Elasticsearch Intro

By Dima Goldin

History

Created by Shay Banon @ 2010 @ Israel

#7 in Database rankings, #1 in Search Engines

Written in Java

Built on top of Apache Lucene (Full Text Search Library in Java)

Current version is 8.10.0

Core DB is open-source but uses Elastic license (Similar to MongoDB license)

Last fully open-source version 7.17

Alternatives to Elasticsearch

Splunk - not open source - costs A LOT

Solr - Not as popular, not as easy but was there before Elasticsearch

OpenSearch - Amazons fork of Elasticsearch 7.10 - same same but different

No other alternatives are even worth mentioning.

Elasticsearch Moto

I’ll give you 70% of what you need at close to **ZERO** work

For the other 30%, you will pay me with interest! :D

What is Elasticsearch

Distributed Search Engine

Document Database optimized for search

JSON in, JSON out

Almost everything in elasticsearch is asynchronous and concurrent

API is REST based, old API was Transport, today Transport is only for internal communication (inside the cluster components)

Elastic has its own query language and its also in JSON

Elastic mostly supports SQL now as well (not all features)

Fun Facts

1. For most databases, if you insert 1gb of data, it will take 1gb or less of storage. For elasticsearch, 1gb of inserted data will take at minimum 1gb and in some cases, even 2 or more GB on disk (100%+ increase)
2. Most known e-commerce websites (Amazon/Ebay/Target/etc.) use Elasticsearch for user searches, but most companies use Elasticsearch for log analytics
3. There are elasticsearch clusters with 200+ Nodes, 4+ PB of data, 600+ billion documents)

Key Concepts of Elasticsearch

Theory

- Documents: Elasticsearch stores data in JSON documents.
- Index: An index is a collection of documents that have similar characteristics.
- Node: A single instance of Elasticsearch running on a machine.
- Cluster: A collection of nodes that work together to store and process data.

Technical General Info

- Shards: an Elasticsearch index can be split into shards, shards hold part of the index data, can and should be on different Nodes
- Replicas: Copies of shards, should be on different Nodes from the original shard
- Indexing: A document inserted into elasticsearch, goes through indexing, making the fields ready for complex search
- Mapping: Schema of the documents in an idnex

Key Concepts of Elasticsearch

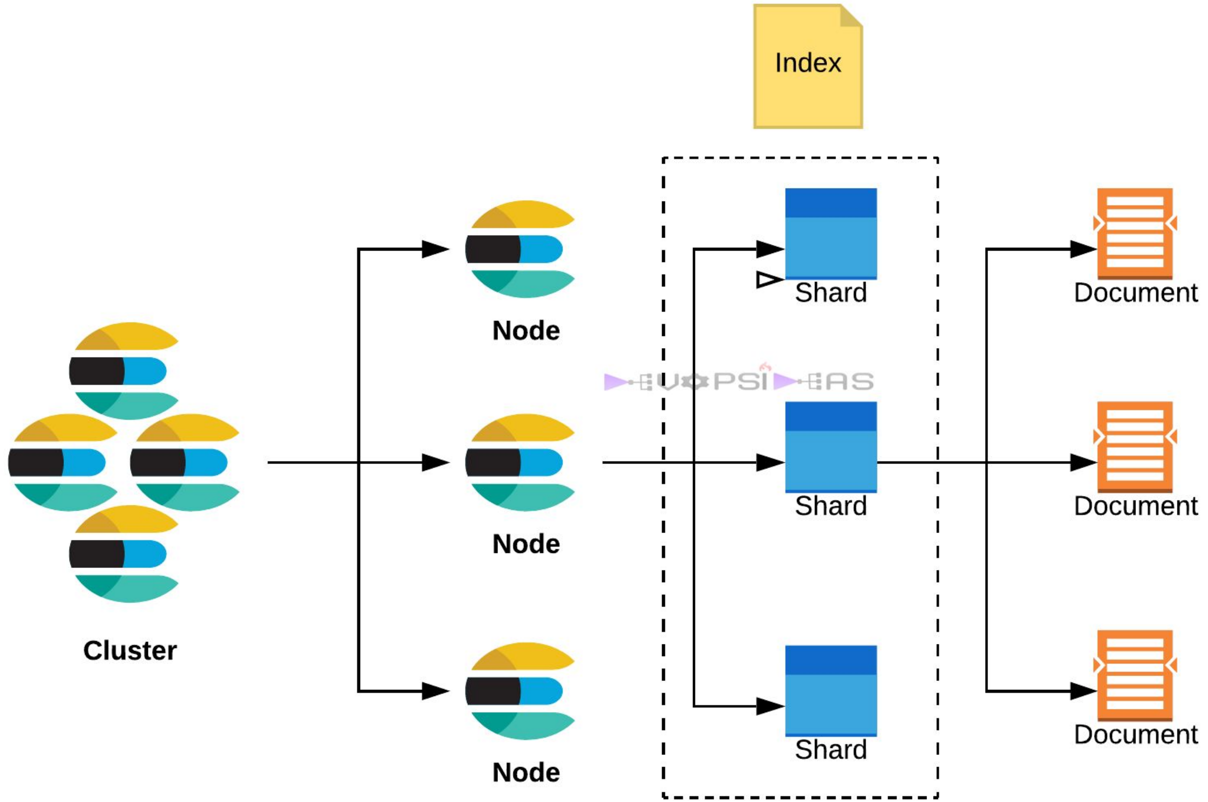
Theory

- Elasticsearch is fully distributed: there is no master or main node, there is no main shard
- Eventually consistent: Documents inserted into elasticsearch are persistent on node but are searchable only after indexing phase.
- CRUD vs Search/Filter Operations: CRUD uses the ID of the document, these operations are fully consistent. Search/Filter operations are not consistent and might show outdated info (for documents not yet indexed)

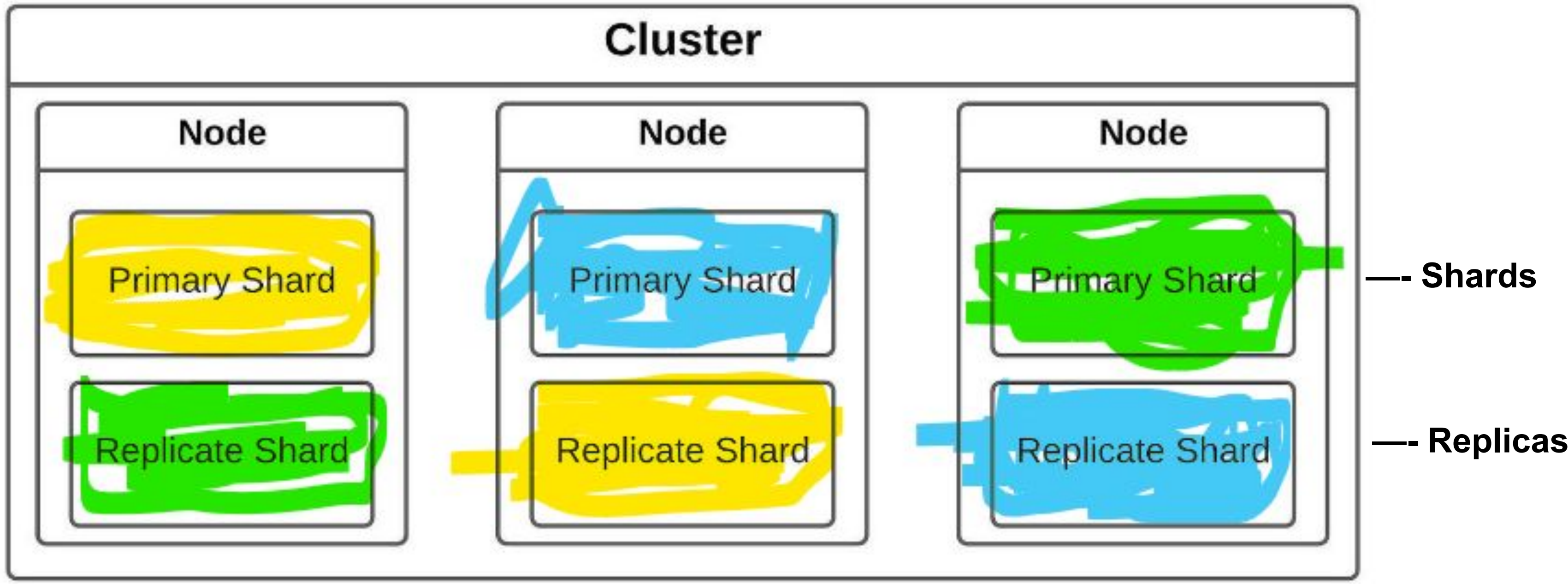
Technical General Info

- Field types: Keyword/Text/Date/Float/etc. They tell elasticsearch, how to index the field.
- Elasticsearch is schemaless - ***No*** elasticsearch can try to infer the schema from the Json on first insert, but schema wont change afterwards, even if it was wrong. You should, set your own schema (mapping) in production and disable “Dynamic Schema”
- Inverted index: hashmap where key is a word and the values are the ids of the documents that contain it

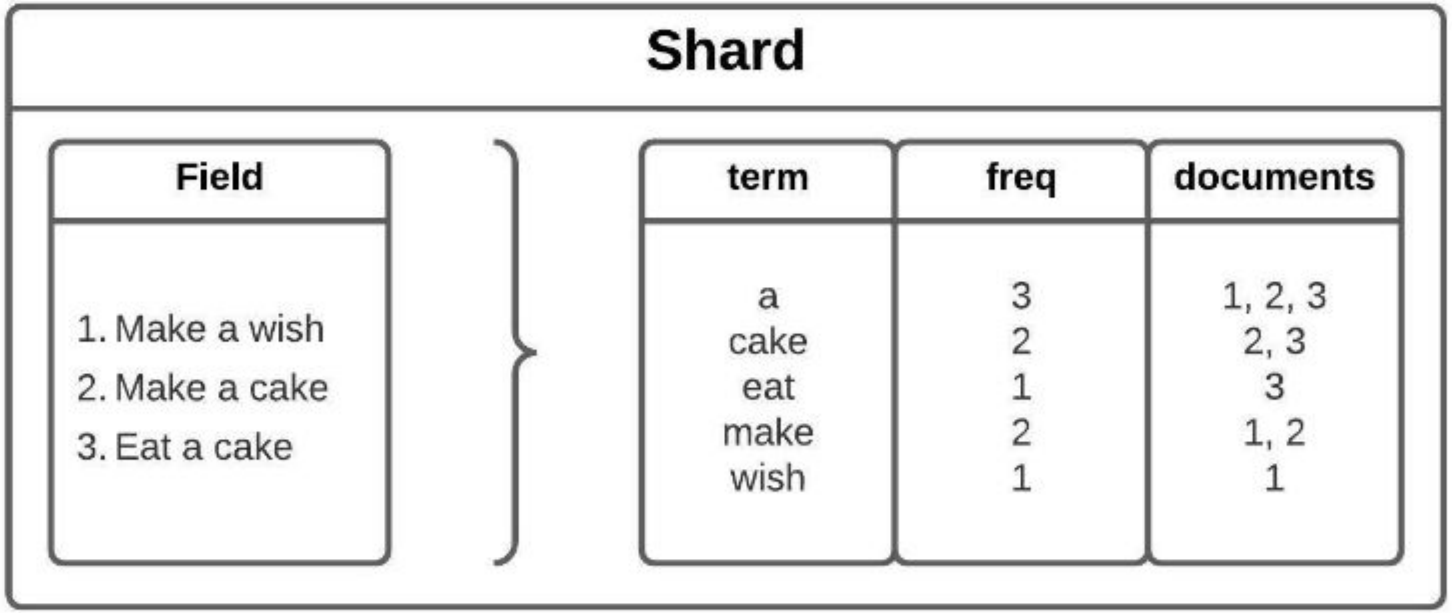
Elasticsearch Component Relation



1 Index - 3 Shards - 1 Replica



Inverted Index



1. Make a wish
2. Make a cake
3. Eat a cake

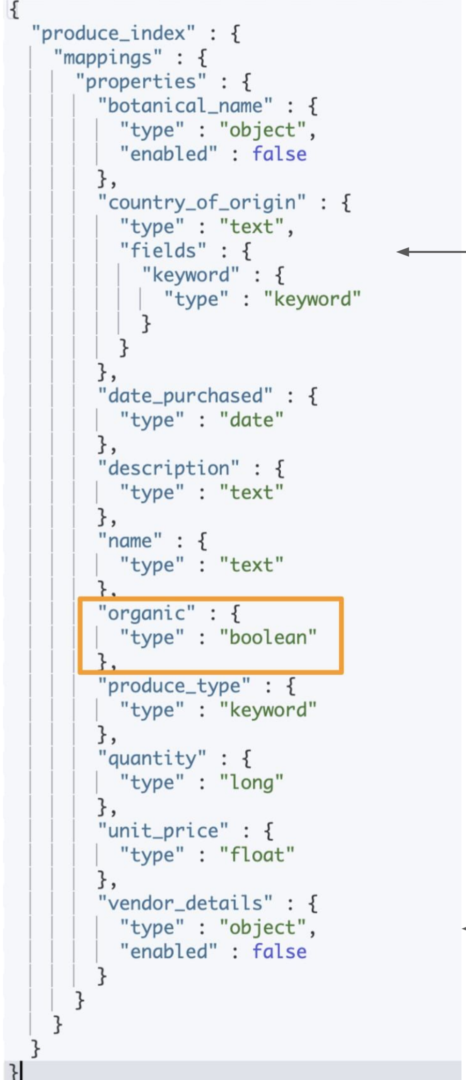
term	freq	documents
a	3	1, 2, 3
cake	2	2, 3
eat	1	3
make	2	1, 2
wish	1	1

Mapping

Defines how the data in the Index is "Indexed" which defines how the data is Searched

What's missing???

_id and @timestamp



```
{
  "produce_index" : {
    "mappings" : {
      "properties" : {
        "botanical_name" : {
          "type" : "object",
          "enabled" : false
        },
        "country_of_origin" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword"
            }
          }
        },
        "date_purchased" : {
          "type" : "date"
        },
        "description" : {
          "type" : "text"
        },
        "name" : {
          "type" : "text"
        },
        "organic" : {
          "type" : "boolean"
        },
        "produce_type" : {
          "type" : "keyword"
        },
        "quantity" : {
          "type" : "long"
        },
        "unit_price" : {
          "type" : "float"
        },
        "vendor_details" : {
          "type" : "object",
          "enabled" : false
        }
      }
    }
  }
}
```

Multi-field
country_of_origin
country_of_origin.keyword

complex sub-object
Enabled

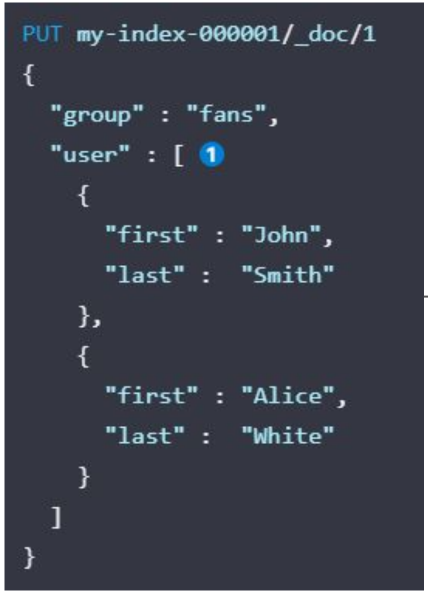
Important facts

1. There are no multi-index JOINs in elasticsearch
2. There is a poor imitation of joins inside 1 index between different types
3. There can only be one type of document per index
4. There is multi-index search
5. There is multi-cluster search
6. 10k magic number in elasticsearch
    a. Cant receive above 10k documents in one result
    b. Cant have arries with more than 10k elements
    c. Cant have more than 10k nested documents (sub documents)

Nested Documents

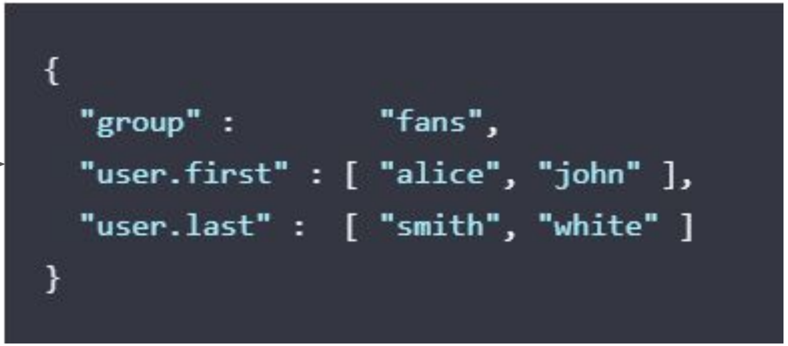
Array and field flattening

Elastic flattens all sub fields



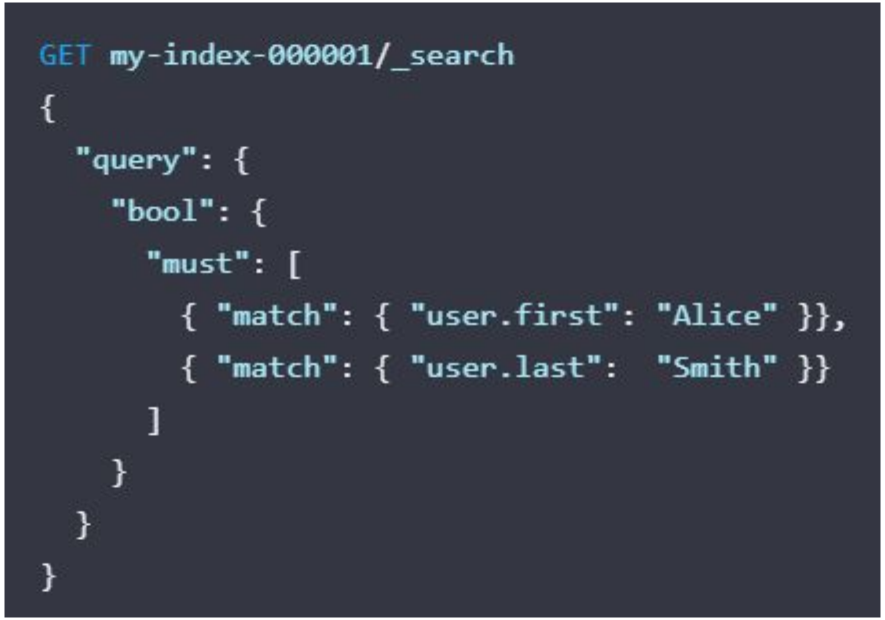
```
PUT my-index-000001/_doc/1
{
  "group" : "fans",
  "user" : [ 1
    {
      "first" : "John",
      "last" :  "Smith"
    },
    {
      "first" : "Alice",
      "last" :  "White"
    }
  ]
}
```

Internally



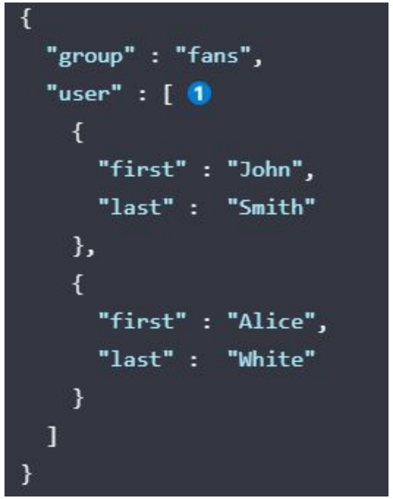
```
{
  "group" :        "fans",
  "user.first" : [ "alice", "john" ],
  "user.last" :  [ "smith", "white" ]
}
```

Search for groups with users having first name Alice and Last name Smith



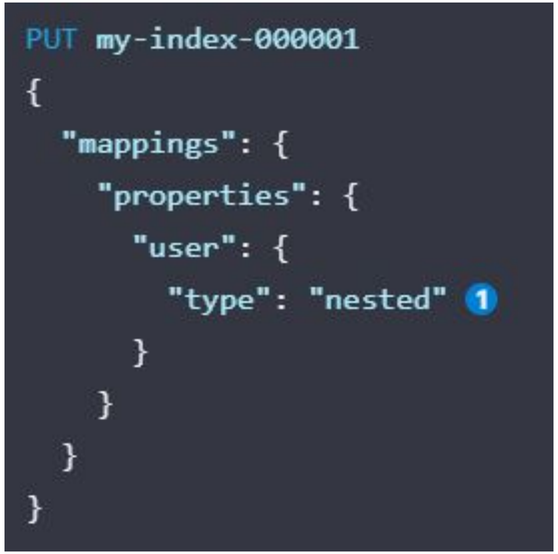
```
GET my-index-000001/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "user.first": "Alice" }},
        { "match": { "user.last":  "Smith" }}
      ]
    }
  }
}
```

matches!



```
{
  "group" : "fans",
  "user" : [ 1
    {
      "first" : "John",
      "last" :  "Smith"
    },
    {
      "first" : "Alice",
      "last" :  "White"
    }
  ]
}
```

Nested type



```
PUT my-index-000001
{
  "mappings": {
    "properties": {
      "user": {
        "type": "nested" 1
      }
    }
  }
}
```

Now users will be nested documents, each with their own ID, and would not be flattened, context is preserved!

(10k nested documents per document limit)

Demo time!

dimagoldin/elasticsearch-tutorial (github.com)

Thanks!