eBook

Elasticsearch

Lecture “New Database Technologies”

Prof. Dr. Carmen Winter

Rykarda Heim

TINF20A

from: Sven Salb
Viola Steiner
Violetta Saibel
Severin Schrettenbrunner

Table of contents

Table of figures

1. Type of Database

Elasticsearch is a full-text search engine based on Apache Lucene, which together with Kibana and Logstash represents the Elastic stack. To store data, the platform is combined with other document-based databases. These are then responsible for data storage, while Elasticsearch takes over the indexing of the documents.

Elasticsearch itself is optimised for read access and enables scaling. Documents are stored here as JSON documents in NoSQL format. Elasticsearch also supports communication with a client, which is implemented using a RESTful architecture. Thus, the full-text search engine is primarily used to search and analyse large amounts of data, which are represented by Big Data, for example (Hopf, 2015).

2. History

2000

Shay Banon develops Elasticsearch with Apache Lucene. For this, he chooses an open-source approach and creates an IRC channel #elasticsearch. Furthermore, he founds a search company together with Steven Schuurman, Uri Boness and Simon Willnauer (Elastic, 2023j).

2012

Logstash, an open-source traffic tool for sending logs, and Kibana, an open source user interface, are being developed. These merge with Elasticsearch to form the Elastic Stack. In addition, the two commercial plugins Marvel and Shield are released. They are responsible for monitoring and security (Elastic, 2023j).

2015

The company is renamed to Elastic and merges with Found, a company that provides hosting and management of Elasticsearch on AWS. Thus, Elasticsearch and Kibana now become available as services through Elastic Cloud on AWS. Beats is also being developed to send network data, logs, metrics and audit data from edge machines to Logstash and Elasticsearch (Elastic, 2023j).

2016

All commercial plug-ins are now bundled in a single extension X-Pack. This contains functions for security, monitoring, alerting and will later be extended by machine learning (Elastic, 2023j).

2017

Filebeat is introduced in the Ealstic stack and provides secure configurations for sending, parsing, storing, analysing and visualising common log formats. Additionally, Elastic Cloud Enterprise is introduced to provide customers with seamless management. Finally, the company joins forces with Opbeat and Swiftype, representing an application performance monitoring company and a search company, respectively (Elastic, 2023j).

2018

Elastic becomes a publicly traded company and the code to all X-Pack features is released to reduce development time through community engagement (Elastic, 2023j).

3. Design

Elasticsearch consists of a logical and a physical part. The logical part is composed of indexes and documents. The physical part of Elasticsearch includes shards, replicashards, clusters and nodes (Hopf, 2015). In this section, all components are explained in more detail.

Index: Documents that have a similar structure and can be logically combined into one unit are grouped in Elasticsearch. Such a grouping is called an index. No data is stored in an index. An index is a virtual parameter that stores the storage locations of the documents. (Hopf, 2015)

Details of mappings are stored in Elasticsearch in main memory. Large mappings can therefore result in a large cluster that takes a long time to update. Elasticsearch performs all update operations in individual threads. Therefore, it is recommended to store data with similar structures in the same index. (Gromley & Zachary, 2015)

Time-based indexes are available for documents that are to be stored for a specific period of time only. Depending on the retention period, data can be stored in daily, weekly, or monthly indexes. This simplifies the administration effort and after the time has expired, the records can be deleted from Elasticsearch again automatically. (Gromley & Zachary, 2015)

Documents: Data is stored in Elasticsearch as documents or more precisely as JSON objects. A document consists of fields. These fields can contain one or more values, and each field has an associated data type. Elasticsearch uses both elementary data types like: Boolean, Byte, Short, Integer, Long, Double, Float etc. as well as complex data types like: Nested, Object an. (Miller & Srivastava, 2019) (Mosaab Asli)

Shard: The data of an index is stored on so-called shards. Shards is the instance in Elasticsearch on which the documents are stored. The data of an index can be stored either on one shard or, if the data volumes exceed the hardware limits of a node/shard, on multiple shards. When an index is divided among several shards, this is also referred to as sharding. By default, each index initially consists of one shard. The number of shards can be defined when setting up the cluster. (Elastic, 2023i)

The number of shards can only be defined at the beginning of the creation of the project, because if the shards are created later, Elasticsearch has no overview on which shards the data was stored. Elasticsearch is able to process multiple shards at the same time. In general, it can be stated that the processing speed depends on the size, the number of shards and also the performance from the network and the storage space. The manufacturer of Elasticsearch refers here to a maximum size of 50 GB per shard. (Dahlqvist, 2022)

To ensure that many documents can be stored in a single node, it is essential to keep the overhead size small and manage the heap usage. The heap space of the node determines how much data and shards it can handle. (Dahlqvist, 2022)

Replicashard: Any number of replicashards can be created from a shard. Replicashards are copies of the stored data. They ensure that the data can still be accessed if individual nodes fail. Replicashards can be created subsequently. (Hopf, 2015) (Mosaab Asli)

Cluster: A cluster in Elasticsearch contains nodes, shards, and replica shards. A cluster can contain one node or multiple nodes. Each cluster has a unique name. (Miller & Srivastava, 2019) (Mosaab Asli)

Node: A node can be considered as a small server unit on which the data is stored in Elasticsearch.

Each node has an ID and a name and belongs to a cluster. If a cluster consists of several nodes, they can be classified into one of the following categories, depending on their purpose:

- Master node: Each cluster has a master node that is selected automatically. It is responsible for controlling the cluster.
- Data node: This node contains data (documents) and is responsible for performing data-related operations, such as create, read, update, and delete. (Hopf, 2015) (Mosaab Asli)
- Machine Learning Node: This node is required if the cluster is to have machine learning functions. (Elastic, 2023a)

Inverted index: Since Elasticsearch runs on Apache-Lucene, it also uses an inverted index. The name "inverted" is based on the fact that instead of pointing documents to words, as is usually the case, a list of words points to documents that contain them. In Elasticsearch, this list contains information about words and values of all documents and an assignment in which document they occur. The inverted index is what makes a fast full-text search possible in the first place. (Gromley & Zachary, 2015; Hopf, 2015)

Term	Index frequency	Document [Frequency in document]
for	1	1[1]
hello	2	4[1]

The term "for" occurs in an index in this example. In total, this term is used once in the document with the number 1. The term "hello" is used in two documents. In document 4, the term "hello" is used once.

4. CAP Theorem (Brewer E. 2000)

The CAP theorem, also known as Brewer's theorem, is a fundamental concept in distributed systems. It states that it is impossible to design a distributed system that simultaneously satisfies the following three guarantees: Consistency, Availability and Partition Tolerance. This means that in a distributed system that faces errors and delays in the network, designing systems that maximise all three aspects simultaneously requires a trade-off.

The CAP theorem doesn’t really apply to Elasticsearch because Elasticsearch isn't normally used as a linearizable register. That said, Elasticsearch would most likely prefer Partition Tolerance and Consistency over Availibility, in the sense that acknowledged writes should never be lost but that some writes may go unacknowledged or may fail.

However, there are some settings and configurations in Elasticsearch that developers can adjust to control the behavior of the system in terms of consistency, availability and partition tolerance. For example, developers can set the number of replicas to have for each shard to improve the availability and consistency of the system. In addition, developers can configure different levels of consistency for system writes and reads.

Overall, developers can customize the settings and configurations of Elasticsearch to control the behavior of the system in terms of consistency, availability and partition tolerance. With the standard settings Elasticsearch would most likely prefer Partiton Tolerance and Consistency over Availibility.

5. API

The great advantage of Elasticsearch is the wide range of REST APIs. These APIs allow the user to integrate, manage and query indexed data in a myriad of ways. Examples of using these APIs in Elasticsearch are numerous and several different use cases (Elastic, 2023b).For a better understanding, the four most used Elasticsearch APIs are discussed.

5.1 Document API

This category of API is used for processing documents in Elasticsearch. The document API is used to create documents in an index, update them, move them to another index or delete them. In addition, the document API also provides the ability to create, update, or delete multiple documents in a single query (Elastic, 2023c).This can be useful when users are processing large amounts of data, making it important to reduce the number of queries to be sent to Elasticsearch.

5.2 Search API

The Search API, as the name suggests, can be used to query indexed data for specific information. For this purpose, the API uses Elasticsearch's own query language DSL (Elastic, 2023d).Filters or sorting criteria can also be used to obtain more accurate search results. Search APIs can be applied globally, across all available indexes and types, or within an index.

5.3 Indices API

Another API in Elasticsearch is the Indices API. This type of API allows the user to manage indexes, mappings and templates. For example, the API can be used to create or delete a new index, check if a specific index exists or set a new mapping for an index (Elastic, 2023e).The search API is also very flexible as it allows users to index data in different formats such as JSON, CSV or TSV (Elastic, 2023g).For this reason, it can be easily deployed in existing infrastructures.

5.4 SQL API

For those who are familiar with SQL queries, Elasticsearch supports a SQL API. This SQL API allows you to perform queries and returns using SQL syntax (Elastic, 2023f). This makes it

possible to run queries with Elasticsearch without having to learn a new query language. This API is especially useful for users who are already experienced with SQL and want to quickly access data and run queries.

6. Improvements regarding relational databases

Elasticsearch provides several improvements over traditional relational databases in terms of data design and query performance. This eBook addresses the most relevant improvements.

Elasticsearch's biggest improvement is the flexible and schema free data model. In traditional relational databases, the data must be structured into tables and columns with predefined fixed data types (Kuc & Rogozinski, 2015).Any changes to this schema can be time-consuming and complex. The improvement of Elasticsearch is that data can be stored as JSON documents, which can be easily indexed and searched (AIMDek Technologies, 2018).This schema-free approach makes it easier to store and search unstructured or semi-structured data.

Elasticsearch is optimized for full-text search, relational databases usually are not. For full-text search, Elasticsearch uses an inverted index that allows users to quickly search texts and get the search results. Therefore, Elasticsearch is ideal for large amounts of data and uses its own query DSL (Domain Specific Language) to perform searches (Patrick Oscity, 2015). DSL allows for complex queries to be executed quickly and efficiently.

Another improvement of Elasticsearch compared to traditional relational databases is its distributed and scalable architecture. Elasticsearch is designed to run on multiple nodes, allowing it to handle large volumes of data and queries with high availability and fault tolerance (Kuc & Rogozinski, 2015). This makes the architecture very easy to scale in enterprises, meaning it is easy to add additional servers to handle the load (AIMDek Technologies, 2018).

The possibility of using APIs in Elasticsearch was explained in the previous chapter. However, these APIs are also an improvement. For example, the dedicated API offers the advantage over relational databases that default values for search queries are generated and displayed for the user. As a result, searches can be greatly simplified, which improves the work process.

Elasticsearch is also very flexible and can handle different data formats, including structured and unstructured data. Relational databases cannot use different formats. It can also be used as a NoSQL database as it does not require a fixed schema definition (Kuc & Rogozinski, 2015). This flexibility allows Elasticsearch to be integrated into existing systems without having to change the data formats.

Elasticsearch should be preferred over relational databases when you need to perform full-text search, data analysis and aggregation. Furthermore, using Elasticsearch is better when you have a large volume of data that needs to be indexed and queried quickly (AIMDek Technologies, 2018).

7. Advantages and Disadvantages

As with any technology, Elasticsearch has its strengths and weaknesses, and it is important to consider them before deciding whether to use it in your project.

First the advantages are explained. Elasticsearch is compatible with any platform as it is developed in Java, for this reason it can be integrated very well into existing infrastructures (JavaTPoint, 2021).In addition, Elasticsearch is also an open-source solution. So, there are no license fees to pay for the download and use. Moreover, it is a distributed, document-oriented system that scales easily in large organizations (Lang, 2022). The developer can easily integrate it into any large organization by scaling it (AIMDek Technologies, 2018). Additionally, Elasticsearch supports all document types except those that do not support text representation. This enables high flexibility.

However, the disadvantages and difficulties should also be noted. Elasticsearch is a flexible and powerful search engine for data storage, but it is somewhat difficult to learn. Especially in terms of using enterprise search, it is not as easy as out-of-the-box search (Lang, 2022). Furthermore, sometimes the problem of "split-brain" situations occurs in Elasticsearch (JavaTPoint, 2021). This problem is an undesirable state of a computer cluster in which all interconnections between the cluster parts are broken at the same time. It should also be mentioned that Elasticsearch does not provide multilingual support for query and response data processing (JavaTPoint, 2021).

Before considering Elasticsearch, the disadvantages and problems should be analyzed in detail so that problems do not arise afterwards.

8. Installation (process, requirements, challenges)

Elasticsearch offers several ways to install it. On the one hand, Elasticsearch offers the option to host Elasticsearch on the Elastic Cloud and on the other hand, Elasticsearch offers several options to manage Elasticsearch yourself. For the self-managed options, there are three options. The first is to install Elasticsearch local on the Linux, Mac or Windows machine. The second option is to set up Elasticsearch using Kubernetes, and the third, which we will discuss in more detail in this section, is to install Elasticsearch in a Docker container. The first step to install Elasticsearch on a Docker container is to install the appropriate Docker application. After that the next step is to pull the Elasticsearch Docker image. To pull the Elasticsearch Docker image, a docker pull command must be executed against the Elastic Docker register. This is shown in

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:8.6.2
```

Figure 1. (Elastic, 2023h)


```
docker pull docker.elastic.co/elasticsearch/elasticsearch:8.6.2
```


Figure 1: Pull the Elasticsearch Docker image (Elastic, 2023h)

With the Docker image it is possible to start a single-node or multi-node cluster. We will investigate starting a single-node cluster with Docker. To create a single-node Elasticsearch cluster, the command from Figure 2 must be executed. (Elastic, 2023h)

```
docker network create elastic
```

Figure 2: Create single-node cluster (Elastic, 2023h)

The Docker container is now created, but to use it productively, the Elasticsearch container must be started in Docker. To do this, use the command from Figure 3 must be executed. (Elastic, 2023h)

```
docker run --name es01 --net elastic -p 9200:9200 -it
docker.elastic.co/elasticsearch/elasticsearch:8.6.2
```

Figure 3: Start Elasticsearch cluster (Elastic, 2023h)

When the cluster is started for the first time, a password is generated for the elastic user and an enrolment token is generated to set up Kibana. These two keys are output in the terminal and should be saved separately by the user. The password, the user, which is elastic by default, and an http certificate are needed to send http queries to Elasticsearch. We already have the password and the user, but we still need the http certificate. This is generated with the command from Figure 4. (Elastic, 2023h)

```
docker cp es01:/usr/share/elasticsearch/config/certs/http_ca.crt .
```

Figure 4: Http certificate generation (Elastic, 2023h)

These were the steps to install Elasticsearch with Docker. In our experience, the installation with Docker works without any problems. We also tested the installation on the local machine, where we had several problems. For example, after several weeks we could no longer access the Kibana interface with the local installation. This problem did not occur with the Docker containers. We therefore recommend the Docker installation.

9. Examples

In this section, we will go into some example requests to show how Elasticsearch works. The examples used are fictitious. First, we need to create an index to store our data in Elasticsearch. To create an index, we need to send a put request to Elasticsearch, one such request is shown in Figure 5: Index creation. For our mappings we enable numeric detection to store numbers like 1.7 as numbers rather than strings. Furthermore, we declare our used dynamic date format to detect strings in that format as dates rather than normal strings. to use date and time-based metrics.

```
PUT persons
{
  "mappings": {
    "numeric_detection": true,
    "dynamic_date_formats": ["dd.MM.yyyy hh:mm:ss"]
  }
}
```

Figure 5: Index creation

Documents can now be stored under this index. For this purpose, post requests are used as shown in Figure 6: Document creation. The data is stored in json format.

```
POST persons/_doc/1
{
    "name" : "John",
    "lastname" : "Doe",
    "job_description" : "Systems administrator and Linux specialist",
    "date_of_birth" : "30.03.2023 12:12:12",
    "height": "1.87"
}
```

Figure 6: Document creation

A get request is used to view the documents. The suffix _search?size declares a search query that is answered with a maximum of 1000 documents. With the help of query, the search criteria can be defined. In this first example, which is shown in Figure 7: Get request to get up to 1000 documents, match_all is used so that all documents under the index persons are displayed.

```
GET persons/_search?size=1000
{
  "query": {
    "match_all": {}
  }
}
```

Figure 7: Get request to get up to 1000 documents

The corresponding response of the Elasticsearch server is shown in the Figure 8: Get response to get up to 1000 documents. Two additional documents were added under the index to show analytical search functions in the other examples.

```
"hits": [
  {
    "_index": "persons",
    "_id": "2",
    "_score": 1,
    "_source": {
      "name": "Sven",
      "lastname": "Salb",
      "job_description": "Administrator",
      "date_of_birth": "01.03.2023 12:12:12",
      "height": "1.90"
    }
  },
  {
    "_index": "persons",
    "_id": "1",
    "_score": 1,
    "_source": {
      "name": "John",
      "lastname": "Doe",
      "job_description": "Systems dministrator and Linux specialist",
      "date_of_birth": "30.03.2023 12:12:12",
      "height": "1.87"
    }
  },
  {
    "_index": "persons",
    "_id": "3",
    "_score": 1,
    "_source": {
      "name": "Severin",
      "lastname": "Schrettenbrunner",
      "job_description": "devops specialist",
      "date_of_birth": "01.03.2023 12:12:12",
      "height": "1.99"
    }
  }
]
```


Figure 8: Get response to get up to 1000 documents

The Figure 9: Full-text search query get request shows a full-text search query. In the full-text search query, Elasticsearch offers several options for searching for exact terms or values within an index. In this search query, all documents with "specialist" under "job_description" in the Json document are output.

```
GET persons/_search
{
  "query": {
    "match": {
      "job_description": "specialist"
    }
  }
}
```

Figure 9: Full-text search query get request

The Figure 10: Full-text search query get response shows the answer to the full-text search query. The score indicates how closely the search text matches the requested term, in this case the "job_description". In addition, we only get the documents in which the search text occurred.


```
    "max_score": 0.52354836,
    "hits": [
      {
        "_index": "persons",
        "_id": "3",
        "_score": 0.52354836,
        "_source": {
          "name": "Severin",
          "lastname": "Schrettenbrunner",
          "job_description": "devops specialist",
          "date_of_birth": "01.03.2023 12:12:12",
          "height": "1.99"
        }
      },
      {
        "_index": "persons",
        "_id": "1",
        "_score": 0.34611148,
        "_source": {
          "name": "John",
          "lastname": "Doe",
          "job_description": "Systems dministrator and Linux specialist",
          "date_of_birth": "30.03.2023 12:12:12",
          "height": "1.87"
        }
      }
    ]
```


Figure 10: Full-text search query get response

In the next example a term-level query is shown. Term-level queries are used to find precise values in documents. With the term-level query, the queries must match the exact terms stored in a field. In the example in the Figure 11: Term-level query get request, every person born within the last day is searched for.

```
GET persons/_search
{
  "size": 100,
  "query": {
    "range": {
      "date_of_birth": {
        "gte":"now-1d"
      }
    }
  }
}
```

Figure 11: Term-level query get request

In the Figure 12: Term-level query get response the corresponding answer is shown. At the time the example was taken, it was 30.03, which is why we only get this one document as a response.

```
"hits": [
  {
    "_index": "persons",
    "_id": "1",
    "_score": 1,
    "_source": {
      "name": "John",
      "lastname": "Doe",
      "job_description": "Systems dministrator and Linux specialist",
      "date_of_birth": "30.03.2023 12:12:12",
      "height": "1.87"
    }
  }
]
```


Figure 12: Term-level query get response

These examples show how to use Elasticsearch and how Elasticsearch responds to different queries. The biggest advantages here are the powerful full-text search, which allows Elasticsearch to search millions of documents quickly and efficiently in real time, and high flexibility, which is demonstrated by the fact that Elasticsearch can store different types of data, including structured and unstructured data as well as geodata. It also offers a variety of configuration options to adapt the behavior of the system to specific requirements.

10. Conclusion

Elasticsearch is a search and analysis engine and is therefore not comparable with conventional SQL databases or NoSQL databases. It is based on Apache Lucene and was developed for real-time search and data analysis. With its distributed and RESTful design, Elasticsearch is horizontally scalable and supports RESTful APIs for data access. With full-text search, Elasticsearch offers powerful search capabilities, including faceted search, filtering, and ranking. From these key points, the main use cases are as follows: Log and event data analysis, full-text search, and monitoring and alerting systems.

The CAP theorem is not fully applicable to Elasticsearch. If one would still have to classify Elasticsearch in the CAP theorem, a compromise between the three guarantees of the theorem consistency, availability and partition tolerance has to be found. However, Elasticsearch was designed to provide high availability and consistency, while the system's partition tolerance may not always be guaranteed. By adjusting Elasticsearch's settings and configurations, developers can control the behavior of the system to meet their consistency, availability and partition tolerance requirements.

Overall, Elasticsearch is a powerful tool for searching and analyzing large amounts of data. Elasticsearch's design and configuration options allow developers to tailor the system's performance and behavior to their specific needs. Thus, Elasticsearch is a valuable tool for companies and organizations that need to access and analyze large amounts of data quickly and efficiently.

References

AIMDek Technologies (2018). What is Elasticsearch? Retrieved on 12.03.2023 from https://medium.com/@AIMDekTech/what-is-elasticsearch-why-elasticsearch-advantages- of-elasticsearch-47b81b549f4d

Dahlqvist, C. (2022). Wie viele Shards brauche ich in meinem Elasticsearch-Cluster? Retrieved on 03.04.2023 from https://www.elastic.co/de/blog/how-many-shards-should-i-have-in-my-elasticsearch-cluster

Elastic (2023a). Node. Retrieved on 12.03.2023 from https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

Elastic (2023b). REST API compatibility. Retrieved on 18.03.2023 from https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-api-compatibility.html

Elastic (2023c). Document API. Retrieved on 10.03.2023from https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html

Elastic (2023d). Search API. Retrieved on 18.03.2023 from https://www.elastic.co/guide/en/elasticsearch/reference/current/search.html

Elastic (2023e). Index API. Retrieved on 12.03.2023 from https://www.elastic.co/guide/en/elasticsearch/reference/current/indices.html

Elastic (2023f). SQL APIs. Retrieved on 18.03.2023 from https://www.elastic.co/guide/en/elasticsearch/reference/current/sql-apis.html

Elastic (2023g). What is an Elasticsearch Index? Retrieved on 12.03.2023 from https://www.elastic.co/de/blog/what-is-an-elasticsearch-index

Elastic (2023h). Elasticsearch Installation guide. Retrieved on 14.03.2023 from https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html

Elastic (2023i). Set up a cluster for high availability. Retrieved on 03.04.2023 from https://www.elastic.co/guide/en/elasticsearch/reference/current/high-availability.html

Elastic (2023j). Our Story. Retrieved on 08.04.2023 from https://www.elastic.co/de/about/history-of-elasticsearch

Gromley, C., & Zachary, T. (2015). *Elasticsearch The Definitive Guide*. Sebastopol: O'Reilly Media. ISBN: 978-1449358549

Hopf, F. (2015). *Elasticsearch: Ein praktischer Einstieg: Ein praktischer Einstieg*. Heidelberg: dpunkt. Retrieved on 12.03.2023 from http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1298363

JavaTPoint (2021). Advantages and Disadvantages of ElasticSearch. Retrieved on 03.04.2023 from https://www.javatpoint.com/advantages-and-disadvantages-of-elasticsearch

Kuc, R., & Rogozinski, M. (2015). *Mastering ElasticSearch* (2nd ed.). Packt Publishing Limited. ISBN: 1783553790

Lang, N. (2022). What is ElasticSearch? Retrieved on 12.03.2023 from https://databasecamp.de/en/ml/elasticsearch-algorithm

Miller, D., & Srivastava, A. (2019). *Elasticsearch 7 Quick Start Guide*. Birmingham: Packt Publishing Limited. ISBN: 9781789803327

Mosaab Asli. *Volltextsuche mit Elasticsearch im Geodaten-Umfeld*, München. Retrieved on 18.03.2023 from https://www.adbv-nuernberg.de/file/pdf/13202/Bachelorarbeit_Asli_Mosaab.pdf

Patrick Oscity (2015). Wer sucht, der findet - Volltextsuche mit PostgreSQL und Elasticsearch. Retrieved on 03.04.2023 from https://www.zweitag.de/blog/wer-sucht-der-findet-volltextsuche-mit-postgresql-und-elasticsearch/