



Elasticsearch Deep Dive

김종민 (Jongmin Kim)
Developer Evangelist @Elastic
jongmin.kim@elastic.co



Elasticsearch is...

an open source, distributed, scalable, highly available, document-oriented, RESTful, full text search engine with real-time search and analytics capabilities

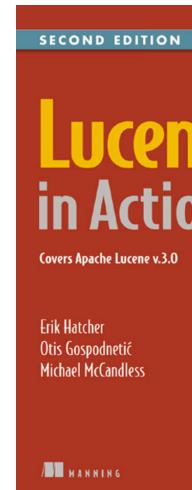
Apache 2.0 License

<https://www.apache.org/licenses/LICENSE-2.0>

아파치 루씬 (Apache Lucene)



- Created by - Doug Cutting
- Written in - Java
- Apache Solr, Elasticsearch



Elasticsearch is...

*An open source, **distributed**, **scalable**, **highly available**, document-oriented, RESTful, full text search engine with real-time search and analytics capabilities*

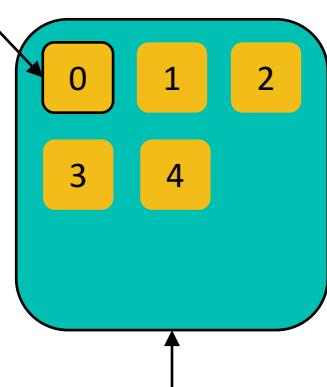


Elasticsearch 클러스터링 과정

대용량 검색을 위해서는 클러스터링이 필요합니다.

Elasticsearch는 데이터를 샤드(Shard) 단위로 분리해서 저장합니다.

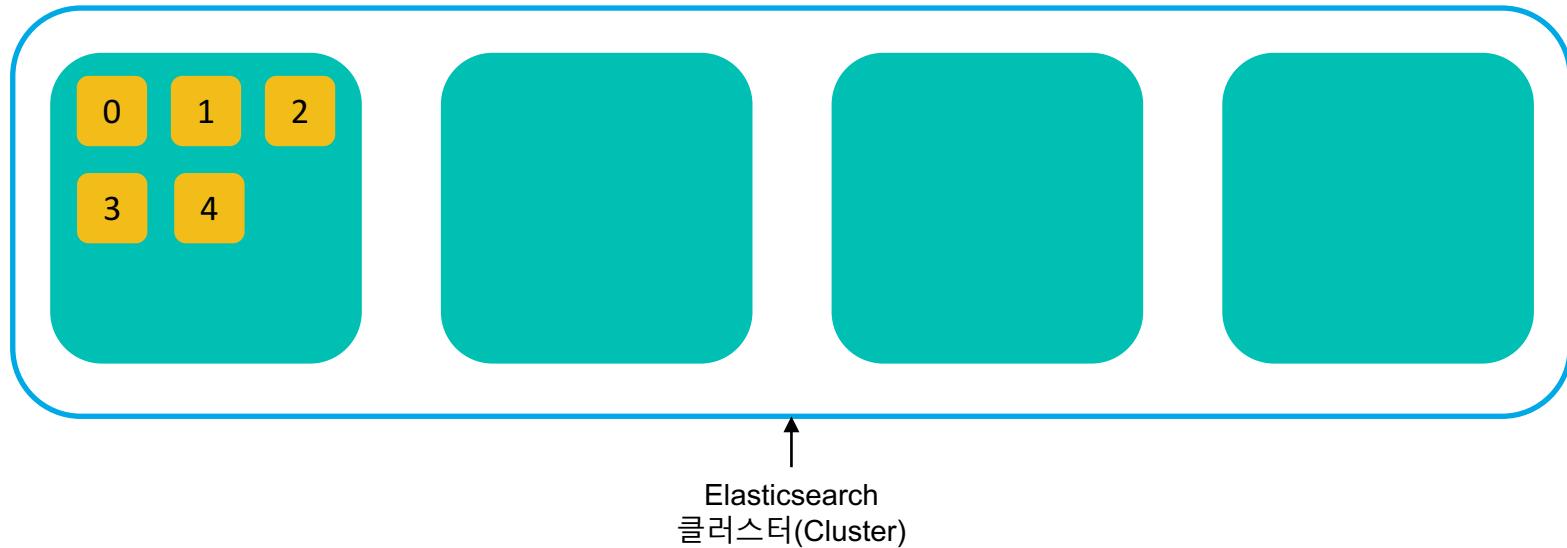
샤드 (Shard)
루씬 검색 쓰레드



노드 (Node)
Elasticsearch 실행 프로세스

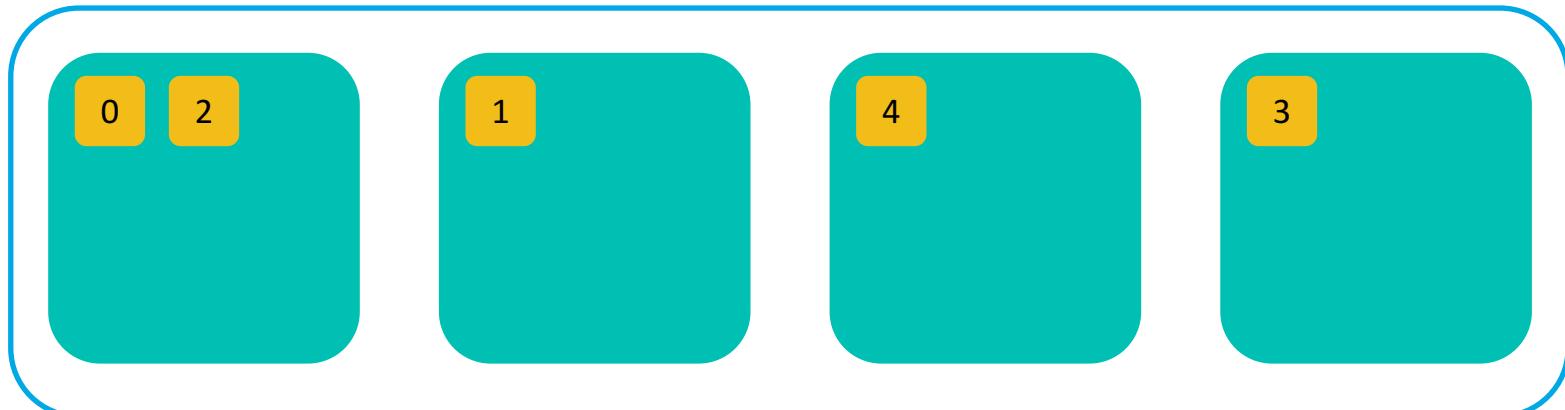
Elasticsearch 클러스터링 과정

노드를 여러개 실행시키면 같은 클러스터로 묶입니다.



Elasticsearch 클러스터링 과정

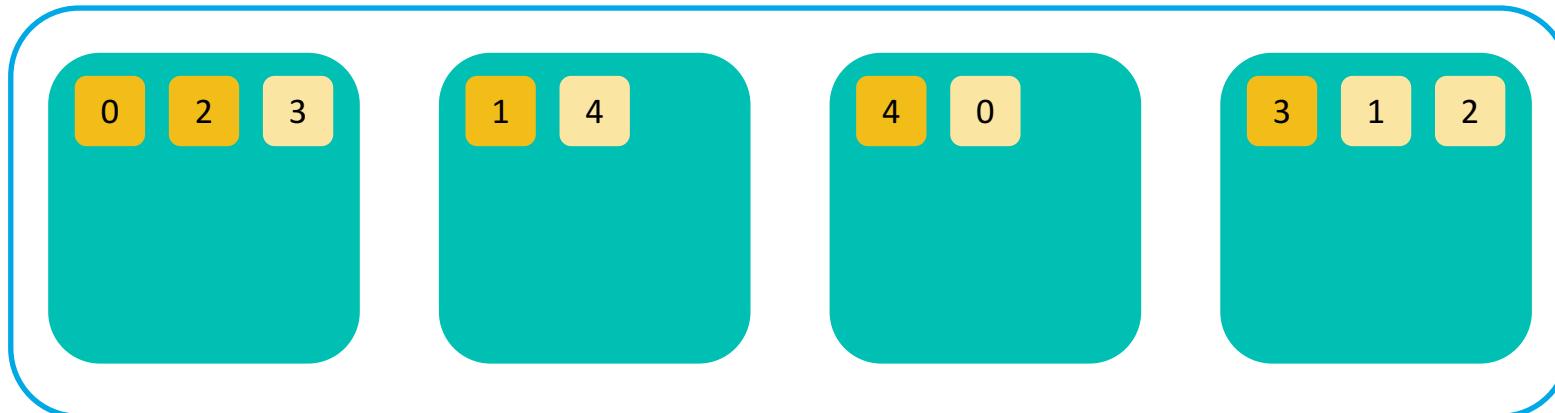
샤드들은 각각의 노드들에 분배되어 저장됩니다.



Elasticsearch 클러스터링 과정

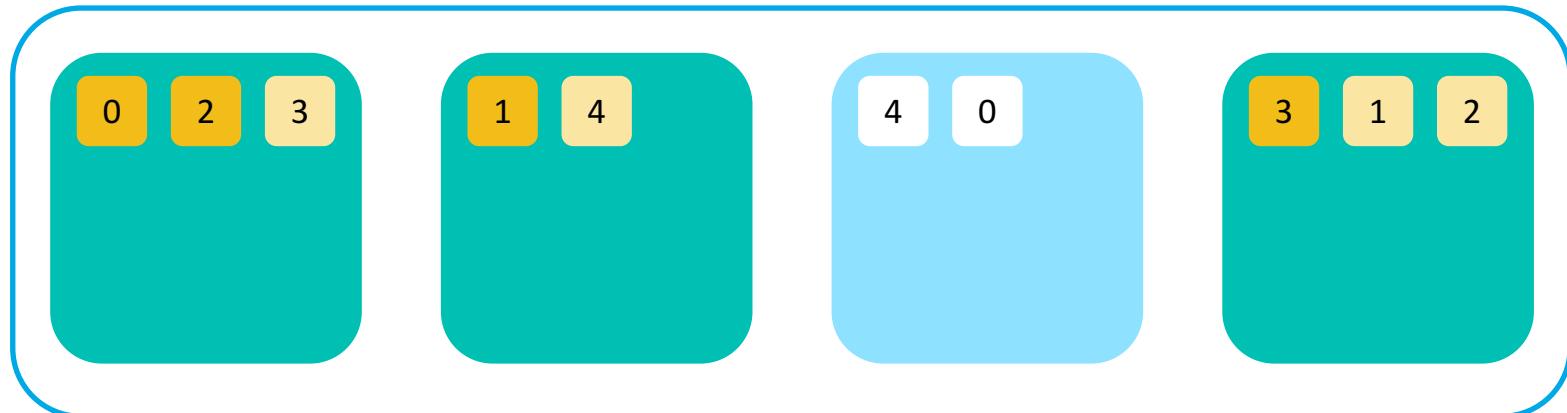
무결성과 가용성을 위해 샤프의 복제본을 만듭니다.

같은 내용의 복제본과 샤프는 서로 다른 노드에 저장됩니다.



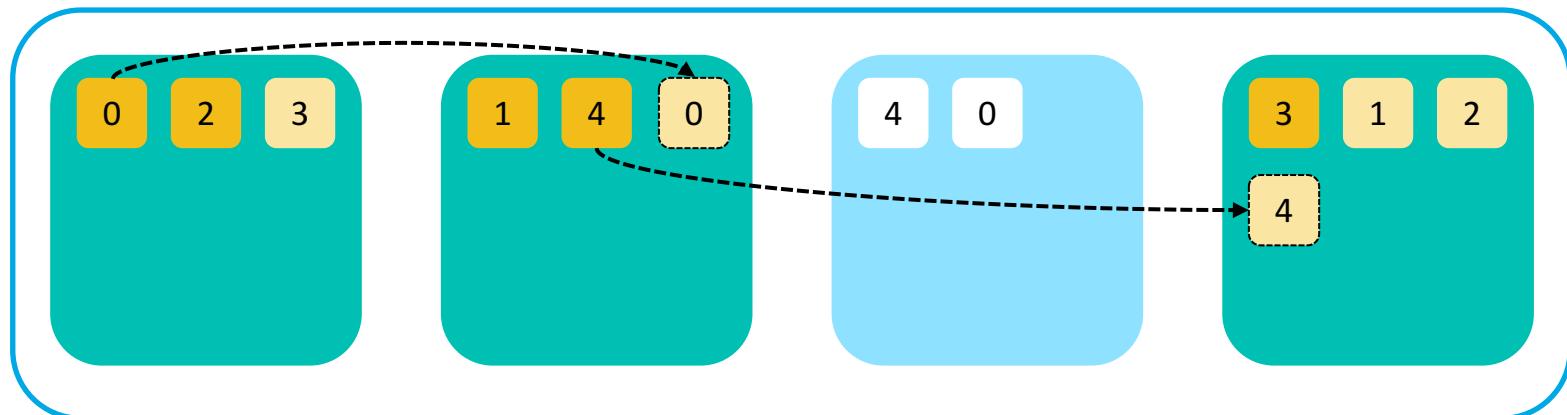
Elasticsearch 클러스터링 과정

시스템 다운이나 네트워크 단절 등으로 유실된 노드가 생기면



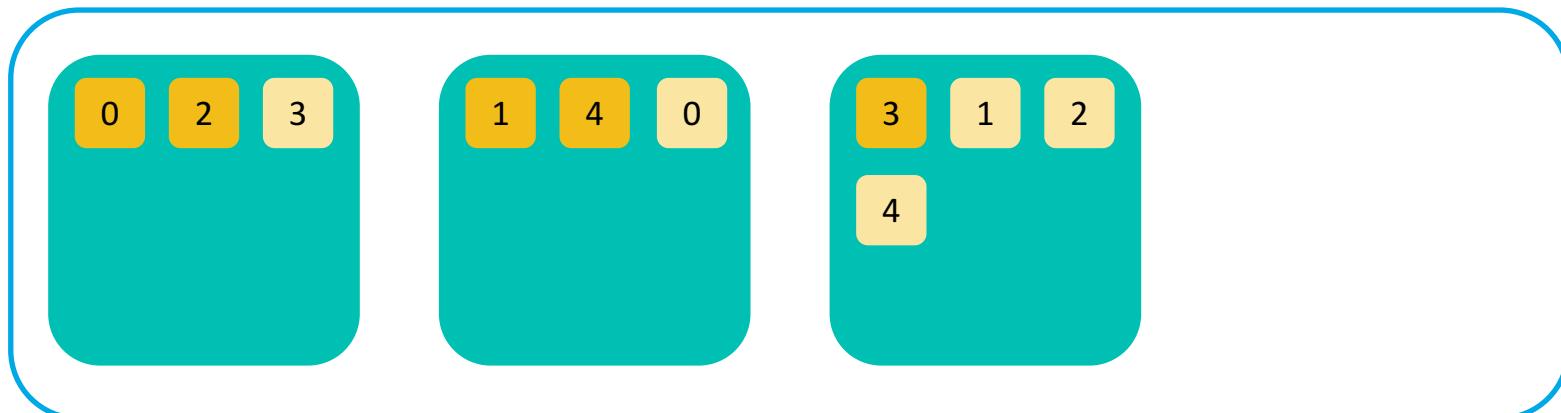
Elasticsearch 클러스터링 과정

복제본이 없는 샤크들은 다른 살아있는 노드로 복제를 시작합니다.



Elasticsearch 클러스터링 과정

노드의 수가 줄어들어도 샤프트의 수는 변함 없이 무결성을 유지합니다.



클러스터 (Cluster)

- 엘라스틱서치 시스템의 가장 큰 단위입니다.
- 하나의 클러스터는 다수의 노드로 구성됩니다.
- 하나의 클러스터를 다수의 서버로 바인딩 해서 운영, 또는 역으로 하나의 서버에서 다수의 클러스터의 운용이 가능합니다.

```
config/elasticsearch.yml
```

```
cluster.name: elasticsearch
```

```
bin/elasticsearch -E cluster.name=elasticsearch
```

노드 (Node)

- 엘라스틱서치를 구성하는 하나의 단위 프로세스입니다.
- 다수의 샤프로 구성됩니다.
- 같은 클러스터명을 가진 노드들은 자동으로 바인딩 됩니다.

```
config/elasticsearch.yml
```

```
node.name: "Node1"
```

```
bin/elasticsearch -E node.name=Node1
```

Master & Data 노드

- 마스터 노드 : 클러스터 상태 정보를 관리합니다.
- 데이터 노드 : 데이터 입/출력, 검색을 수행합니다.

config/elasticsearch.yml

```
node.name: "Node1"  
node.master: true  
node.data: false
```

```
node.name: "Node2"  
node.master: false  
node.data: true
```



샤드(Shard) & 레플리카(Replica)

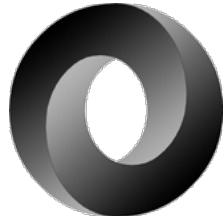
- 샤드 : 루씬 검색 인스턴스입니다.
- 레플리카 : 데이터 무결성 유지를 위한 샤드의 복사본입니다.

```
curl -XPUT "http://localhost:9200/books" -H 'Content-Type: application/json' -d'  
{  
  "settings": {  
    "number_of_shards": 5,  
    "number_of_replicas": 1  
  }  
}'
```

```
curl -XPUT "http://localhost:9200/books/_settings" -H 'Content-Type: application/json' -d'  
{  
  "number_of_replicas": 0  
}'
```

Elasticsearch is...

*An open source, distributed, scalable, highly available, **document-oriented**, RESTful, full text search engine with real-time search and analytics capabilities*



Source: <http://json.org>

```
{  
  "name" : "Craft"  
  "geo" : {  
    "city" : "Budapest",  
    "lat" : 47.49, "lon" : 19.04  
  }  
}
```

Elasticsearch is...

An open source, distributed, scalable, highly available, document-oriented, RESTful, full text search engine with real-time search and analytics capabilities

```
method      host     port    index   type   document id
$ curl -XPUT http://localhost:9200/books/book/1 -d '
{
  "title" : "Elasticsearch Guide",
  "author" : "Kim",
  "date" : "2014-05-01",
  "pages" : 250
}'
{"_index":"books","_type":"book","_id":"1","_version":1,"created":true}
```

데이터 입력

PUT 메소드를 이용해서 본문으로 JSON 문서를 입력합니다.

```
curl -XPUT 'http://localhost:9200/books/book/1' -d '  
{ "title": "Romeo and Juliet", "author": "William Shakespeare", "category": "Tragedies",  
"written": "1562-12-01T20:40:00", "pages" : 125 }' -H 'Content-Type: application/json'
```

```
curl -XPUT 'http://localhost:9200/books/book/2' -d '  
{ "title" : "Hamlet", "author": "William Shakespeare", "category": "Tragedies",  
"written": "1599-06-01T12:34:00", "pages" : 172 }' -H 'Content-Type: application/json'
```

```
curl -XPUT 'http://localhost:9200/books/book/3' -d '  
{ "title": "The Prince and the Pauper", "author": "Mark Twain", "category": "Children book",  
"written": "1881-08-01T10:34:00", "pages" : 79}' -H 'Content-Type: application/json'
```

데이터 조회 (접근)

GET 메소드 (생략 가능) 으로 도큐먼트에 접근합니다.
pretty 매개변수를 이용해서 보기 좋게 출력이 가능합니다.

```
curl -XGET 'http://localhost:9200/books/book/1?pretty' -H 'Content-Type: application/json'
```

```
{  
    "_index" : "books",  
    "_type" : "book",  
    "_id" : "1",  
    "_version" : 3,  
    "found" : true,  
    "_source" : {  
        "title" : "Romeo and Juliet",  
        "author" : "William Shakespeare",  
        "category" : "Tragedies",  
        "written" : "1562-12-01T20:40:00",  
        "pages" : 125  
    }  
}
```

데이터 삭제

DELETE 메소드를 이용해서 도큐먼트 또는 인덱스 단위로 삭제합니다.

```
curl -XDELETE localhost:9200/books/book/1 -H 'Content-Type: application/json'
```

```
curl -XDELETE localhost:9200/books -H 'Content-Type: application/json'
```

데이터 배치 입력

_bulk API 를 이용해서 여러개의 문서를 배치로 입력합니다.

```
curl -XPOST "http://localhost:9200/books/book/_bulk" -d '  
{"index":{"_id":"1"}}  
{"title":"Romeo and Juliet","author":"William Shakespeare","category":"Tragedies","written":"1562-12-01T20:40:00","pages":125}  
{"index":{"_id":"2"}}  
{"title":"Hamlet","author":"William Shakespeare","category":"Tragedies","written":"1599-06-01T12:34:00","pages":172}  
{"index":{"_id":"3"}}  
{"title":"The Prince and the Pauper","author":"Mark Twain","category":"Children book","written":"1881-08-01T10:34:00","pages":79}  
' -H 'Content-Type: application/json'
```

데이터 검색

_search API를 사용해서 인덱스 단위로 검색합니다.

```
curl -XGET "http://localhost:9200/books/_search?pretty=true"
```

```
{ ... 중략 ...
},
  "hits" : {
    "total" : 3,
    "max_score" : 1.0,
    "hits" : [ {
      ... 중략 ...
      "_source": {
        "title": "Romeo and Juliet", "author": "William Shakespeare", "category": "Tragedies", "written": "1562-12-01T20:40:00", "pages" : 125 }
    },
    ... 중략 ...
  }
```

데이터 검색 (URI)

URI 의 q 매개변수로 검색이 가능합니다.

```
curl -XGET "http://localhost:9200/books/_search?pretty=true"
```

```
curl -XGET "http://localhost:9200/books/_search?q=author:wiliam&pretty=true"
```

```
curl -XGET "http://localhost:9200/books/_search?q=wiliam&df=author&pretty=true"
```

```
curl -XGET "http://localhost:9200/books/_search?q=wiliam%20AND%20romeo&pretty=true"
```

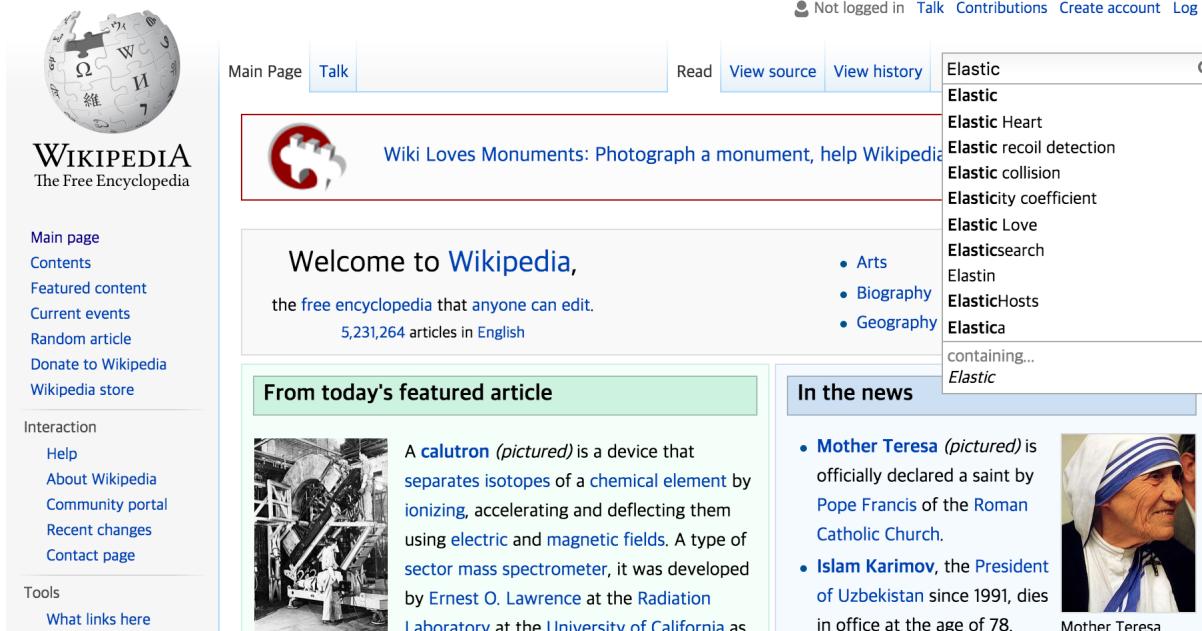
데이터 검색 (Request Body)

http 의 본문에 쿼리를 삽입합니다. 더욱 복잡한 질의를 할 수 있습니다.

```
curl -XGET "http://localhost:9200/books/_search?pretty=true" -d'  
{  
  "query": {  
    "match": {  
      "author": "william"  
    }  
  }  
}' -H 'Content-Type: application/json'
```

Elasticsearch is...

An open source, distributed, scalable, highly available, document-oriented, RESTful, full text search engine with real-time search and analytics capabilities



The screenshot shows the Wikipedia homepage with a search bar in the top right corner containing the word "Elastic". The search bar includes a magnifying glass icon and a dropdown menu listing suggestions such as "Elastic", "Elastic Heart", "Elastic recoil detection", etc. Below the search bar, the main content area features the "Welcome to Wikipedia" banner, the featured article about a calutron, and a news section about Mother Teresa.

Not logged in Talk Contributions Create account Log in

Main Page Talk Read View source View history

Elastic

Elastic

Elastic Heart

Elastic recoil detection

Elastic collision

Elasticity coefficient

Elastic Love

Elasticsearch

Elastin

ElasticHosts

Elastica

containing...

Elastic

Wiki Loves Monuments: Photograph a monument, help Wikipedia

Welcome to Wikipedia,

the free encyclopedia that anyone can edit.

5,231,264 articles in English

• Arts

• Biography

• Geography

In the news

• Mother Teresa (pictured) is officially declared a saint by Pope Francis of the Roman Catholic Church.

• Islam Karimov, the President of Uzbekistan since 1991, dies in office at the age of 78.

Mother Teresa

From today's featured article

A calutron (pictured) is a device that separates isotopes of a chemical element by ionizing, accelerating and deflecting them using electric and magnetic fields. A type of sector mass spectrometer, it was developed by Ernest O. Lawrence at the Radiation Laboratory at the University of California as

Interaction

- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact page

Tools

- What links here

RDBMS 에서는 데이터를 테이블 형태로 저장합니다.

열을 기준으로 인덱스를 만듭니다.
책의 맨 앞에 있는 제목 리스트와 같습니다.

DOC	TEXT
1	The quick brown fox jumps over the lazy dog
2	Fast jumping rabbits

검색엔진에서는 inverted index 라는 구조로 저장합니다.

RDBMS 와 반대 구조입니다.

텍스트를 다 뜯어서 검색어 사전을 만듭니다. (Term 이라고 합니다)
책의 맨 뒤에 있는 페이지를 가리키는 키워드 같습니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
Fast	2	jumps	1
The	1	lazy	1
brown	1	over	1
dog	1	quick	1
fox	1	rabbits	2
jumping	2	the	1

실제로는 이렇게 저장됩니다.

텍스트를 저장할 때 몇가지 처리 과정을 거칩니다.
이 과정을 텍스트 분석 (Text Analysis) 라고 합니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
brown	1	lazi	1
dog	1	over	1
fast	1 , 2	quick	1 , 2
fox	1	rabbit	2
jump	1 , 2		

텍스트 분석 과정

문장을 분리합니다. 이 과정을 Tokenizing 이라고 합니다.
보통은 **Whitespace Tokenizer** 가 사용됩니다.

TEXT

The quick brown fox jumps over the lazy dog

Fast jumping rabbits



TOKEN (TERM)	TOKEN (TERM)	TOKEN (TERM)	TOKEN (TERM)
Fast	dog	jumps	quick
The	fox	lazy	rabbits
brown	jumping	over	the

텍스트 분석 과정

TOKENIZED 된 Term 들을 가공합니다. 이 과정을 Token Filtering 이라고 합니다.
먼저 Lowercase Token Filter로 대소문자를 변환 합니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
Fast → fast	2	jumps	1
The → the	1	lazy	1
brown	1	over	1
dog	1	quick	1
fox	1	rabbits	2
jumping	2	the	1

텍스트 분석 과정

토큰을 (보통 ascii 순서로) 재 정렬합니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
brown	1	lazy	1
dog	1	over	1
fast	2	quick	1
fox	1	rabbits	2
jumping	2	the	1
jumps	1	the	1

텍스트 분석 과정

불용어(stopwords, 검색어로서의 가치가 없는 단어들)를 제거합니다.

a, an, are, at, be, but, by, do, for, i, no, the, to ... 등등

Stop Token Filter 가 사용됩니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
brown	1	lazy	1
dog	1	over	1
fast	2	quick	1
fox	1	rabbits	2
jumping	2	the	4
jumps	1	the	4

텍스트 분석 과정

형태소 분석 과정을 거칩니다. 보통 ~s, ~ing 등을 제거하는 과정입니다.

보통 **Snowball Token Filter** 를 사용합니다.

한글은 의미 분석을 해야 해서 좀 더 복잡합니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
brown	1	lazy → lazi	1
dog	1	over	1
fast	2	quick	1
fox	1	rabbits → rabbit	2
jumping → jump	2		
jumps → jump	1		

텍스트 분석 과정

jumping, jumps가 jump로 똑같이 바뀌었으므로 토큰을 병합 해 줍니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
brown	1	lazi	1
dog	1	over	1
fast	2	quick	1
fox	1	rabbit	2
jump	1 , 2		

텍스트 분석 과정

동의어를 처리합니다.

Synonym Token Filter 를 이용해 동의어 사전을 정의할 수 있습니다.

TOKEN (TERM)	DOC	TOKEN (TERM)	DOC
brown	1	lazi	1
dog	1	over	1
fast	1 , 2	quick	1 , 2
fox	1	rabbit	2
jump	1 , 2		

_analyze API

분석된 텍스트를 미리 볼 수 있습니다.

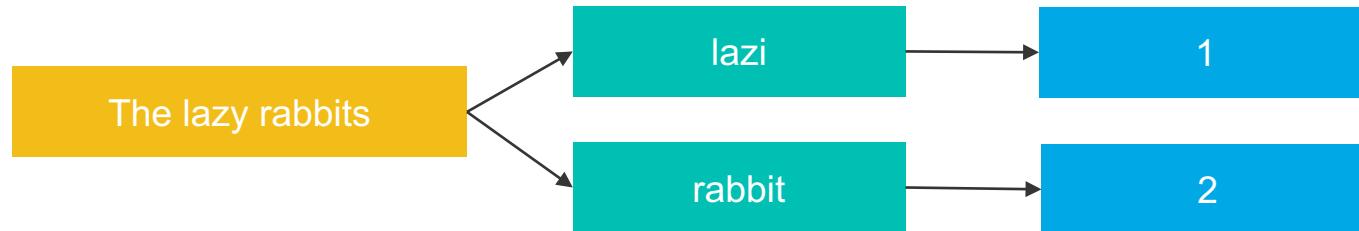
analyzer 는 1개의 tokenizer 와 n개의 token filter로 구성됩니다.

```
curl -XPOST "http://localhost:9200/_analyze?pretty" -d '  
{  
  "tokenizer": "whitespace",  
  "filter": ["lowercase", "stop", "snowball"],  
  "text": [  
    "The quick brown fox jumps over the lazy dog",  
    "Fast jumping rabbits"  
  ]  
}' -H 'Content-Type: application/json'
```

검색 과정

검색어도 똑같이 텍스트 처리를 합니다.

“The lazy spiders” 라고 검색하면



검색엔진은

	RDBMS	검색엔진
데이터 저장 방식	정규화	역정규화
전문(Full Text) 검색 속도	느림	빠름
의미 검색	불가능	가능
Join	가능	불가능
수정 / 삭제	빠름	느림

텀(Term) 확인

termvectors 를 이용해서 저장된 데이터의 확인이 가능합니다.

```
curl -XGET "http://localhost:9200/books/book/1/_termvectors?fields=author&pretty"
```

```
"terms": {  
    "shakespeare": {  
        "term_freq": 1,  
        "tokens": [ ... ]  
    },  
    "william": {  
        "term_freq": 1,  
        "tokens": [ ... ]  
    }  
}
```

match 검색

검색어도 analyze 과정을 거칩니다.

```
curl -XGET "http://localhost:9200/books/_search?pretty=true" -d'  
{  
  "query": {  
    "match": {  
      "author": "William"  
    }  
  }  
}' -H 'Content-Type: application/json'
```

term 검색

검색어의 analysis 과정을 거치지 않습니다.

검색어가 저장된 템과 정확히 일치해야 결과가 나타납니다.

```
curl -XGET "http://localhost:9200/books/_search?pretty=true" -d'  
{  
  "query": {  
    "term": {  
      "author": "William"  
    }  
  }  
}' -H 'Content-Type: application/json'
```

매핑 (Mapping)

- 각 필드별로 데이터를 저장하는 스키마 명세이며 인덱스/타입 별로 구분됩니다.
- 매핑이 없는 경우 데이터를 처음 입력하면 매핑이 자동 생성됩니다.
- 자동 생성된 매핑의 텍스트 필드는 기본적으로 standard analyzer 가 적용되며 **keyword** 타입의 멀티 필드가 생깁니다.

```
curl -XGET "http://localhost:9200/books/_mappings?pretty"
```

```
"properties" : {  
    "author" : {  
        "type" : "text",  
        "fields" : {  
            "keyword" : {  
                "type" : "keyword",  
                "ignore_above" : 256  
            }  
        }  
    }  
}
```

Keyword

- Keyword 타입은 Analyze 과정을 거치지 않은 해당 필드의 데이터의 원본 데이터입니다.
- Term 검색으로 검색시 대소문자까지 정확히 일치해야 합니다.

```
curl -XGET "http://localhost:9200/books/book/1/_termvectors?fields=author.keyword&pretty"
```

```
"terms" : {  
    "William Shakespeare" : {  
        "term_freq" : 1,  
        "tokens" : [ ... ]  
    }  
}
```

Keyword

- Term 검색으로 검색시 대소문자까지 정확히 일치해야 합니다.

```
curl -XGET "http://localhost:9200/books/_search?pretty=true" -d'  
{  
  "query": {  
    "term": {  
      "author.keyword": "William Shakespeare"  
    }  
  }  
}' -H 'Content-Type: application/json'
```

Aggregation

Search

Search by property name

Go

aggregation

Filter properties by

Property Class

- ★★★★★ 5 Stars (9)
- ★★★★ 4 Stars (55)
- ★★★ 3 Stars (122)
- ★★ 2 Stars (195)
- ★ 1 Star (10)

Price Per Night

- Less than \$75 (1)
- \$75 to \$124 (38)
- \$125 to \$199 (95)
- \$200 to \$299 (76)
- Greater than \$300 (51)

Neighborhood

- San Francisco (and vicinity)
- Bernal Heights
- Castro
- Chinatown
- Chinatown

Show more

Amenities

- High-speed Internet (384)
- Air conditioning (257)
- Swimming pool (81)
- Babysitting service (10)
- Business services (180)

Pier 2620 Hotel Fisherman's Wharf ★★★★

Fisherman's Wharf [Map](#)

Book Now - Save 15% in Fisherman's Wharf

Free WiFi & 50" TVs. Next to Cable Car stop, steps from Ghirardelli Square, Alcatraz Ferry, Pier 39. Newly renovated Lobby Level.

1-866-264-5744 Free Cancellation

38 people booked this property in the last 48 hours

Very good! 4.2/5
(1,607 reviews)

Expedia+ points applied

\$336 **\$332**

rate per night
Sponsored
 Reserve Now, Pay Later
 Earn 2,322 points

Villa Florence ★★★★

Union Square [Map](#)

1-866-267-9053 • Expedia Rate Free Cancellation

16 people booked this property in the last 48 hours

Good! 3.8/5
(391 reviews)

Expedia+ points applied

\$306 **\$237**

avg/night
member price

 Reserve Now, Pay Later
 Earn 1,661 points

Handlery Union Square Hotel ★★★★

Union Square [Map](#)

1-866-272-4856 Free Cancellation

26 people booked this property in the last 48 hours

Good! 3.9/5
(1,355 reviews)

Expedia+ points applied

\$299 **\$195**

rate per night
member price

 Reserve Now, Pay Later
 Earn 1,364 points

Stanford Court San Francisco ★★★★

Union Square [Map](#)

1-866-276-6393 Free Cancellation

Very good! 4.1/5
(1,004 reviews)

Expedia+ points applied

In high demand!

hits

Aggregation

_search API에서 query 와 함께 사용이 가능합니다.

```
curl 'localhost:9200/_search' -d '  
{  
  "query" : {  
    // query  
  },  
  "aggregations" : {    // or "aggs"  
    "aggs_name" : {  
      // a set of aggregation  
    }  
  }  
}'
```

Aggregation

_search API에서 query와 함께 사용이 가능합니다.

```
curl -XGET "http://localhost:9200/books/_search?pretty" -d'  
{  
  "query": {  
    "match_all": {}  
  },  "aggs": {  
    "authors": {  
      "terms": {  
        "field": "author.keyword"  
      }  
    }  
  }  
}' -H 'Content-Type: application/json'
```

```
...  
"aggregations" : {  
  "authors" : {  
    "doc_count_error_upper_bound" : 0,  
    "sum_other_doc_count" : 0,  
    "buckets" : [  
      {  
        "key" : "William Shakespeare",  
        "doc_count" : 2  
      },  
      {  
        "key" : "Mark Twain",  
        "doc_count" : 1  
      }  
    ]  
  }  
}
```

Aggregation

Bucket, Metric, Pipeline

- Bucket
 - 도큐먼트 집단 단위인 버킷을 생성합니다.
 - 또 다른 Bucket 또는 Metric 의 sub-aggregation을 포함합니다.
- Metric
 - 도큐먼트 집단에 대한 하나 또는 그 이상의 계산된 수치를 포함합니다.
- Pipeline
 - agg 결과에 대한 새로운 계산을 실행합니다.

Aggregation

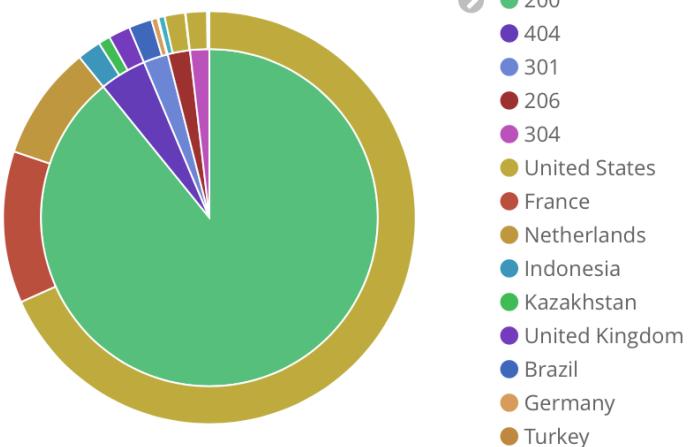
- percentile
- percentile_ranks
- cardinality
- significant_terms
- top hits
- scripted_metric
- filters
- range
- geohash
- terms
- histogram
- date_histogram
- stats
- extended stats
- min / max
- sum
- pipeline aggregations

Agg. Combination

```
curl -XGET "http://localhost:9200/books/_search?pretty" -d'
{
  "query": {
    "match_all": {}
  },
  "aggs": {
    "authors": {
      "terms": {
        "field": "author.keyword"
      },
      "aggs": {
        "pages_per_author": {
          "sum": { "field": "pages" }
        }
      }
    }
  }
}' -H 'Content-Type: application/json'
```

```
...
"aggregations" : {
  "authors" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "William Shakespeare",
        "doc_count" : 2,
        "pages_per_author" : {
          "value" : 297.0
        }
      },
      {
        "key" : "Mark Twain",
        "doc_count" : 1,
        "pages_per_author" : {
          "value" : 79.0
        }
      }
    ]
  }
}
```

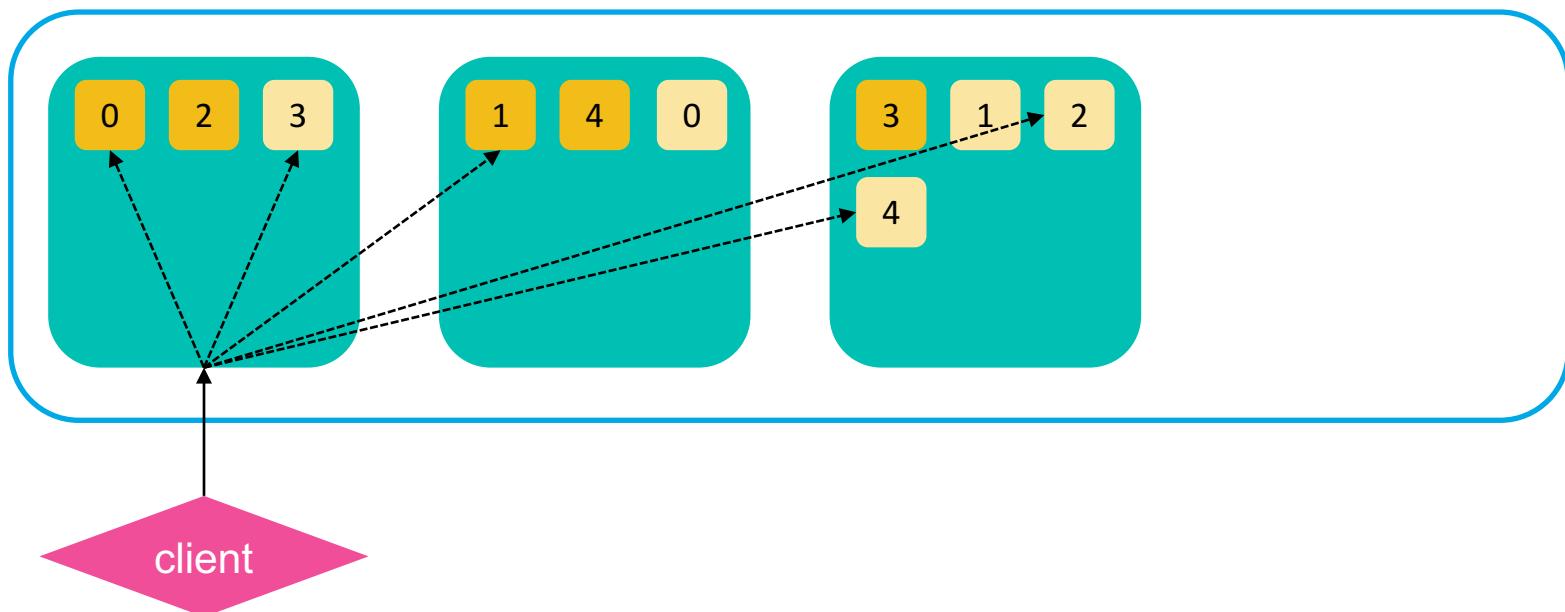
Agg. Combination in Kibana



```
"aggregations": {
  "2": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 145,
    "buckets": [
      {
        "3": {
          "doc_count": 267343,
          "buckets": [
            {
              "key": "United States",
              "doc_count": 105999,
              "score": 0.010441924596790206,
              "bg_count": 115895
            },
            {
              "key": "France",
              "doc_count": 18450,
              "score": 0.004923687375361934,
              "bg_count": 19325
            },
            {
              "key": "Netherlands",
              "doc_count": 13905,
              "score": 0.004078378127678101,
              "bg_count": 14469
            }
          ]
        },
        "key": 200,
        "doc_count": 267343
      },
      {
        "3": {
          "doc_count": 145
        }
      }
    ]
  }
}
```

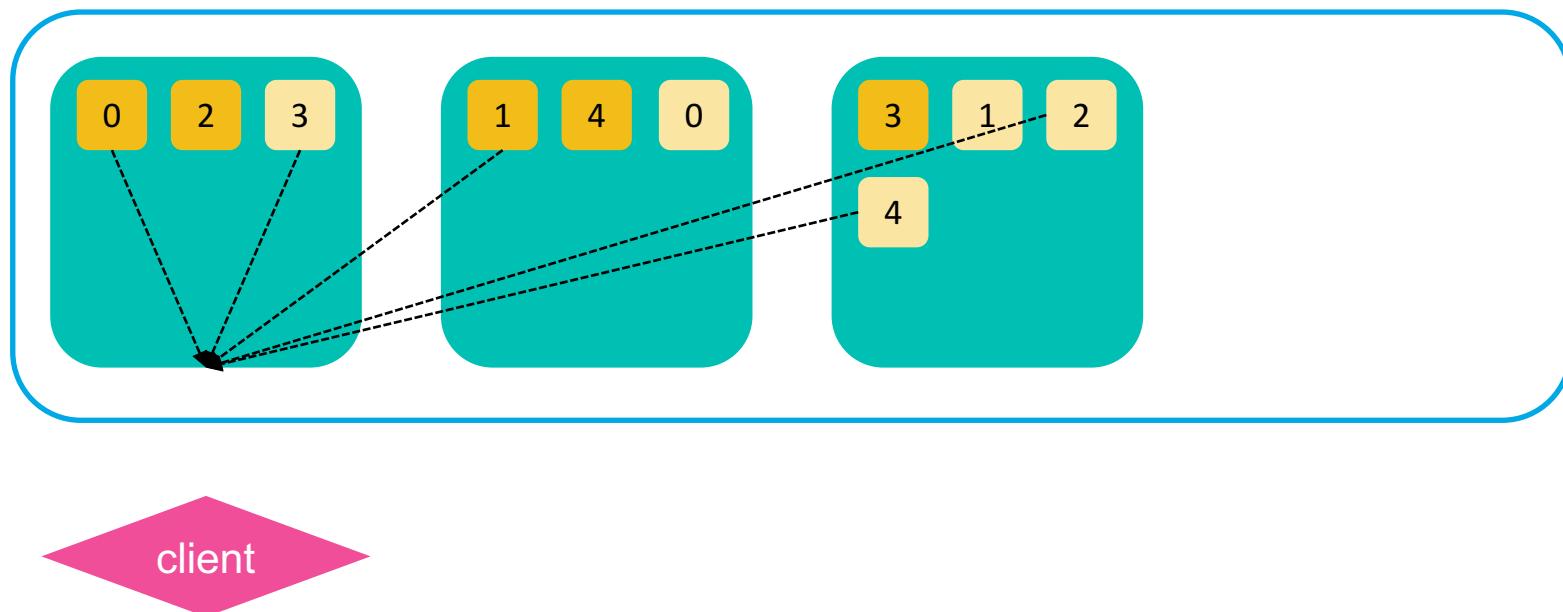
검색 과정 – 1. Query Phase

처음 쿼리 수행 명령을 받은 노드는 모든 샤드에게 쿼리를 전달합니다.
1차적으로 모든 샤드(또는 복제본에서) 검색을 실행합니다.



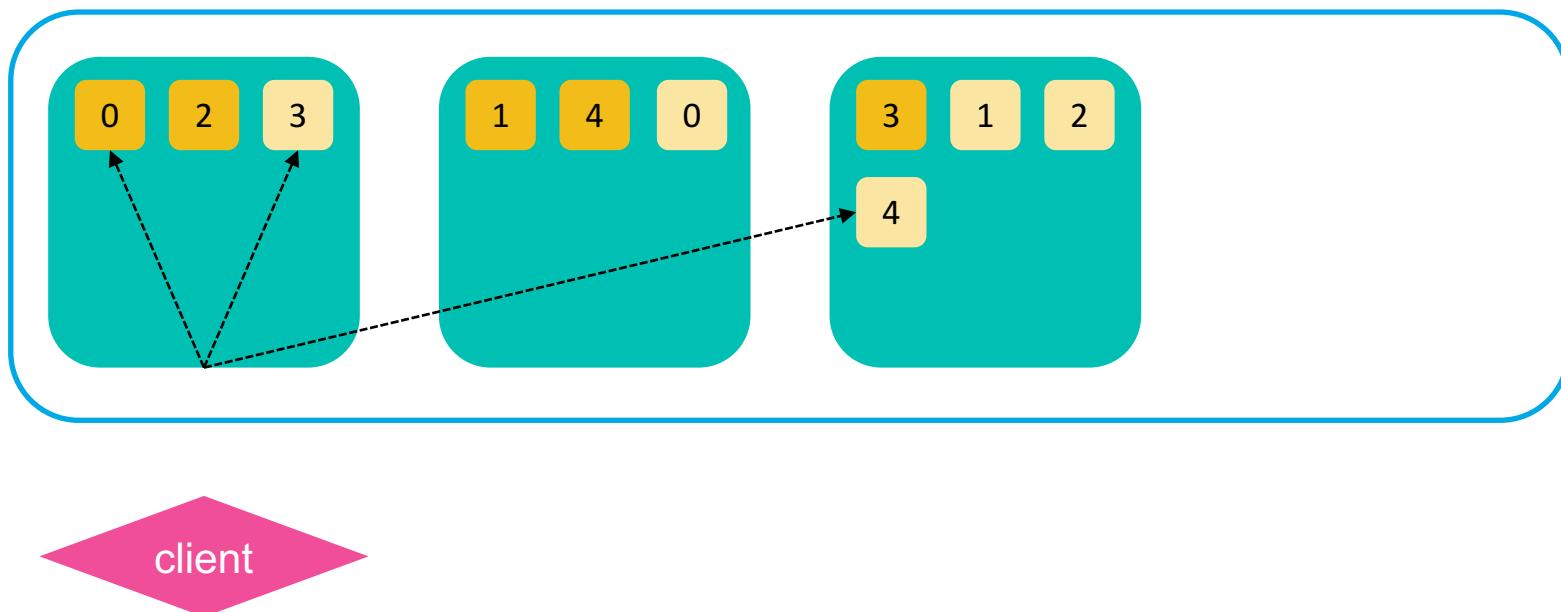
검색 과정 – 1. Query Phase

각 샤드들은 요청된 크기만큼의 검색 결과 큐를 노드로 리턴합니다.
리턴된 결과는 루씬 doc id 와 랭킹 점수만 가지고 있습니다.



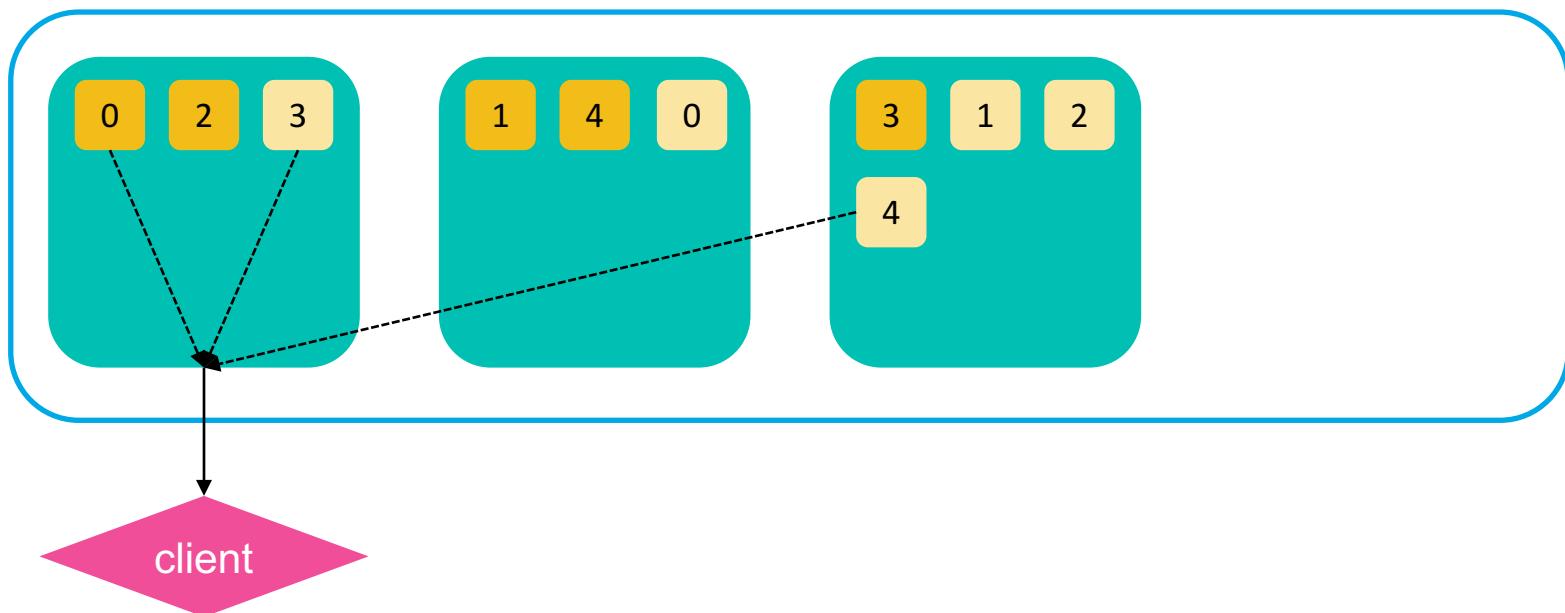
검색 과정 – 2. Fetch Phase

노드는 리턴된 결과들의 랭킹 점수를 기반으로 정렬한 뒤 유효한 색드들에게 최종 결과들을 다시 요청합니다.



검색 과정 – 2. Fetch Phase

전체 문서 내용(_source) 등의 정보가 리턴되어 클라이언트로 전달됩니다.



“

검색엔진에서는
정확한 검색을 위한
랭킹 알고리즘이
정말 정말 중요합니다.

랭킹 알고리즘

보통은 TF/IDF 를 많이 씁니다.

$$\text{tf}(t, d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

Elasticsearch 5.0 부터는
BM25 라는 알고리즘을 사용합니다.

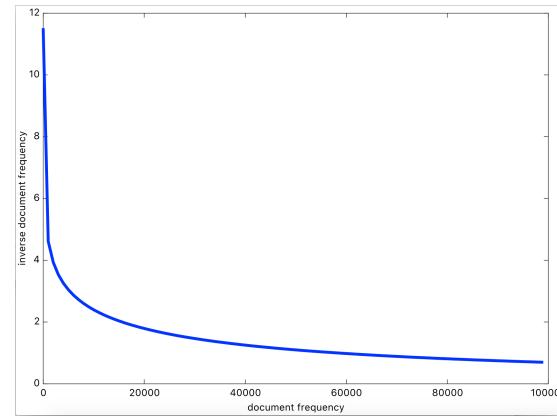
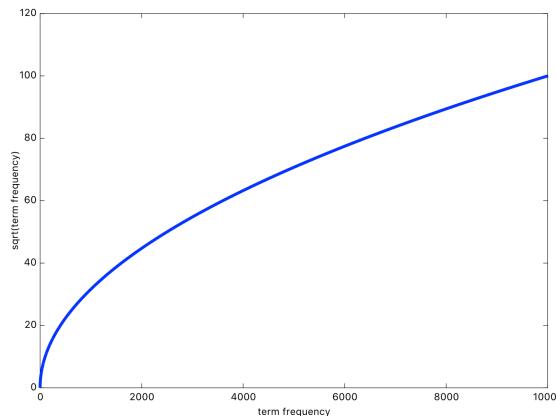
$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)},$$



TF / IDF

Term Frequency / Inverse Document Frequency

- Term Frequency
 - 찾는 검색어가 문서에 많을수록 해당 문서의 정확도가 높습니다.
- Inverse Document Frequency
 - 전체 문서에서 많이 출현한 (흔한) 단어일수록 점수가 낮습니다.

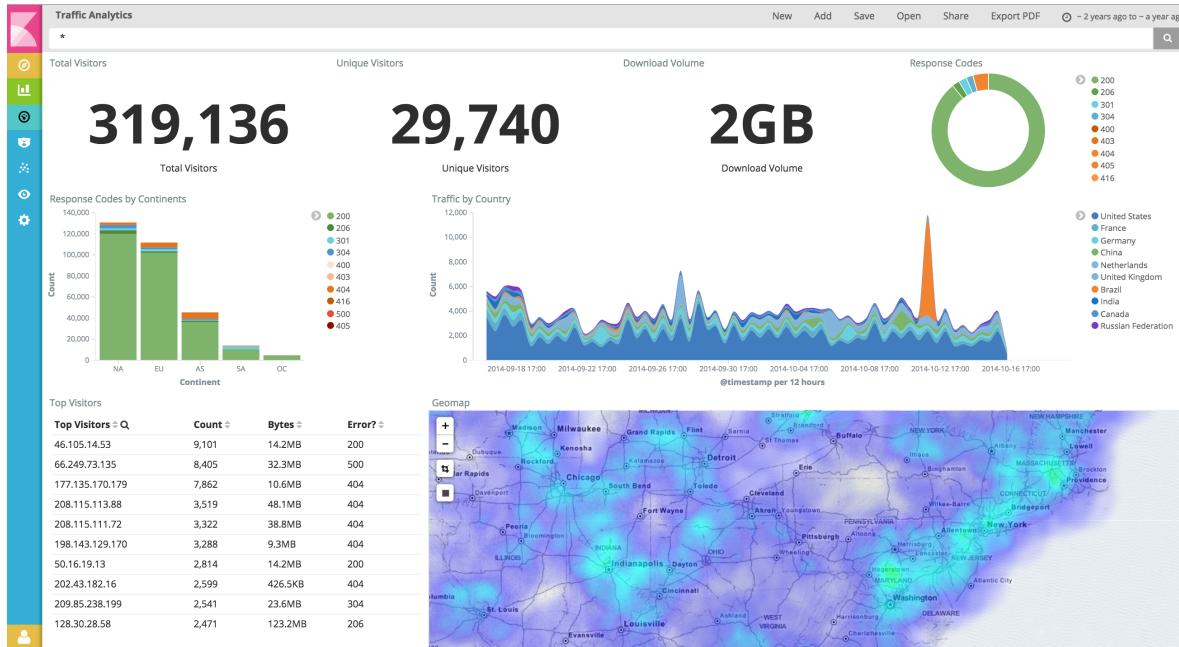


검색 랭킹이 중요한 이유

- 사용자들은 대부분 처음 나온 결과만 봅니다.
- 결과값이 큰 내용을 fetch 하는 것은 상당히 부하가 큽니다.
- 1~1,000 을 fetch 하는 것이나 990~1,000 을 fetch 하는 것이나 쿼리 작업 규모가 비슷합니다.
- 구글도 랭킹이 중요하긴 마찬가지...

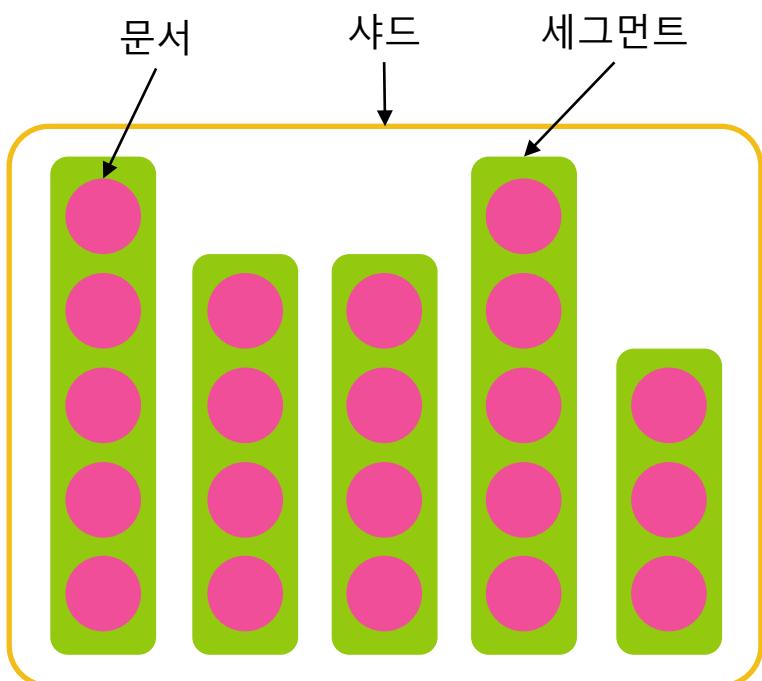
Elasticsearch is...

An open source, distributed, scalable, highly available, document-oriented, RESTful, full text search engine with real-time search and analytics capabilities



루씬 세그먼트(Segment)

Inverted Index, Doc Value, 원본 문서 등등을 저장하고 있는 단위 파일입니다.



- 루씬은 inverted index를 하나의 거대한 파일이 아니라 여러개의 작은 파일 단위로 저장합니다.
- 입력 버퍼가 가득 차거나, 1초 마다 하나씩 생성됩니다.
- 한번 생성된 세그먼트는 변경되지 않습니다. (immutable)

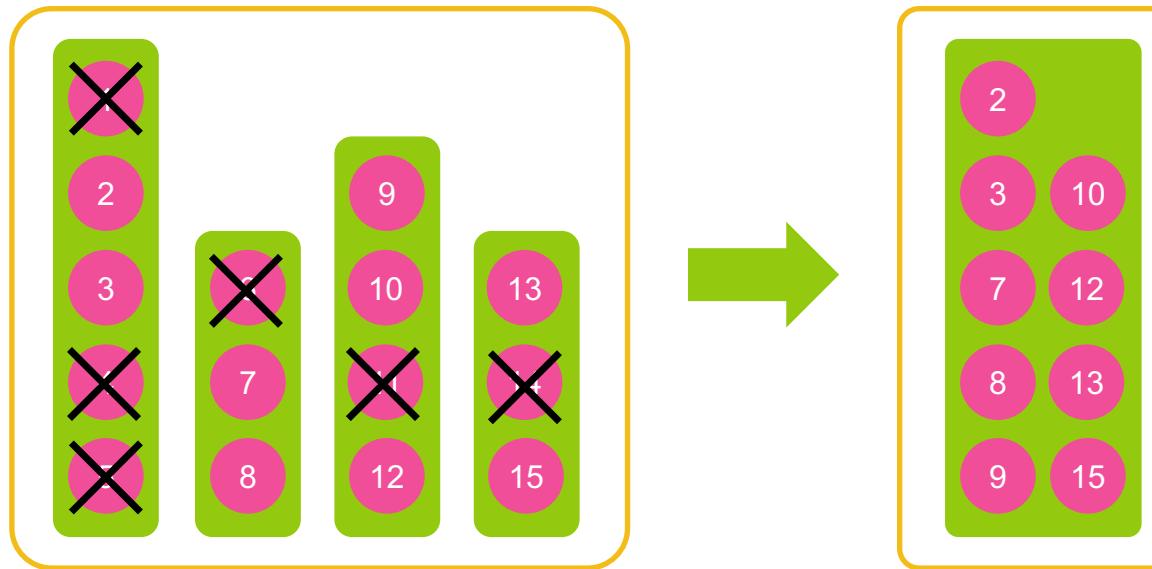
문서 변경/삭제 과정

한번 생성된 세그먼트는 변경되지 않습니다.

- update는 없습니다. 모두 delete & insert 입니다.
- 문서를 삭제하면 삭제되었다는 상태만 표시하고 검색에서 제외합니다.
- 나중에 세그먼트 병합 과정에서 삭제된 문서를 빼고 나머지 문서들을 모아 새로운 세그먼트를 만듭니다.
 - 그래서 문서를 삭제 하더라도 세그먼트 병합을 하기 전 까지 스토리지 용량은 줄어들지 않습니다.
- 세그먼트 병합은 비용이 큰 동작입니다.
 - 디스크 I/O 작업입니다.
 - 시스템이 느려집니다.
 - 가능하면 사용자들이 적은 시간에 하는것이 좋습니다.

세그먼트 병합(Segment Merge)

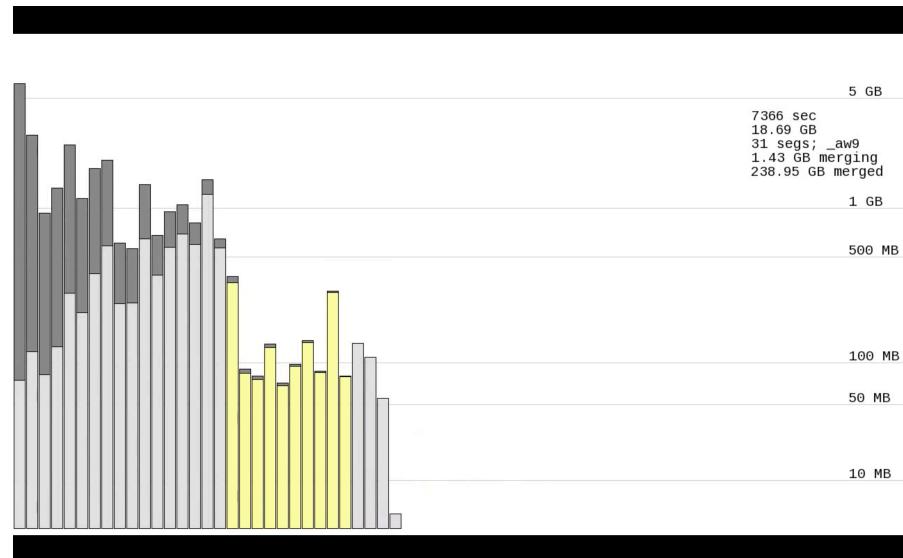
<http://blog.mikemccandless.com/2011/02/visualizing-lucenes-segment-merges.html>



세그먼트 병합(Segment Merge)

<http://blog.mikemccandless.com/2011/02/visualizing-lucenes-segment-merges.html>

- 오래된 세그먼트는 크기가 크고 최근 생성된 세그먼트는 상대적으로 크기가 작습니다.
- 오래된 문서를 삭제하는 것은 더욱 비용이 큽니다.
- 날짜별로 저장 영역(인덱스)을 구분하는 것이 바람직합니다.
 - Elasticsearch 에서는 여러 인덱스를 묶어서 검색할 수 있는 멀티테넌시를 지원합니다.



Elasticsearch 사용하시려면

- 로그는 가능하면 날짜별로 나눠서 저장하세요.
- 원본 데이터는 항상 잘 가지고 계세요. 새로 부어야 하는 경우가 많습니다.
- 세그먼트 병합은 사용하지 않는 시간에.

Snapshot / Restore - backup

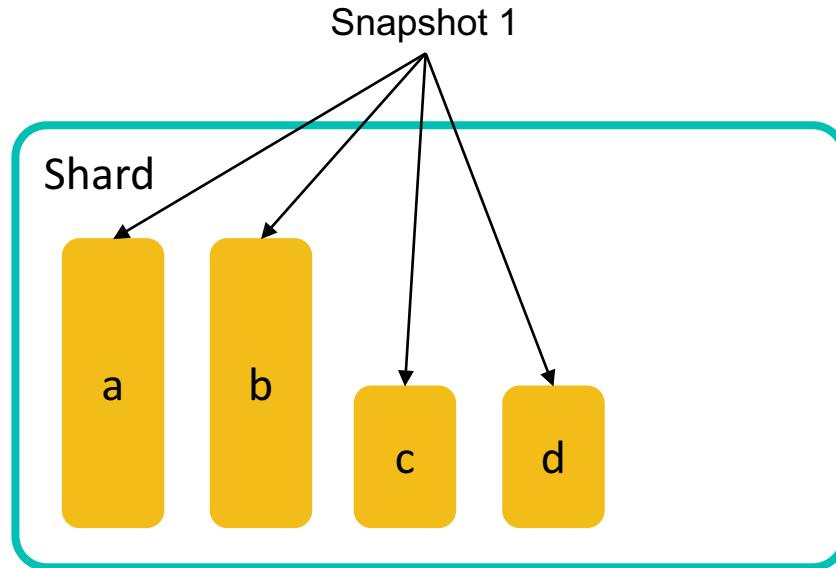
- `_snapshot`
- Segment 들을 백업 저장소에 카피
- 사용 가능한 저장소
 - File System
 - HDFS
 - AWS S3
 - Azure
- `_reindex API` 를 지원하지 않는 버전에서 cluster 간 데이터 복사에 용이

```
curl -XPUT 'localhost:9200/_snapshot/main_backup/snap1?wait_for_completion=true'
```

```
curl -XGET 'localhost:9200/_snapshot/main_backup/snap1'
```

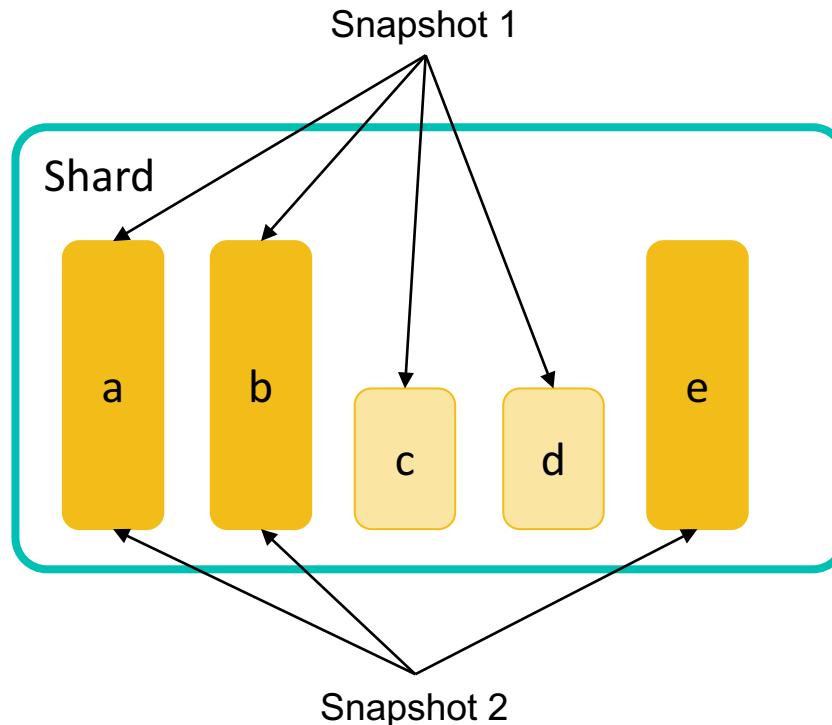
Snapshot / Restore

- 스냅샷 찍은 시점의 세그먼트 저장



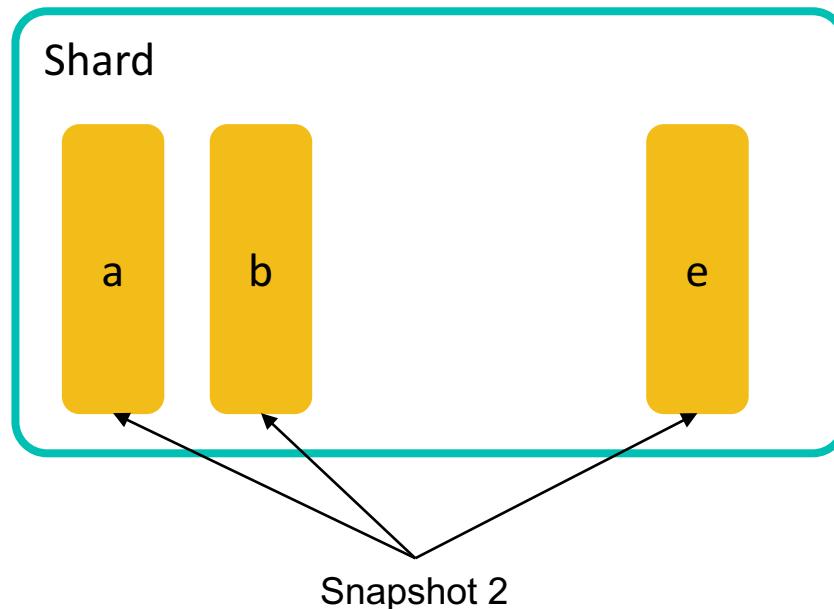
Snapshot / Restore

- 병합 & 삭제 된 세그먼트들 보관



Snapshot / Restore

- 삭제된 스냅샷에 해당되는 세그먼트 삭제



Reference

Elasticsearch 인덱싱에 대한 성능 고려 사항

- <https://www.elastic.co/blog/performance-indexing-2-0>
- <https://www.elastic.co/kr/blog/performance-considerations-elasticsearch-indexing>
 - Java 프로세스 스왑하지 않도록 설정
 - Bulk API 사용 – Logstash, Beats 는 모두 bulk 로 처리함.
 - SSD 사용 – 동시 색인, 세그먼트 병합 처리 속도 우수함
 - AWS 같은 가상화 스토리지는 로컬 스토리지에 비해 느림
 - 세그먼트 병합 정책 활용

Reference

Java Heap 메모리 설정

- <https://www.elastic.co/guide/en/elasticsearch/guide/2.x/heap-sizing.html>
 - Java Heap 메모리는 32GB 넘지 않도록 설정. (어떤 문서는 30.5GB로 나옴)
 - 32GB 이하에서 Java는 오브젝트 포인터 압축을 사용함
 - 절반의 시스템 메모리는 루씬의 캐쉬 사용을 위해 할당.
 - → 1 노드 당 최적 설정은 64GB 시스템 메모리에 32GB Java Heap
- <https://www.elastic.co/blog/a-heap-of-trouble>
 - 색인 데이터 용량 대비 Heap 메모리가 너무 적게 할당 된 경우
 - 색인 버퍼, 캐시 등의 메모리 부족
 - OutOfMemoryException 발생으로 이어질 위험
 - 색인 데이터 용량 대비 Heap 메모리가 너무 많게 할당 된 경우
 - Garbage Collector 시간이 오래 걸릴 수 있음.
 - IO 병목 발생으로 이어질 수 있음



감사합니다.

<https://www.elastic.co/kr/>

<https://www.facebook.com/groups/elasticsearch.kr>

