

ElasticSearch

- Büyük verileri analiz etme, bunların arasında arama yapma ve önerme gibi işlemleri gerçekleştiren Java ile geliştirilmiş güçlü, esnek ve performanslı bir teknolojidir.
- Apache Lucene altyapısına sahip olduğu için mükemmel bir full-text search yeteneğine sahiptir.
- Document-based bir yapıya sahiptir ve bu nedenle verileri JSON formatında tutar.
- Tüm search, CRUD ve analiz işlemleri sağladığı Rest API üzerinden rahatlıkla yapılabilir.

Search Engine Candidates:

- Elasticsearch
- Solr
- InfluxDB
- TypeSense
- Algolia
- Sphinx

For more:

- <https://www.educba.com/elasticsearch-alternatives/>
- <https://www.linuxlinks.com/searchengines/>

Solr vs Elasticsearch

Both Solr and Elasticsearch write indexes in Lucene. But, since differences exist in sharding and replication (among other features), there are also differences in their files and architectures. Additionally, Elasticsearch has native DSL support while Solr has a robust Standard Query Parser that aligns to Lucene syntax.

For more: <https://logz.io/blog/solr-vs-elasticsearch/>

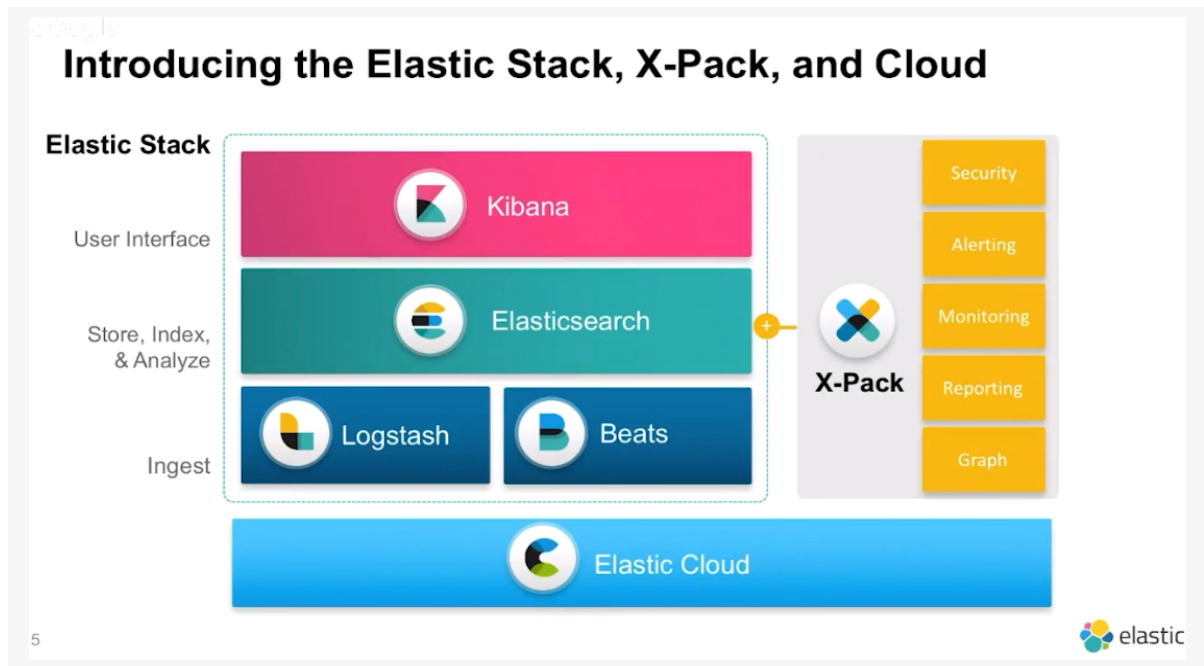
ElasticSearch Features:

- **Lots of search options**
- **Document-oriented**
- **Speed**
- **Scalability**

- **Data record**
- **Query fine tuning**
- **RESTful API**
- **Distributed approach**
- **Multi-tenancy**

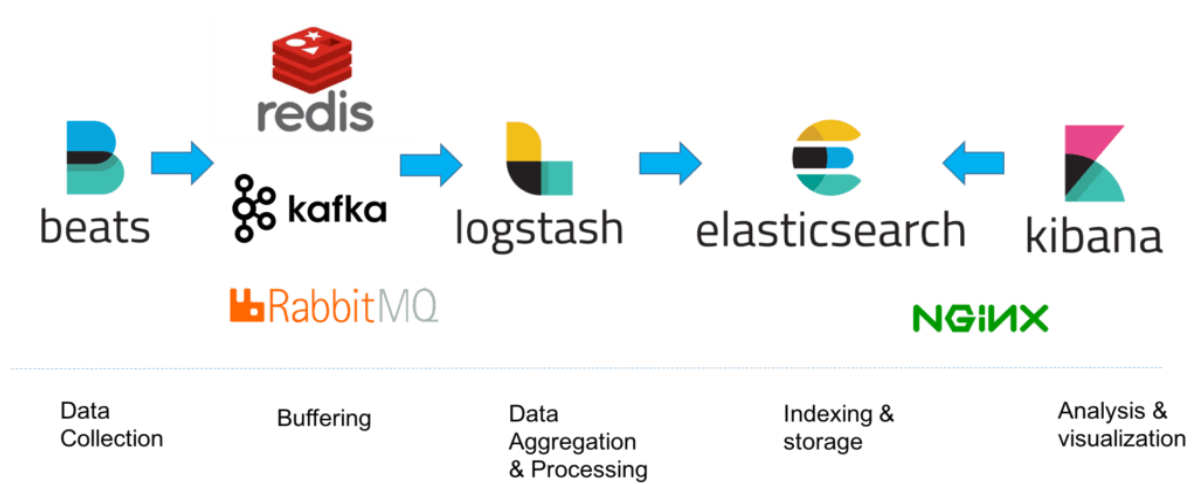
For more:

- <https://www.elastic.co/elastic-stack/features>
- <https://dzone.com/articles/elastic-search-advantages-case-studies-amp-books>



Senaryomuz üzerinde clientlardan topladıkları logları elkstack sunucumuza yollamakla yükümlü aracımız **Beats** , toplanan bu logları bir sunucu içinde derleyip üzerinde yapılması gereken bir dönüştürme ve anlam kazandırma işlemini gerçekleştiren aracımız **Logstash** ,bu logları indexlendirerek aranabilir ve analiz edilebilir hale dönüştüren aracımız **Elasticsearch** ve tüm bu yapılan detaylı çalışmayı bizlere görselleştiren aracımız **Kibana** olarak karşımıza çıkmaktadır.

- Her node'da 1 adet ElasticSearch çalışmaktadır. Her node birbiri ile haberleşir.
- **Kibana** konfigürasyonlarını Elastic Search üzerinde tutar.
- **Logstash** pipeline: Inputs -> Filters -> Outputs
- **Beats**: Dosyaları, log'ları toplar LogStash veya ElasticSearch'e aktarır. FileBeat, MetricBeat, PacketBeat, WinLogBeat, AuditBeat, HeartBeat.

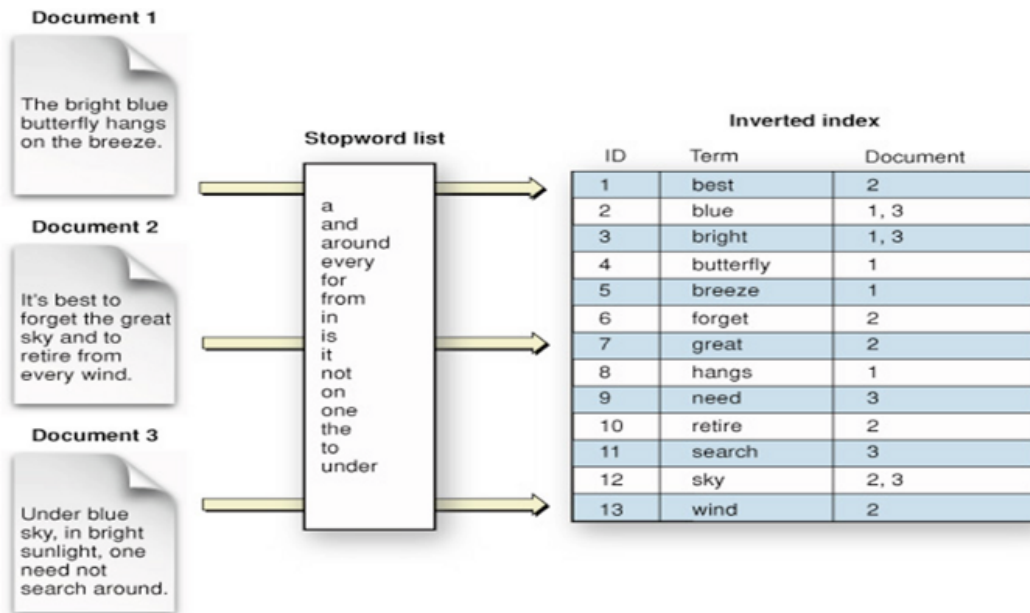


Terimler

Index: Index'ler, Json veriler topluluğudur. Index'leri biz belirleriz. Bir kelime veya cümle olarak tanımlayabiliriz. Eklenen her dökümandaki her bir kelime için hangi döküman ya da dökümanlarda o kelimenin olduğu bilgisini tutan bir indeksleme sistemi vardır.

Aramayı çok kolay bir hale getiren Apache Lucene'in **Inverted Index**'leri ve içerisindeki verinin şemasını tanımlayan **mapping**'i içerirler.

Bir veri insert edilirken **Apache Lucene** alt yapısı kullanılarak index'lenir.



Mapping: İlgili dökümanın arama motoruna nasıl aktarılacağını tanımlar. Index'ler iken o verinin türünü bu şekilde tanımlarız.

```
{
  "animal": {
    "badger": {
      "properties": {
        "bio": {
          "type": "string"
        },
        "name": {
          "type": "string"
        },
        "role": {
          "type": "string"
        }
      }
    }
  }
}
```

Types:

- Explicit Mapping: Field'lar ve type'ları önceden tanımlıdır.
- Dynamic Mapping: Field'lar ve type'ları Elasticsearch tarafından otomatik olarak tanımlanır.

Field types,

- Basit Veri tipleri: String, byte, short, integer, long, float, double, boolean, date
- Kompleks veri tipleri: object, nested
- Özel veri tipleri: geo_point, geo_shape, completion

Field types	Field Index	Field Analyzer
String, byte, short, integer, long, float, double, boolean, date	Do you want this field indexed for full-text search? analyzed / not_analyzed / no	Define your tokenizer and token filter. standard / whitespace / simple / english etc.
<pre>{ "properties": { "user_id": { "type": "long" } } }</pre>	<pre>{ "properties": { "genre": { "index": "not_analyzed" } } }</pre>	<pre>{ "properties": { "description": { "analyzer": "english" } } }</pre>

Document: Elasticsearch'teki her bir kayda denir. Her döküman kendi unique Id'sine sahiptir. Her bir document bir type ve unique id'ye sahiptir ve bir index içerisinde tutulur.

Field: Her döküman içerisindeki alana field denir. RDBMs'teki column'a denk gelir. Bir document birden fazla field'a sahip olabilir.

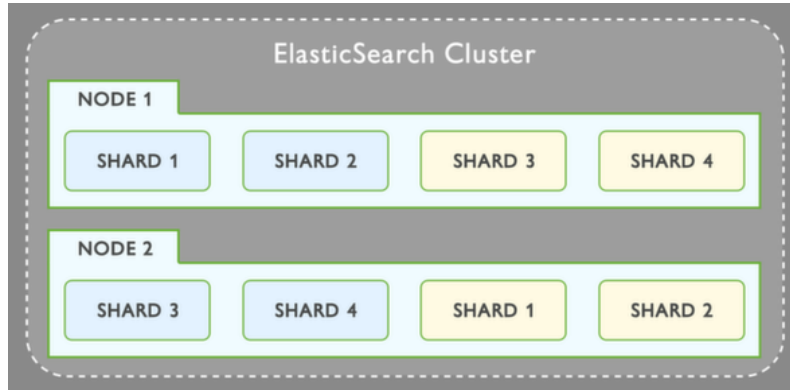
Type: Aslında Relational Database'lerde tablo' ya denk gelir. product ve user gibi. ES 7.0 dan sonra kaldırılmıştır. Type'ın RDBMs'de tablolara denk geldiğini söylesek de durum tam olarak öyle değil. İlişkisel veritabanında tablolar birbirinden

bağımsızdır. Örnek olarak user tablosundaki name field'ı ile product tablosundaki name field'ının bir bağlantısı yoktur. Fakat Elasticsearch'te aynı index altında bulunan field'lar arka planda aynı Lucene field'ına denk gelmektedir. Yani user type'ının name field'ı ile product type'ının name field'ı aynı alanda saklanır. Bu da bir takım sorunlara yol açmaktadır.

Relational Database	Elasticsearch
Schema	Mapping
Database	Index
Table	Type
Row	Document
Column	Field

Cluster: Birden fazla Node'un tutulduğu yapıya denir.

Node: Verilerin depolandığı makinelerin her biridir. Cluster'ların indexleme ve arama yetenekleri, bu node'lar ile gerçekleştirilir.



Shard: Shard'lar **Apache Lucene**'in ta kendisidir. Index'lerde çok fazla veri olması performans, ölçeklenebilirlik ve bakım sorunlarına yol açabilir. Bu yüzden index'ler birer Lucene instance'ı olan shard'lara bölünebilir. Index oluşturulurken shard'ların sayısı belirlenir ve veriler shard'lara dağıtılır.

Shard'lar node'lara bölünür. Böylece işlemler hem dağıtık hem de paralel hale gelmiş olur ve performans artar. İçerik hacmini yatay olarak bölmeye ve ölçeklendiremeye olanak tanır.

Primary shard'ların miktarı daha sonradan değıştirilemiyor. Bunun yerine read için daha fazla replica oluşturulabilir. En kötü durumda yeni bir index oluşturulabilir.

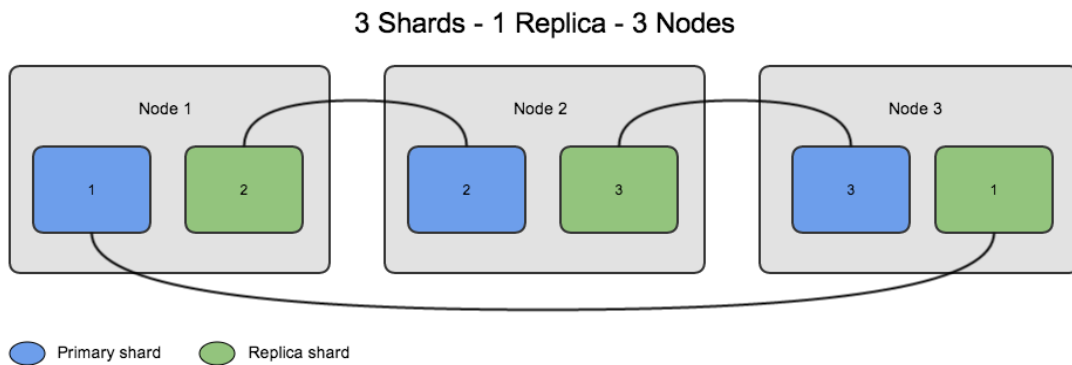
```
PUT /testindex
{
  "settings": {
    "number_of_shards": 3
    , "number_of_replicas": 1
  }
}
```

Bu ayarda 3 tane primary 3 tane replica shard olmak üzere 6 tane shard oluşturuluyor. 5 primary 3 replica için ise toplam 20 shard oluşur.

Replica: Shard'ın devre dışı kalma ihtimaline karşılık index'lerin tutulduğu shard'lar farklı node'lar içerisinde çoklanır.

Write request'leri primary shard'lara yönlendirilir. Oradan replica shard'lara iletilir.

Read request'ler, hem primary hem de replica shard'lara yönlendirilir.



APIs

- Cat API: index, cluster, node, shard gibi bir çok şey ilgili bilgiler görüntülenebilir.
- Cluster API: Cluster ile ilgili bilgiler
- Index API: index bazında create, delete, monitoring, mapping, settings gibi işlerden sorumludur.
- Document API: Dökümanlar üzerinde CRUD operasyonlarını yapmamızı sağlar.
- Search API: Arama yapmak için kullanılır. URI Query ve Query DSL.

Analyzer

ElasticSearch'e yolladığımız dökümanlar hem performans hem de istediğimiz şekilde çalışmasını sağlamak için indexlenmeden önce bir dizi işlemten geçer. Bu işlemlerin bütünü **analysis**'tir. Bu işlemi yapan kısma ise **Analyzer** denir.



1. Karakter Filtreleme: Daha doğru search sonuçları elde edebilmek için belirli karakterleri farklı karakterlere çevirme işlemidir.

you & me -> you and me

2.Text'i Token'lara Ayırıştırma: Text'e karakter filtresi uygulandıktan sonra, ilgili dökümanın kelime setlerine ayrıştırılması gerekir. Lucene inverted index yapısı gereği dökümanı token adı verilen kelimelere ayırıştırarak işlem yapar. Örneğin; standart tokenizer kullanırsak boşluklara göre token'lara ayırıştırır.

"I like ES" -> tokens: "I", "like", "ES"

3.Token Filtreleme: Token'ları alarak onları ihtiyaca göre modifiye edilmesi işlemidir. lowercase token filterda tüm token'lar küçük harfe çevrilir.

4.Token indexleme: Token'lar token filter'lar üzerinden geçirilir ve inverted index'e yerleştirilir.

Built-in Analyzers

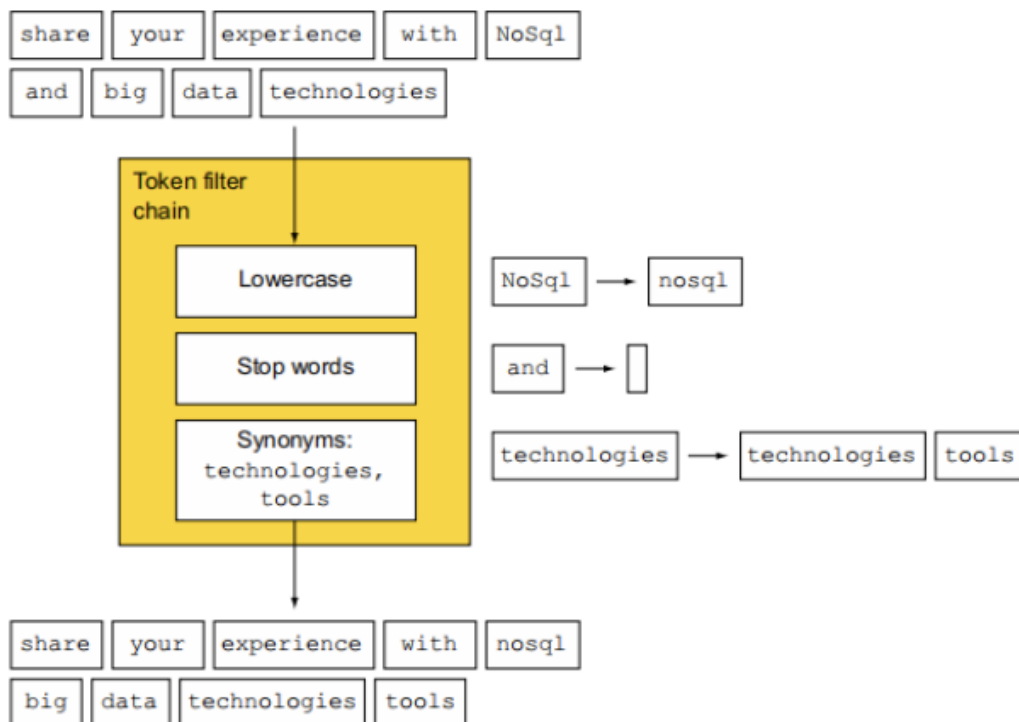
- Standart Analyzer, default geçerli olandır. Standart lowercase token filter'ını ve stok token filter'ını içerir.
- Simple Analyzer,
- Whitespace Analyzer,
- ...

Built-in Tokenizers

- Standart Tokenizer, Gammer based tokenizer'dır. "I like, ES" -> "I", "like", "ES"
- Keyword Tokenizer,
- Letter Tokenizer,
- ...

Built-in Token Filters

- Standart
- Lowercase
- Length
- Stop
- Trim



Fuzzy Searching (i.e. similar words also work):

- Kelimelerin birkaçının farklı olmasının arama yaparken tolere edilmesini sağlar.
- link: <https://hackernoon.com/how-to-use-fuzzy-query-matches-in-elasticsearch-dh1h3167>
- Bunun için arama yaparken fuzziness attribute'ünü ekleriz.

```
"match": {  
  "title.default": {  
    "query": "eRçort",  
    "fuzziness": "AUTO",  
    "fuzzy_transpositions": "true"  
  }  
}
```

Auto-Complete suggestion

- Auto-complete prefix ile çalışır. Dökümantasyona göre eğer kibir suggestion'a ihtiyacımız var ise o zaman matching yaklaşımı değil de suggester kullanmalıyız. Suggester mekanizması çok hızlıdır. Şu şekilde çalışır:

