



# elasticsearch

Τεχνολογία Λογισμικού 2017-2018

Αναγνωστίδης Σωτήρης-Κωνσταντίνος

# Who uses it



NETFLIX



StumbleUpon

theguardian

Etsy

JUST EAT

GitHub



mozilla



flickr

accenture



# Search Engine

- Indexing of data
- Analyzing of data
  - Text Search
    - Tokenize
    - Analyze
  - Locations, Date, Score value, ...

## NoSQL

- Doesn't have a structured schema
- Drops the acid requirements

# Elasticsearch is a document oriented search engine

Insert documents

Delete documents

Retrieve documents

Analyze documents

Search documents

# Appendix

328

## Index

Canvas of life turned upside down, 68  
 "Carbonate of pork," 315  
 Carracci, the, 147  
*Casina di Banda*, 119  
 Castelletto, 265  
 Cavagnolo, 76  
 Cenere, Monte, narcissuses on, 228  
 Ceres, 161  
*Cerrea*, 133  
 Chalk, Conté, the Italian for whom this was the one thing needful, 136  
 Chalk eggs, 43  
 Chamois, foot of, 283  
 Change, repudiation of desire for sudden, 186  
 — importance of, depends on the rate of introduction, 196  
 — either the circumstances or the sufferer will, 196  
 Changes, sweeping, to be felt hereafter as vibrations, 60  
*Cheapissimo*, 165  
 Cheese and the *alpi*, 289  
 Cherries, 33, 35, 46  
 Chestnuts, 115  
 Chicory and seed onions, weary utterness in, 227  
 Children, subalpine, 301  
 — what becomes of the clever, 149  
 Chinese, the examination-ridden, 151  
 Chironico, 75  
 "Chow," 52  
 Church-going, subalpine, 303  
 Circulation of people like blood, 20  
 Ciseri, his picture at Locarno, 271  
 Civilisation, antiquity of Italian, 124  
 — stationary, of ants and bees, 195  
 Class distinction inevitable, 195  
 Classification only possible through sense of shock, 63  
 Clergy, our English, and S. Michele priests, 106  
 Cloisters at Locarno, 271  
 — at Oltrona, 258  
 Club, the, the true university, 155

Cocking, Wednesbury, 55, 305  
 Collecta, unsympathetic priest, 111  
 Colleone, Medea, 231  
 Colma di San Giovanni, 163  
 Comba di Susa, 119  
 Comfort as a moral influence, 185  
 Comic song, the landlord's, 128  
 Common sense, the safest guide, 108  
 Consistent, who ever is? 153  
 Contradictory principles, there must be a harmonious fusing of, 152  
 Converting things by eating them, 153  
 Corpses, desiccated, at S. Michele, 97  
 Cousins, my, the lower animals, 100  
 Cows fighting in farmyard, 120  
 Cricco, 125  
 Cristoforo, S., church of, at Mesocco, 208  
 — — — at Castello, 234  
 Crossing, efficacy of, 152  
 — unexpected results of, 53  
 — useless if too wide, 157  
 Crucifixion, fresco at Fusio, 140  
 Culture and priggishness, 141  
 — a mode of concealing weakness, 192  
 Current feeling, the safest guide, 108  
 Cutlets, burnt, and the waiter, 143  
 Dalpe, 38  
 Dante a humbug, 156  
 Darwin, Charles, no place for meeting, 69  
 Darwin, Erasmus, 23, 151  
 Dazio, Signor Pietro, of Fusio, 119  
 Death, no man can die to himself, 277  
 Deceit a necessary alloy of truth, 259  
 Dedomenici da Rossa, 137  
 Demand and supply, 108  
 D'Enrico, the brothers, 189  
 Dentist's show-case mistaken for relics, 43

Deportment, good technique resembles, 156  
 Desire and power, 108  
 Development of power to know our own likes and dislikes, 22  
 Devil's Bridge, 23  
 Diatonic scale, and song of birds in New Zealand, 212  
 Dirt, eating a peck of moral, 71  
*Disgrazia* and misfortune, 58  
 D'Israeli, Isaac, quotations from, 67  
 Dissenters all narrow-minded, 153  
 Distribution of plants and animals often inexplicable, 135  
 Diversion of mental images, 54  
 Duera, fresco at, 145, 221  
 Dogs, 156, 202, 260, 313  
 Doing, the only mode of learning, 151  
 Doors, how they open in time, 151  
 Doubt, "There lives more doubt in honest faith," 67  
 Downs, the South, like Monte Generoso, 230  
 Draughtsman, first business of a, 145  
 Drawing, the old manner of teaching, 150  
 Dream, my, at Lago di Cadagno, 82  
 Drunkenness and imagination, 46  
*Dunque*, 133  
 Duso, Agostino, his fresco at Sta. Maria in Calanca, 225  
 Earnestness, 142, 192  
 Eating, a mode of bigotry, 153  
 Echo at Graglia, 192  
 Edelweiss, 291  
 Electricity and Alpine roads, 60  
 Elephant brays a third, 233  
 "Elongated" honey, 293  
 Embryonic stages, the artist must go through, 148  
 Eponym, Lord Beaconsfield's, 23, 141  
 English as tract-distributors, 65  
 — language, its ultimate supremacy, 41

English priests and Italian, 106  
 — why introspective, 18  
 Equilibrium only attainable at the cost of progress, 195  
*Eritis*, a panic concerning, 204  
 Eternal punishment, 111, 196  
 Eusebius, St., 178  
 Evolution and illusion, 43  
 — essence of, consists in not shocking too much, 110  
 Extreme, every, an absurdity, 153  
 Faido, 22  
 Faith, doubt lives in honest, 67  
 — more assured in the days of spiritual Saturnalia, 68  
 — foundations of our system based on, 107, 277  
 — and reason, 108  
 — catholic, of protoplasm, 152  
 — a mode of impudence, 283  
 Falsehood turning to truth, 71  
 Famine prices at Locarno, 276  
 Feeling, current, the safest guide, \* 108  
 Fertile, rich and poor rarely fertile *inter se*, 195  
 Fires, how Italians manage their, 117  
 Fishmonger choosing a bloater, 23  
 Flats and sharps, a maze of metaphysical, 21  
 Fleet Street, beauties of, 19  
 Flowers, names of, 291  
 Fossil-soul, 234  
 Foundations of action lie deeper than reason, 107  
 — of a durable system laid on faith, 277  
 Francis, St., and Insurance Co.'s plate, 191  
 Friction, which prevents the unduly rapid growth of inventions, 60  
 Fucine, 166  
 Fun, Italian love of, 243  
 Fusing and confusing of ideas and structures, 44  
 — faith and reason, necessity of, 108

## Index

329

# Origin

Elastic search is built on the of a search software called lucene

Underline data structure;

- inverted index, designed to allow very fast full-text searches

Organizes documents into the inverted index

# Example (documentation)

Sentences break into terms or tokens.

1. The quick brown fox jumped over the lazy dog
2. Quick brown foxes leap over lazy dogs in summer

Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

# What are documents

Most entities or objects can be serialized into JSON!

With fields and values

- Simple
- Fast
- Easy to read

```
{
  "name":      "John Smith",
  "age":       42,
  "confirmed": true,
  "join_date": "2014-06-01",
  "home": {
    "lat":     51.5,
    "lon":     0.1
  },
  "accounts": [
    {
      "type": "facebook",
      "id":   "johnsmith"
    },
    {
      "type": "twitter",
      "id":   "johnsmith"
    }
  ]
}
```



# Insert

Inserting = indexing

```
PUT /{index}/{type}/{id}
{
  "data1": data1,
  "data2": data2
}
```

# Methods

GET

HEAD

PUT

POST

DELETE

It's all about REST semantics.

POST basically that you are posting a request which is going to modify the server state

```
POST index/type
{
  "foo": "bar"
}
```

will generate an `_id` server side and will index the document with this `_id`.

PUT is used to send a resource to the server.

```
PUT index/type/id
{
  "foo": "bar"
}
```

will put or update a document named `index/type/id` in the server.

# Index structure - schema

Mappings; field and its type

Settings; shards, replicas, ...

```
PUT localhost:9200/vehicles
{
  "settings": {
    ...
  },
  "mappings": {
    ...
  }
}
```

# Elasticsearch structure

Elasticsearch is distributed technology

One elasticsearch cluster is composed of multiple nodes

Nodes has different replicas

Physical representation: shards

# Elasticsearch structure

Shards; splits of the index

Every node has a primary shard and replicas of others.

Document goes to each node based on its id after passing through a hash function. By this hash value the new document is sent to the node where the primary shard can be found

After a successful index, replicate to the other replicas

Every node knows where all the documents are in the cluster. Each request goes to the node that has the primary shard.

# Elasticsearch structure

On get request (read request) the coordinator balances the requests equally to all nodes.

The requests are balanced in a round robin way.

# Shards

Shard; lucene index

Elasticsearch makes lucene distributed

Each shard is a container of inverted indexes. It has multiple segments

Each segment is an inverted index.

# Indexing

Analysis (more on this later)

The inverted index is created.

This information goes to a buffer.

Once this buffer gets filled up it gets committed to a segment

Segments are immutable. Once they are populated they can't be changed



# Review

**cluster** – An Elasticsearch cluster consists of one or more nodes and is identifiable by its cluster name.

**node** – A single Elasticsearch instance. In most environments, each node runs on a separate box or virtual machine.

**index** – In Elasticsearch, an index is a collection of documents.

**shard** – Because Elasticsearch is a distributed search engine, an index is usually split into elements known as shards that are distributed across multiple nodes. Elasticsearch automatically manages the arrangement of these shards. It also rebalances the shards as necessary, so users need not worry about the details.

**replica** – By default, Elasticsearch creates five primary shards and one replica for each index. This means that each index will consist of five primary shards, and each shard will have one copy.

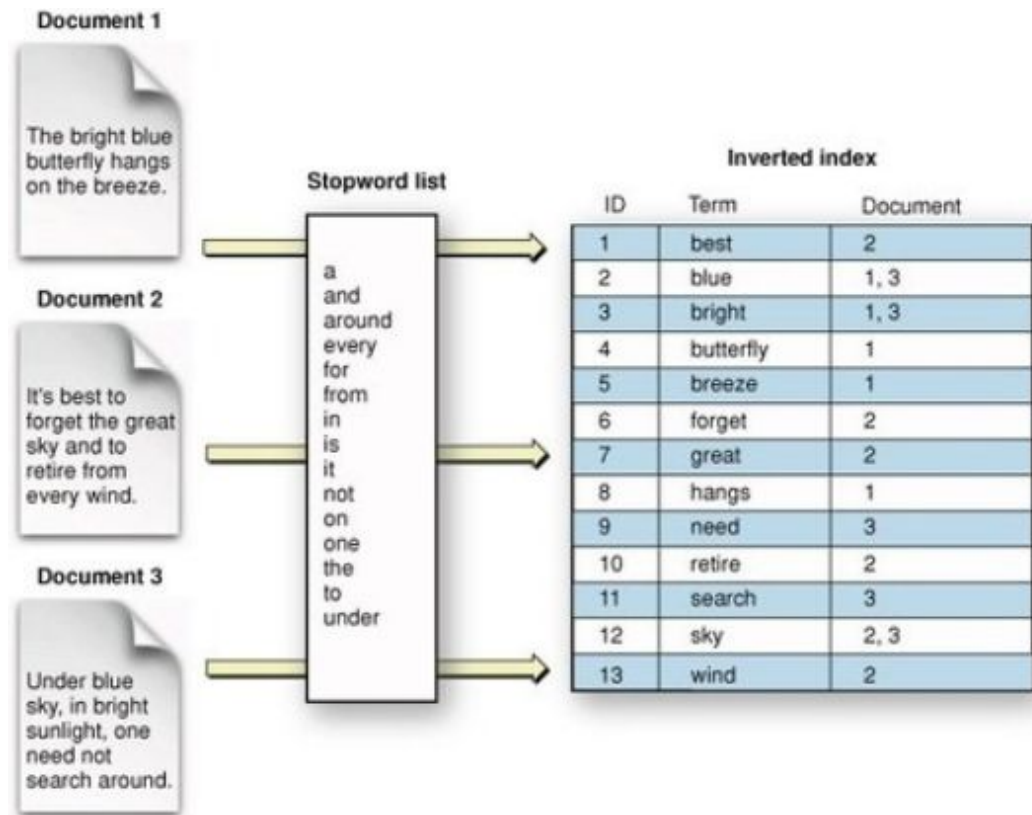
The more segments there are, the longer each search takes. So Elasticsearch will merge a number of segments of a similar size ("tier") into a single bigger segment, through a background merge process. Once the new bigger segment is written, the old segments are dropped. This process is repeated on the bigger segments when there are too many of the same size.

Segments are immutable. When a document is updated, it actually just marks the old document as deleted, and indexes a new document. The merge process also expunges these old deleted documents.

# Analysis

Document is made into  
inverted index and stored into a  
shard segment

Create tokens so they can be  
found



# Inverted index

- Tokenizer
- Filtering
  - Remove stop words (non important words)
  - Case insensitive
  - Stemming
  - Synonyms

In searching the same process happens

# Index

## Create index

```
PUT vehicles
{
  "settings": {
    "number_of_shards": 5
    , "number_of_replicas": 1
  },
  "mappings": {}
}
```

## Create mapping

```
PUT vehicles/_mapping/cars
{
  "properties": {
    "name": {
      "type": "text",
      "analyzer": "standard"
    }
  }
}
```

# GET vehicles

Also

New fields are automatically indexed  
by finding the datatype

... unless you set dynamic to false  
(error will be thrown)

```
{
  "vehicles": {
    "aliases": {},
    "mappings": {
      "cars": {
        "properties": {
          "name": {
            "type": "text",
            "analyzer": "standard"
          }
        }
      }
    },
    "settings": {
      "index": {
        "creation_date": "1519937080703",
        "number_of_shards": "5",
        "number_of_replicas": "1",
        "uuid": "bWYkqIasRvqGJfY_st8w5g",
        "version": {
          "created": "6000099"
        }
      },
      "provided_name": "vehicles"
    }
  }
}
```

# Datatypes

string

text and keyword

## Numeric datatypes

long, integer, short, byte, double, float, half\_float, scaled\_float

## Date datatype

date

## Boolean datatype

boolean

## Binary datatype

binary

## Range datatypes

integer\_range, float\_range, long\_range, double\_range, date\_range

Complex (array, object, nested)

Geo (geopoint geoshape)

Specialised (ip, completion, token count, ...)

# Analizers

1. Standard Analyzer
2. Simple Analyzer
3. Whitespace Analyzer
4. Stop Analyzer
5. Keyword Analyzer
6. Pattern Analyzer
7. Language Analyzers
8. Custom analyzers ...

# Query DSL (Domain Specific Language)

Query context

Relevancy score

Filter context



# Query Examples

Different score base on: number of letters in each field, frequency of the word in the document, in all documents.

```
GET vehicles/_search
{
  "query": {
    "match": {
      "name": "honda"
    }
  }
}
```

```
GET vehicles/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": {
          "FIELD1": "TEXT1"
        } },
        { "match": {
          "FIELD2": "TEXT2"
        } }
      ]
    }
  }
}
```

Other key words:

Exists, must\_not, multi\_match

should, minimum\_should\_match

Numbers: range, gt, lt, gte, lte

Field boosting..

# Aggregations

- Size, from
- Sort results by field
  - descending
  - ascending
- Stats
- Combination with queries, filters

```
GET vehicles/cars/_search
{
  "aggs": {
    "popular_manufacturer": {
      "terms": {
        "field": "manufacturer.keyword",
        "size": 10
      },
      "aggs": {
        "avg_price": {
          "avg": {
            "field": "price"
          }
        }
      }
    }
  }
}
```



elastic



**logstash**

kibana

# References

<https://www.elastic.co/guide/index.html>

<https://stackoverflow.com>