

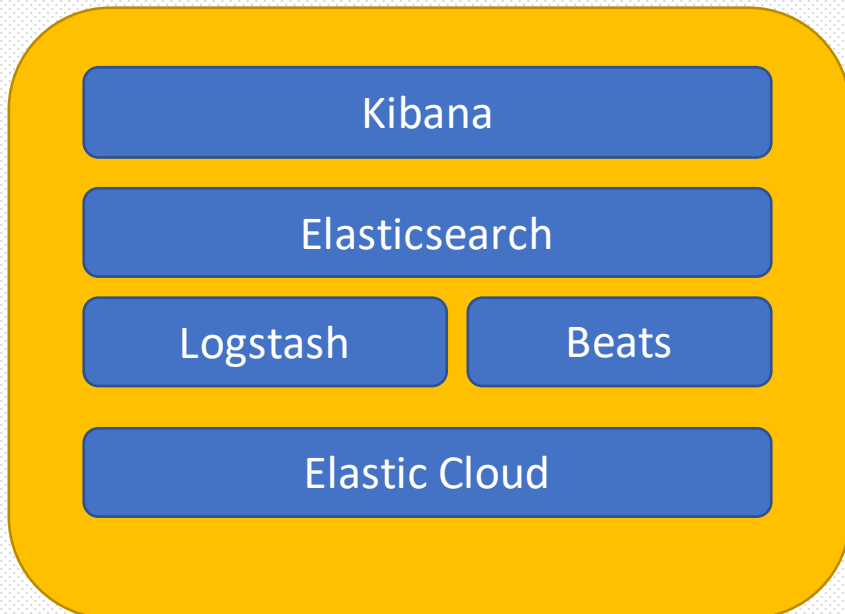
Asp.Net Core + Elastic Stash

Elastic Search nedir ?

Java Programlama dili ile yazılmış, open source olarak geliştirilmiş, distributed search(horizontally scalable) ve analytics engine'dir.

Elastic Stack'in en önemli bileşenidir.

Elastic Stack



Asp.Net Core + Elastic Stash

Elastic Lucene nedir ?

Java Programlama dili ile yazılmış, open source olarak geliştirilmiş, distributed search(horizontally scalable) ve analytics engine'dir.

Elastic Stack'in en önemli bileşenidir.

Asp.Net Core + Elastic Search

Elastic Stack Components

- Elastic Search
- Logstash
- Beats
- Kibana
- X-Path

Asp.Net Core + Elastic Search

Features

- Schemaless
- Searching
- Analytics
- Rich Library
- Fault-tolerant

Asp.Net Core + Elastic Search

Use Cases

- Search
- Logs
- Metric Analytis
- Website Search

Asp.Net Core + Elastic Search

Kibana ve Elasticsearch Kurulumu

Docker Hub

Container olarak ayağa kaldırma

Asp.Net Core + Elastic Search

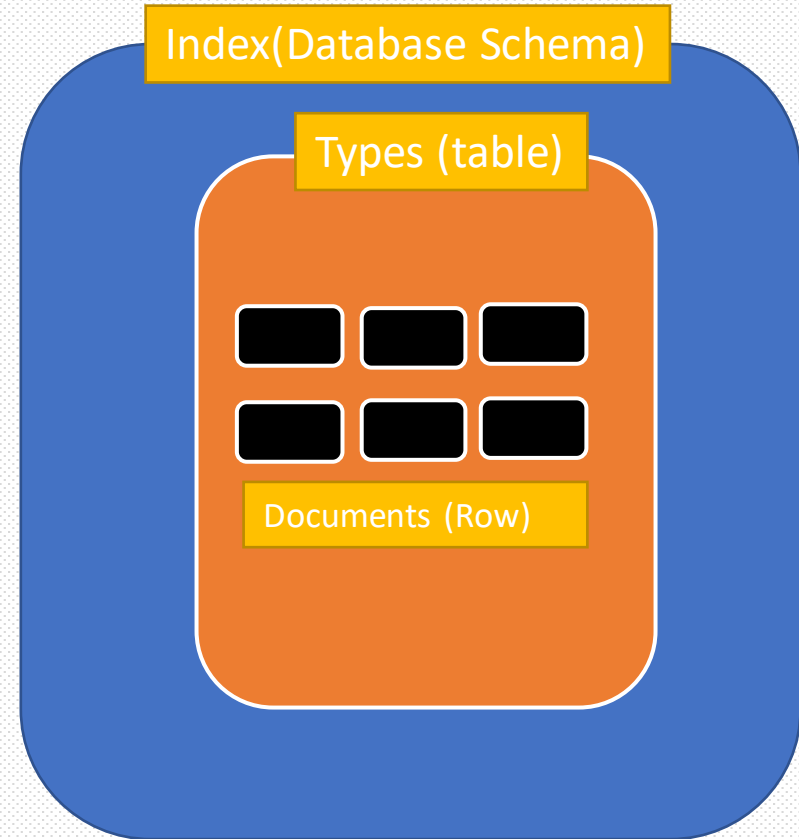
Kibana Dev Tools

Restful isteklerini gerçekleştirme

Asp.Net Core + Elastic Search

Kibana Core Concepts

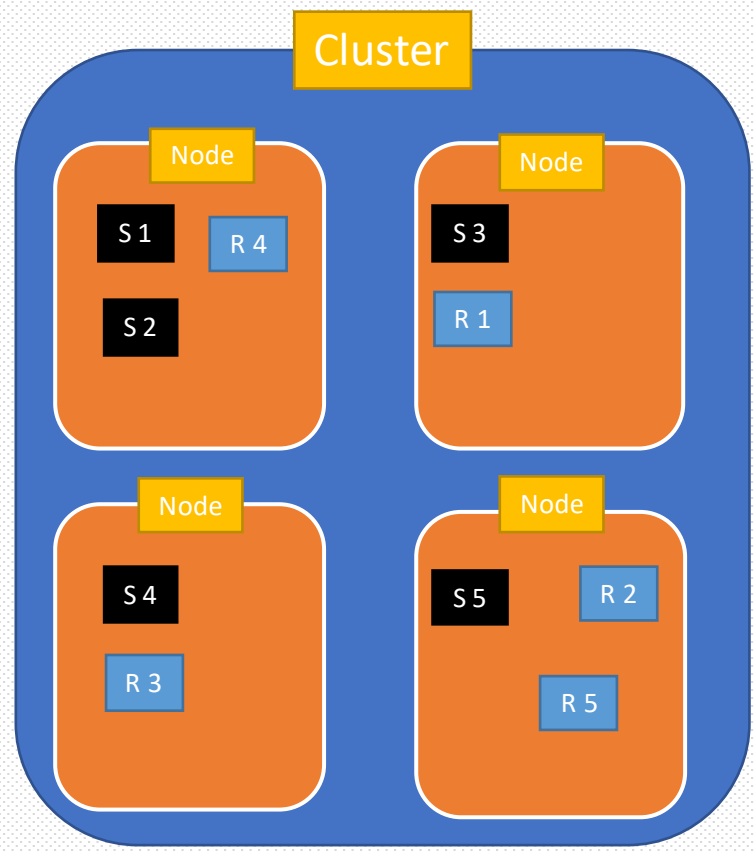
Index
Type
Document
Cluster
Node
Shard
Replica
Mapping
Inverted indexes



Asp.Net Core + Elastic Search

Kibana Core Concepts

- Index
- Type
- Document
- Cluster
- Node
- Shard
- Replica
- Mapping
- Inverted indexes



Asp.Net Core + Elastic Search

Elastic Search Data Types

String (Text-Keyword)

Numeric (byte,short,integer,long)

Date

Boolean

Binary

Array datatypes

Asp.Net Core + Elastic Search

Inverted indexes

Document Id	Document
1	Bugün güzel bir gün
2	Nasılsın Fatih, güzel görünüyorsun

Term	Frequency	Documents
bugün	1	1
güzel	2	1,2
bir	1	1
gün	1	1
nasılsın	1	2

Asp.Net Core + Elastic Search

Terminology

RDMS	Elastic Search
Datatable schema	Index
Table	Type
Row	Document
Column	Field
Schema	Mapping

Asp.Net Core + Elastic Search

CRUD Operations

Index (Save)

Get

Update

Delete

Asp.Net Core + Elastic Search

Structured Data Search (Term-level-query)

Range Query

Exist Query

Fiyatı 100'den büyük olanlar

Id'si 10'dan büyük olan datalar

Asp.Net Core + Elastic Search

Aggregations types

1. Bucket aggregations (group by)
2. Metric aggregations (avg,sum)
3. Matrix aggregations
4. Pipeline aggregations

Asp.Net Core + Elastic Search

E-commerce sample data load

Analiz yapabilmek için örnek bir data yüklemek

Asp.Net Core + Elastic Search

Metric aggregations

1. Sum
2. Min
3. Max
4. Avg
5. Stats
6. Cardinality

Asp.Net Core + Elastic Search

Bucket aggregations

1. String Data (keyword type /Terms aggregations)
2. Numeric Data (Histogram-Range)

Asp.Net Core + Elastic Search

Log Data Analyzing

Log = Timestamp + Data

1. Troubleshooting
2. Application behavior
3. Auditing

Asp.Net Core + Elastic Search

Log Data Analyzing

Logstash Pipe



Elasticsearch | Get Started

Kurs boyunca neler öğreneceğiz?

Elasticsearch
Architecture

Elasticsearch
API

Elasticsearch
With .Net
Core

Devops

Elasticsearch | Get Started



elasticsearch

What is Elasticsearch?

Elasticsearch is an open source search and analytics engine. Developed in Java it is an ultra-fast, highly available search engine built on the popular full-text library, Apache Lucene

What is Apache Lucene?

Apache Lucene is high performance and full-text searching library developed in Java
Apache Lucene is not complete application that you can download

Elasticsearch | Get Started

Search Engine Capability

- Search for unstructured and structured data
- did-you-mean recommendations
- Forgiveness for users' spelling mistakes
- Search capabilities on geolocations and GeoPoints
- Easy scalability based on the fluctuating demands
- speedy indexing and search capabilities
- Architecture that's a high-availability and fault-tolerant distributed system

Elasticsearch | Get Started

Elasticsearch Cores (Elastic, the company behind the Elasticsearch)

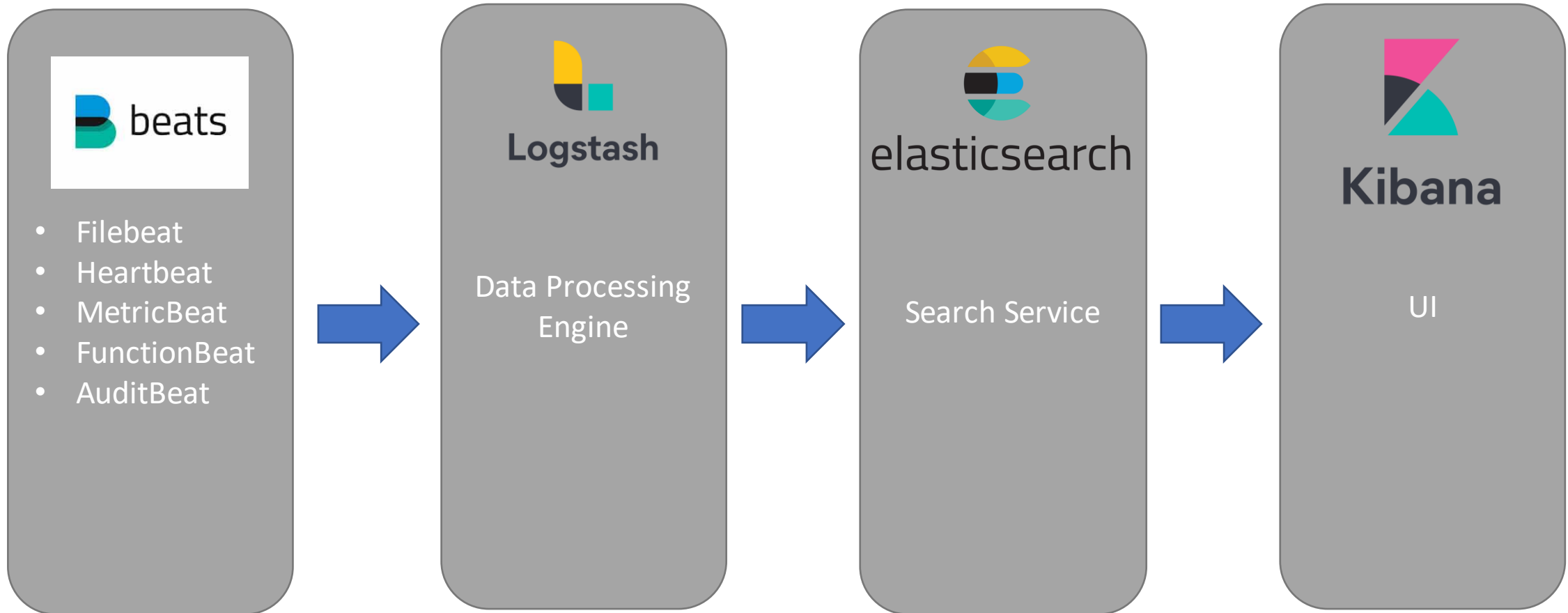
Elastic Enterprise Search

Elastic Observability

Elastic Security

Elasticsearch | Get Started

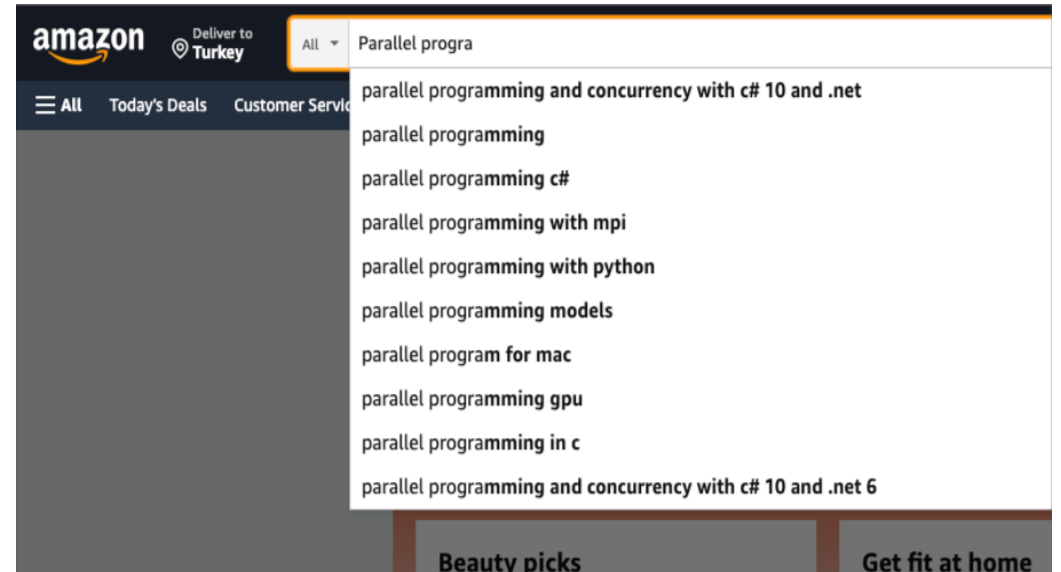
Elastic Stack



Elasticsearch | Get Started

Use Case

- Search Engine (Application Search, Enterprise Search, Site Search-SAAS)
- Business Analytics
- Logging and Monitoring



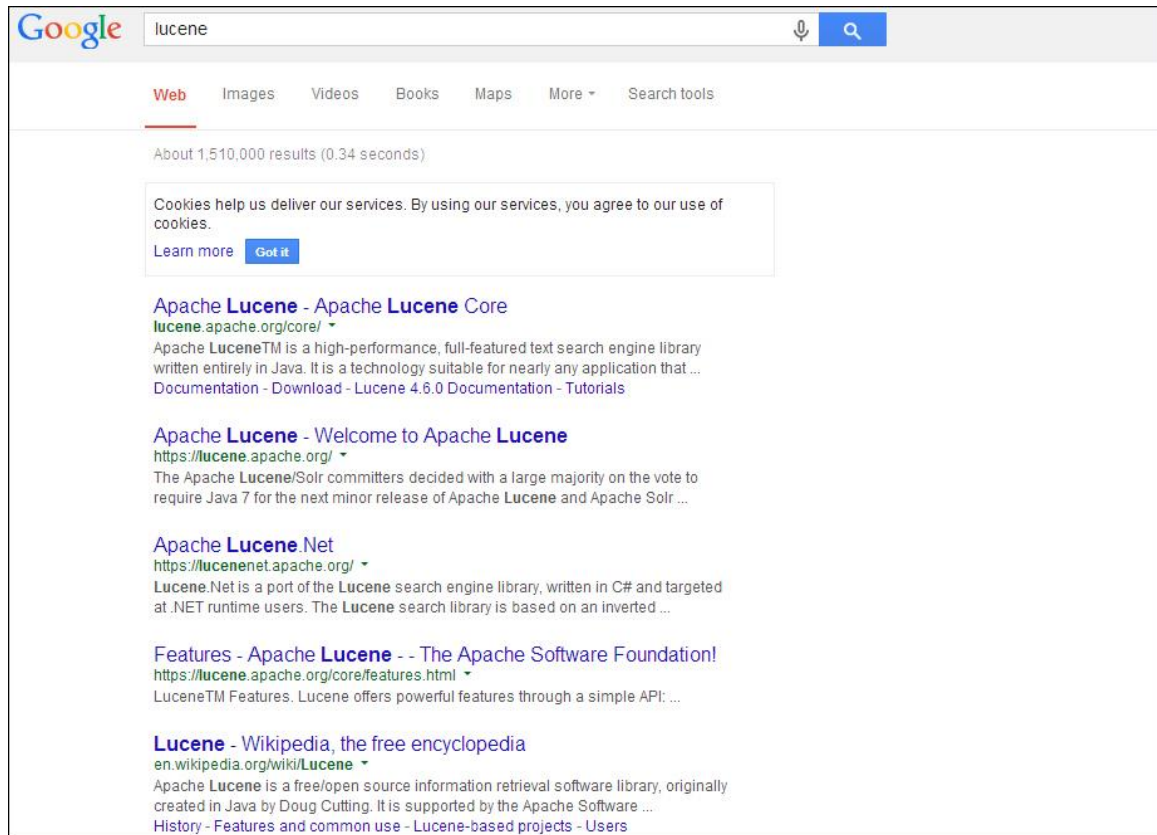
Elasticsearch | Get Started

Features

- Search engine based on the NoSQL model
- Search on structured and unstructured data
- Analytics
- Schema free engine (JSON documents)
- RESTful (communication)

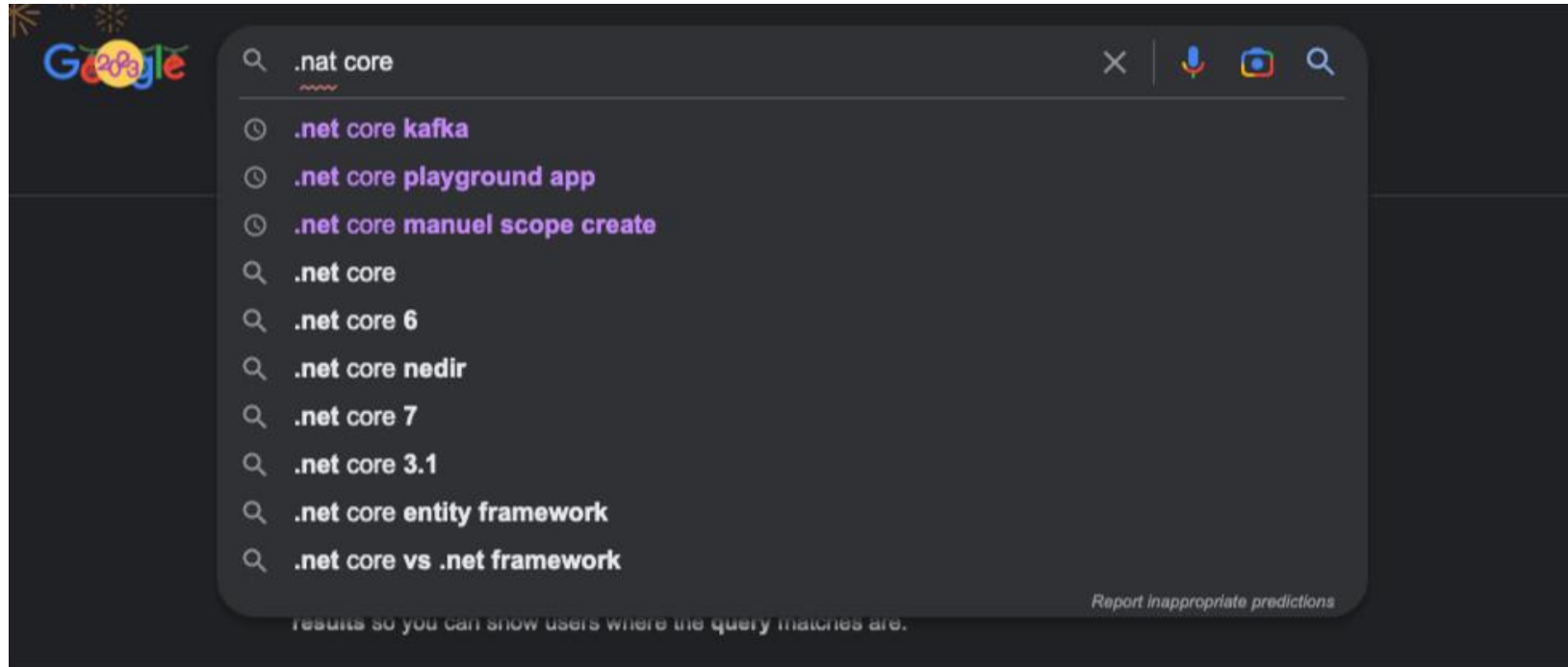
Elasticsearch | Get Started

Highlighting result



Elasticsearch | Get Started

Fuzzy queries (Levenshtein edit distance algorithm) (tamamlanmadı)



Elasticsearch | Architecture

Building Block

- Document
- Index
- Data Stream
- Shard
- Replicas
- Node
- Cluster
- Inverted Index
- Relevancy

Elasticsearch | Architecture

Document

- The most basic information indexed by Elasticsearch
- Elasticsearch expects documents to be in json format

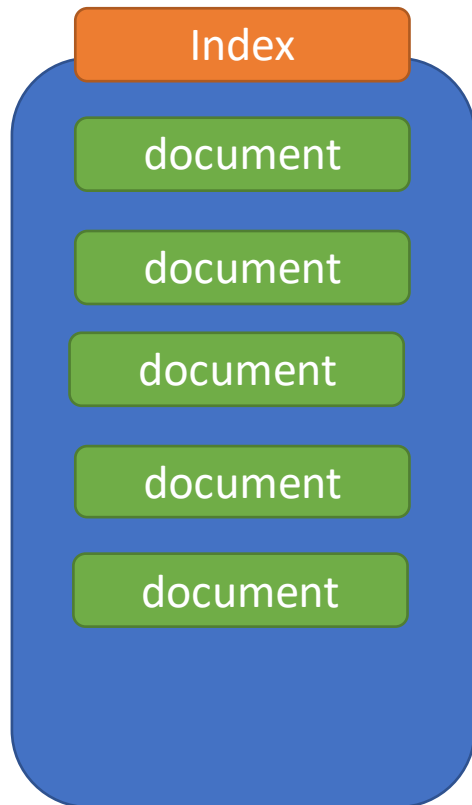
```
{  
  "id":1  
  "Name:"Blue Pencil"  
  "Price" : 100  
  "Category":"Pencils"  
}
```

Elasticsearch json datayı okur ve onu kendi özel data tiplerine dönüştürür.

Elasticsearch | Architecture

Index

A logical collection of documents(Logical groups)



Elasticsearch | Architecture

Data Streams

Time-based data (example: logs)

Microservice1.logs.12-12-2022

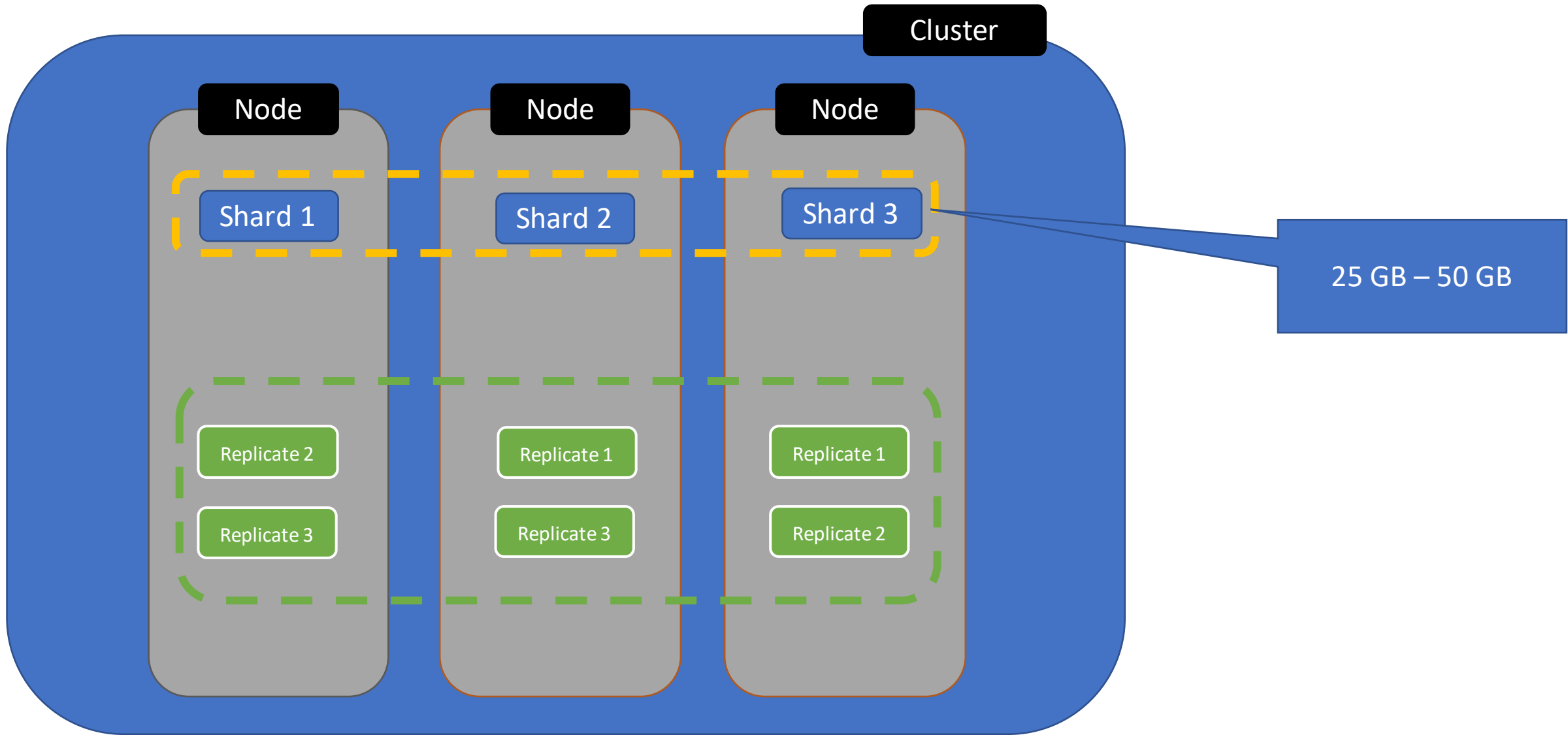
Microservice1.logs.12-13-2022

Microservice1.logs.12-14-2022

Microservice1-logs-data-stream

Alias = multiple index files

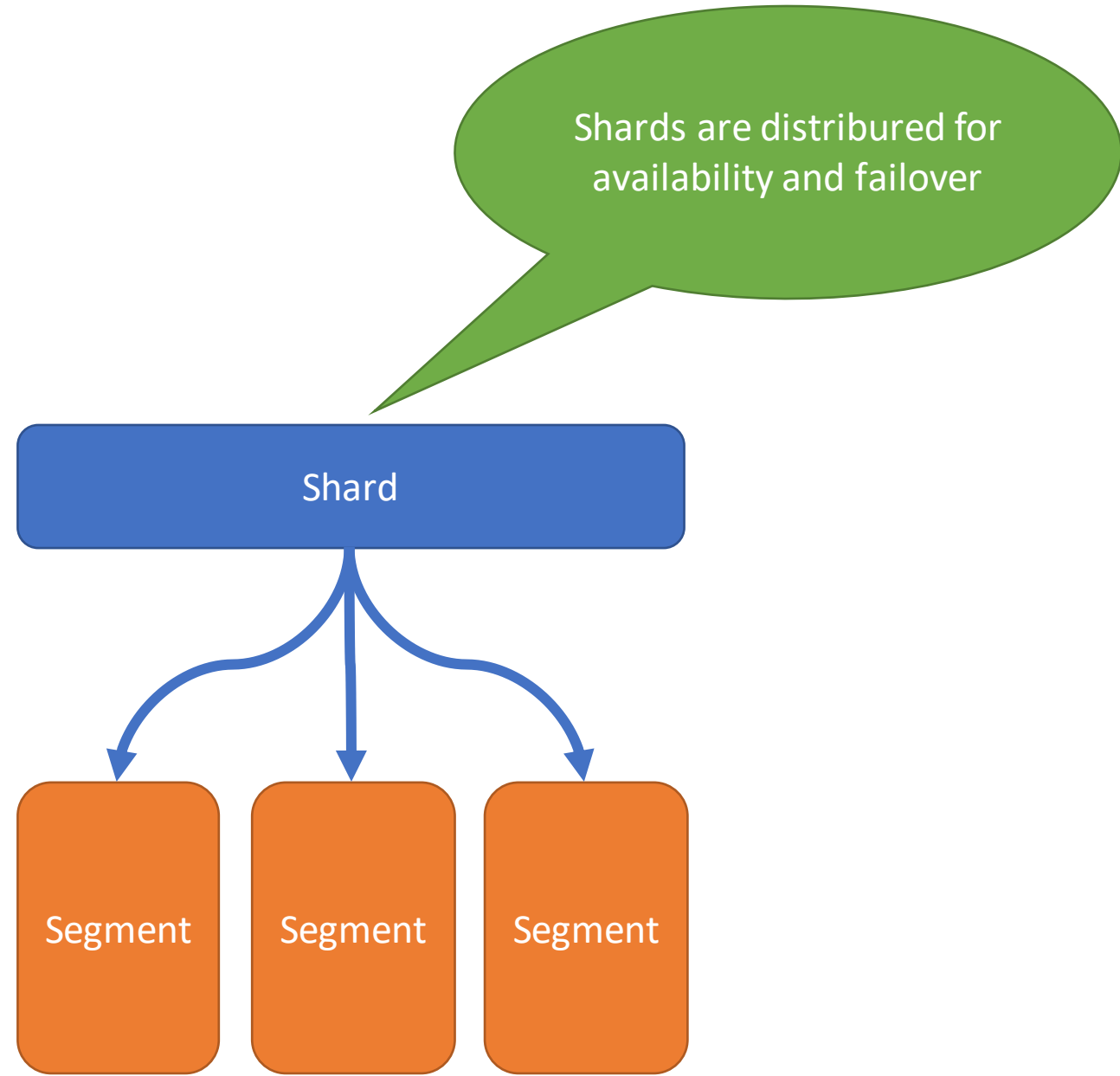
Elasticsearch | Architecture



Elasticsearch | Architecture

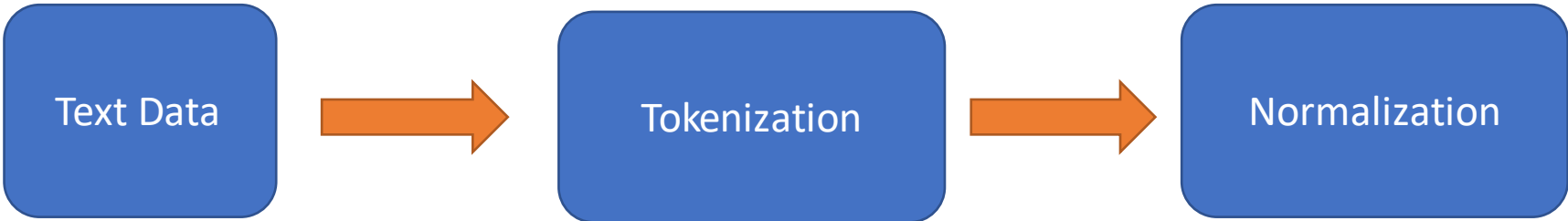
Shard

- Shards are component holding data
- Apache Lucene instances
- Immutable file segments



Elasticsearch | Architecture

Text Analyzing



Elasticsearch | Architecture

Tokenization

Untokenized string	Bugün çok sıcaktı. Sıcak havalarda çalışılmaz.
Tokenized string	[Bugün,çok,sıcaktı,sıcak,havalarda,çalışılmaz]



Elasticsearch | Architecture

Normalization

Normalization aşamasında data eş anlamlı veya kök cümleler ile zenginleştirilir.

- Sıcaktı => Sıcak (Kelime kökü
- Havalarda => Hava (Kelime kökü)
- Isim => ad (Eş anlamlı)
- Covid => Korona, Koronavirüs (Eş anlamlı)



Advanced data structure (inverted index)

Elasticsearch | Architecture

Normalization

The screenshot shows a Google search for 'koronavirüs'. The results include a table from 'Genel Koronavirüs Tablosu - Covid19 (saglik.gov.tr)', a link to 'COVID Live - Coronavirus Statistics - Worldometer', a link to 'Corona Virüsü (Koronavirüs) Son Dakika Haberleri - Hürriyet', a link to 'Koronavirüs Nedir? - Acıbadem', and a link to 'Corona Virüsü (Koronavirüs) Nedir, Belirtileri Nelerdir?'. The table from 'Genel Koronavirüs Tablosu - Covid19 (saglik.gov.tr)' is as follows:

Tarih	Vaka Sayısı	Vefat Sayısı	İyileşen Sayısı	Toplam Vaka Sayısı	To...
14 - 27 Kasım 2022	37.185	92	21.052	17.042.722	10...
31 Ekim - 13 Kasım 20...	28.808	73	16.897	17.005.537	10...
17- 30 Ekim 2022	22.887	64	23.101	16.976.729	10...

The table from 'COVID Live - Coronavirus Statistics - Worldometer' is as follows:

Country, Other	Total Cases	New Cases	Total Deaths	Total Recovered	New Recovered
World	668,174,437	+367,253	6,710,544	639,524,357	+236,534
Europe	243,536,918	+15,908	1,986,873	238,203,556	+67,736
Asia	209,016,510	+346,581	1,515,933	194,443,089	+161,678

Elasticsearch | Architecture

Inverted Index

Her full-text field karşılıkbir inverted-index oluştur

Word	Doc Number
Hello	1,5,6
World	5,8,9
Friends	1,2

INDEX	
ABC, 164, 321 <i>n</i>	Anello, Douglas, 60
academic journals, 262, 280–82	animated cartoons, 21–24
Adobe eBook Reader, 148–53	antiretroviral drugs, 257–61
advertising, 36, 45–46, 127, 145–46, 167–68, 321 <i>n</i>	Apple Corporation, 203, 264, 302
Africa, medications for HIV patients in, 257–61	architecture, constraint effected through, 122, 123, 124, 318 <i>n</i>
Agee, Michael, 223–24, 225	archive.org, 112
agricultural patents, 313 <i>n</i>	<i>see also</i> Internet Archive
Aibo robotic dog, 153–55, 156, 157, 160	archives, digital, 108–15, 173, 222, 226–27
AIDS medications, 257–60	Aristotle, 150
air traffic, land ownership vs., 1–3	Armstrong, Edwin Howard, 3–6, 184, 196
Akerlof, George, 232	Arrow, Kenneth, 232
Alben, Alex, 100–104, 105, 198–99, 295, 317 <i>n</i>	art, underground, 186
alcohol prohibition, 200	artists:
<i>Alice's Adventures in Wonderland</i> (Carroll), 152–53	publicity rights on images of, 317 <i>n</i>
	recording industry payments to, 52, 58–59, 74, 195, 196–97, 199, 301, 329 <i>n</i> –30 <i>n</i>

Elasticsearch | Architecture

Inverted Index

<h1>Kırmızı kitap</h1>

HTML strip filter

Kırmızı kitap

Whitespace Tokenizer

[Kırmızı,kitap]

Lowercasing Tokenizer

[kırmızı,kitap]

Elasticsearch | Architecture

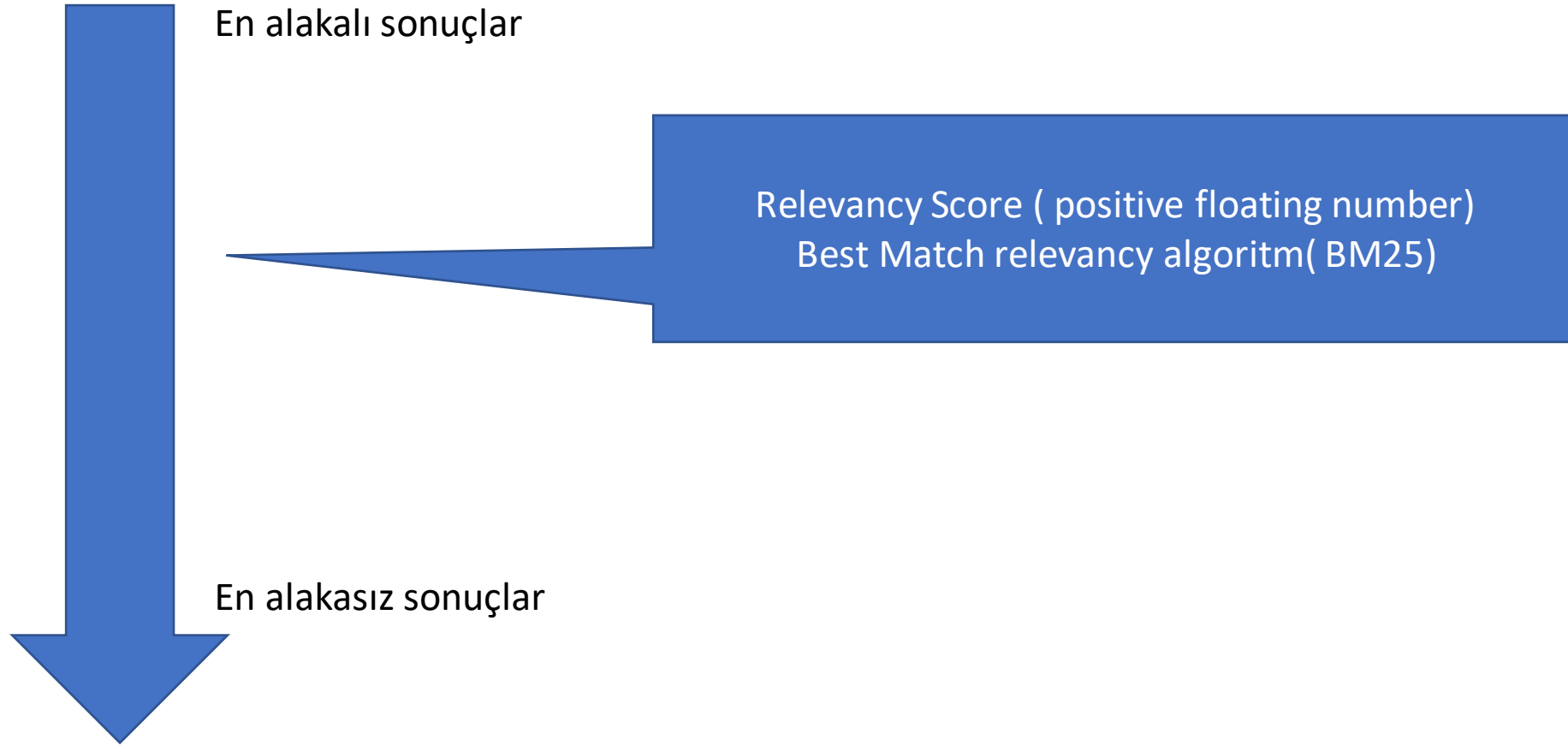
Inverted Index

Kırmızı kitap
Kırmızı kalem

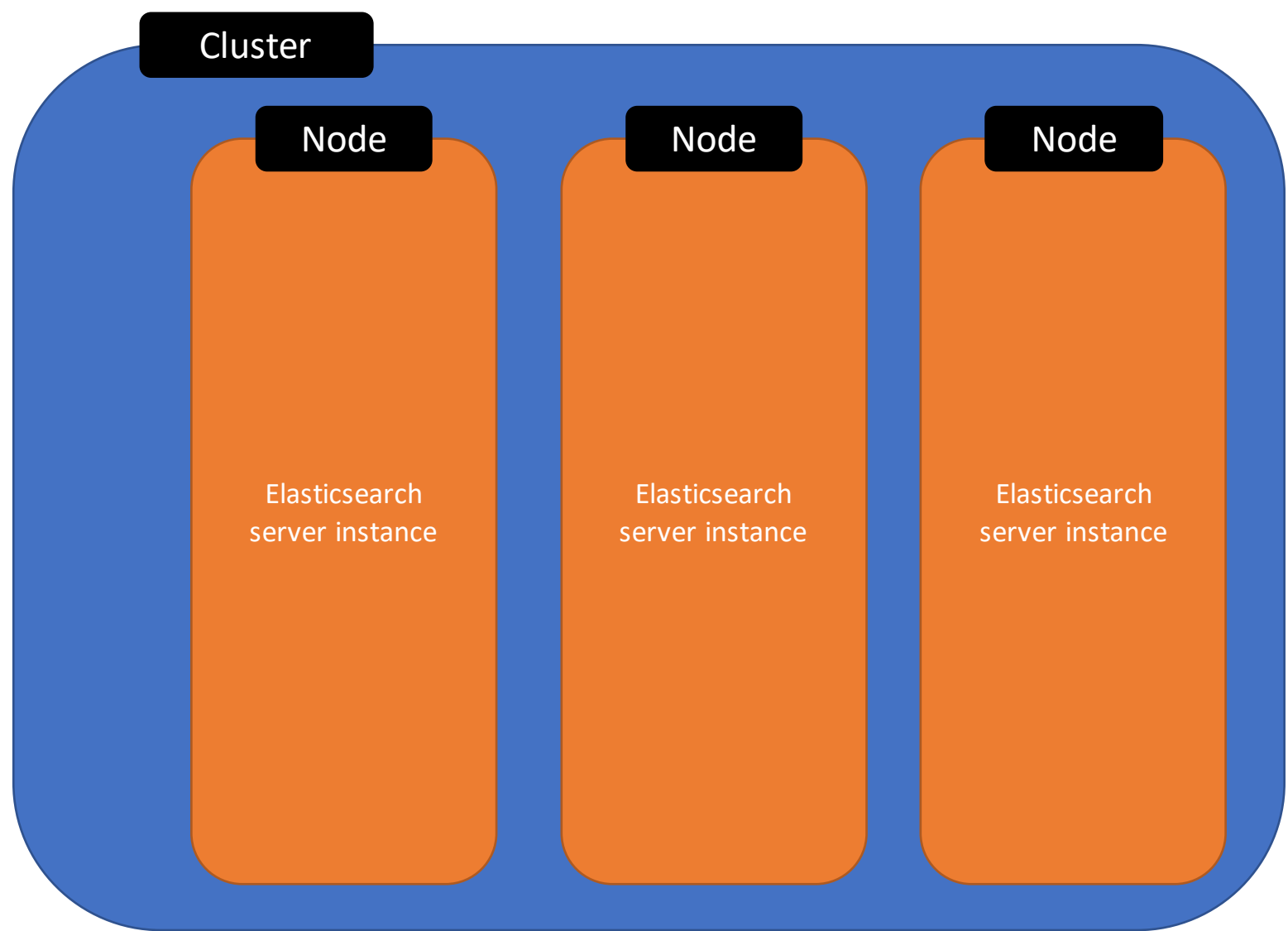
Word	Frequency	Document Id
kırmızı	2	1,2
kitap	1	1
Kalem	1	1

Elasticsearch | Architecture

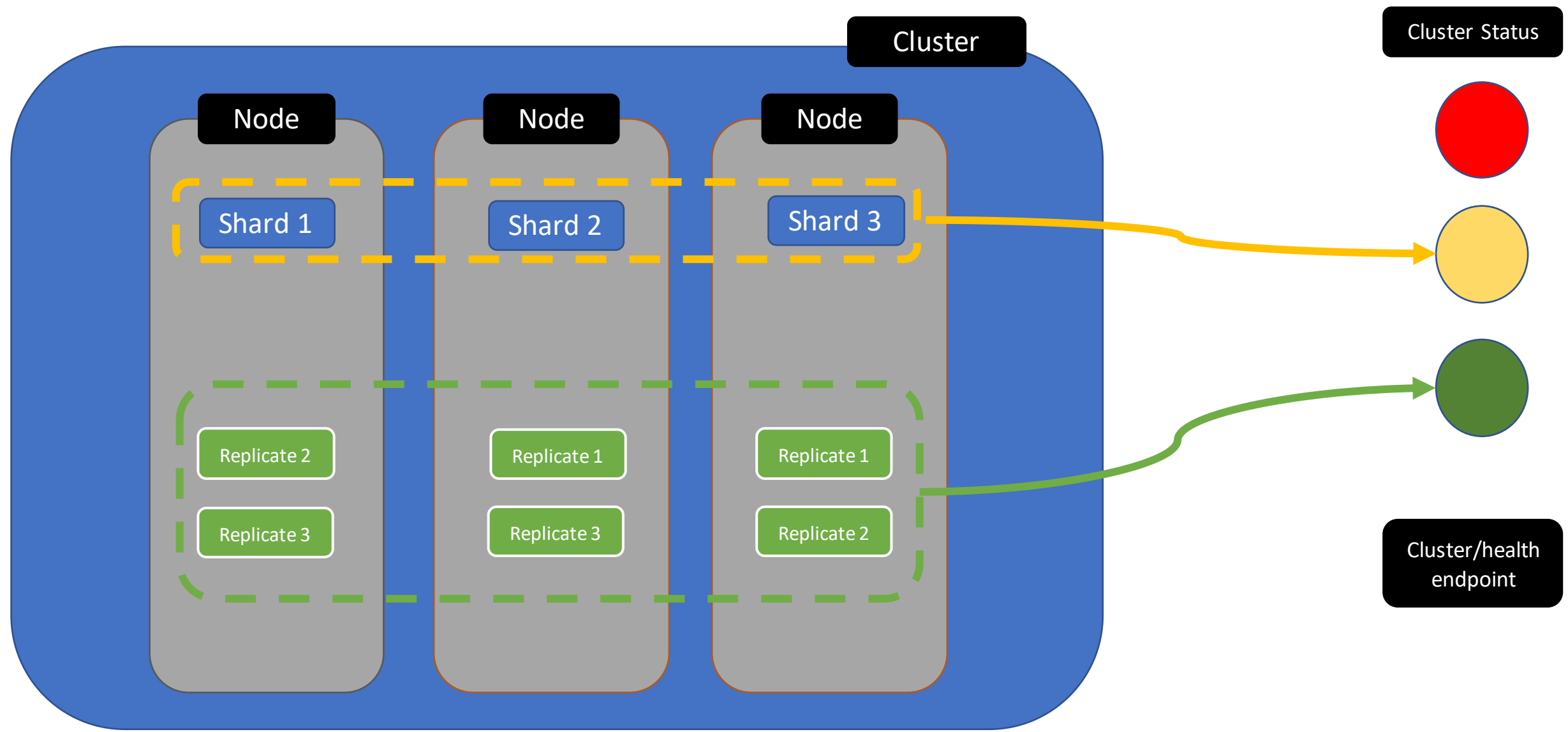
Relevancy



Elasticsearch | Architecture



Elasticsearch | Architecture



Elasticsearch | Setup

Docker



Elasticsearch
Container

Kibana
Container

Elasticsearch | Setup

Kibana UI

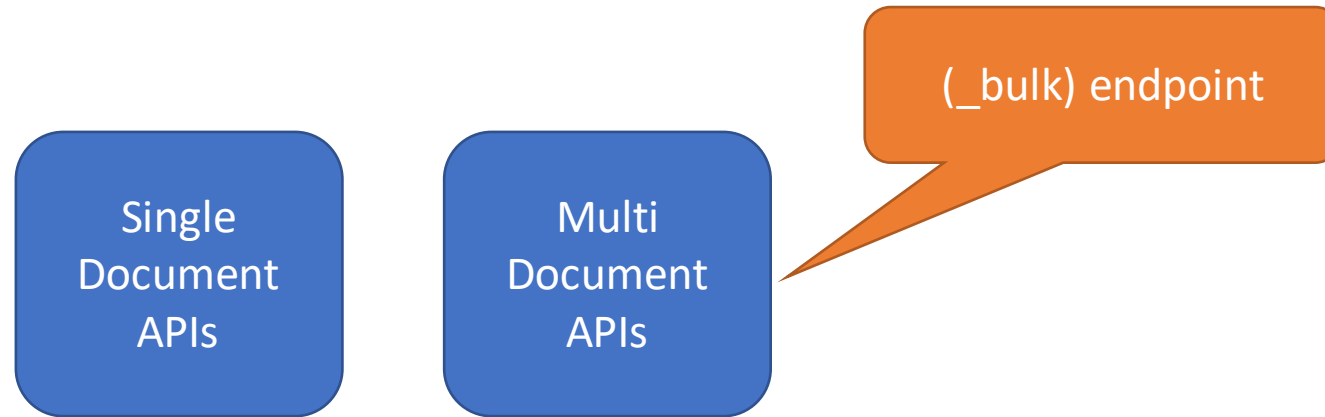
Elasticsearch | Setup

Örnek data yükleme

Elasticsearch | API

API

- Indexing API
- Searching API
- Reading API
- Updating API
- Deleting API



Elasticsearch | API

Create Document (indexing)

<HTTP_Method_Type> <Server:Port>/<IndexName>/_doc/<Doc_ID>

Table

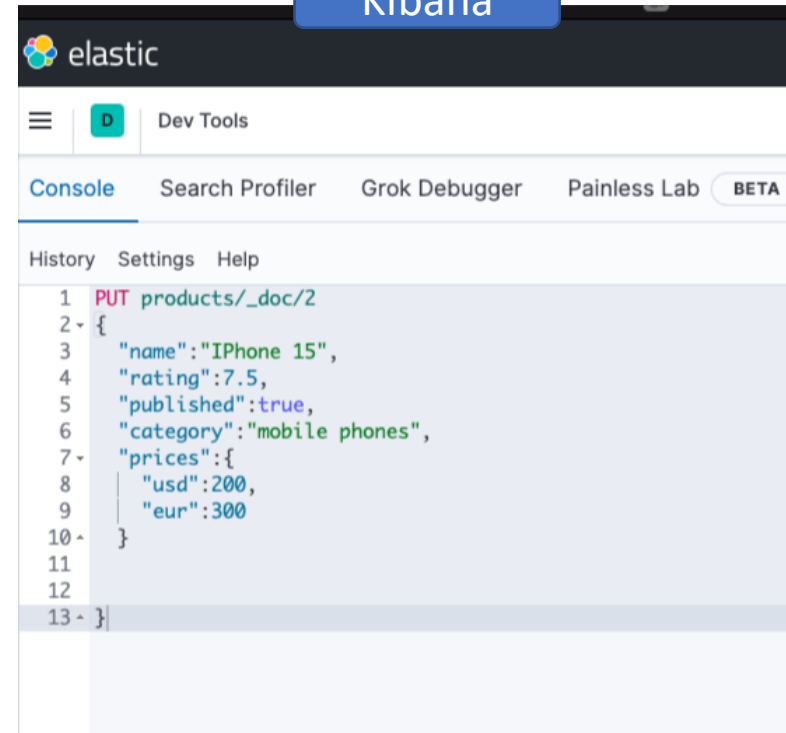
Primary Key

RESTful

http://localhost:9200/products/_doc/1

```
{
  "name": "iPhone 14",
  "rating": 8.5,
  "published": true,
  "category": "mobile phones",
  "prices": {
    "usd": 900,
    "eur": 800
  }
}
```

Kibana



Elasticsearch | Get Started

Retrieve Document

GET products/_doc/1

```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "1",  
5   "_version" : 1,  
6   "_seq_no" : 0,  
7   "_primary_term" : 1,  
8   "found" : true,  
9   "_source" : {  
10    "name" : "iPhone 14",  
11    "rating" : 8.5,  
12    "published" : true,  
13    "category" : "mobile phones",  
14    "prices" : {  
15     "usd" : 900,  
16     "eur" : 800  
17    }  
18  }  
19 }  
20
```

MetaData

Document

GET products/_source/1

Elasticsearch | Get Started

What is score field

```
"hits" : [  
  {  
    "_index" : "products",  
    "_type" : "_doc",  
    "id" : "1",  
    "_score" : 0.5260966,  
    "_source" : {  
      "name" : "Nokia mobile phone 1",  
      "rating" : 8.5,  
      "published" : true,  
      "category" : "mobile phones",  
      "prices" : {  
        "usd" : 400,  
        "eur" : 500  
      }  
    }  
  },  
  {  
    "_index" : "products",  
    "_type" : "_doc",  
    "id" : "2",  
    "_score" : 0.43210987,  
    "_source" : {  
      "name" : "Nokia mobile phone 2",  
      "rating" : 7.5,  
      "published" : false,  
      "category" : "mobile phones",  
      "prices" : {  
        "usd" : 350,  
        "eur" : 450  
      }  
    }  
  }  
]
```

`_score` is a positive floating number
A score indicates how well the documents matched the query(Okapi Best Match 25 algorithm)

Elasticsearch | Documents

Identifiers

Client
(HTTP PUT)

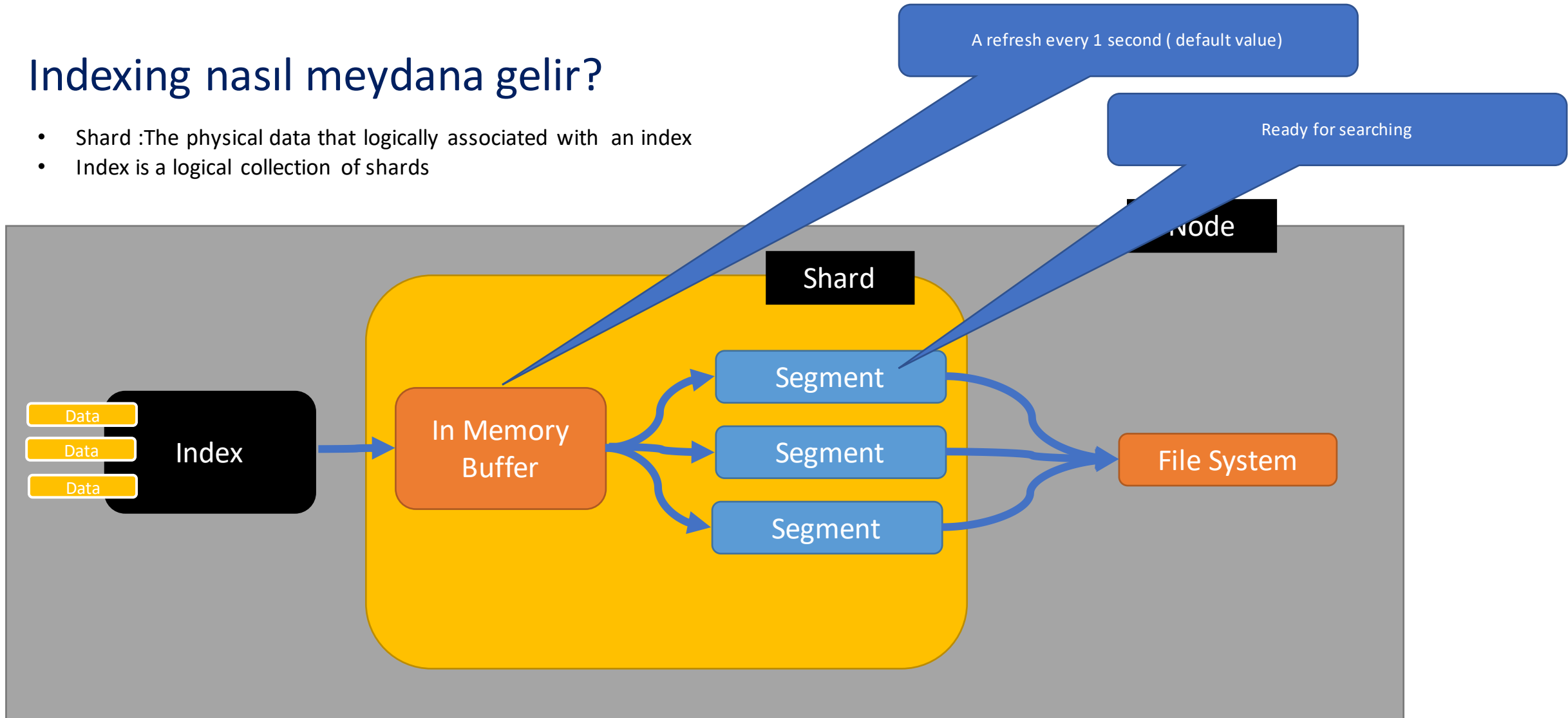
Elasticsearch
(HTTP POST)

[PUT] : products/_doc/1
[POST] : products/_doc
[PUT] : products/_create/1

Elasticsearch | Documents

Indexing nasıl meydana gelir?

- Shard :The physical data that logically associated with an index
- Index is a logical collection of shards



Elasticsearch | Documents

Interval Indexing

```
PUT products/_settings
{
  "index":{ "refresh_interval":"30s"
}
```

```
PUT products/_doc/1?refresh
```

Searching after interval
refresh

Searching immediately

Refresh=true
Refresh=false (default)
Refresh=wait_for

Elasticsearch | Documents

Indexing documents with client Id

[PUT] : <index_name>/_doc/<identifier>

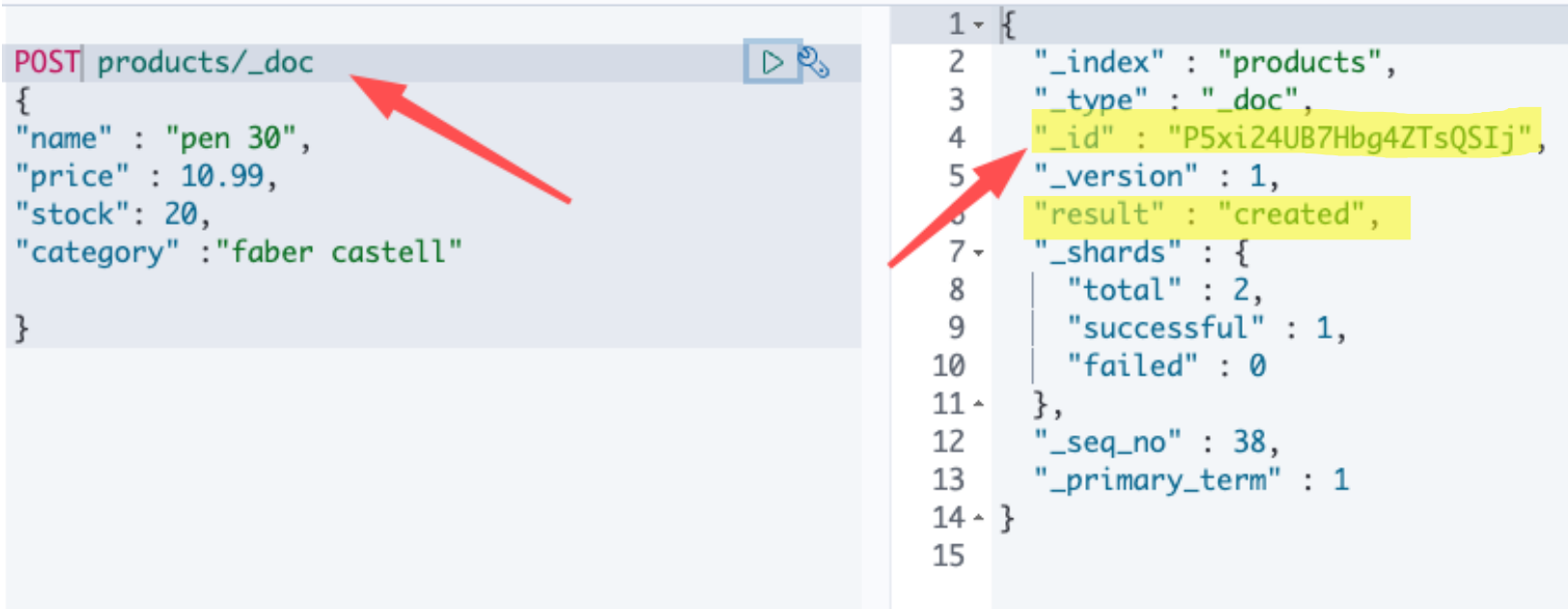
```
PUT products/_doc/11
{
  "name" : "pen 11",
  "price" : 10.99,
  "stock": 20,
  "category" : "faber castell"
}
```

```
1 {
2   "_index" : "products",
3   "_type" : "_doc",
4   "_id" : "11",
5   "_version" : 1,
6   "result" : "created",
7   "_shards" : {
8     "total" : 2,
9     "successful" : 1,
10    "failed" : 0
11  },
12   "_seq_no" : 37,
13   "_primary_term" : 1
14 }
15
```


Elasticsearch | Documents

Indexing documents with elasticsearch (auto generated ID)

[POST] : <index_name>/_doc



The screenshot displays the Elasticsearch client interface. On the left, a POST request is shown with the endpoint `products/_doc` and a JSON body: `{ "name": "pen 30", "price": 10.99, "stock": 20, "category": "faber castell" }`. A red arrow points from the `_doc` part of the endpoint to the response. On the right, the response is shown as a JSON object: `{ "_index": "products", "_type": "_doc", "_id": "P5xi24UB7Hbg4ZTsQSIj", "_version": 1, "result": "created", "_shards": { "total": 2, "successful": 1, "failed": 0 }, "_seq_no": 38, "_primary_term": 1 }`. The `_id` and `result` fields are highlighted in yellow. A red arrow points from the `_id` field in the response back to the `_doc` part of the endpoint.

```
POST products/_doc
{
  "name": "pen 30",
  "price": 10.99,
  "stock": 20,
  "category": "faber castell"
}
```

```
{
  "_index": "products",
  "_type": "_doc",
  "_id": "P5xi24UB7Hbg4ZTsQSIj",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 38,
  "_primary_term": 1
}
```

Elasticsearch | Documents

Indexing documents with _create keyword

[POST] : <index_name>/_create/100

POST products/_create/100

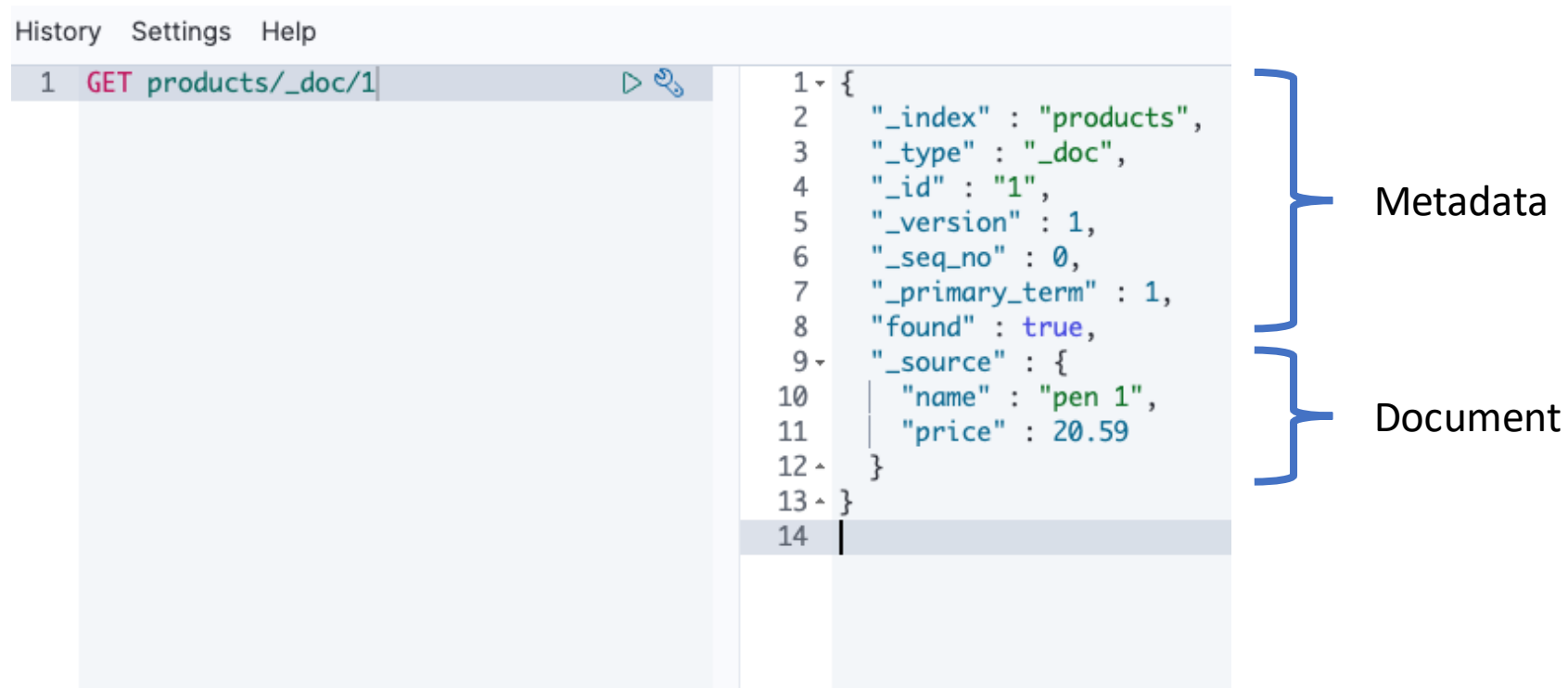
```
{
  "name" : "pen 30",
  "price" : 10.99,
  "stock": 20,
  "category" : "faber castell"
}
```

```
1 {
2   "error" : {
3     "root_cause" : [
4       {
5         "type" : "version_conflict_engine_exception",
6         "reason" : "[100]: version conflict, document already exists (current version [1])",
7         "index_uuid" : "rM9WKOVRS-6FgwdDclLTEA",
8         "shard" : "0",
9         "index" : "products"
10      }
11    ],
12    "type" : "version_conflict_engine_exception",
13    "reason" : "[100]: version conflict, document already exists (current version [1])",
14    "index_uuid" : "rM9WKOVRS-6FgwdDclLTEA",
15    "shard" : "0",
16    "index" : "products"
17  },
18  "status" : 409
19 }
```

Elasticsearch | Documents

Retrieving documents

[GET] : <index_name>/_doc/<id>



The screenshot shows a REST client interface with a top navigation bar containing 'History', 'Settings', and 'Help'. The main area displays a single request: '1 GET products/_doc/1'. To the right of the request bar, a JSON response is shown, with line numbers 1 through 14 on the left margin. The JSON is as follows:

```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "1",  
5   "_version" : 1,  
6   "_seq_no" : 0,  
7   "_primary_term" : 1,  
8   "found" : true,  
9   "_source" : {  
10    "name" : "pen 1",  
11    "price" : 20.59  
12  }  
13 }  
14
```

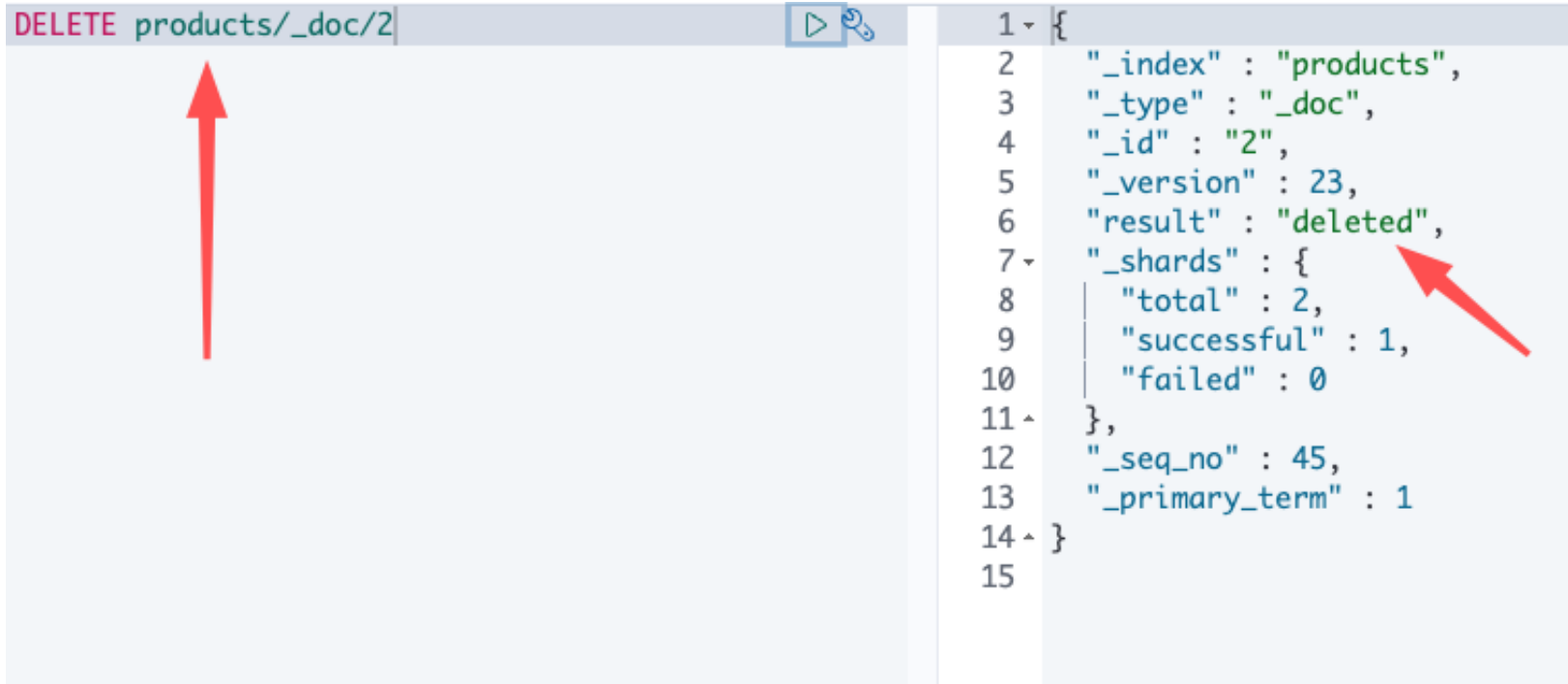
Blue curly braces on the right side of the JSON response group the fields into two categories:

- Metadata:** Lines 2 through 8, which include fields like `"_index"`, `"_type"`, `"_id"`, `"_version"`, `"_seq_no"`, `"_primary_term"`, and `"found"`.
- Document:** Lines 9 through 12, which include the `"_source"` object containing `"name"` and `"price"`.

Elasticsearch | Documents

Deleting documents

[DELETE] : <index_name>/_doc/<id>



The screenshot shows an Elasticsearch client interface. On the left, a text input field contains the DELETE request: `DELETE products/_doc/2`. A red arrow points from the text below the input field up to the `products/_doc/2` part of the request. On the right, the JSON response is displayed, with line numbers 1 through 15 on the left margin. A red arrow points from the `"result" : "deleted"` line (line 6) to the `"successful" : 1` line (line 9) within the `_shards` object.

```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "2",  
5   "_version" : 23,  
6   "result" : "deleted",  
7   "_shards" : {  
8     "total" : 2,  
9     "successful" : 1,  
10    "failed" : 0  
11  },  
12  "_seq_no" : 45,  
13  "_primary_term" : 1  
14 }  
15
```

Elasticsearch | Documents

Retrieving documents with Head Method Type

```
1 HEAD products/_doc/100 1 {"statusCode":404,"error":"Not Found","message":"404 - Not Found"}
```

```
1 HEAD products/_doc/1 1 200 - OK
```

Elasticsearch | Documents

Retrieving multiple documents with _mget

```
1 GET products/_mget
2 {
3   "ids":["1","2"]
4 }
```

||

```
1 {
2   "docs" : [
3     {
4       "_index" : "products",
5       "_type" : "_doc",
6       "_id" : "1",
7       "_version" : 1,
8       "_seq_no" : 0,
9       "_primary_term" : 1,
10      "found" : true,
11      "_source" : {
12        "name" : "pen 1",
13        "price" : 20.59
14      }
15    },
16    {
17      "_index" : "products",
18      "_type" : "_doc",
19      "_id" : "2",
20      "_version" : 22,
21      "_seq_no" : 22,
22      "_primary_term" : 1,
23      "found" : true,
24      "_source" : {
25        "name" : "pen 2",
26        "price" : 20.59
27      }
28    }
29  ]
30 }
```

Elasticsearch | Documents

Custom response with no metadata

```
1 GET products/_source/1
2
3
4
5
```

```
1 {
2   "name" : "pen 1",
3   "price" : 20.59
4 }
5
```

Elasticsearch | Documents

Retrieving multiple documents with `_mget` (no document)

```
GET _mget
{
  "docs": [
    {
      "_index": "products",
      "_id": 1
    },
    {
      "_index": "products",
      "_id": 2
    }
  ]
}
```

```
1 {
2   "docs" : [
3     {
4       "_index" : "products",
5       "_type" : "_doc",
6       "_id" : "1",
7       "_version" : 1,
8       "_seq_no" : 0,
9       "_primary_term" : 1,
10      "found" : true,
11      "_source" : {
12        "name" : "pen 1",
13        "price" : 20.59
14      }
15    },
16    {
17      "_index" : "products",
18      "_type" : "_doc",
19      "_id" : "2",
20      "_version" : 22,
21      "_seq_no" : 22,
22      "_primary_term" : 1,
23      "found" : true,
24      "_source" : {
25        "name" : "pen 2",
26        "price" : 20.59
27      }
28    }
29  ]
30 }
```


Elasticsearch | Documents

Custom response with no source

```
GET products/_doc/1?_source=false
```

```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "1",  
5   "_version" : 1,  
6   "_seq_no" : 0,  
7   "_primary_term" : 1,  
8   "found" : true  
9 }  
10
```

Elasticsearch | Documents

Custom response with `_source_includes` or `_source_excludes`

```
GET products/_doc/1?_source_includes=name,stock,category
```



```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "1",  
5   "_version" : 2,  
6   "_seq_no" : 25,  
7   "_primary_term" : 1,  
8   "found" : true,  
9   "_source" : {  
10    "name" : "pen 1",  
11    "stock" : 60,  
12    "category" : "pens"  
13  }  
14 }  
15
```

Elasticsearch | Documents

Updating documents

[POST] : <index_name>/_update/<id>

```
POST products/_update/1
{
  "doc":{
    "name" : "pen 222",
    "stock" : 600,
    "category" : "faber castell"
  }
}
```

```
1 - {
2   "_index" : "products",
3   "_type" : "_doc",
4   "_id" : "1",
5   "_version" : 4,
6   "result" : "noop",
7 -  "_shards" : {
8     "total" : 0,
9     "successful" : 0,
10    "failed" : 0
11 - },
12   "_seq_no" : 27,
13   "_primary_term" : 1
14 - }
15
```

```
POST products/_update/1
{
  "doc":{
    "type":1
  }
}
```

Elasticsearch | Documents

Updating documents with scripts

"source" : "ctx._source.colors.add()"

"source" : "ctx._source.colors.remove()"

```
POST products/_update/1
{
  "script": {
    "source": "ctx._source.colors.add('pink')"
  }
}
```

```
1
2  "_index" : "products",
3  "_type" : "_doc",
4  "_id" : "1",
5  "_version" : 8,
6  "_seq_no" : 31,
7  "_primary_term" : 1,
8  "found" : true,
9  "_source" : {
10   "name" : "pen 222",
11   "price" : 100.99,
12   "stock" : 600,
13   "category" : "faber castell",
14   "type" : 1,
15   "colors" : [
16     "red",
17     "blue",
18     "pink"
19   ]
}
```

Elasticsearch | Documents

Replacing documents with new one

[PUT] : <index_name>/_doc/<id>

```
PUT products/_doc/222
```

```
{  
  "name": "pen 2222"  
}
```

```
}
```

```
1- {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "222",  
5   "_version" : 2,  
6   "result" : "updated",  
7-  "_shards" : {  
8     "total" : 2,  
9     "successful" : 1,  
10    "failed" : 0  
11-  },  
12  "_seq_no" : 33,  
13  "_primary_term" : 1  
14- }  
15
```

Elasticsearch | Documents

Update or Add documents with upset keyword

[POST] : <index_name>/_update/<id>

POST products/_update/200

```
{
  "script": {
    "source": "ctx._source.name='pen 200'"
  },
  "upsert": {
    "name": "pen 2000",
    "price": 10.99,
    "stock": 20,
    "category": "faber castell"
  }
}
```

```
1 {
2   "_index" : "products",
3   "_type" : "_doc",
4   "_id" : "200",
5   "_version" : 3,
6   "result" : "updated",
7   "_shards" : {
8     "total" : 2,
9     "successful" : 1,
10    "failed" : 0
11  },
12   "_seq_no" : 42,
13   "_primary_term" : 1
14 }
15
```

Elasticsearch | Documents

Update or Add documents with upset keyword

[POST] : <index_name>/_update/<id>

POST products/_update/200

```
{
  "script": {
    "source": "ctx._source.name='pen 200'"
  },
  "upsert": {
    "name": "pen 2000",
    "price": 10.99,
    "stock": 20,
    "category": "faber castell"
  }
}
```

```
1 {
2   "_index" : "products",
3   "_type" : "_doc",
4   "_id" : "200",
5   "_version" : 3,
6   "result" : "updated",
7   "_shards" : {
8     "total" : 2,
9     "successful" : 1,
10    "failed" : 0
11  },
12   "_seq_no" : 42,
13   "_primary_term" : 1
14 }
15
```

Elasticsearch | Documents

Update or Add documents without errors

[POST] : <index_name>/_update/<id>

```
POST products/_update/400
```

```
{  
  "doc": {  
    "price": 400  
  },  
  "doc_as_upsert": true  
}
```

```
GET products/_doc/300
```

```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "400",  
5   "_version" : 1,  
6   "result" : "created",  
7   "_shards" : {  
8     "total" : 2,  
9     "successful" : 1,  
10    "failed" : 0  
11  },  
12   "_seq_no" : 44,  
13   "_primary_term" : 1  
14 }  
15
```


Elasticsearch | Documents

Updating by query

[POST] : nasıl query yazılacağı tam bilinmediği için sonra doldurulacak

Elasticsearch | Documents

Deleting documents with _delete_by_query keyword

[DELETE] : <index_name>/_delete_by_query

```
POST products/_delete_by_query
{
  "query": {
    "match": {
      "name": "pen"
    }
  }
}
```

Elasticsearch | Documents

Reindexing documents

[POST] : _reindex

```
POST _reindex
{
  "source": {"index": "products"},
  "dest": {"index": "products_new"}
}
```

Elasticsearch | Data Types

Data types

Common types

- text
- boolean
- date
- numeric
- binary

Complex and relational types

- object
- nested
- flattened
- join

Structured types

- geo_shape
- geo_point
- ip
- date_range
- ip_range

Elasticsearch | Data Types

Common Data Types (totally 29 data types)

Type	Description	Examples
text	string values	log message, blog content, product description, product title
byte/short/integer/long	number	
float/double	floating-point number	
boolean	true/false	
object	Json object	
keyword	text must not be broken down or analyzed	Email address, phone number, SGK NO

Elasticsearch | Data Types

Core Data Types

Text type (knows as **full text** and **unstructured text**)

- unstructured text : blog post, blog title,article
- analysis process

Elasticsearch | Data Types

Core Data Types

Keyword type (structured text)

- structured text : bank account,phone number,pincodes,sgk no, tc no
- don't analysis process

Elasticsearch | Data Types

Core Data Types

Date type (structured text)

- ISO 8601
- yyyy-MM-dd

Elasticsearch | Data Types

Core Data Types

Date type – custom format

```
PUT products2
{
  "mappings":
  {
    "properties": {
      "updated":{
        "type":"date",
        "format":"dd-MM-yyyy"
      }
    }
  }
}
```

```
GET products2/_mapping
```

```
PUT products2/_doc/1
{
  "updated":"06-06-2010"
}
```

```
GET products2/_doc/1
```

```
1 {
2   "products2" : {
3     "mappings" : {
4       "properties" : {
5         "updated" : {
6           "type" : "date",
7           "format" : "dd-MM-yyyy"
8         }
9       }
10    }
11  }
12 }
13
```

Elasticsearch | Data Types

Core Data Types

Numeric types

- byte (8 bit)
- short (16 bit)
- integer (32 bit)
- long (64 bit)
- float (32 bit)
- double (64 bit)
- half_float(16 bit- half precision)
- half_double(32 bit- half precision)

Elasticsearch | Data Types

Core Data Types

Boolean type

- true/false

Elasticsearch | Data Types

Core Data Types

range type / date_range

Example : 25-50

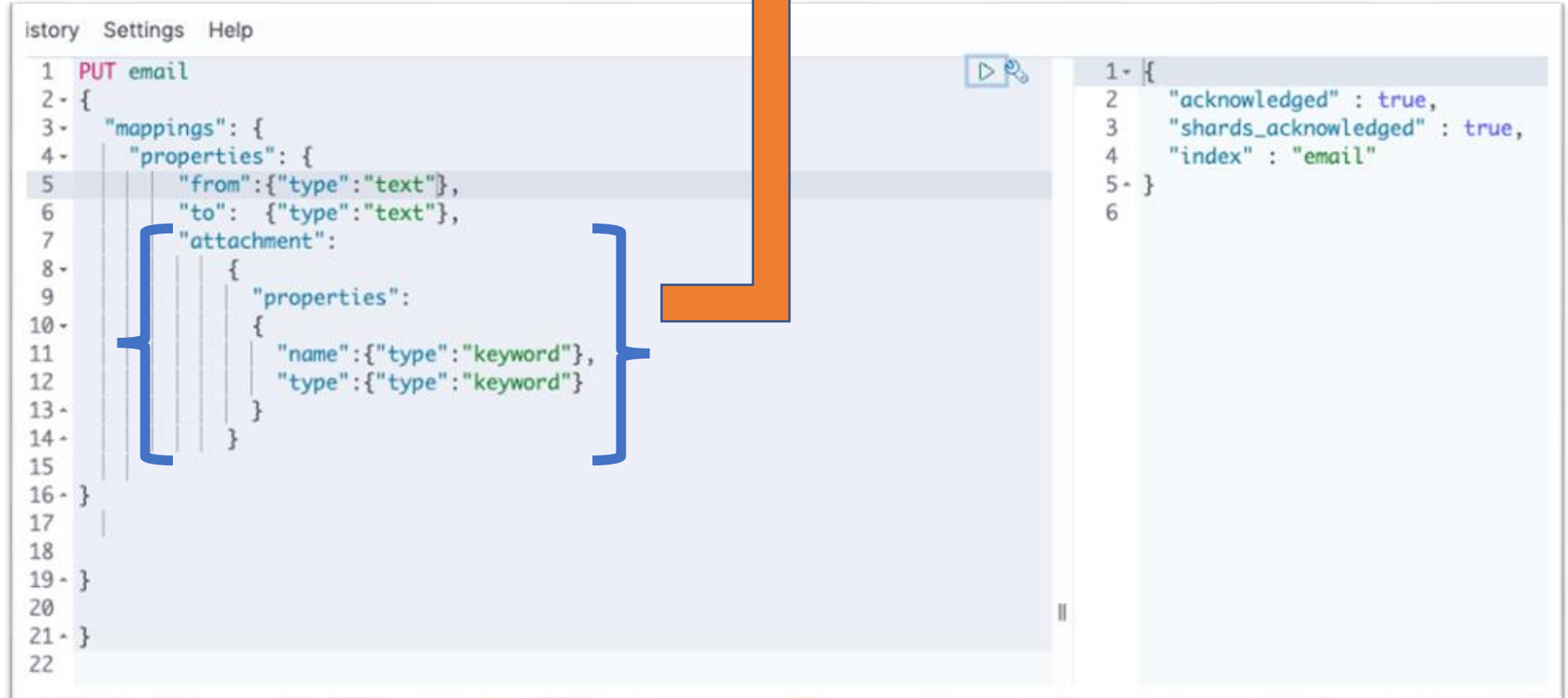
Example : 05/05/2021-07/07/2022

Elasticsearch | Data Types

Complex and Relational Type

Object data type

Hierarchical object



Elasticsearch | Data Types

Complex and Relational Type

Flattened data type

- specialized object type
- all sub fields are keyword type

```
PUT email3
{
  "mappings": {
    "properties": {
      "from": {"type": "text"},
      "to": {"type": "text"},
      "attachment": {
        "type": "flattened"
      }
    }
  }
}
```

```
PUT email2/_doc/5
{
  "from": "a@outlook.com",
  "to": "b@outlook.com",
  "attachment": [
    {
      "name": "example file",
      "type": ".png",
      "length": 2000
    },
    {
      "name": "example2 file",
      "type": ".png",
      "length": 1000
    }
  ]
}
```

```
GET email2/_doc/5
```

keyword type

```
1- {
2  "_index": "email2",
3  "_type": "_doc",
4  "_id": "5",
5  "_version": 2,
6  "_seq_no": 3,
7  "_primary_term": 1,
8  "found": true,
9  "_source": {
10   "from": "a@outlook.com",
11   "to": "b@outlook.com",
12   "attachment": [
13     {
14       "name": "example file",
15       "type": ".png",
16       "length": 2000
17     },
18     {
19       "name": "example2 file",
20       "type": ".png",
21       "length": 1000
22     }
23   ]
24 }
25 }
26
```

Elasticsearch | Data Types

Complex and Relational Type

Nested data type

specialized object type

```
PUT user/_doc/1 {  
  "userId" : "1",  
  "user" : [  
    { "name" : "ahmet", "surname" : "yılmaz" },  
    { "name" : "mehmet", "surname" : "güneş" }  
  ]  
}
```

```
user.name="ahmet"  
user.surname="güneş"
```

Elasticsearch | Data Types

Complex and Relational Type

Nested data type

specialized object type

Attachment.name="example file"
Attachment.type=".png"

```
PUT email2
{
  "mappings": {
    "properties": {
      "from": {"type": "text"},
      "to": {"type": "text"},
      "attachment": {
        {
          "type": "nested",
          "properties": {
            {
              "name": {"type": "keyword"},
              "type": {"type": "keyword"}
            }
          }
        }
      }
    }
  }
}

POST email2/_doc
{
  "from": "a@outlook.com",
  "to": "b@outlook.com",
  "attachment": [
    {
      "name": "example file",
      "type": ".png"
    },
    {
      "name": "example2 file",
      "type": ".png"
    }
  ]
}
```


Elasticsearch | Data Types

Complex and Relational Type

Join data type

Parent

```
PUT questions/_doc/1?refresh
{
  "id": "1",
  "text": "soru 1"
}
```

```
PUT questions/_doc/1?refresh
{
  "id": "2",
  "text": "soru 2"
}
```



Child

```
PUT questions/_doc/1?refresh
{
  "id": "3",
  "text": "soru 1 cevap"
}
```

```
PUT questions/_doc/1?refresh
{
  "id": "4",
  "text": "soru 2 cevap"
}
```



Elasticsearch | Data Types

Advance Data Types

Search data type

- auto-complete
- suggestions

```
PUT email3
{
  "mappings": {
    "properties": {
      "from": {"type": "text"},
      "to": {"type": "text"},
      "attachment": {
        {
          "type": "flattened"
        }
      }
    }
  }
}

PUT email2/_doc/5
{
  "from": "a@outlook.com",
  "to": "b@outlook.com",
  "attachment": [
    {
      "name": "example file",
      "type": ".png",
      "length": 2000
    },
    {
      "name": "example2 file",
      "type": ".png",
      "length": 1000
    }
  ]
}

GET email2/_doc/5
```

```
1- {
2  "_index" : "email2",
3  "_type" : "_doc",
4  "_id" : "5",
5  "_version" : 2,
6  "_seq_no" : 3,
7  "_primary_term" : 1,
8  "found" : true,
9  "_source" : {
10   "from" : "a@outlook.com",
11   "to" : "b@outlook.com",
12   "attachment" : [
13     {
14       "name" : "example file",
15       "type" : ".png",
16       "length" : 2000
17     },
18     {
19       "name" : "example2 file",
20       "type" : ".png",
21       "length" : 1000
22     }
23   ]
24 }
25 }
26 }
```

keyword type

Elasticsearch | Data Types

Advance Data Types

multiple data type



```
History Settings Help
1 PUT article
2 {
3   "mappings": {
4     "properties": {
5       "title": {
6         "type": "text",
7         "fields": {
8           "kw": { "type": "keyword" }
9         }
10      }
11    }
12  }
13 }
14 }
15
16 GET article/_mapping
17 {
18   "article": {
19     "mappings": {
20       "properties": {
21         "title": {
22           "type": "text",
23           "fields": {
24             "kw": {
25               "type": "keyword"
26             }
27           }
28         }
29       }
30     }
31   }
32 }
```

Elasticsearch | Mapping

What is mapping?

Schema(kaydedilmek istenen data tipleri) oluşturma süreci **mapping** olarak adlandırılır.

History Settings Help

```
1 PUT products/_doc/1
2 {
3   "name": "pen 1",
4   "price": 38.99,
5   "created": "2023-15-02",
6   "stocks": {
7     "warehouse1": 100,
8     "warehouse2": 200
9   }
10 }
11
12 }
```

string field

number field

date field

inner field

Elasticsearch | Mapping

first indexing

```
History Settings Help
1 PUT products/_doc/1
2 {
3   "name": "pen 1",
4   "price": 38.99,
5   "created": "2023-15-02",
6   "stocks": {
7     "warehouse1": 100,
8     "warehouse2": 200
9   }
10 }
11
12 }
```

```
1 {
2   "_index" : "products",
3   "_type" : "_doc",
4   "_id" : "1",
5   "_version" : 1,
6   "result" : "created",
7   "_shards" : {
8     "total" : 2,
9     "successful" : 1,
10    "failed" : 0
11  },
12   "_seq_no" : 0,
13   "_primary_term" : 1
14 }
15
```

- products isminde yeni bir index oluşur.
- Yeni bir schema oluşur.
- document kalıcı olarak kaydedilir.
- Daha sonraki işlemlerde ilk 2 adım gerçekleşmez.

Elasticsearch | Mapping

Mapping Types

- Dynamic Mapping
- Explicit Mapping

Elasticsearch | Mapping

Dynamic mapping

The screenshot displays an Elasticsearch client interface with two main panels. The left panel shows a REST client with a PUT request to `pens/_doc/1` containing a document with `name`, `price`, and `created_date` fields. The right panel shows the resulting mapping for the `pens` index, where the `name` field is mapped as a `text` type with a `keyword` sub-field, and the `price` field is mapped as a `float` type. A blue callout box labeled "Multi types" points to the `keyword` sub-field of the `name` field.

History Settings Help

```
1 PUT pens/_doc/1
2 {
3   "name": "pen 1",
4   "price": 38.99,
5   "created_date": "2010/05/05"
6 }
7
8 GET pens/_mapping
```

Click to send request

```
1 {
2   "pens": {
3     "mappings": {
4       "properties": {
5         "created_date": {
6           "type": "date",
7           "format": "yyyy/MM/dd HH:mm:ss||yyyy/MM/dd||epoch_millis"
8         },
9         "name": {
10          "type": "text",
11          "fields": {
12            "keyword": {
13              "type": "keyword",
14              "ignore_above": 256
15            }
16          }
17        },
18        "price": {
19          "type": "float"
20        }
21      }
22    }
23  }
24 }
25
```

Multi types

Elasticsearch | Mapping

Explicit mapping

- Indexing API's (during index creation)
- Mapping API's (alter schema)

Elasticsearch | Mapping

Explicit mapping

- Indexing API's (during index creation)



The screenshot shows a web-based interface for interacting with Elasticsearch. At the top, there are tabs for 'History', 'Settings', and 'Help'. The main area is split into two panels. The left panel contains a JSON document for a PUT request to the '/products' index. The right panel shows the response from the server, which is a JSON object indicating the index was created successfully.

```
History Settings Help

1 PUT products
2 {
3   "mappings": {
4     "properties": {
5       "name": {"type": "text"},
6       "price": {"type": "integer"},
7       "stock_no": {"type": "keyword"},
8       "category_name": {"type": "text"},
9       "warehouse": {
10        "properties": {
11          "germany": {"type": "long"},
12          "turkey": {"type": "long"}
13        }
14      }
15    }
16  }
17 }
18 }
19
20 }
```

```
1 {
2   "acknowledged" : true,
3   "shards_acknowledged" : true,
4   "index" : "products"
5 }
6
```

Elasticsearch | Mapping

Explicit mapping

- Mapping API's

```
1 PUT products/_mapping
2 {
3   "properties":
4   {
5     "color":{
6       "type":"keyword"
7     }
8   }
9 }
```

```
1 {
2   "acknowledged" : true
3 }
4
```

Elasticsearch | Mapping

Re-indexing

- Mapping API's

Elasticsearch doesn't allow us to update data types of existing fields

```
GET products/_mapping
{
  "products" : {
    "mappings" : {
      "properties" : {
        "category_name" : {
          "type" : "text"
        },
        "color" : {
          "type" : "keyword"
        },
        "name" : {
          "type" : "text"
        },
        "price" : {
          "type" : "integer"
        },
        "stock_no" : {
          "type" : "keyword"
        },
        "warehouse" : {
          "properties" : {
            "germany" : {
              "type" : "long"
            },
            "turkey" : {
              "type" : "long"
            }
          }
        }
      }
    }
  }
}
```

Elasticsearch | Searching

Searching Types

- structured search
- unstructured search

Elasticsearch | Searching

Structured Types

a term-level query functionality

- date between
- rating (1-5)
- price (>1000 or <5000)

Elasticsearch | Searching

Unstructured Types

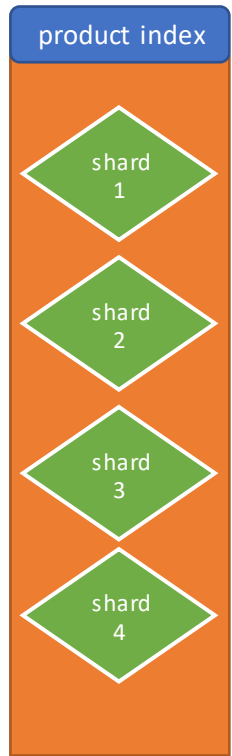
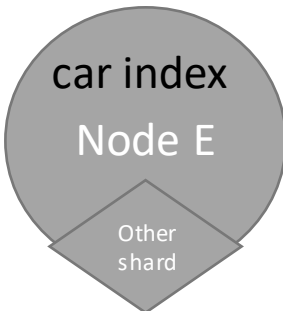
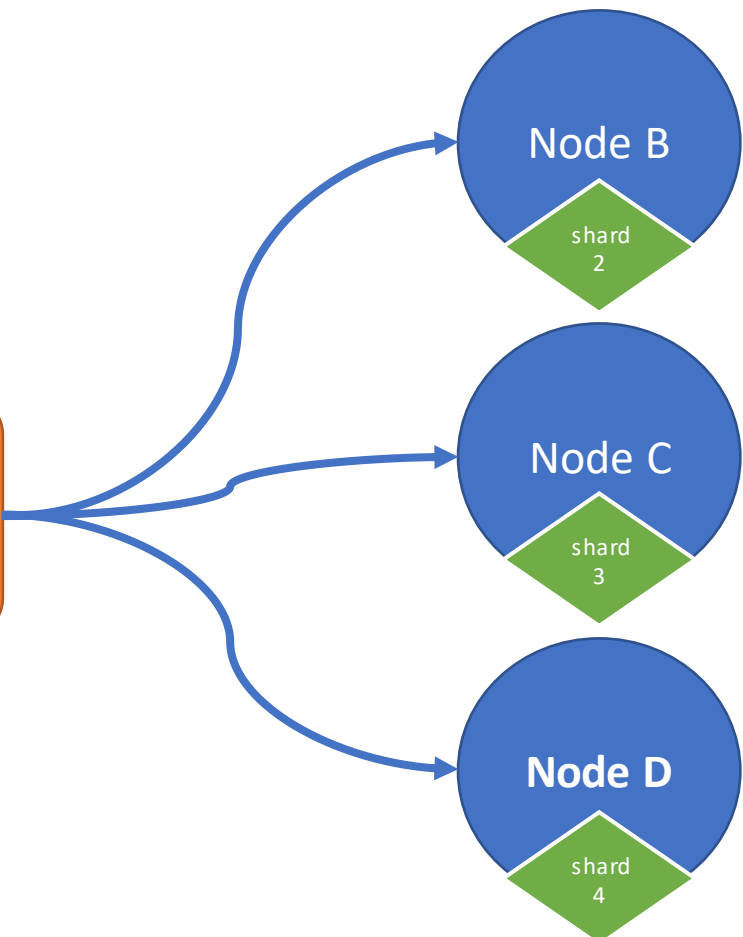
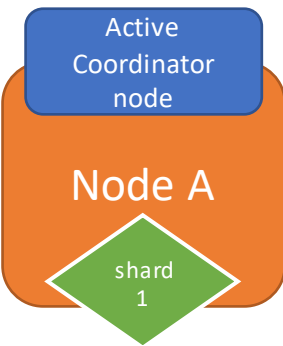
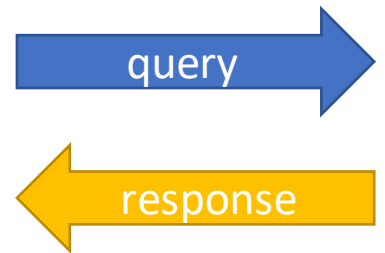
full text searching functionality

- based score (how closely)
- relevancy score (the higher the score,the higher relevancy)

Elasticsearch | Searching

How does search work?

search product index



Elasticsearch | Searching

Rest API

- Special query syntax called [Query DSL](#)(domain specific language)
- Url standart called [Query requests](#)

Elasticsearch | Searching

Search Endpoint

- GET/POST
- search parameter by query string or request body

query string : URI request method

request body : Query DSL ([Best practices](#))

Elasticsearch | Searching

Search Endpoint

```
1 GET kibana_sample_data_ecommerce/_search?q=user:yahya
```

```
4 GET kibana_sample_data_ecommerce/_search
```

```
5 {  
6   "query": {  
7     "match": {  
8       "user": "yahya"  
9     }  
10  }  
11 }
```

```
1 {  
2   "took" : 3,  
3   "timed_out" : false,  
4   "_shards" : {  
5     "total" : 1,  
6     "successful" : 1,  
7     "skipped" : 0,  
8     "failed" : 0  
9   },  
10  "hits" : {  
11    "total" : {  
12      "value" : 79,  
13      "relation" : "eq"  
14    },  
15    "max_score" : 4.0744414,  
16    "hits" : [  
17      {  
109    },  
201    },  
293    },  
385    },  
478    },  
569    },  
662    },  
754    },  
847    },  
939    ],  
940  }  
941 }  
942
```

Query string

Request body

Elasticsearch | Searching

Search Context

- Query context = relevancy score
- Filter context = no relevancy score

history Settings Help

```

1 GET kibana_sample_data_ecommerce/_search
2 {
3   "query": {
4     "match": {
5       "customer_first_name": "Al"
6     }
7   }
8 }
9 GET kibana_sample_data_ecommerce/_search
10 {
11   "query": {
12     "bool": {
13       "filter": [
14         {
15           "match": {"customer_first_name": "Al"}
16         }
17       ]
18     }
19   }
20 }
21 }

```

1- {
2 "took" : 1,
3 "timed_out" : false,
4 "_shards" : {
5 "total" : 1,
6 "successful" : 1,
7 "skipped" : 0,
8 "failed" : 0
9 },
10 "hits" : {
11 "total" : {
12 "value" : 394,
13 "relation" : "eq"
14 },
15 "max_score" : 0.0,
16 "hits" : [
17 {
154 },
291 },
428 },
566 },
704 },
796 },
889 },
982 },
1074 },
1167],
1168 },
1169 }
1170 }

Query context

Filter Context

- No score calculate
- Caching for better performance

Elasticsearch | Searching

Search Response

```
POST kibana_sample_data_ecommerce/_search
{
  "query": {
    "match": {
      "customer_first_name": "Al"
    }
  }
}
```

```
1- {
2-   "took" : 2,
3-   "timed_out" : false,
4-   "_shards" : {
5-     "total" : 1,
6-     "successful" : 1,
7-     "skipped" : 0,
8-     "failed" : 0
9-   },
10-  "hits" : {
11-    "total" : {
12-      "value" : 394,
13-      "relation" : "eq"
14-    },
15-    "max_score" : 1.8717334,
16-    "hits" : [
17-      {
18-        "_index" : "kibana_sample_data_ecommerce",
19-        "_type" : "_doc",
20-        "_id" : "RyK1HIYBSEUAIXirpSoG",
21-        "_score" : 1.8717334,
22-        "_source" : { }
23-      },
24-      { },
25-      { },
26-      { },
27-      { },
28-      { },
29-      { },
30-      { },
31-      { },
32-      { }
33-    ]
34-  }
35-}
```

execution time
(millisecond)

How many
shard returned

number of
result

Highest score
across result

Highest score
across result

Elasticsearch | Searching

URI request searching

GET | POST <index-name>/_search?q=<name:value> and / or <name:value>

```
GET kibana_sample_data_ecommerce/_search?q=user:mary
```

```
GET kibana_sample_data_ecommerce/_search?q=user:mary jim gwen
```

```
GET kibana_sample_data_ecommerce/_search?q=customer_first_name  
:Jason and customer_last_name:Hansen
```

//**sadece** boşluk bırakılırsa or görevini görür.

```
GET kibana_sample_data_ecommerce/_search?q=customer_first_name  
:Jason or customer_last_name:Hansen
```



```
GET kibana_sample_data_ecommerce/_search?q=customer_first_name  
:Jason OR customer_last_name:(Hansen OR Ahmet ) AND  
total_quantity:(>2 AND <10)
```

Elasticsearch | Searching

Query DSL (Domain Spesific Language)

```
GET | POST <index_name>/_search
{
  "query":
    {
      "match": {}
    }
}
```

Elasticsearch | Searching

Search Request

Index name
Multiple index(index1,index2,alias)
No index =search for all index

search
query type

fields

```
1
2 GET kibana_sample_data_ecommerce/_search
3 {
4   "query": {
5     "match": {
6       "customer_first_name": "AL"
7     }
8   }
9 }
10
```

Elasticsearch | Searching

Response (fields)

```
GET kibana_sample_data_ecommerce/_search
{
  "_source": false,
  "size": 20,
  "query": {
    "match": {
      "manufacturer": "Pyramidustries"
    }
  },
  "fields": ["customer_id", "taxful_total_price", "manufacturer"]
}

{
  "took": 3,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1044,
      "relation": "eq"
    },
    "max_score": 2.0555186,
    "hits": [
      {
        "_index": "kibana_sample_data_ecommerce",
        "_type": "_doc",
        "_id": "UyK1HIYBSEUAIXirpSoG",
        "score": 2.0555186,
        "fields": {
          "customer_id": [
            "24"
          ],
          "taxful_total_price": [
            55.98
          ],
          "manufacturer": [
            "Pyramidustries"
          ]
        }
      },
      {
        "_index": "kibana_sample_data_ecommerce",
        "_type": "_doc",

```


Elasticsearch | Searching

Highlighting

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 30,
  "highlight": {
    "fields": {
      "manufacturer": {}
    }
  },
  "query": {
    "multi_match": {
      "query": "Curvy",
      "fields": ["manufacturer"]
    }
  }
}
```

```
378      "Z00711307113"
379    ],
380    "taxful_total_price" : 53.98,
381    "taxless_total_price" : 53.98,
382    "total_quantity" : 2,
383    "total_unique_products" : 2,
384    "type" : "order",
385    "user" : "yasmine",
386    "geoip" : {
387      "country_iso_code" : "SA",
388      "location" : {
389        "lon" : 45,
390        "lat" : 25
391      },
392      "continent_name" : "Asia"
393    },
394    "event" : {
395      "dataset" : "sample_ecommerce"
396    }
397  },
398  "highlight" : {
399    "manufacturer" : [
400      "Spheredrecords <em>Curvy</em>"
401    ]
402  },
403  },
404 }
```

Elasticsearch | Searching

Sorting

```
GET kibana_sample_data_ecommerce/_search
{
  "size": 20,
  "query": {
    "match": {
      "manufacturer": "Pyramidustries"
    }
  },
  "track_scores": true,
  "sort": [
    {
      "taxful_total_price": {
        "order": "desc"
      }
    }
  ]
}
```

Elasticsearch | Term-level Query

Query DSL

- Term-level queries
- Full-text queries

Elasticsearch | Term-level Query

What is Term-level Query?

You can use **term-level queries** to find documents based on precise values in structured data. Examples of structured data include date ranges, IP addresses, prices, or product IDs.

Unlike [full-text queries](#), term-level queries do not analyze search terms. Instead, term-level queries match the exact terms stored in a field.

Elasticsearch | Term-level Query

Searching Types

- Numbers,
- Dates
- IP
- Keyword Types(SGK No,TC No)



Structured searching
not analysis

Elasticsearch | Term-Level Searching

Term query

Returns documents that contain an **exact** term in a provided field.

You can use the term query to find documents based on a precise value such as a price, a product ID, or a username.

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "term": {
      "customer_first_name.keyword": {
        "value": "Sonya"
      }
    }
  }
}
```

```
1 {
2   "took" : 2,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 106,
13      "relation" : "eq"
14    },
15    "max_score" : 3.782053,
16    "hits" : [ ]
939  }
940 }
941
```

Elasticsearch | Term-Level Searching

Terms query

```
1
2
3 GET kibana_sample_data_ecommerce/_search
4 {
5   "query": {
6     "terms": {
7       "customer_first_name.keyword": ["Youssef", "Sonya"]
8     }
9   }
10 }
11
12
```

```
1 {
2   "took" : 3,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 164,
13      "relation" : "eq"
14    },
15    "max_score" : 1.0,
16    "hits" : [ ]
17  }
18 }
19
```

Elasticsearch | Term-level Query

IDs

Returns documents based on their IDs.


```
1
2
3 GET kibana_sample_data_ecommerce/_search
4 {
5   "query": {
6     "ids": {
7       "values": ["1-1FP4YBkq0oY_v0cRqY", "2e1FP4YBkq0oY_v0bhTR"]
8     }
9   }
10 }
```

```
1 {
2   "took": 2,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 2,
13      "relation": "eq"
14    },
15    "max_score": 1.0,
16    "hits": [
17      { },
18      { }
19    ]
20  }
21 }
```


Elasticsearch | Term-level Query

Exist Query

Returns documents that contain an indexed value for a field.



The screenshot displays the Kibana search interface. On the left, the query editor shows a GET request to `kibana_sample_data_ecommerce/_search` with a query object containing an `exists` clause for the `user` field. A red arrow points from the `"field": "user"` line in the query to the `"value": 4675` line in the response. The right pane shows the JSON response, which includes a `hits` array with a `total` object indicating 4675 matches.

```
1 GET kibana_sample_data_ecommerce/_search
2
3 {
4   "query": {
5     "exists": {
6       "field": "user"
7     }
8   }
9 }
10
```

```
1 {
2   "took" : 4,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 4675,
13      "relation" : "eq"
14    },
15    "max_score" : 1.0,
16    "hits" : [ ]
17  }
18 }
```

Elasticsearch | Term-level Query

Prefix

Returns documents that contain a specific prefix in a provided field.

```
1 GET kibana_sample_data_ecommerce/_search
2
3 {
4   "query": {
5     "prefix": {
6       "customer_first_name.keyword": {
7         "value": "Son"
8       }
9     }
10  }
11 }
12
13
14
```

```
1 {
2   "took" : 2,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 106,
13      "relation" : "eq"
14    },
15    "max_score" : 1.0,
16    "hits" : [
17      {
18        "_index" : "kibana_sample_data_ecommerce",
19        "_type" : "_doc",
20        "_id" : "9e1FP4YBkq0oY_v0bA-l",
21        "_score" : 1.0,
22        "_source" : {
23          "customer_first_name" : "Son",
24          "customer_last_name" : "Doe",
25          "customer_email" : "son.doe@example.com",
26          "customer_phone" : "1234567890",
27          "customer_address" : "123 Main St, San Francisco, CA 94102",
28          "customer_created_at" : "2019-01-01T00:00:00Z",
29          "customer_updated_at" : "2019-01-01T00:00:00Z",
30          "customer_status" : "active",
31          "customer_role" : "customer",
32          "customer_verified" : true,
33          "customer_password" : "12345678",
34          "customer_username" : "son.doe",
35          "customer_avatar" : "https://i.pravatar.cc/320?img=1",
36          "customer_bio" : "I am a software engineer and I love to code.",
37          "customer_hobbies" : "reading, hiking, swimming",
38          "customer_languages" : "English, Spanish",
39          "customer_timezone" : "America/Los_Angeles",
40          "customer_currency" : "USD",
41          "customer_locale" : "en-US",
42          "customer_gender" : "male",
43          "customer_age" : 30,
44          "customer_weight" : 70,
45          "customer_height" : 180,
46          "customer_blood_type" : "O+",
47          "customer_marital_status" : "single",
48          "customer_employment_status" : "employed",
49          "customer_education_level" : "graduate",
50          "customer_professional_field" : "software engineering",
51          "customer_skills" : "JavaScript, Python, Java, C++, JavaScript, Python, Java, C++",
52          "customer_certifications" : "AWS Certified Solutions Architect, AWS Certified Solutions Architect",
53          "customer_projects" : "Building a web application, Building a web application",
54          "customer_open_source_contributions" : "Contributed to several open source projects, Contributed to several open source projects",
55          "customer_publications" : "Published several articles on web development, Published several articles on web development",
56          "customer_conferences" : "Spoke at several conferences, Spoke at several conferences",
57          "customer_mentors" : "Mentored several junior developers, Mentored several junior developers",
58          "customer_mentees" : "Mentored several junior developers, Mentored several junior developers",
59          "customer_volunteering" : "Volunteered at several community events, Volunteered at several community events",
60          "customer_donations" : "Made several donations to charity, Made several donations to charity",
61          "customer_rewards" : "Received several awards for my work, Received several awards for my work",
62          "customer_honors" : "Received several awards for my work, Received several awards for my work",
63          "customer_achievements" : "Received several awards for my work, Received several awards for my work",
64          "customer_interests" : "Interested in web development, Interested in web development",
65          "customer_preferences" : "Prefers working in a team, Prefers working in a team",
66          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
67          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
68          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
69          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
70          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
71          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
72          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
73          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
74          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
75          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
76          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
77          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
78          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
79          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
80          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
81          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
82          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
83          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
84          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
85          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
86          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
87          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
88          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
89          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
90          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
91          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
92          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
93          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
94          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
95          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
96          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
97          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
98          "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
99          "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
100         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
101         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
102         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
103         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
104         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
105         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
106         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
107         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
108         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
109         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
110         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
111         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
112         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
113         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
114         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
115         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
116         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
117         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
118         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
119         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
120         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
121         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
122         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
123         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
124         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
125         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
126         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
127         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
128         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
129         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
130         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
131         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
132         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
133         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
134         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
135         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
136         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
137         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
138         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
139         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
140         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
141         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
142         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
143         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
144         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
145         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
146         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
147         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
148         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
149         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
150         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
151         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
152         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
153         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
154         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
155         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
156         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
157         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
158         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
159         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
160         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
161         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
162         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
163         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
164         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
165         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
166         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
167         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
168         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
169         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
170         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
171         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
172         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
173         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
174         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
175         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
176         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
177         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
178         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
179         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
180         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
181         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
182         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
183         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
184         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
185         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
186         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
187         "customer_favorites" : "Favorite food is pizza, Favorite food is pizza",
188         "customer_dislikes" : "Dislikes working in a team, Dislikes working in a team",
189         "customer_favorites
```

Elasticsearch | Term-level Query

Range

Returns documents that contain terms within a provided range.

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "range": {
      "taxless_total_price": {
        "gte": 33.98,
        "lte": 40
      }
    }
  }
}
```

gt	greater than
gte	greater than or equal
lt	less then
lte	less then or equal

Elasticsearch | Term-Level Searching

Wildcard Query

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "wildcard": {
      "customer_last_name.keyword": {
        "value": "Lam*"
      }
    }
  }
}

GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "wildcard": {
      "customer_last_name.keyword": {
        "value": "Lam*rt"
      }
    }
  }
}

GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "wildcard": {
      "customer_last_name.keyword": {
        "value": "Lambe?t"
      }
    }
  }
}
```

*	multiple characters
?	single character

Elasticsearch | Term-Level Searching

Fuzzy Query

```
GET kibana_sample_data_ecommerce/_search
```

```
{
  "query": {
    "fuzzy": {
      "customer_first_name.keyword": {
        "value": "tephanie" //Stephanie
        , "fuzziness": 1
      }
    }
  }
}
```

```
GET kibana_sample_data_ecommerce/_search
```

```
{
  "query": {
    "fuzzy": {
      "customer_first_name.keyword": {
        "value": "Stepnie" //Stephanie
        , "fuzziness": 2
      }
    }
  }
}
```

cake
take
bake
lake
make
nake

Elasticsearch | Term-Level Searching

Fuzzy Query With Auto

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "fuzzy": {
      "customer_first_name.keyword": {
        "value": "Sphanie" //Stephanie
      }
    }
  }
}
```

Characters length	Fuzziness
0-2	0
3-5	1
5	2

Elasticsearch | Searching

Pagination

```
GET kibana_sample_data_ecommerce/_search
```

```
{  
  "size": 3,  
  "from": 2,  
  "query": {  
    "multi_match": {  
      "query": "Pyramidustries",  
      "fields": ["manufacturer"]  
    }  
  }  
}
```

Elasticsearch | Searching

Response (includes,excludes)

```
GET kibana_sample_data_ecommerce/_search
{
  "_source":{
    "includes": ["customer_id","taxful_total_price","manufacturer"]
  },
  "size": 20,
  "query": {
    "match": {
      "manufacturer": "Pyramidustries"
    }
  }
}
```

```
1- {
2  "took" : 4,
3  "timed_out" : false,
4  "_shards" : {
5    "total" : 1,
6    "successful" : 1,
7    "skipped" : 0,
8    "failed" : 0
9  },
10 "hits" : {
11   "total" : {
12     "value" : 1044,
13     "relation" : "eq"
14   },
15   "max_score" : 2.0555186,
16   "hits" : [
17     {
18       "_index" : "kibana_sample_data_ecommerce",
19       "_type" : "_doc",
20       "_id" : "UyK1HIYBSEUAIxrpSoG",
21       "score" : 2.0555186,
22       "_source" : {
23         "customer_id" : 24,
24         "manufacturer" : [
25           "Pyramidustries"
26         ],
27         "taxful_total_price" : 55.98
28       }
29     },
30     ...
31   ]
32 }
```


Elasticsearch | Term-Level Searching

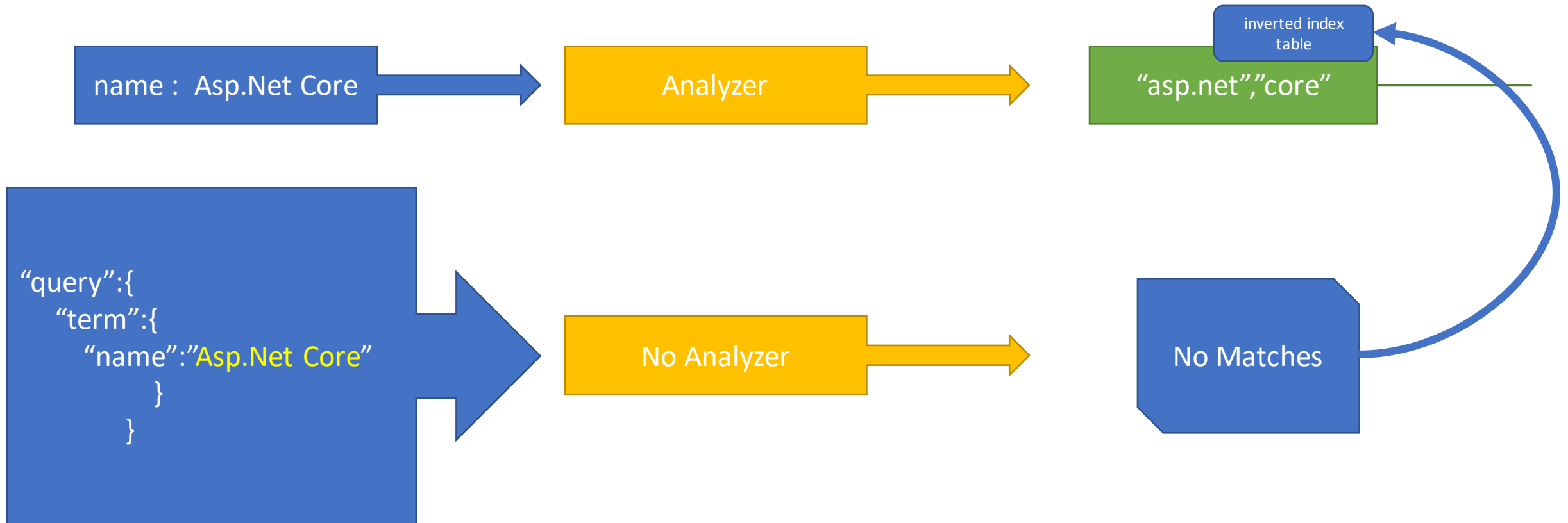
Range with sort query

```
GET kibana_sample_data_ecommerce/_search
```

```
{
  "query": {
    "range": {
      "taxless_total_price": {
        "gte": 33.98,
        "lte": 40
      }
    }
  },
  "sort": [
    {
      "taxless_total_price": {
        "order": "desc"
      }
    }
  ]
}
```

Elasticsearch | Term-level Query

Searching Types



Elasticsearch | Term-level Query

Term Query

```
1 GET kibana_sample_data_ecommerce/_search
2 {
3   "query": {
4     "term": {
5       "customer_gender": "MALE"
6     }
7   }
8 }
9
10
11 GET kibana_sample_data_ecommerce/_search
12 {
13   "query": {
14     "term": {
15       "customer_gender": "Male"
16     }
17   }
18 }
19 }
```

keyword type

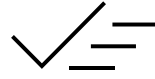
inverted index table

MALE		



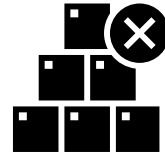
GET kibana_sample_data_ecommerce/_search

```
{
  "query": {
    "term": {
      "customer_full_name.keyword": "Elyssa Bryan"
    }
  }
}
```



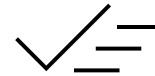
GET kibana_sample_data_ecommerce/_search

```
{
  "query": {
    "term": {
      "customer_full_name.keyword": "elyssa bryan"
    }
  }
}
```



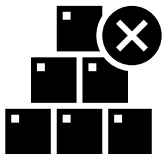
GET kibana_sample_data_ecommerce/_search

```
{
  "query": {
    "term": {
      "customer_full_name": "elyssa bryan"
    }
  }
}
```



GET kibana_sample_data_ecommerce/_search

```
{
  "query": {
    "term": {
      "customer_full_name": "elyssa"
    }
  }
}
```



inverted index table

Elyssa Bryan (keyword type)

elyssa (text type)

bryan (text type)

Elasticsearch | Term-Level Searching

Range with expression format

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "range": {
      "order_date": {
        "lte": "2025-03-03T11-2d"
      }
    }
  }
}

GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "range": {
      "order_date": {
        "lte": "now-2d"
      }
    }
  }
}
```

year	y
day	d
month	M
week	w
hour	h
minute	m
s	

Elasticsearch | Term-Level Searching

Match all query

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "match_all": {}
  }
}
```

```
1- {
2   "took" : 5,
3   "timed_out" : false,
4-  "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9-  },
10-  "hits" : {
11-    "total" : {
12      "value" : 4675,
13      "relation" : "eq"
14-    },
15    "max_score" : 1.0,
16-    "hits" : [ ]
1069  }
1070 }
1071
```

Elasticsearch | Full text queries

Match Query

Returns documents that match a provided text, number, date or boolean value. The provided text is analyzed before matching.

The match query is the standard query for performing a full-text search

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "match": {
      "category": "shoes"
    }
  }
}

GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "match": {
      "category": "shoes clothing"
    }
  }
}
```

```
1 {
2   "took" : 2,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2080,
13      "relation" : "eq"
14    },
15    "max_score" : 0.9538229,
16    "hits" : [
17      {
18        "_index" : "kibana_sample_data_ecommerce",
19        "_type" : "_doc",
20        "_id" : "9-1FP4YBkq0oY_v0bA-l",
21        "_score" : 0.9538229,
22        "_source" : {
23          "category" : [
24            "Women's Shoes"
25          ],
26          "currency" : "EUR",
27          "customer_first_name" : "Brigitte",
28          "customer_full_name" : "Brigitte King",
29          "customer_gender" : "FEMALE",
30          "customer_id" : 12,
```

Elasticsearch | Full text queries

Match Query-2 (And keyword)

```
GET kibana_sample_data_ecommerce/_search
```

```
{
  "query": {
    "match": {
      "category": {
        "query": "Women's Clothing",
        "operator": "and"
      }
    }
  }
}
```

```
1 {
2   "took" : 2,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 1968,
13      "relation" : "eq"
14    },
15    "max_score" : 0.976707,
16    "hits" : [
17      {
18        "_index" : "kibana_sample_data_ecommerce",
19        "_type" : "_doc",
20        "_id" : "3e1FP4YBkq0oY_v0bA-l",
21        "_score" : 0.976707,
22        "_source" : {
23          "category" : [
24            "Women's Shoes",
25            "Women's Clothing"
26          ],
27          "currency" : "EUR",
28          "customer_first_name" : "Gwen",
29          "customer_full_name" : "Gwen Butler",
30          "customer_gender" : "FEMALE",
31          "customer_id" : 26,
32          "customer_last_name" : "Butler",
33          "customer_phone" : "",
34          "day_of_week" : "Sunday",
35          "day_of_week_i" : 6,
36          "email" : "gwen@butler-family-zzz"
```


Elasticsearch | Full-text query

Multi-match Query

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "multi_match": {
      "query": "gwen",
      "fields": ["customer_first_name", "customer_last_name", "customer_full_name"]
    }
  }
}
```

```
1 {
2   "took" : 2,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 122,
13      "relation" : "eq"
14    },
15    "max_score" : 3.8098755,
16    "hits" : [ ]
17  }
18 }
```

Elasticsearch | Full-text query

Match Boolean Prefix

```
1 GET kibana_sample_data_ecommerce/_search
2 {
3   "query": {
4     "match_bool_prefix": {
5       "category": "Wo"
6     }
7   }
8 }
9 }
```

```
1 GET kibana_sample_data_ecommerce/_search
2 {
3   "query": {
4     "match_bool_prefix": {
5       "category": "Clothing acc"
6     }
7   }
8 }
9 }
```

Elasticsearch | Full-text query

Match Phrase Query

Elasticsearch | Full-text query

Match Phrase Prefix Query

Elasticsearch | Compound Query

Compound Query (bool)

Occur	Description
must	The clause (query) must appear in matching documents and will contribute to the score.
filter	The clause (query) must appear in matching documents. However unlike must the score of the query will be ignored.
should	The clause (query) should appear in the matching document. it is not mandatory (like OR)
must_not	The clause (query) must not appear in the matching documents

Elasticsearch | Compound Query

Compound Query (bool)

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "geoip.city_name": "New York"
          }
        }
      ],
      "must_not": [
        {
          "range": {
            "taxful_total_price": {
              "lte": 100.00
            }
          }
        }
      ],
      "should": [
        {
          "term": {
            "category.keyword": "Women's Clothing"
          }
        }
      ],
      "filter": [
        {
          "term": {
            "manufacturer.keyword": "Tigress Enterprises"
          }
        }
      ]
    }
  }
}
```

```
1 - {
2   "took" : 1,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 74,
13      "relation" : "eq"
14    },
15    "max_score" : 2.5775695,
16    "hits" : [
17      {
18        "_index" : "kibana_sample_data_ecommerce",
19        "_type" : "_doc",
20        "_id" : "K01FP4YBkq0oY_v0bBCl",
21        "_score" : 2.5775695,
22        "_source" : {
23          "category" : [
24            "Women's Shoes",
25            "Women's Clothing"
26          ],
27          "currency" : "EUR",
28          "customer_first_name" : "Brigitte",
29          "customer_full_name" : "Brigitte Morris",
30          "customer_gender" : "FEMALE",
31          "customer_id" : 12,
32          "customer_last_name" : "Morris",
33          "customer_phone" : "",
34          "day_of_week" : "Saturday",
35          "day_of_week_i" : 5,
36          "email" : "brigitte@morris-family.zzz",
```

Elasticsearch | Aggregations

Aggregations Types

1. Bucket aggregations
2. Metric aggregations

Elasticsearch | Aggregations

Bucket aggregations

Terms Aggregations

Elasticsearch | Aggregations

Bucket aggregations

Range Aggregations

Elasticsearch | Aggregations

Aggregations

ipucu

Elasticsearch | Aggregations

Metric aggregations

Avg Aggregations

Elasticsearch | Aggregations

Metric aggregations

Sum/Max/Min Aggregations

Elasticsearch | Example Proje(Blog)

Avg aggregation

Elasticsearch | .Net Core Web API

Projenin oluşturulması ve **NEST** kütüphanesinin yüklenmesi

Elasticsearch | .Net Core Web API

ElasticClient sınıfının oluşturulması

Elasticsearch | .Net Core Web API

Product ve **ProductFeature** sınıflarının oluşturulması

Blog içeriği için schema belirleme

Elasticsearch | .Net Core Web API

ProductRepository ve **Save** methodunun oluşturulması (CRUD)

Elasticsearch | .Net Core Web API

ProductService ve **Save** methodunun oluşturulması (CRUD)

Elasticsearch | .Net Core Web API

ProductsController sınıfının oluşturulması

Elasticsearch | .Net Core Web API

Client tarafından ID'nin belirlenmesi ve BaseController sınıfının oluşturulması

Elasticsearch | .Net Core Web API

GetAll method

Elasticsearch | .Net Core Web API

GetByID method

Elasticsearch | .Net Core Web API

Update method

Elasticsearch | .Net Core Web API

Delete method

Elasticsearch | .Net Core Web API

Hataları ele almak

Elasticsearch | .Net Core Web API

Elasticsearch kütüphaneleri



NEST Library
7.17

Elastic.Clients.Elasticsearch
8.1.1

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch



Elastic.Clients.Elasticsearch
8.1.1

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Term Level Query

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Term Query-2

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Terms Query

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Prefix Query

```
1 GET kibana_sample_data_ecommerce/_search
2 {
3   "query": {
4     "prefix": {
5       "customer_first_name.keyword": {
6         "value": "Son"
7       }
8     }
9   }
10 }
11
12
13
14
15 {
16   "took" : 2,
17   "timed_out" : false,
18   "_shards" : {
19     "total" : 1,
20     "successful" : 1,
21     "skipped" : 0,
22     "failed" : 0
23   },
24   "hits" : {
25     "total" : {
26       "value" : 106,
27       "relation" : "eq"
28     },
29     "max_score" : 1.0,
30     "hits" : [
31       {
32         "_index" : "kibana_sample_data_ecommerce",
33         "_type" : "_doc",
34         "_id" : "9e1FP4YBkq0oY_v0bA-1",
35         "_score" : 1.0,
36         "_source" : {
37           "customer_first_name": "Son",
38           "customer_last_name": "Doe",
39           "customer_email": "son.doe@example.com",
40           "customer_phone": "1234567890",
41           "customer_address": "123 Main St, San Francisco, CA 94102",
42           "customer_created": "2019-01-01T00:00:00Z",
43           "customer_updated": "2019-01-01T00:00:00Z",
44           "customer_deleted": "2019-01-01T00:00:00Z",
45           "customer_status": "active",
46           "customer_verified": true,
47           "customer_password": "12345678",
48           "customer_username": "son.doe",
49           "customer_avatar": "https://i.pravatar.cc/150?img=1",
50           "customer_bio": "I am a software engineer and I love to code.",
51           "customer_hobbies": "reading, hiking, playing guitar",
52           "customer_languages": "English, Spanish",
53           "customer_skills": "JavaScript, Python, Java",
54           "customer_experience": "5 years",
55           "customer_education": "Bachelor's degree",
56           "customer_employment": "Full-time",
57           "customer_salary": 100000,
58           "customer_benefits": "Health insurance, 401k",
59           "customer_job_title": "Software Engineer",
60           "customer_company": "Example Corp",
61           "customer_department": "Engineering",
62           "customer_manager": "John Doe",
63           "customer_team": "Frontend",
64           "customer_projects": "Project X, Project Y",
65           "customer_tasks": "Task A, Task B",
66           "customer_status_history": [
67             {
68               "status": "active",
69               "start": "2019-01-01T00:00:00Z",
70               "end": "2019-01-01T00:00:00Z"
71             }
72           ],
73           "customer_verification_code": "123456",
74           "customer_reset_password_token": "1234567890",
75           "customer_login_attempts": 0,
76           "customer_last_login": "2019-01-01T00:00:00Z",
77           "customer_created_at": "2019-01-01T00:00:00Z",
78           "customer_updated_at": "2019-01-01T00:00:00Z",
79           "customer_deleted_at": "2019-01-01T00:00:00Z",
80           "customer_status_updated_at": "2019-01-01T00:00:00Z",
81           "customer_verified_at": "2019-01-01T00:00:00Z",
82           "customer_password_updated_at": "2019-01-01T00:00:00Z",
83           "customer_username_updated_at": "2019-01-01T00:00:00Z",
84           "customer_avatar_updated_at": "2019-01-01T00:00:00Z",
85           "customer_bio_updated_at": "2019-01-01T00:00:00Z",
86           "customer_hobbies_updated_at": "2019-01-01T00:00:00Z",
87           "customer_languages_updated_at": "2019-01-01T00:00:00Z",
88           "customer_skills_updated_at": "2019-01-01T00:00:00Z",
89           "customer_experience_updated_at": "2019-01-01T00:00:00Z",
90           "customer_education_updated_at": "2019-01-01T00:00:00Z",
91           "customer_employment_updated_at": "2019-01-01T00:00:00Z",
92           "customer_salary_updated_at": "2019-01-01T00:00:00Z",
93           "customer_benefits_updated_at": "2019-01-01T00:00:00Z",
94           "customer_job_title_updated_at": "2019-01-01T00:00:00Z",
95           "customer_company_updated_at": "2019-01-01T00:00:00Z",
96           "customer_department_updated_at": "2019-01-01T00:00:00Z",
97           "customer_manager_updated_at": "2019-01-01T00:00:00Z",
98           "customer_team_updated_at": "2019-01-01T00:00:00Z",
99           "customer_projects_updated_at": "2019-01-01T00:00:00Z",
100          "customer_tasks_updated_at": "2019-01-01T00:00:00Z",
101          "customer_status_history_updated_at": "2019-01-01T00:00:00Z",
102          "customer_verification_code_updated_at": "2019-01-01T00:00:00Z",
103          "customer_reset_password_token_updated_at": "2019-01-01T00:00:00Z",
104          "customer_login_attempts_updated_at": "2019-01-01T00:00:00Z",
105          "customer_last_login_updated_at": "2019-01-01T00:00:00Z",
106          "customer_created_at_updated_at": "2019-01-01T00:00:00Z",
107          "customer_updated_at_updated_at": "2019-01-01T00:00:00Z",
108          "customer_deleted_at_updated_at": "2019-01-01T00:00:00Z",
109          "customer_status_updated_at_updated_at": "2019-01-01T00:00:00Z",
110          "customer_verified_at_updated_at": "2019-01-01T00:00:00Z",
111          "customer_password_updated_at_updated_at": "2019-01-01T00:00:00Z",
112          "customer_username_updated_at_updated_at": "2019-01-01T00:00:00Z",
113          "customer_avatar_updated_at_updated_at": "2019-01-01T00:00:00Z",
114          "customer_bio_updated_at_updated_at": "2019-01-01T00:00:00Z",
115          "customer_hobbies_updated_at_updated_at": "2019-01-01T00:00:00Z",
116          "customer_languages_updated_at_updated_at": "2019-01-01T00:00:00Z",
117          "customer_skills_updated_at_updated_at": "2019-01-01T00:00:00Z",
118          "customer_experience_updated_at_updated_at": "2019-01-01T00:00:00Z",
119          "customer_education_updated_at_updated_at": "2019-01-01T00:00:00Z",
120          "customer_employment_updated_at_updated_at": "2019-01-01T00:00:00Z",
121          "customer_salary_updated_at_updated_at": "2019-01-01T00:00:00Z",
122          "customer_benefits_updated_at_updated_at": "2019-01-01T00:00:00Z",
123          "customer_job_title_updated_at_updated_at": "2019-01-01T00:00:00Z",
124          "customer_company_updated_at_updated_at": "2019-01-01T00:00:00Z",
125          "customer_department_updated_at_updated_at": "2019-01-01T00:00:00Z",
126          "customer_manager_updated_at_updated_at": "2019-01-01T00:00:00Z",
127          "customer_team_updated_at_updated_at": "2019-01-01T00:00:00Z",
128          "customer_projects_updated_at_updated_at": "2019-01-01T00:00:00Z",
129          "customer_tasks_updated_at_updated_at": "2019-01-01T00:00:00Z",
130          "customer_status_history_updated_at_updated_at": "2019-01-01T00:00:00Z",
131          "customer_verification_code_updated_at_updated_at": "2019-01-01T00:00:00Z",
132          "customer_reset_password_token_updated_at_updated_at": "2019-01-01T00:00:00Z",
133          "customer_login_attempts_updated_at_updated_at": "2019-01-01T00:00:00Z",
134          "customer_last_login_updated_at_updated_at": "2019-01-01T00:00:00Z",
135          "customer_created_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
136          "customer_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
137          "customer_deleted_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
138          "customer_status_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
139          "customer_verified_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
140          "customer_password_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
141          "customer_username_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
142          "customer_avatar_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
143          "customer_bio_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
144          "customer_hobbies_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
145          "customer_languages_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
146          "customer_skills_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
147          "customer_experience_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
148          "customer_education_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
149          "customer_employment_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
150          "customer_salary_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
151          "customer_benefits_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
152          "customer_job_title_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
153          "customer_company_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
154          "customer_department_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
155          "customer_manager_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
156          "customer_team_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
157          "customer_projects_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
158          "customer_tasks_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
159          "customer_status_history_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
160          "customer_verification_code_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
161          "customer_reset_password_token_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
162          "customer_login_attempts_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
163          "customer_last_login_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
164          "customer_created_at_updated_at_updated_at_updated_at": "2019-01-01T00:00:00Z",
165          "customer_updated_at_updated_at_updated
```

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Range Query

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "range": {
      "taxless_total_price": {
        "gte": 33.98,
        "lte": 40
      }
    }
  }
}
```


Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

MathAll Query

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Pagination

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Wildcard

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "wildcard": {
      "customer_last_name.keyword": {
        "value": "Lam*"
      }
    }
  }
}

GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "wildcard": {
      "customer_last_name.keyword": {
        "value": "Lam*rt"
      }
    }
  }
}

GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "wildcard": {
      "customer_last_name.keyword": {
        "value": "Lambe?t"
      }
    }
  }
}
```

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Fuzzy

```
GET kibana_sample_data_ecommerce/_search
```

```
{
  "query": {
    "fuzzy": {
      "customer_first_name.keyword": {
        "value": "tephanie" //Stephanie
        , "fuzziness": 1
      }
    }
  }
}
```

```
GET kibana_sample_data_ecommerce/_search
```

```
{
  "query": {
    "fuzzy": {
      "customer_first_name.keyword": {
        "value": "Stepnie" //Stephanie
        , "fuzziness": 2
      }
    }
  }
}
```

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Match Query

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "match": {
      "category": "shoes"
    }
  }
}

GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "match": {
      "category": "shoes clothing"
    }
  }
}
```

```
1 {
2   "took" : 2,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2080,
13      "relation" : "eq"
14    },
15    "max_score" : 0.9538229,
16    "hits" : [
17      {
18        "_index" : "kibana_sample_data_ecommerce",
19        "_type" : "_doc",
20        "_id" : "9-1FP4YBkq0oY_v0bA-l",
21        "_score" : 0.9538229,
22        "_source" : {
23          "category" : [
24            "Women's Shoes"
25          ],
26          "currency" : "EUR",
27          "customer_first_name" : "Brigitte",
28          "customer_full_name" : "Brigitte King",
29          "customer_gender" : "FEMALE",
30          "customer_id" : 12,
```

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Match Boolean Prefix

```
1 GET kibana_sample_data_ecommerce/_search
2 {
3   "query": {
4     "match_bool_prefix": {
5       "category": "Wo"
6     }
7   }
8 }
9
1 GET kibana_sample_data_ecommerce/_search
2 {
3   "query": {
4     "match_bool_prefix": {
5       "category": "Clothing acc"
6     }
7   }
8 }
9
10
```

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Match Phrase Query

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Match Phrase Prefix Query

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Compound Query Example-1

```
GET kibana_sample_data_ecommerce/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "geoip.city_name": "New York"
          }
        }
      ],
      "must_not": [
        {
          "range": {
            "taxful_total_price": {
              "lte": 100.00
            }
          }
        }
      ],
      "should": [
        {
          "term": {
            "category.keyword": "Women's Clothing"
          }
        }
      ],
      "filter": [
        {
          "term": {
            "manufacturer.keyword": "Tigress Enterprises"
          }
        }
      ]
    }
  }
}
```

```
1- {
2  "took" : 1,
3  "timed_out" : false,
4  "_shards" : {
5    "total" : 1,
6    "successful" : 1,
7    "skipped" : 0,
8    "failed" : 0
9  },
10 "hits" : {
11   "total" : {
12     "value" : 74,
13     "relation" : "eq"
14   },
15   "max_score" : 2.5775695,
16   "hits" : [
17     {
18       "_index" : "kibana_sample_data_ecommerce",
19       "_type" : "_doc",
20       "_id" : "K01FP4YBkq0oY_v0bBc1",
21       "_score" : 2.5775695,
22       "_source" : {
23         "category" : [
24           "Women's Shoes",
25           "Women's Clothing"
26         ],
27         "currency" : "EUR",
28         "customer_first_name" : "Brigitte",
29         "customer_full_name" : "Brigitte Morris",
30         "customer_gender" : "FEMALE",
31         "customer_id" : 12,
32         "customer_last_name" : "Morris",
33         "customer_phone" : "",
34         "day_of_week" : "Saturday",
35         "day_of_week_i" : 5,
36         "email" : "brigitte@morris-family.zzz",
```

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Compound Query Example-2

Elasticsearch | .Net Core Web API

Elastic.Clients.Elasticsearch

Multi Match Query

GET kibana_sample_data_ecommerce/_search

```
{
  "query": {
    "multi_match": {
      "query": "gwen",
      "fields": ["customer_first_name", "customer_last_name", "customer_full_name"]
    }
  }
}
```



```
1 {
2   "took" : 2,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 122,
13      "relation" : "eq"
14    },
15    "max_score" : 3.8098755,
16    "hits" : [ ]
17  }
18 }
```



Elasticsearch | Example Project (Blog App)

1. Adım

Blog schema'sının belirlenmesi (Explicit Mapping)

Elasticsearch | Example Project (Blog App)

2. Adım

- MVC projemizin oluşturulması
- Blog model class'ının oluşturulması

Elasticsearch | Example Project (Blog App)

3. Adım

- ElasticsearchClient sınıfının oluşturulması
- Save methodunun kodlanması

Elasticsearch | Example Project (Blog App)

4. Adım

- BlogSevice sınıfının oluşturulması

Elasticsearch | Example Project (Blog App)

5. Adım

- BlogController sınıfının oluşturulması

Elasticsearch | Example Project (Blog App)

6. Adım

- Search Methodunun oluşturulması

Elasticsearch | Example Project (Blog App)

7. Adım

- Search sayfasının oluşturulması

Elasticsearch | Example Project (Blog App)

8. Adım

- Search sonuçlarının gösterilmesi.

Elasticsearch | Example Project (Blog App)

9. Admin

- View Best Practices

Elasticsearch | Example Project (ECommerce App)

1. Adım

- SearchFormViewModel class

Elasticsearch | Example Project (ECommerce App)

2. Adım

- SearchAsync Method

Elasticsearch | Example Project (ECommerce App)

3. Adım

- Service : SearchAsync Method

Elasticsearch | Example Project (ECommerce App)

4. Adım

- EcommerceController and SearchPageViewModel

Elasticsearch | Example Project (ECommerce App)

5. Adım

- Search Page

Elasticsearch | Example Project (ECommerce App)

6. Admin

- Pagination