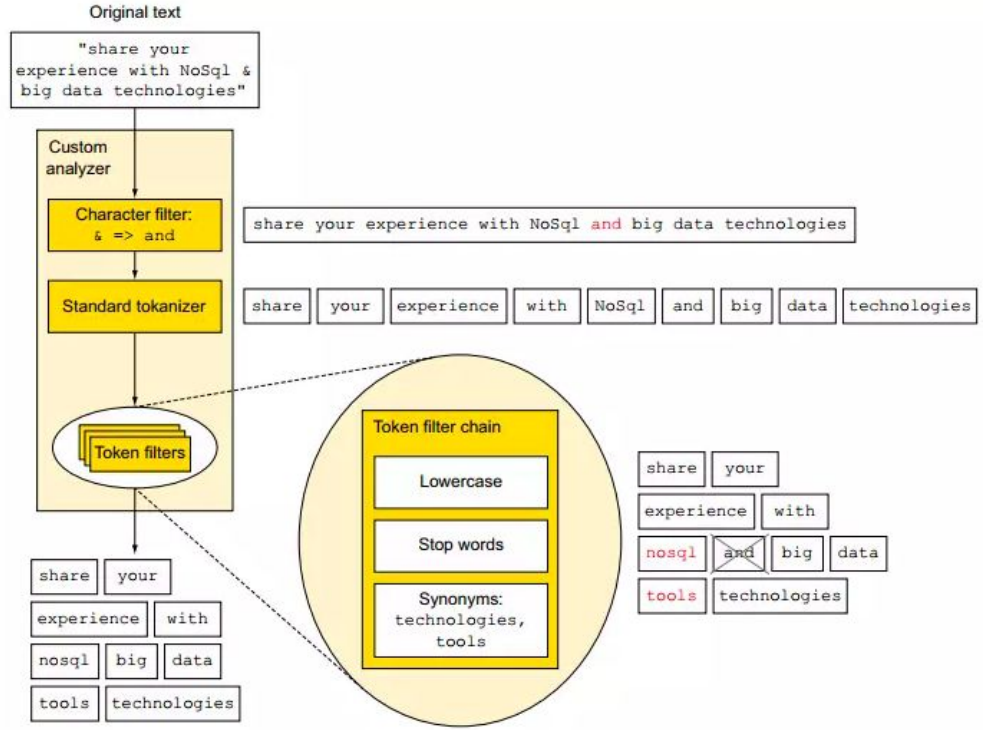
- Một analyzer có thể có không hoặc nhiều token filters

VD: Chuyển thành chữ thường, loại bỏ các stop word, thêm từ đồng nghĩa ...

"share", "your", "experience", "with", "NoSql", "and", "big", "data", "technologies"

=> "share", "your", "experience", "with", "nosql", "big", "data", "tools", "technologies"

Phần 4: Text analysis



Phần 4: Text analysis

2. Các loại Character filtering

- HTML Strip Character Filter: Bộ lọc ký tự html_strip loại bỏ các phần tử HTML

- Mapping Character Filter: Thay thế từ, chuỗi bằng từ văn bản khác

- **Pattern Replace Character Filter:** Bộ lọc ký tự pattern_replace thay thế bất kỳ ký tự nào khớp với biểu thức chính quy bằng ký tự thay thế được chỉ định.

Phần 4: Text analysis

3. Các loại Tokenizer

- Word Oriented Tokenizers (phân tách theo từng từ): Standard Tokenizer, Whitespace Tokenizer, Letter Tokenizer, Lowercase Tokenizer, UAX URL Email Tokenizer, Classic Tokenizer

- Partial Word Tokenizers (phân tách bên trong từ): N-Gram Tokenizer, Edge N-Gram Tokenizer

- **Structured Text Tokenizers (phân tách cấu trúc từ):** Keyword Tokenizer, Pattern Tokenizer, Simple Pattern Tokenizer, Char Group Tokenizer, Simple Pattern Split Tokenizer, Path Tokenizer

Phần 4: Text analysis

4. Các loại Token filter

ASCII folding:
- Chuyển đổi các ký tự chữ cái, số và ký tự tượng trưng không có trong khối Unicode Latinh cơ bản (127 ký tự ASCII đầu tiên) thành tương đương ASCII của chúng, nếu có.

Length:
- Loại bỏ token ngắn hơn hoặc dài hơn độ dài ký tự được chỉ định. Ví dụ: có thể sử dụng bộ lọc độ dài để loại trừ token ngắn hơn 2 ký tự và token dài hơn 5 ký tự

Lowercase:
- Đưa token về dạng không viết hoa

Phần 4: Text analysis

4. Các loại Token filter

Stemmer:
- Liệt kê thêm các biến thể của từ. VD: quickly => quickli | jumping => jump

Stop:
- Liệt kê các stopword token để loại bỏ

Synonym:
- Liệt kê các từ đồng nghĩa, bổ sung vào token

Phần 5: Index

Tạo chỉ mục để thêm chỉ mục mới vào cụm Elasticsearch. Khi tạo chỉ mục, có thể chỉ định những phần sau:

Settings: tạo trên mỗi chỉ mục và kiểm soát tất cả các cài đặt liên quan đến index

Mappings: cung cấp định nghĩa định các trường được index

Aliases: cung cấp tập hợp các bí danh

Phần 5: Create Index

5.1 Settings

- Tĩnh => Chỉ có thể được đặt tại thời điểm tạo chỉ mục hoặc trên một chỉ mục đã stop.

- Động => Có thể được thay đổi trực tiếp trên một index bằng cách sử dụng API

- Cài đặt cần quan tâm:

- Index: Cài đặt thời gian update sau index, số lượng shards, replicas
- Analysis: Cài đặt và xác định analyzers, tokenizers, token filters and character filters.

Xem ví dụ:
https://github.com/dangquangha/elasticsearch-training/blob/master/training/mapping/es_mapping.txt

```
PUT jobs_test
{
  "settings": {
    "index": {
      "refresh_interval": "2s",
      "number_of_shards": 3,
      "number_of_replicas": 2
    },
    "analysis": {⟺}
  }
}
```

Phần 5: Create Index

5.2 Mapping

- Là quá trình xử lý cách mà các Document sẽ được index và lưu trữ như thế nào. Mapping giúp chúng ta cùng lúc khởi tạo 1 field & định nghĩa cách field đó được index

- Mapping rất quan trọng trong quá trình xây dựng lưu trữ index, cần tổ chức xây dựng và tối ưu

- Tĩnh:
- Tự định cấu hình từ trước cho mỗi trường khi được đánh index vào elasticsearch
- Các trường sẽ được chỉ định tên, kiểu dữ liệu và kiểu phân tích dữ liệu,...

- Động:
- Các trường và kiểu ánh xạ không cần phải xác định trước khi sử dụng
- Nhờ mapping động, các tên trường mới sẽ được thêm tự động khi lập chỉ mục
- Các quy tắc ánh xạ động có thể được định cấu hình để tùy chỉnh sử dụng cho các trường mới

Phần 5: Create Index

5.2 Mapping

- Kiểu dữ liệu: Alias, Arrays, Binary, Boolean, Date, Date nanoseconds, Dense, Vector, Flattened, Geo-point, Geo-shape, Histogram, IP, Join, Keyword, Nested, Numeric, Object, Percolator, Point, Range, Rank feature, Rank features, Search-as-you-type, Shape, Sparse vector, Text, Token count

PUT: ENDPOINT/jobs_test

```
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "shirt": {
      "properties": {
        "name": {
          "type": "text"
        },
        "address": {
          "type": "text"
        },
        "salary": {
          "type": "long"
        }
      }
    }
  }
}
```

Phần 6: Mapping

1 số vấn đề

- Khi có nhiều properties

- Khi có indices khác thì với cách làm trên khi tổng hợp và truy vấn thì sẽ chỉ thực hiện được trên từng loại một

Phần 6: Mapping

```
{
  "name": "tshirt",
  "size": "S",
  "color": "black",
  "fabric": "cotton",
  "price": 1000
}
```



```
{
  "entity": {
    "name": "tshirt"
  },
  "keyword_facets": [
    {
      "facet_name": "size",
      "facet_value": "S"
    },
    {
      "facet_name": "color",
      "facet_value": "black"
    },
    {
      "facet_name": "fabric",
      "facet_value": "cotton"
    }
  ],
  "long_facets": [
    {
      "facet_name": "price",
      "facet_value": 1000
    }
  ]
}
```

- Chia làm 2 khía cạnh: keyword, long

- Những thuộc tính chưa xác định được khía cạnh thì sẽ cho vào “entity"

Phần 6: Mapping

PUT: ENDPOINT/shirts

```
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "shirt": {
      "properties": {
        "name": {
          "type": "text"
        },
        "size": {
          "type": "keyword"
        },
        "color": {
          "type": "keyword"
        },
        "fabric": {
          "type": "keyword"
        }
        "price": {
          "type": "long"
        }
      }
    }
  }
}
```


PUT: ENDPOINT/shirts

```
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "shirt": {
      "properties": {
        "entity": {
          "properties": {
            "name": {
              "type": "text"
            }
          }
        },
        "keyword_facets": {
          "type": "nested",
          "properties": {
            "facet_name": {
              "type": "keyword"
            },
            "facet_value": {
              "type": "keyword"
            }
          }
        },
        "long_facets": {
          "type": "nested",
          "properties": {
            "facet_name": {
              "type": "keyword"
            },
            "facet_value": {
              "type": "long"
            }
          }
        }
      }
    }
  }
}
```

Phần 7: Query DSL

- Elasticsearch cung cấp một bộ Query DSL (Domain Specific Language) dựa trên JSON để định nghĩa các truy vấn. Bao gồm hai loại mệnh đề:

- **Leaf query clauses**: Những mệnh đề này tìm những giá trị cụ thể của những trường cụ thể. Ví dụ như các truy vấn match, term và range. => **Truy vấn đơn**

- **Compound query clauses**: Những mệnh đề này kết hợp các leaf query và các compound query khác để thu được kết quả mong muốn. => **Truy vấn kép**

Phần 7: Query DSL

```
{
  "users" : {
    "mappings" : {
      "user" : {
        "properties" : {
          "name" : {
            "type" : "string"
          }
               }
          }
        }
    }
}
```

Document 1

```
{name: "Nguyen Thi Ngoc"}
```

Document 2

```
{name: "Ruby Nguyen"}
```

Document 3

```
{name: "Ngoc Nguyen Thi"}
```

Phần 7: Query DSL

1. Full text queries

- Loại truy vấn này thường được sử dụng cho các trường full text

1.1 Match query

- Cơ bản

```
{
  "query": {
    "match": {
      "name": "Nguyen"
    }
  }
}
```

Query này sẽ search tất cả user có name là Nguyen

Phần 7: Query DSL

1.1 Match query

- Yêu cầu độ khớp tối thiểu: số hoặc %

```
{
  "query": {
    "match": {
      "name": {
        "query": "Nguyen",
        "minimum_should_match" : 1,
       }
    }
  }
}
```

- Toán tử “and", “or"

```
{
  "query": {
    "match": {
      "name": {
        "query": "Nguyen Thi",
        "operator" : "and"
       }
    }
  }
}
```

Phần 7: Query DSL

1.2 Match Phrase Query

- Truy vấn cụm từ: Được sử dụng khi chúng ta muốn match các từ với thứ tự giống như query chúng ta mong muốn. Ví dụ search user có name Thi Ngoc

```
{
    "query": {
        "match_phrase" : {
            "name" : "Thi Ngoc"
        }
    }
}
```

- Khớp với tiền tố của từ
=> với truy vấn này thì "Thi Ngoc" cũng match mà "Thi Nguyen" cũng đúng.

```
{
    "query": {
        "match_phrase_prefix" : {
            "name" : "Thi Ng"
        }
    }
}
```

Phần 7: Query DSL

1.3 Multi Match Query

- Sử dụng từ truy vấn match và cho phép search nhiều trường và có độ ưu tiên khác nhau

```
{
  "query": {
    "multi_match" : {
      "query":  "Ngoc",
      "fields": ["name", "address"]
    }
  }
}
```

```
{
  "query": {
    "multi_match" : {
      "query":  "Ngoc",
      "fields": ["name", "address^3"]
    }
  }
}
```

- Field có thể chỉ định bằng ký tự đại diện

```
{
  "query": {
    "multi_match" : {
      "query":  "Ngoc",
      "fields": ["*_name", "address" ]
    }
  }
}
```

Khi này ta có thể search cả field first_name và last_name

Phần 7: Query DSL

2. Term level queries

- Truy vấn này thường được sử dụng cho các dữ liệu kiểu số, ngày tháng, enum. Ngoài ra cho phép bạn tạo truy vấn cấp thấp, bỏ qua bước phân tích.

2.1 Term query

- Truy vấn term tìm những record có chứa cụm từ chính xác trong query

```
{
      term: {
         name: "Ngoc"
      }
}
```

Với term query nó sẽ match document nào có chính xác term “Ngoc" và trả về kết quả

Phần 7: Query DSL

2.2 Range Query

- Trả về các record với trường khớp với phạm vi nhất định

- gte: Lớn hơn hoặc bằng
- gt: Lớn hơn
- lte: Nhỏ hơn hoặc bằng
- lt: Nhỏ hơn

```
{
  "query": {
    "range" : {
        "age" : {
          "gte" : 12,
          "lte" : 20
        }
    }
  }
}
```

Phần 7: Query DSL

2.3 Date format in range queries

```
{
    "query": {
        "range" : {
            "date_of_birth" : {
                "gte": "01/01/2000",
                "lte": "2013",
                "format": "dd/MM/yyyy||yyyy"
            }
        }
    }
}
```

Phần 7: Query DSL

2.4 Wildcard Query

- Trả về các record khớp với các ký tự đại diện được đưa ra.

```
{
    "query": {
        "wildcard" : { "name" : "*hon*" }
    }
}
```

Với query thế này thì record có name là "Phong" cũng đúng mà có name là "Thong" cũng đúng.

Phần 7: Query DSL

3. Bool Query

- Cho phép kết hợp các câu truy vấn khác để tạo ra một logic hợp lý. Có các loại:

- must: phải phù hợp với tất cả các điều kiện và đóng góp vào điểm số.
- filter: giống với must nhưng bỏ qua điểm số. (Tham khảo)
- should: Chỉ cần phù hợp vs một trong các điều kiện.
- must_not: Người lại với must, phải không phù hợp với tất cả các điều kiện.

Phần 7: Query DSL

3. Bool Query

```
POST _search
{
  "query": {
    "bool" : {
      "must" : {
        "term" : { "user" : "kimchy" }
      },
      "filter": {
        "term" : { "tag" : "tech" }
      },
      "must_not" : {
        "range" : {
          "age" : { "gte" : 10, "lte" : 20 }
        }
      },
      "should" : [
        { "term" : { "tag" : "wow" } },
        { "term" : { "tag" : "elasticsearch" } }
      ],
      "minimum_should_match" : 1,
      "boost" : 1.0
    }
  }
}
```

Phần 7: Query DSL

Query Name	Chức năng	Query mẫu	Matching Text	Not Matching Text
match	- Matches nếu 1 term trong query đó match - Càng nhiều term match thì score của document đó sẽ lớn - Query search sẽ được apply analyzer config cho field search đó hoặc để mặc định (nếu không set)	cat dog	- cat and dog - The blue cat - The cat is blue - The dog is white	- The blue - The white
phrase_match	- Chỉ matches nếu các term match có cùng thứ tự và liên tiếp nhau	cat dog	- cat dog are green - green and cat dogs	- dog cat are green - cat and dog are green
prefix	Nó tương tự như pharse_match query nhưng nó sẽ match chính xác term	FR VN	- FR VN2017 - FR VN-2017	- FRVN2017 - FRVN-2017

Phần 7: Query DSL

Query Name	Chức năng	Query mẫu	Matching Text	Not Matching Text
term	- Query trên truy vấn mình truyền vào, analyzer sẽ không được apply	dog	- this is a dog	- there are many dogs here - Dog is green
multi_match	- apply match query trên nhiều field khác nhau	-field1: dog - field2: cat	match với field1 chứa dog hoặc field2 chứa cat	- Nếu cả 2 fields không chứa bất cứ từ liên quan đến dog và cat
bool	Applies nhiều queries khác nhau	must: {field1: dog}, must_not: {field2: cat}	match với document với field1 chứa dog và field2 không chứa cat	Nếu document có field1 chứa dog và field2 chứa cat - Hoặc field2 chứa cat - Hoặc field1 không chứa dog

Phần 8: Aggregations

Aggregation query của elasticsearch cũng giống như những gì chúng ta làm với group by query trong MySQL nhưng những gì mà MySQL đem lại là không đủ với những thống kê phức tạp.

Dưới đây là 2 ví dụ ta có thể sử dụng aggregations:

- Bạn đang thực hiện kinh doanh quần áo online và muốn biết trung bình cộng giá của tất cả sản phẩm trong danh mục hàng hóa.

- Bạn muốn kiểm tra xem có bao nhiêu sản phẩm có giá trong khoảng 100$ và giá trong khoảng từ 100$ đến 200$.

Phần 8: Aggregations

Cú pháp của Aggregations

aggs: Từ khóa này cho biết bạn muốn thực hiện truy vấn aggregation.

name_of_aggregation: Đây là tên của aggregation mà bạn định nghĩa.

type_of_aggregation: Loại aggregation được sử dụng.

field: field của document

```
"aggs": {
    "name_of_aggregation": {
      "type_of_aggregation": {
        "field": "document_field_name"
}
```

Phần 8: Aggregations

Cú pháp của Aggregations

aggs: Từ khóa này cho biết bạn muốn thực hiện truy vấn aggregation.

name_of_aggregation: Đây là tên của aggregation mà bạn định nghĩa.

type_of_aggregation: Loại aggregation được sử dụng.

field: field của document

```
"aggs": {
    "name_of_aggregation": {
      "type_of_aggregation": {
        "field": "document_field_name"
}
```

Phần 8: Aggregations

Các loại Aggregation chính

Aggregations có thể được chia thành 4 nhóm: bucket aggregations, metric aggregations, matrix aggregations, và pipeline aggregations.

- **Metric aggregations:** Loại aggregation này tính toán số liệu từ các giá trị thu được từ các document đang được tổng hợp.

- **Bucket aggregations:** Không tính toán các số liệu từ các field như Metric aggregations mà tạo ra các bucket của các document. Mỗi bucket tương ứng với một tiêu chí mà dựa vào đó để xác định xem 1 document có thuộc bucket đó trong bối cảnh hiện tại không.

- **Pipeline aggregations:** Loại aggregation này lấy input từ output của các aggregation khác.

- **Matrix aggregations:** Những aggregation này làm việc trên nhiều hơn một field và cung cấp kết quả thống kê dựa trên các document thu được từ các trường được sử dụng.

Phần 8: Aggregations

Buckets

- Là một họ của aggregations query xây dựng lên các buckets, mỗi buckets tương đương với một key và một tiêu chí nào đó.

- Khi chạy aggregation query, tất cả các tiêu chí của từng buckets được kiểm tra trên toàn bộ các documents, khi document thỏa mãn một điều kiện của một bucket, nó được cân nhắc là thuộc bucket đó.

- Kết quả của aggregation dạng này là một list các buckets, mỗi bucket chứa tập các documents thuộc về bucket đó.

Phần 8: Aggregations

Một số Aggregation quan trọng

5 aggregation quan trọng trong Elasticsearch là:

1. Cardinality aggregation
2. Stats aggregation
3. Filter aggregation
4. Terms aggregation
5. Nested aggregation

Phần 8: Aggregations

8.1 Cardinality aggregation

Aggregation này là một single-value aggregation thuộc loại Metric aggregations, sử dụng để tính toán số lượng các giá trị khác nhau của một field cụ thể.

```
{
 "aggs": {
  "unique_name": {
    "cardinality": {
      "field": "name"
    }
  }
}
```

=> Để xem có bao nhiêu tên áo khác nhau có trong dữ liệu

Phần 8: Aggregations

8.2 Stats Aggregation

Đây là 1 multi-value Metric aggregations, tính toán số liệu thống kê từ các giá trị số từ các document tổng hợp được.

Số liệu thống kê trả về bao gồm min, max, sum, count và avg.

```
{
  "aggs": {
    "quantity_stats": {
      "stats": {
        "field": "id"
      }
    }
  }
}
```



```
{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 5960,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [...]
  },
  "aggregations" : {
    "quantity_stats" : {
      "count" : 5960,
      "min" : 1.0,
      "max" : 5960.0,
      "avg" : 2980.5,
      "sum" : 1.776378E7
    }
  }
}
```

Phần 8: Aggregations

8.3 Filter Aggregation

Aggregation này thuộc Bucket aggregations, định nghĩa một bucket duy nhất chứa các documents thỏa mãn điều kiện filter, và có thể thực hiện tính toán số liệu trong bucket này.

Ví dụ: ta filter các documents có username "quang ha" và tính trung bình cộng giá của các sản phẩm người đó đã mua.

```
{
 "aggs": {
     "user_based_filter" : {
         "filter" : {
             "term" : {
                 "user" : "quang ha"
              }
          },
          "aggs" : {
              "avg_price" : {
                  "avg" : {
                      "field" : "products.price"
                   }
               }
           }
      }
   }
}
```


```
{
  "took" : 774,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 6045,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "user_based_filter" : {
      "doc_count" : 46,
      "avg_price" : {
        "value" : 6022.5
      }
    }
  }
}
```

Phần 8: Aggregations

8.4 Terms Aggregation

Là 1 loại Bucket aggregations, tạo ra các bucket từ các giá trị của field, số lượng bucket là động, mỗi giá trị khác nhau của field được chỉ định sẽ tạo ra 1 bucket.

Trong ví dụ dưới đây, ta sẽ thực hiện terms aggregation trên field "user". Ở kết quả, ta sẽ có các bucket cho mỗi user, mỗi bucket sẽ chứa số lượng document.

```
POST products/_search
{
  "aggs": {
    "term_aggregations" : {
      "terms": {
        "field": "category"
      }
    }
  }
}
```

Phần 8: Aggregations

8.5 Nested Aggregation

Đây là một trong những loại quan trọng nhất trong Bucket Aggregations. Một Nested Aggregation cho phép tổng hợp một field với nested documents một field mà có nhiều sub-fields

Một field phải có type là "nested" trong index mapping nếu bạn muốn sử dụng Nested Aggregation trên field đó.

```
POST products/_search
{
  "aggs": {
    "agg_integer_facet": {
      "nested": {
        "path": "integer_facets"
      },
      "aggs": {
        "integer_facets": {
          "filter": {
            "term": {
              "integer_facets.facet_name": "trade_type"
            }
          },
          "aggs": {
            "facet_value": {
              "terms": {
                "field": "integer_facets.facet_value"
              }
            }
          }
        }
      }
    }
  },
  "size": 0
}
```

```
},
"aggregations" : {
  "agg_integer_facet" : {
    "doc_count" : 90216,
    "integer_facets" : {
      "doc_count" : 10024,
      "facet_value" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 0,
        "buckets" : [
          {
            "key" : 1,
            "doc_count" : 5162
          },
          {
            "key" : 2,
            "doc_count" : 4862
          }
        ]
      }
    }
  }
}
```

1 số tài liệu tham khảo

https://gitlab.com/training-o-t-o/elasticsearch

https://topdev.vn/blog/elasticsearch-la-gi/

https://xuanthulab.net/elasticsearch/