elasticsearch

Vítězslav Jíra

Overview

1. Intro & Mental Model
2. Core Concepts & Architecture
3. Create an Index
4. Inserting Documents
5. Field Data Types
6. Delete Documents
7. Get Document
8. Count Documents

Overview

9. The Exists API
10. Update Document
11. Bulk API
12. The Search API – Part 1
13. The Search API – Part 2 (Query DSL)
14. The Search API – Part 3
15. The Search API – Part 4 (Query Types Overview)
16. Dense Vectors

Overview

17. Embedding Documents
18. k-Nearest Neighbor (kNN) Search
19. Deep Pagination
20. Ingest Pipelines
21. Ingest Processors
22. Filters in Depth
23. SQL Search API
24. Time Series Data Stream

Overview

25. Analyzers
26. Synonyms
27. Common Options
28. Change Heap Size
29. Collapse Search Results
30. Pre-filtering + kNN Search
31. Hybrid Search
32. Examples

1 Intro & Mental Model

What Elasticsearch Is (and Is Not)

- Distributed search engine
- Built on Apache Lucene
- Not a primary database

Why Elasticsearch Exists

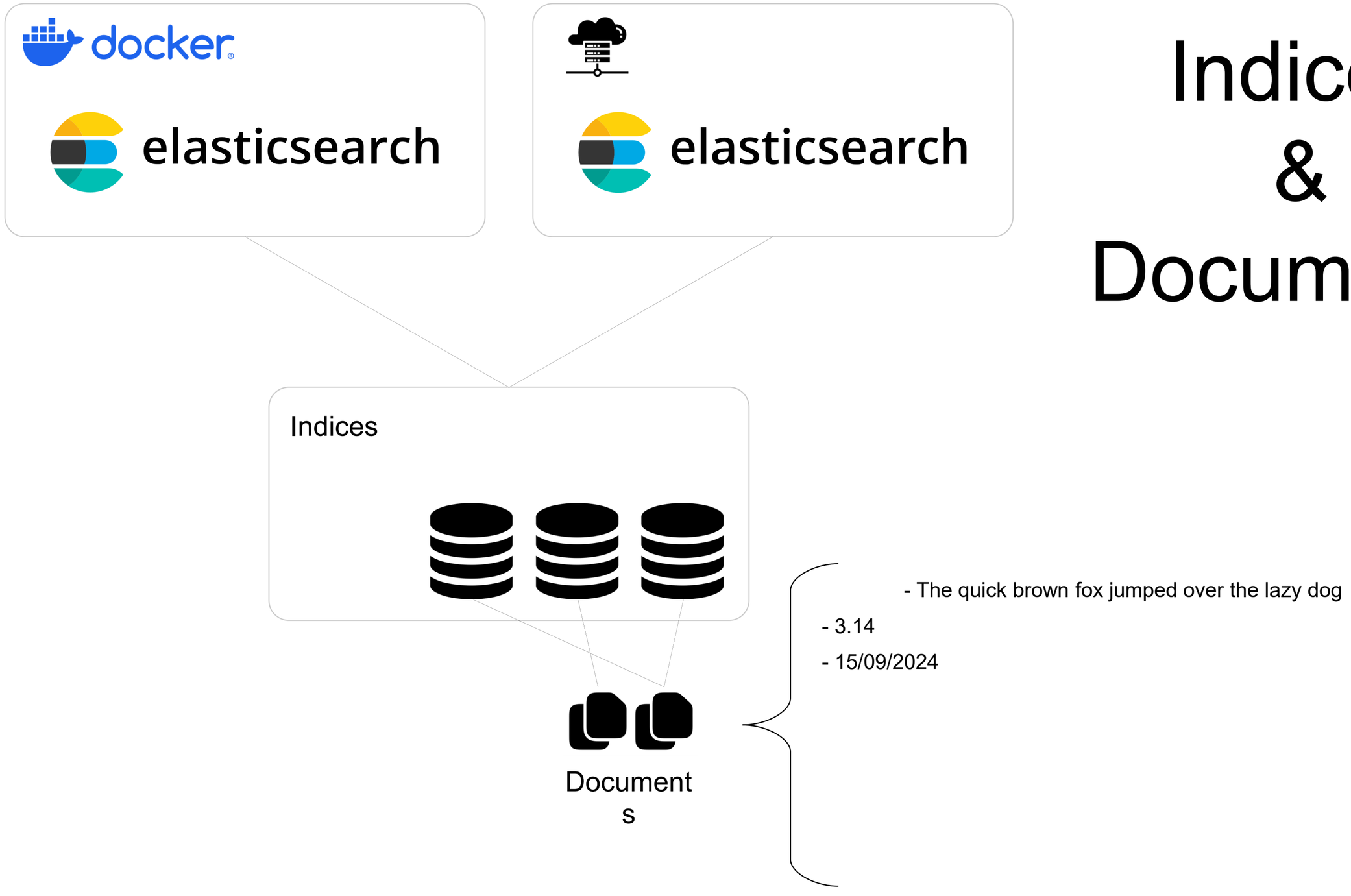
- Full-text search
- Relevance scoring
- Aggregations
- Horizontal scalability

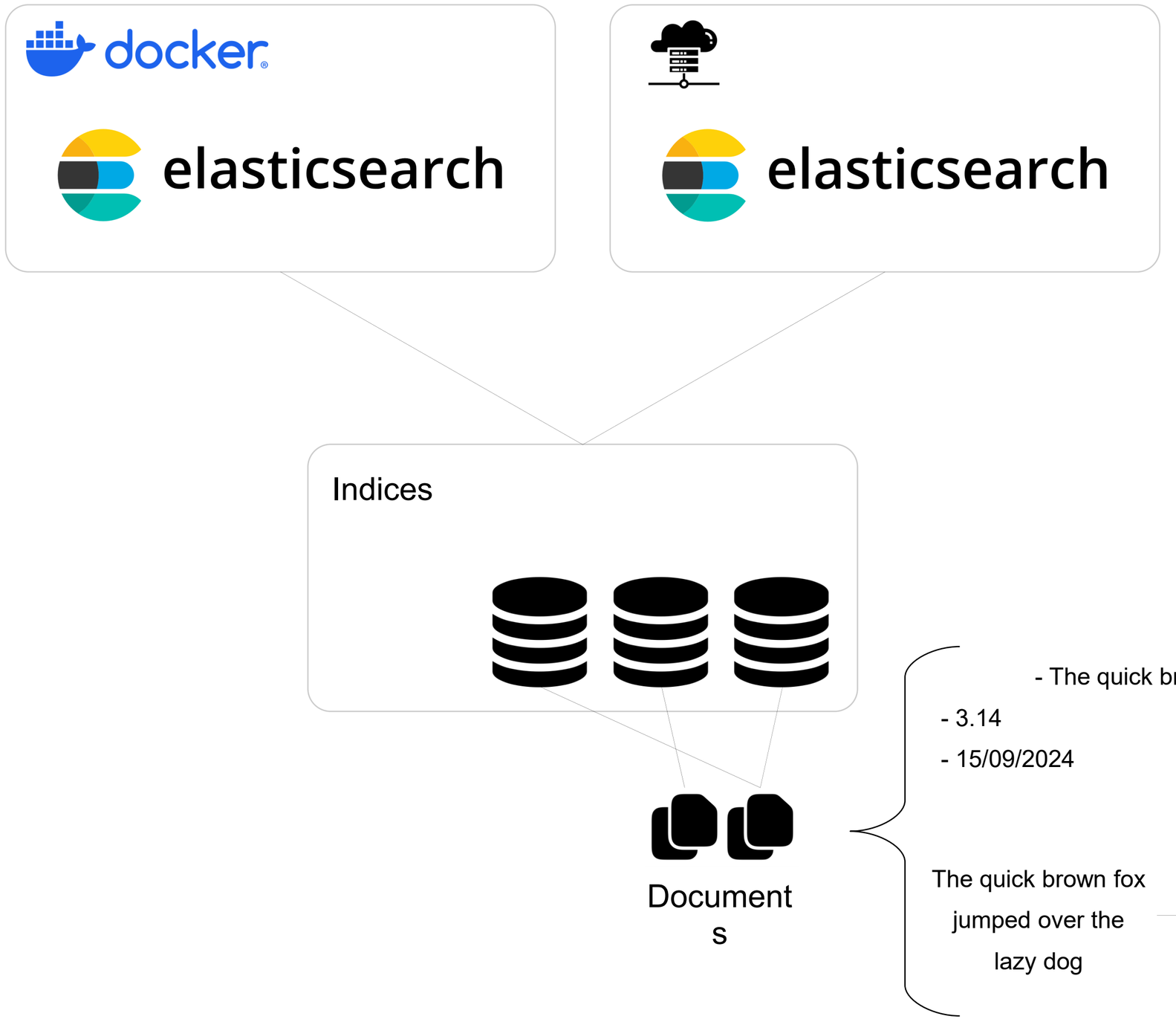
Mental Model: Index, Not Database

- Data is copied into an index
- Optimized for search
- Source of truth elsewhere

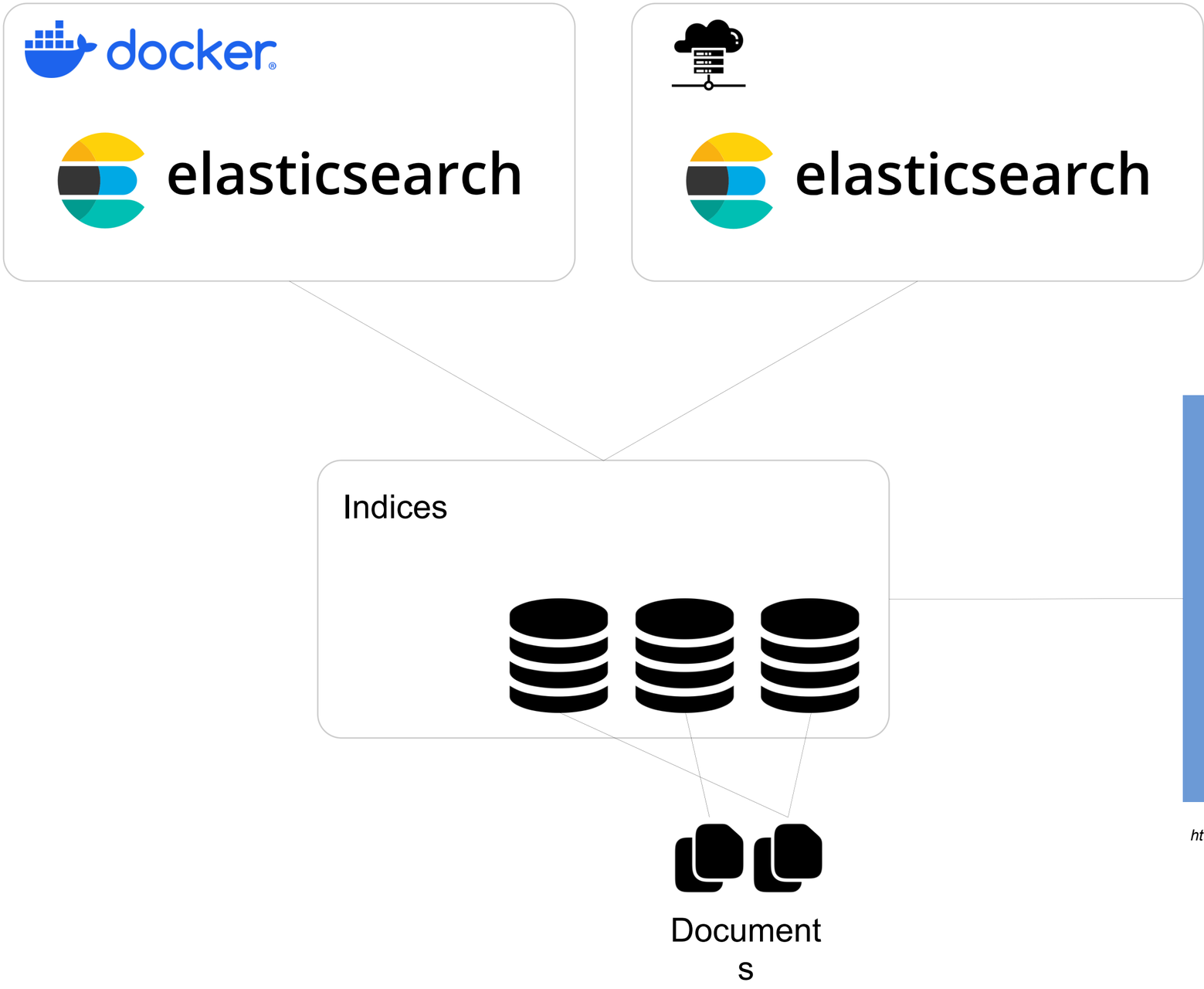
Deployment Options (Local vs Cloud)

Indices & Documents



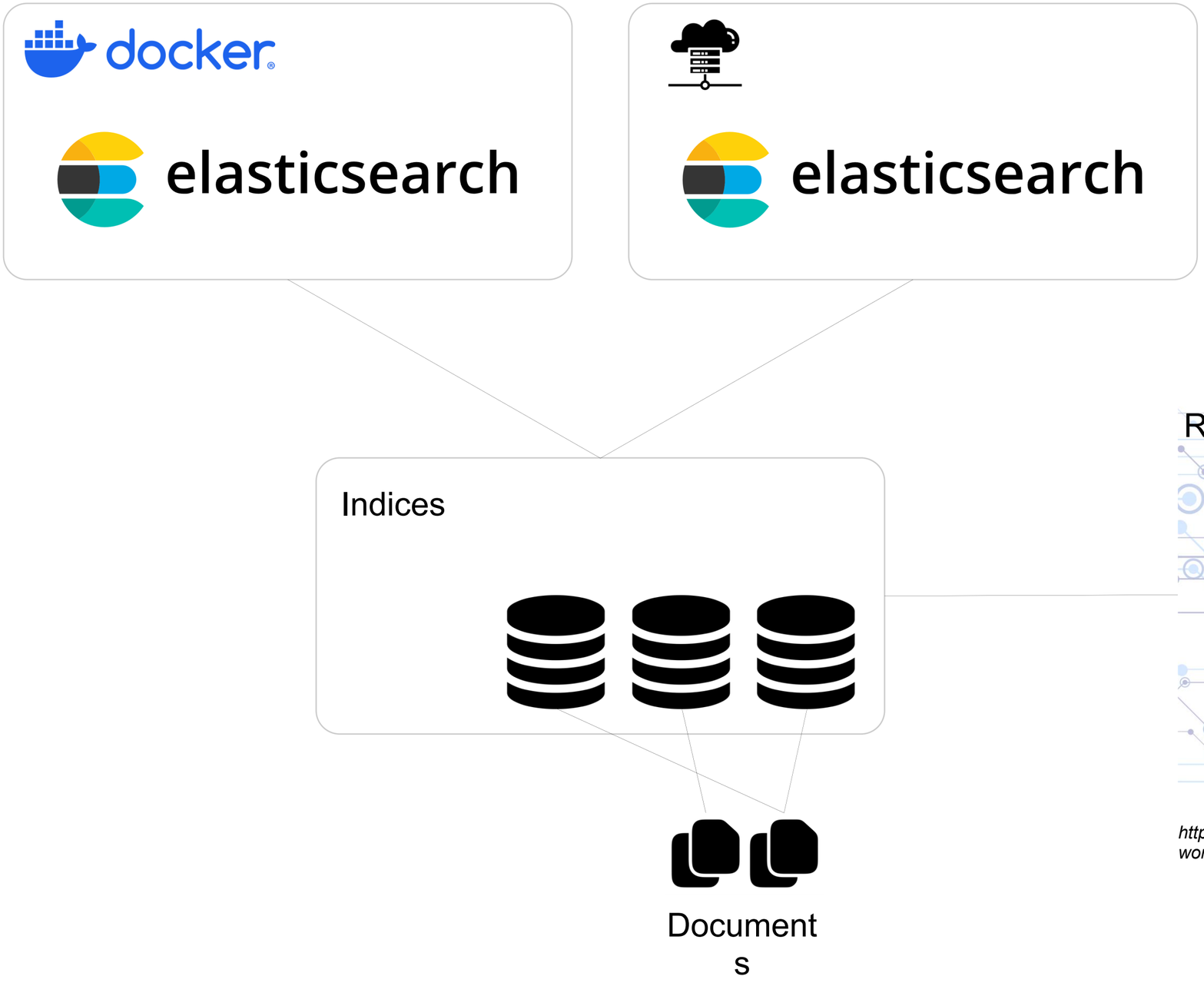


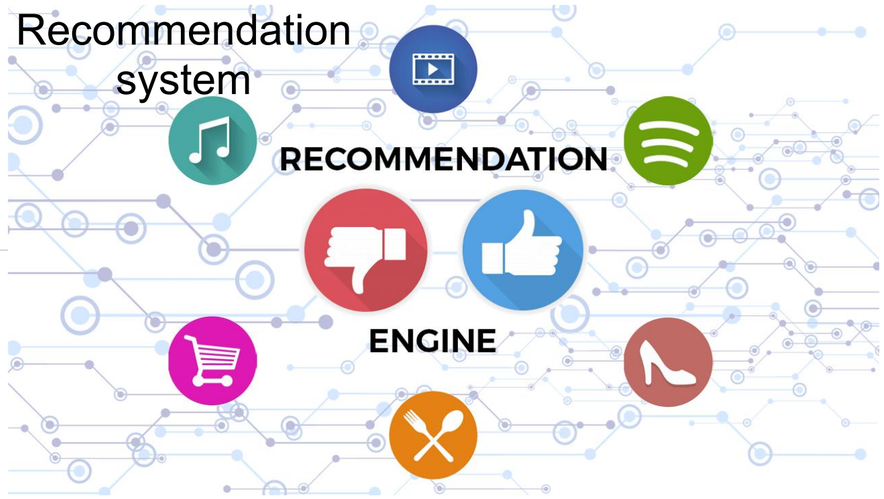
Text → Embeddings → Vectors



https://www.travelmediagroup.com/the-power-of-facebook-as-a-search-engine-2/

What Can You Build on Top of Elasticsearch? Search Systems





https://www.shopagain.com/blog/product-recommendation-engines-what-is-it-how-it-work/

What Can You Build on Top of Elasticsearch? Recommendation Systems

2 Core concepts & Architecture

Core Concepts

- Index
- Document
- Field
- Mapping
- Analyzer

Inverted Index

- Term → document list
- Fast search
- Expensive writes

Cluster Architecture

- Cluster
- Node
- Shard
- Replica

Shard Distribution

- Primary shards
- Replica shards
- Parallel execution

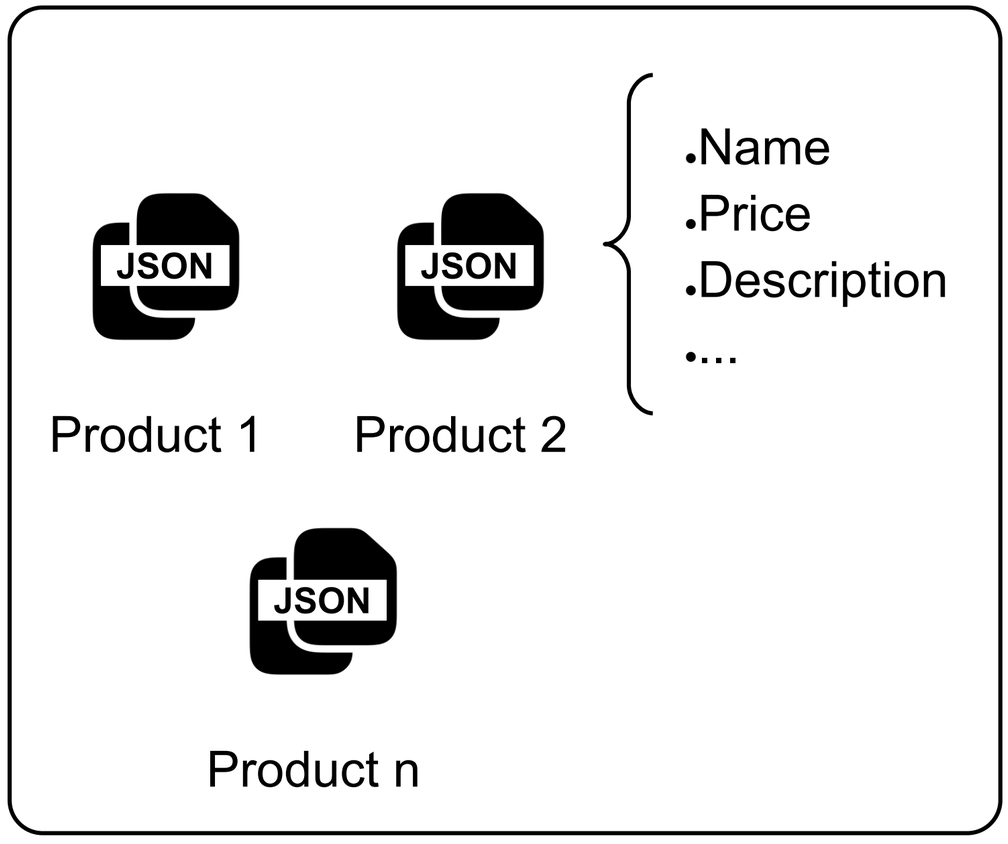
Query Flow

- Client → Coordinating node
- Fan-out to shards
- Merge & scoring

BLOCK 2
INDEXING & MAPPING

3 Create an index

What is an index ?



Product Index

```python
from pprint import pprint
from elasticsearch import Elasticsearch

es = Elasticsearch('http://localhost:9200')
client_info = es.info()
print('Connected to Elasticsearch!')
pprint(client_info.body)
```

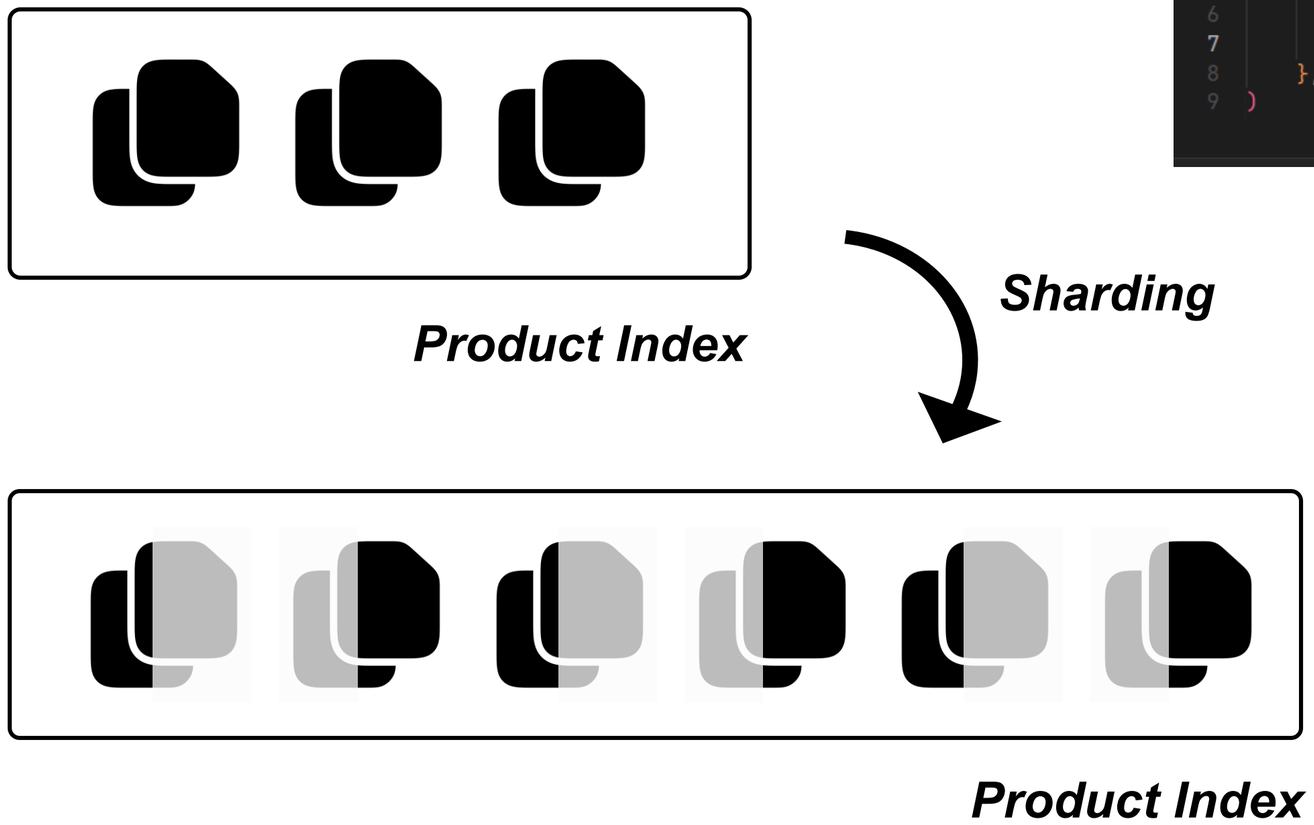
✓ 0.0s

```
Connected to Elasticsearch!
{'cluster_name': 'docker-cluster',
 'cluster_uuid': 'DlYG5m9gR3upn7qgaYyAJA',
 'name': '3d37442d2591',
 'tagline': 'You Know, for Search',
 'version': {'build_date': '2024-08-05T10:05:34.233336849Z',
             'build_flavor': 'default',
             'build_hash': '1a77947f34deddb41af25e6f0ddb8e830159c179',
             'build_snapshot': False,
             'build_type': 'docker',
             'lucene_version': '9.11.1',
             'minimum_index_compatibility_version': '7.0.0',
             'minimum_wire_compatibility_version': '7.17.0',
             'number': '8.15.0'}}
```

```python
es.indices.create(index='my_index')
```

Shards & replicas

Number of shards = 2



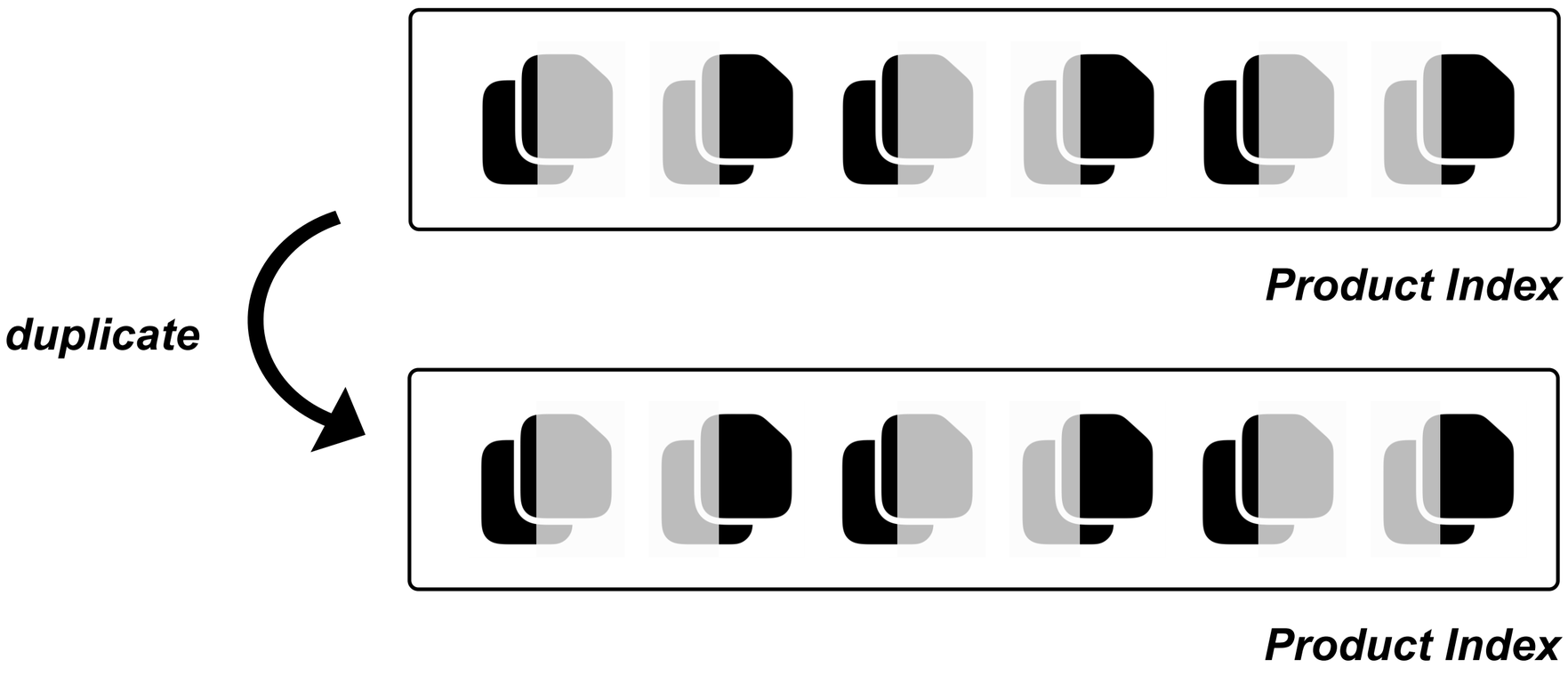

```python
es.indices.delete(index='my_index', ignore_unavailable=True)
es.indices.create(
    index="my_index",
    settings={
        "index": {
            "number_of_shards": 3, # how many pieces the data is split into
        }
    },
)
```

Shards & replicas

Number of shards = 2

Number of replicas = 1

```python
es.indices.delete(index='my_index', ignore_unavailable=True)
es.indices.create(
    index="my_index",
    settings={
        "index": {
            "number_of_shards": 3, # how many pieces the data is split into
            "number_of_replicas": 2 # how many copies of the data
        }
    },
)
```

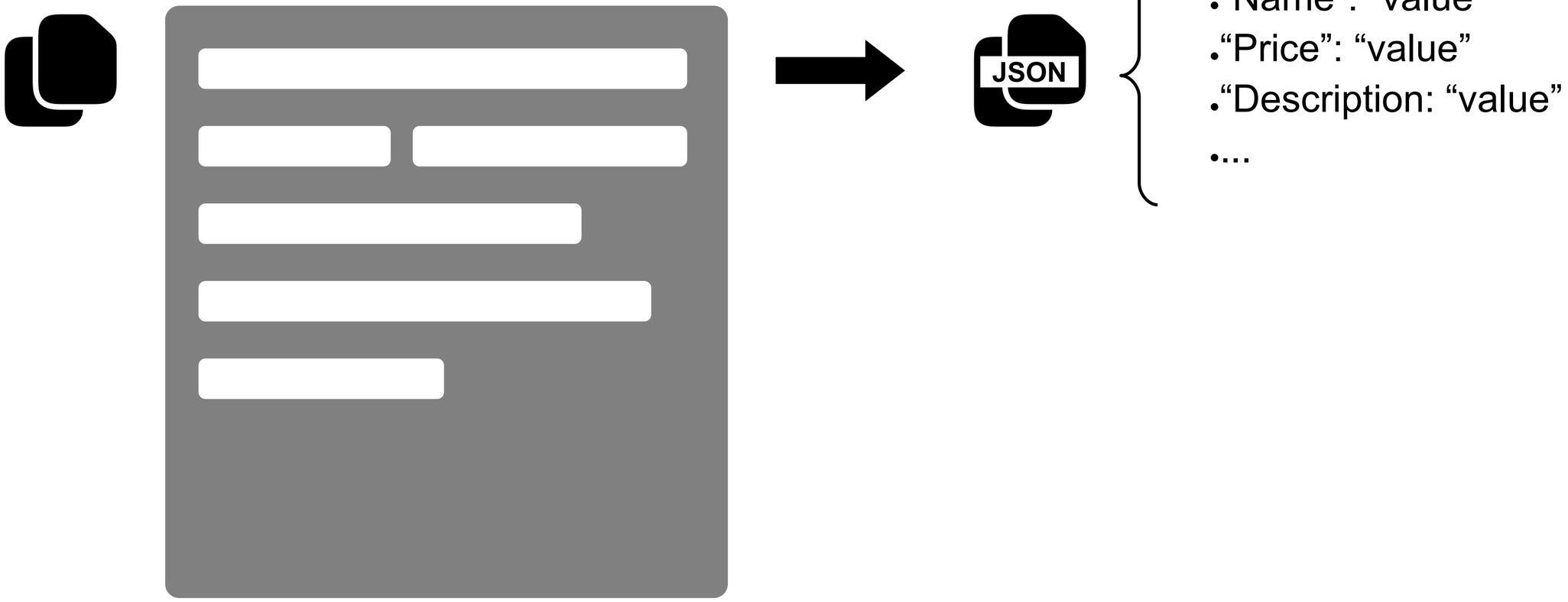


4 Inserting documents

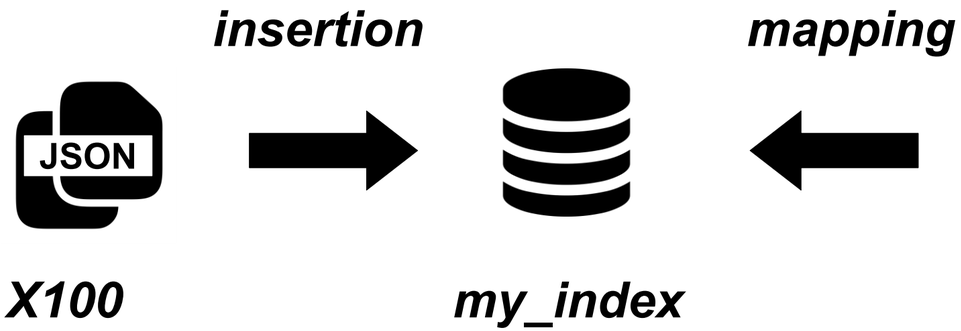
Index & Mapping

- Explicit vs dynamic mapping
- text vs keyword
- Data types

Document



Document




```python
document = {
    'title': 'title',
    'text': 'text',
    'created_on': '2024-09-22',
}
response = es.index(index='my_documents', body=document)
```

Field	Type
created_on	date
text	text
title	text

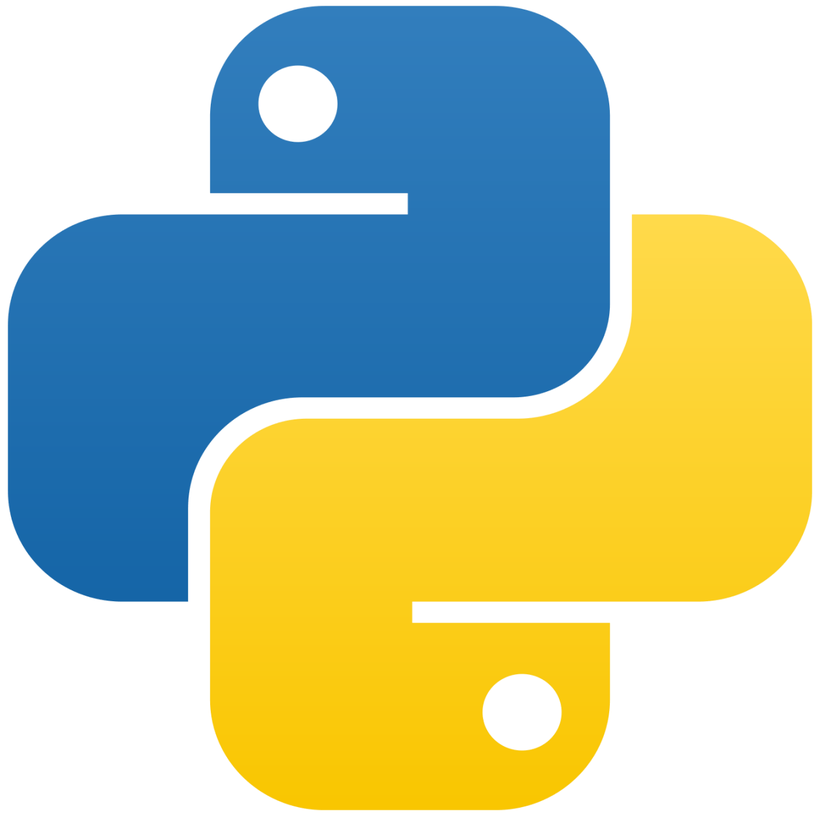
ⓘ This process is called ***mapping*** and can be done ***automatically*** or ***manually***. By ***default***, ElasticSearch does it ***automatically***.

Why Mapping Matters

- Defines indexing behavior
- Affects search
- Requires reindex

Indexing Documents

- JSON documents
- Near real-time
- Refresh interval



How to run Elasticsearch locally

- Docker Desktop (recommended for workshop)
- Local install (advanced / not needed today)
- Cloud / managed service (prod-like)

Run Elasticsearch with Docker (single-node)

- Expose HTTP API on *localhost:9200*
- Disable security for local demo
- Persist data in a Docker volume
- Version pinned (reproducible)

Run Elasticsearch with Docker (single-node)

```
docker run -d --name elasticsearch `
  -p 9200:9200 `
  -e "discovery.type=single-node" `
  -e "xpack.security.enabled=false" `
  -e "xpack.license.self_generated.type=trial" `
  -v elasticsearch-data:/usr/share/elasticsearch/data `
  docker.elastic.co/elasticsearch/elasticsearch:8.15.0
```

Inline version for copy and paste:

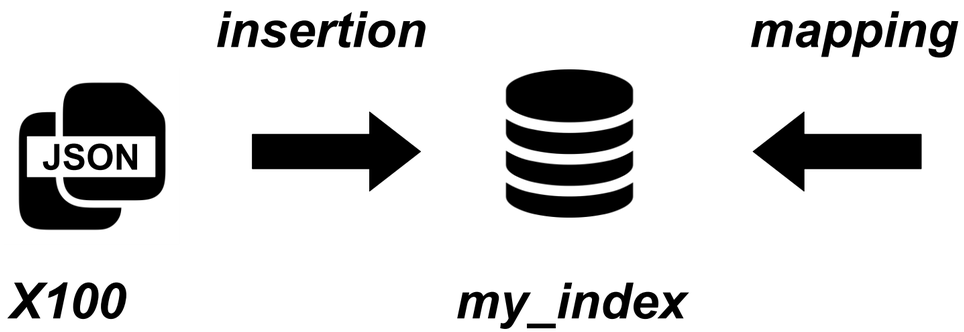
```
docker run -d --name elasticsearch -p 9200:9200 -e "discovery.type=single-node" -e "xpack.security.enabled=false" -e "xpack.license.self_generated.type=trial" -v elasticsearch-data:/usr/share/elasticsearch/data docker.elastic.co/elasticsearch/elasticsearch:8.15.0
```

Run Elasticsearch with Docker (single-node)

- *py -m venv .venv*
- *.\.venv\Scripts\Activate.ps1*
- *pip install "elasticsearch==8.15.1"*

5 Field data types

Field data types



Field	Type
created_on	**date**
text	**text**
title	**text**

Field data types

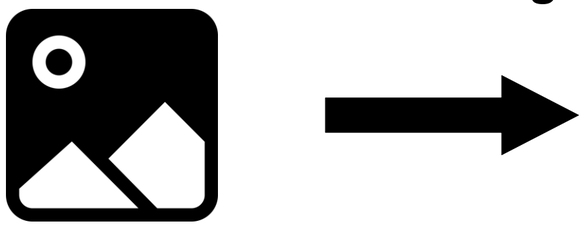
1)Common types

.Binary

.Accepts a binary value as a **Base64** encoded string.

.Is not searchable and is not stored.

.Use **_source (i.e., document)** to get the data back.



encoding

IVBORw0KGgoAAAANSUhEUgAAA
OEAAADhCAIAAADiVBORw0KGgo
AAAANSUhEUgAAAOEAAADhCAI
A….

Base64 representation

Field data types

1)Common types

.Binary

.Boolean (True / False)

.Numbers (long, integer, byte, short, etc)

.Dates

.Keyword (IDs, email addresses, status codes, zip codes, etc)

Field data types

2)Objects types (JSON)

.Object


```
{
   "region": "US",
   "manager": {
      "age": 30,
      "name": {
         "first": "John",
         "last": "Smith"
      }
   }
}
```

indexation

```
{
   "region": "US",
   "manager.age": 30,
   "manager.name.first": "John",
   "manager.name.last": "Smith"
}
```

Field data types

2)Objects types (JSON)

.Object

.Flattened

- .Efficient for deeply nested JSON objects.
- .Hierarchical structure is not preserved.

.Nested

- .Cases where you have array of objects.
- .Maintains relationship between the object’s fields.

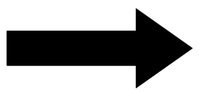
Field data types

2)Objects types (JSON)

•Flattened / Nested object example


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

indexation



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

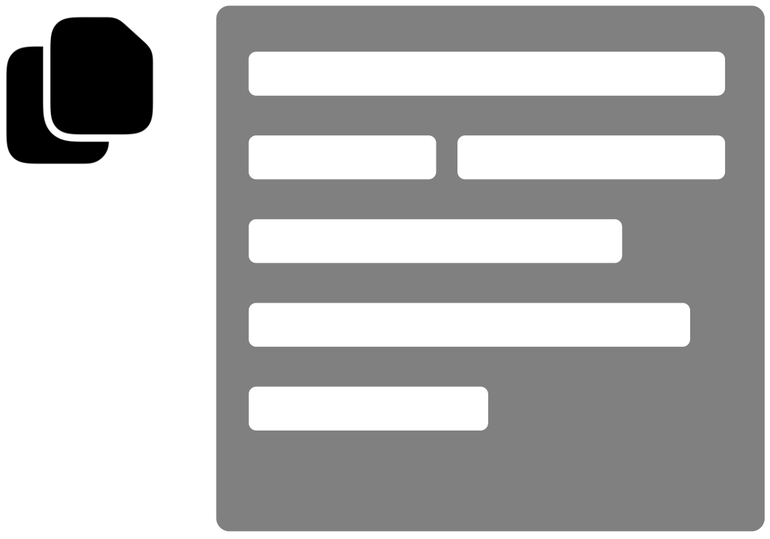
Field data types

3)Text search types

.Text

.Used for full-text content.

.Examples: Body of an email or the description of a product.



Unstructured format

analyzer

Structured format that’s optimized for search.

Field data types

3)Text search types

.Text

- .Used for full-text content.
- .Examples: Body of an email or the description of a product.

.Completion

.Search as you type

.Annotated text

Field data types

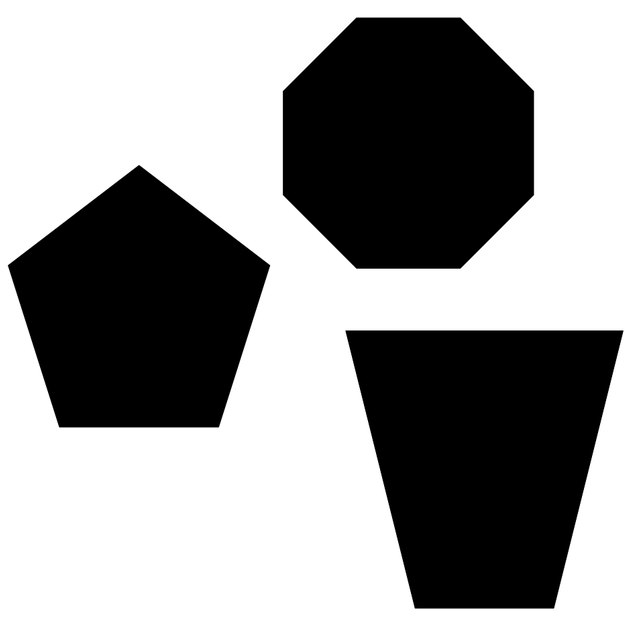
4)Spatial data types

.Geo point

.Geo shape

.Point (Cartesian point)

.Shape (Cartesian geometry)



Field data types

elastic Platform Solutions Customers Resources Pricing Docs Start free trial Contact Sales

Elasticsearch Guide:
8.15 (current)

- What's new in 8.15
- Elasticsearch basics
- Quick starts
- Set up Elasticsearch
- Upgrade Elasticsearch
- Index modules
- Mapping

Elastic Docs › Elasticsearch Guide [8.15] › Mapping

Field data types

edit

Each field has a *field data type*, or *field type*. This type indicates the kind of data the field contains, such as strings or boolean values, and its intended use. For example, you can index strings to both `text` and `keyword` fields. However, `text` field values are analyzed for full-text search while `keyword` strings are left as-is for filtering and sorting.

Field types are grouped by *family*. Types in the same family have exactly the same search behavior but may have different space usage or performance characteristics.

Currently, there are two type families, `keyword` and `text`. Other type families have only a single field type. For example, the `boolean` type family consists of one field type: `boolean`.

On this page

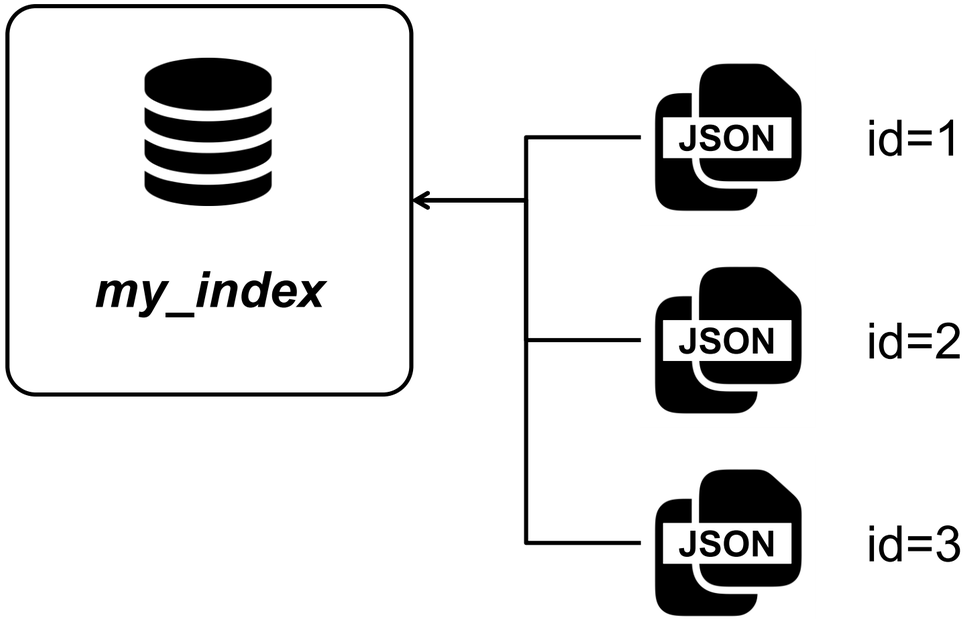
- Common types
- **Objects and relational types**
- Structured data types
- Aggregate data types
- Text search types
- Document ranking types
- Spatial data types
- Other types
  - Arrays
  - Multi-fields

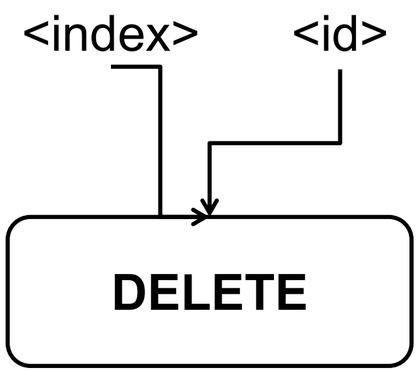
Most Popular

VIDEO

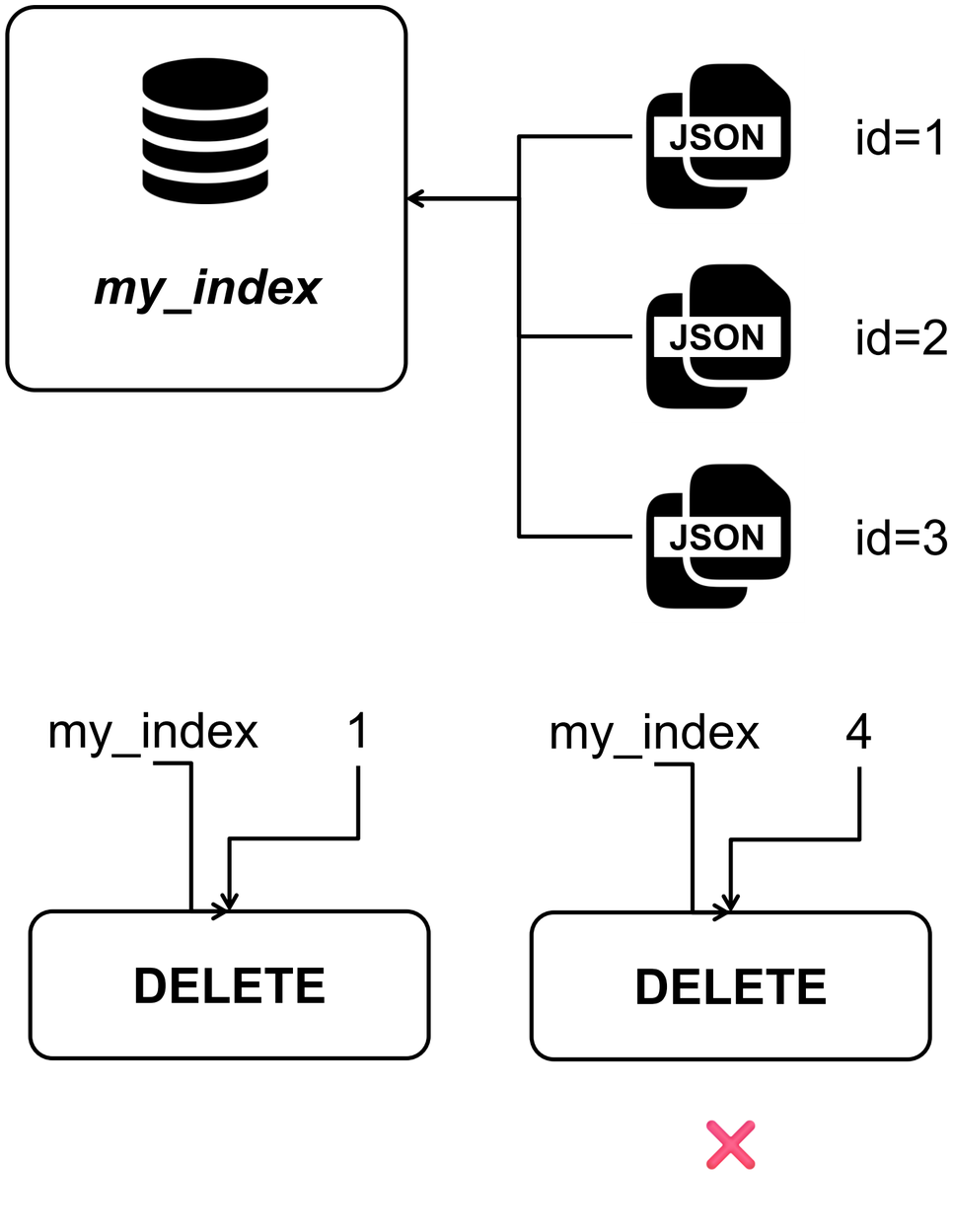
6 Delete documents

Delete documents



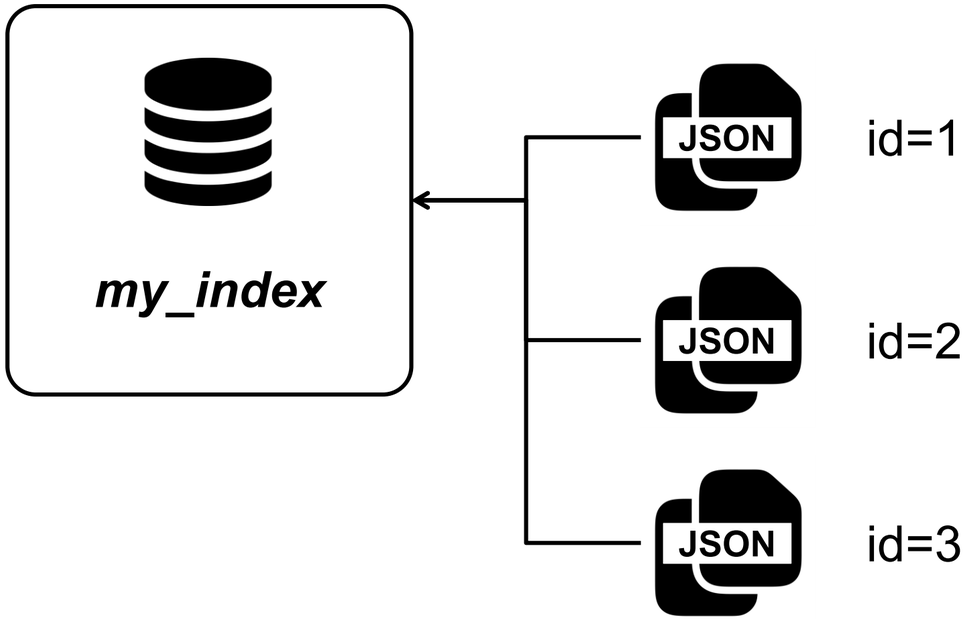


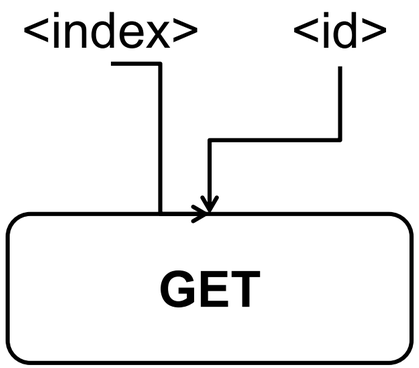
Delete documents



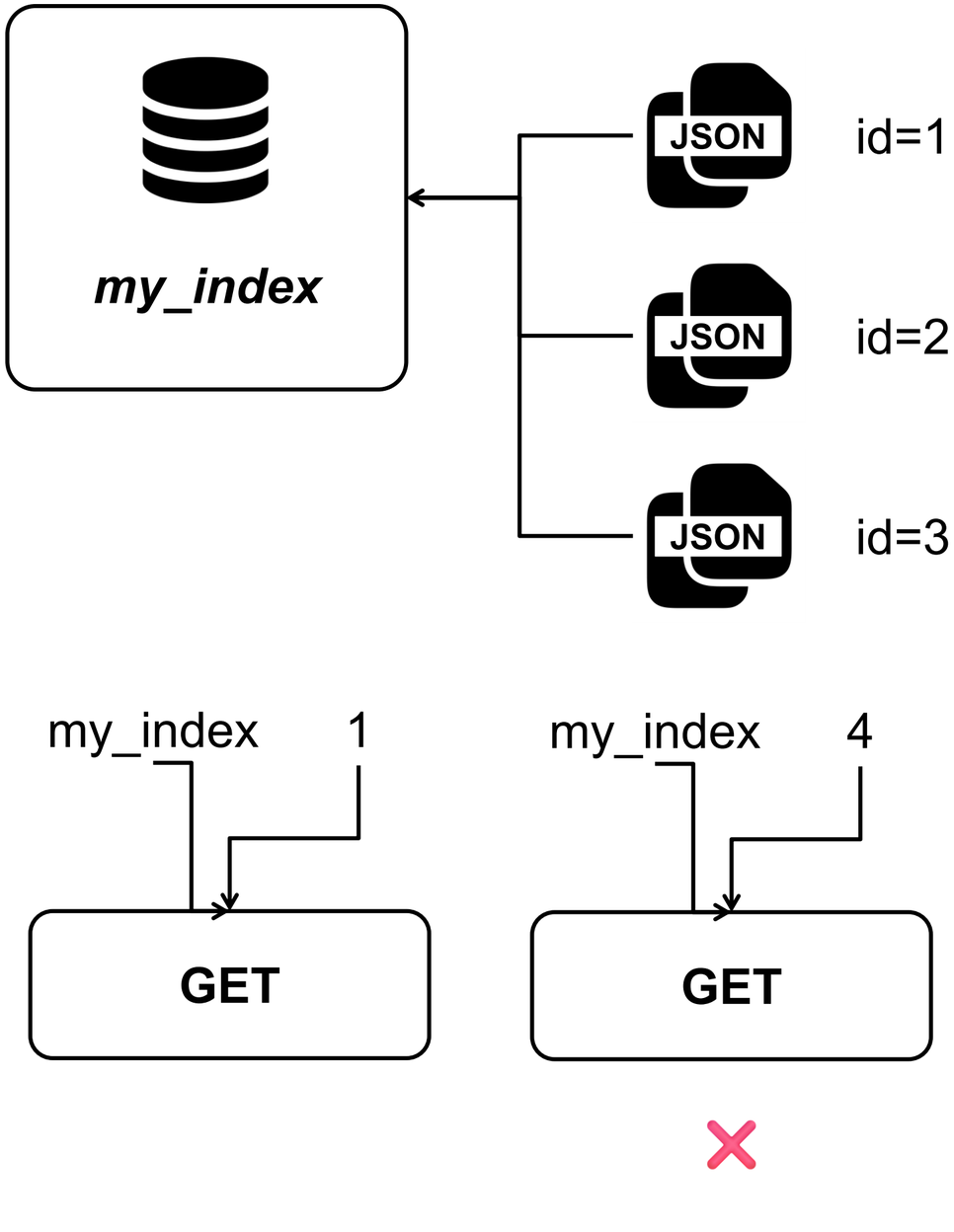
7 Get document

Get documents



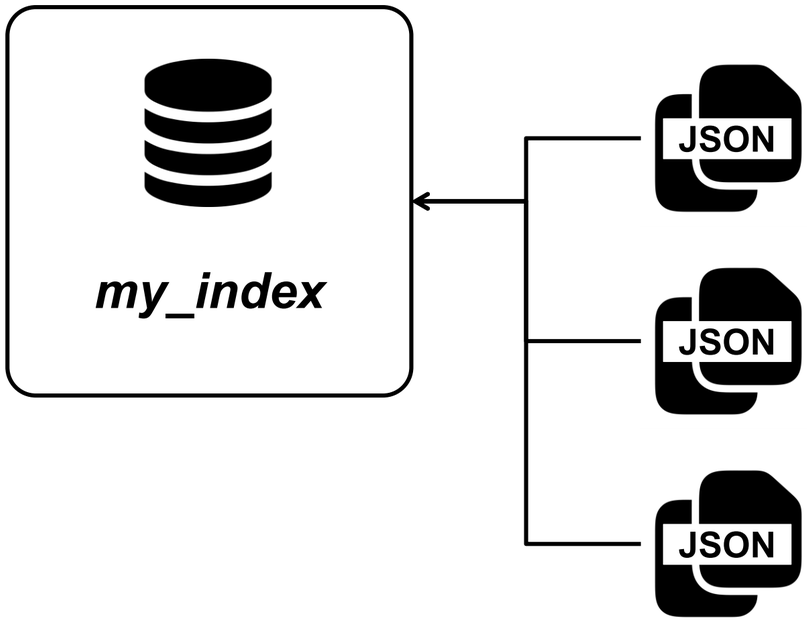


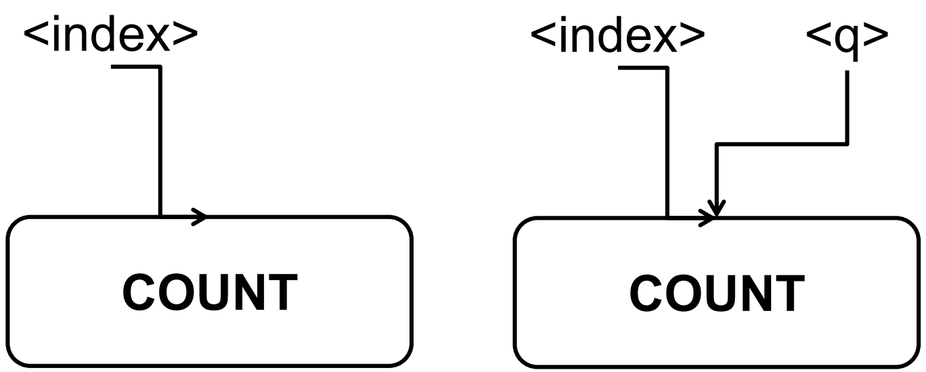
Get documents



8 Count documents

Count documents

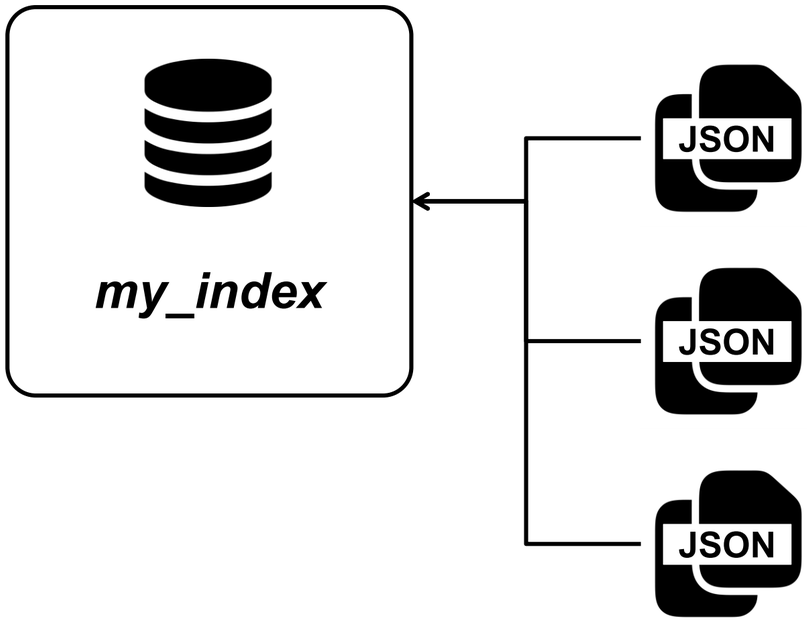


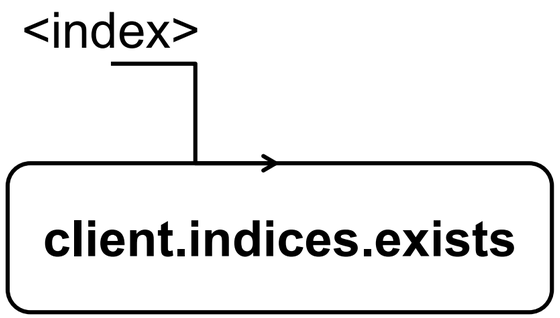


ⓘ The **query** parameter is used to match certain criteria

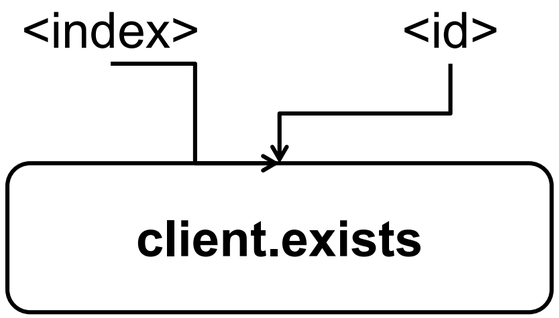
9 The exists API

The exists API





ⓘ This checks if an **index exists** in ElasticSearch.

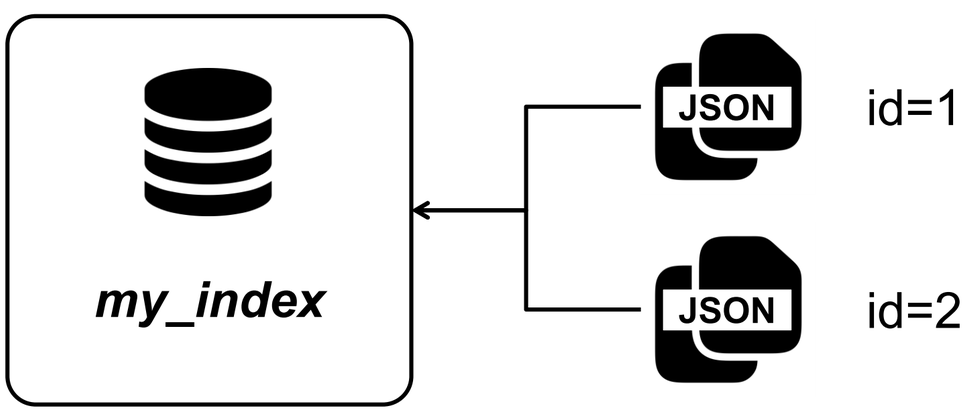


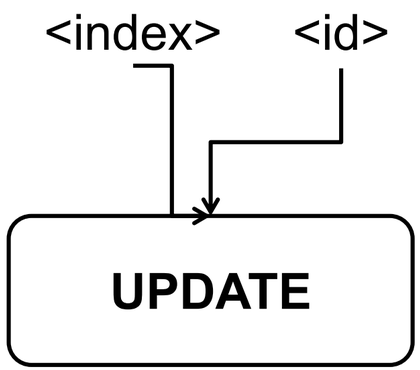
ⓘ This checks if a **document exists** in an index.

10 Update document

Update documents

1) Documents exists in the index





ⓘ The update operation follows these steps:

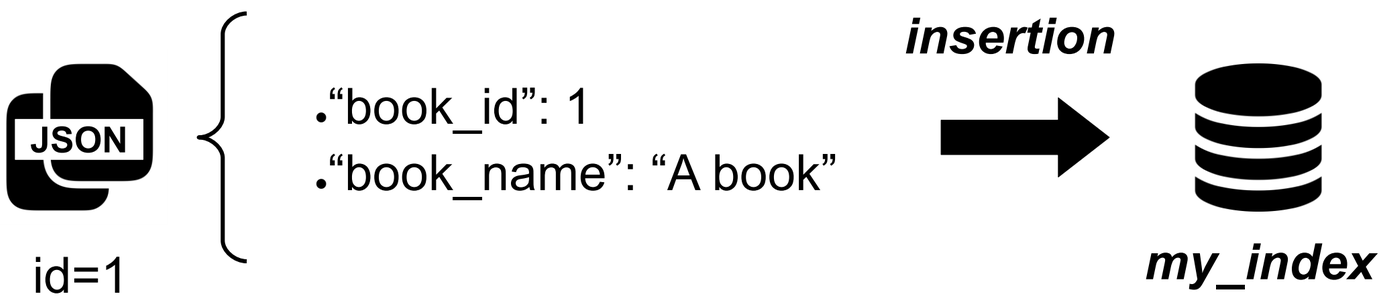
1Get the document.

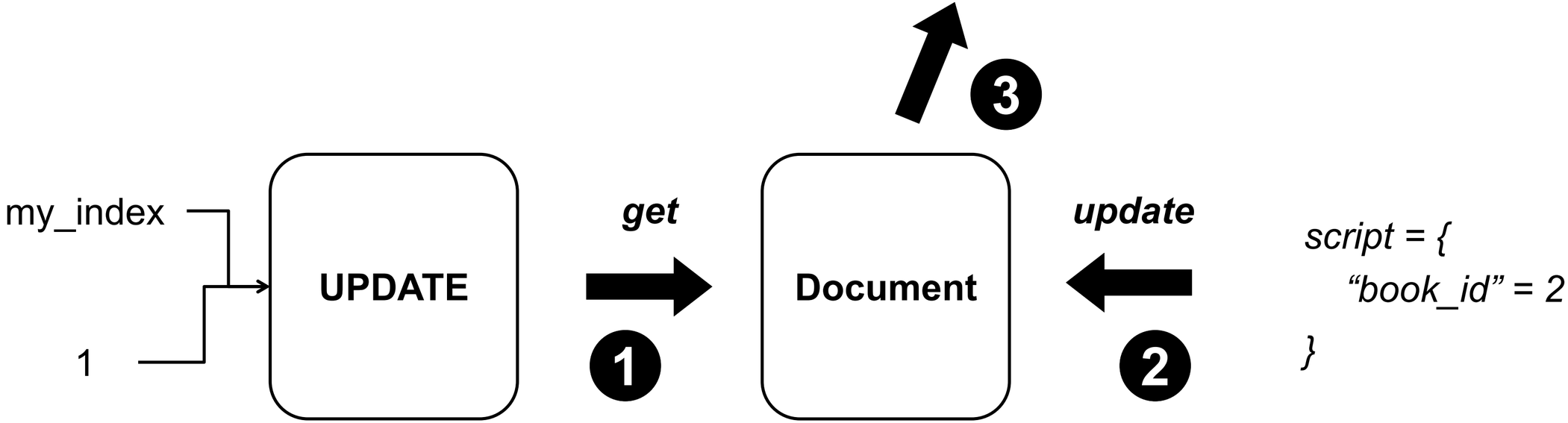
2Update it (e.g. **Add a new field**, **remove a field**, or **update a field**)

3Re-index the result.

Update documents

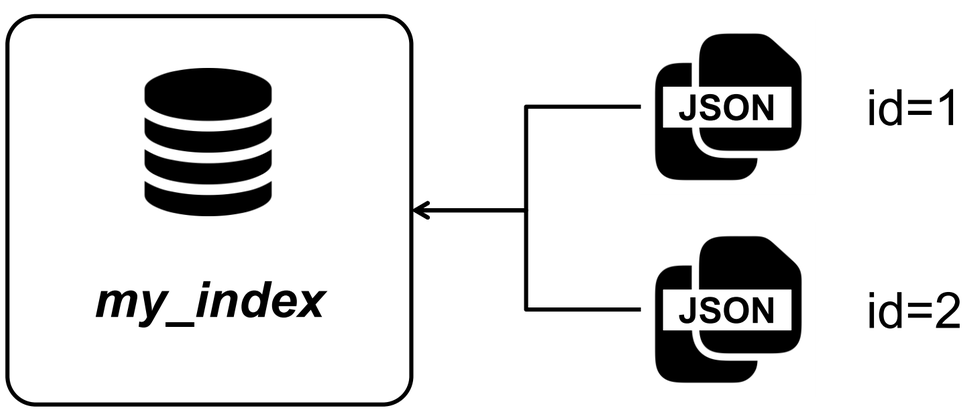
But how do you update the document?

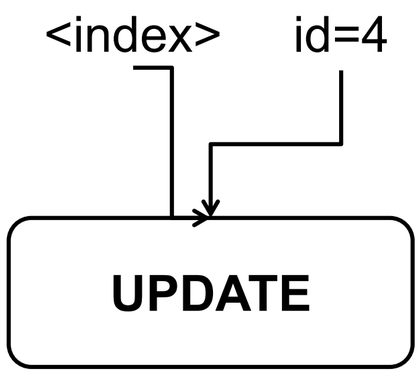




Update documents

2) Document doesn’t exist





ⓘ The update operation can create the document if it doesn’t exist

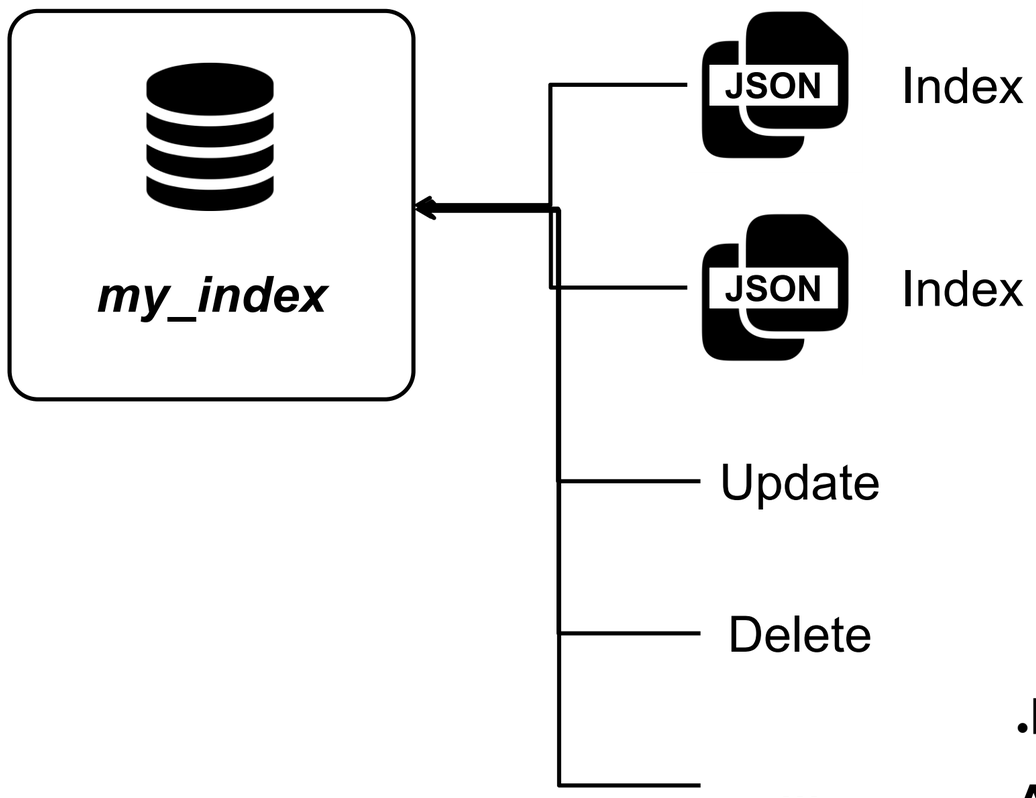
1Add the values you want to insert.

2Set ***doc_as_upsert*** to true.

Read more

11 Bulk API

The bulk API



- Each operation (**index, update, delete**) makes a **separate API call**.
- The **bulk API** performs multiple operations in **one API call**. This increases indexing speed.

The bulk API - Syntax

- action and metadata\n
- optional source\n
- action and metadata\n
- optional source\n
- …
- action and metadata\n
- optional source\n

ⓘ The source is required for:

- update
- index
- create

ⓘ The action can be one of the following:

- index
- update
- create
- delete

The bulk API - Example

```
response = es.bulk(
    operations=[
        {
            "index": {
                "_index": "test",
                "_id": "1"
            }
        },
        {
            "field1": "value1"
        },
        {
            "delete": {
                "_index": "test",
                "_id": "2"
            }
        },
        {
            "create": {
                "_index": "test",
                "_id": "3"
            }
        },
        {
            "field1": "value3"
        },
        {
            "update": {
                "_id": "1",
                "_index": "test"
            }
        },
        {
            "doc": {
                "field2": "value2"
            }
        }
    ],
)
```

ⓘ The source is required for:

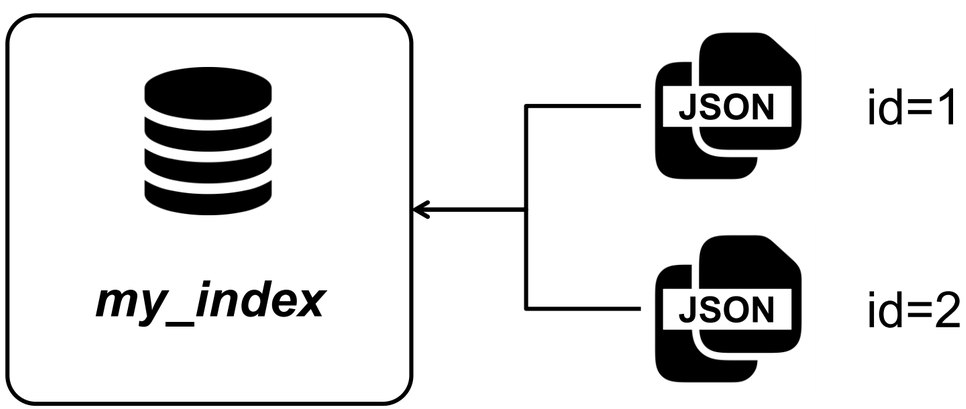
- update
- index
- create

Action

Source

12 The search API – Part 1

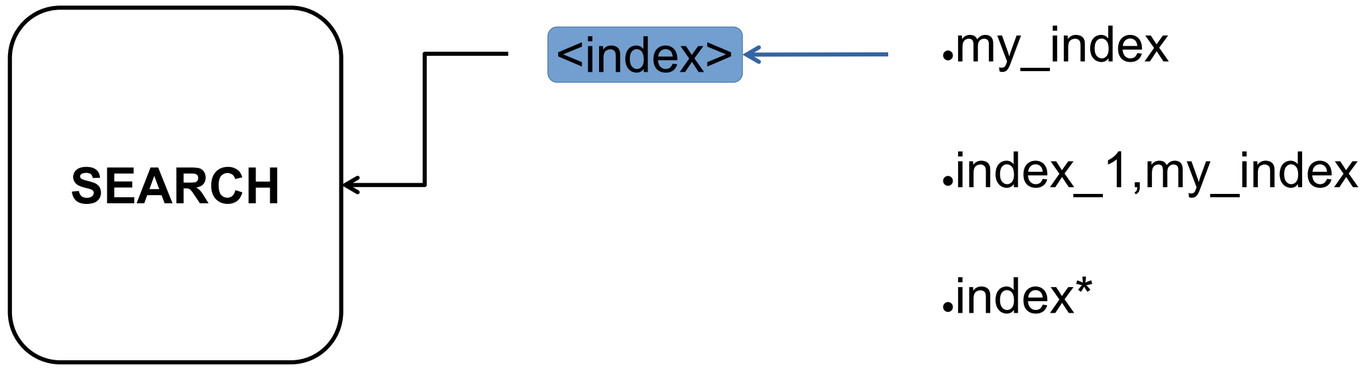
The search API



You use the search API to build:

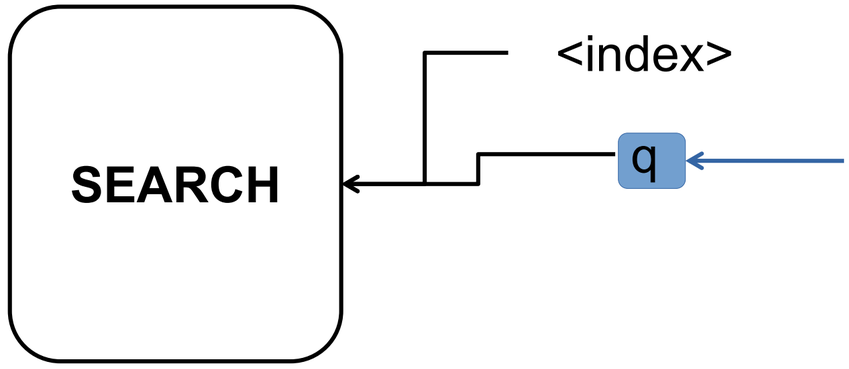
- Search engines
- Recommendation systems
- Real-time dashboards
- Log data analysis
- ...

The search API



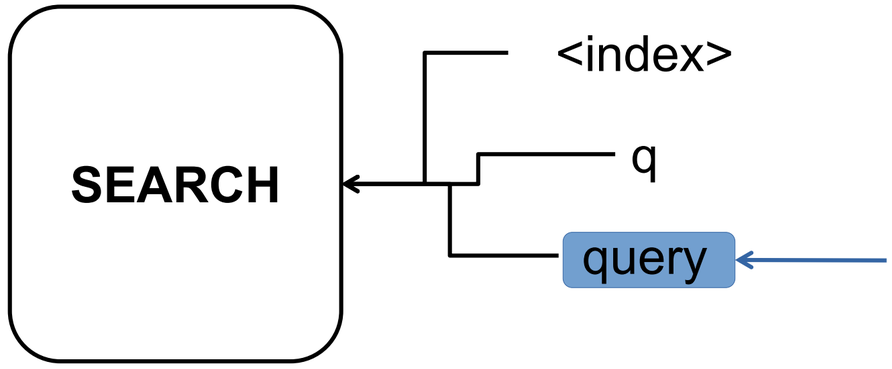
- .my_index
- .index_1,my_index
- .index*
- _all

The search API



- Use it for simple searches.
- Uses the **Lucene** syntax.

The search API

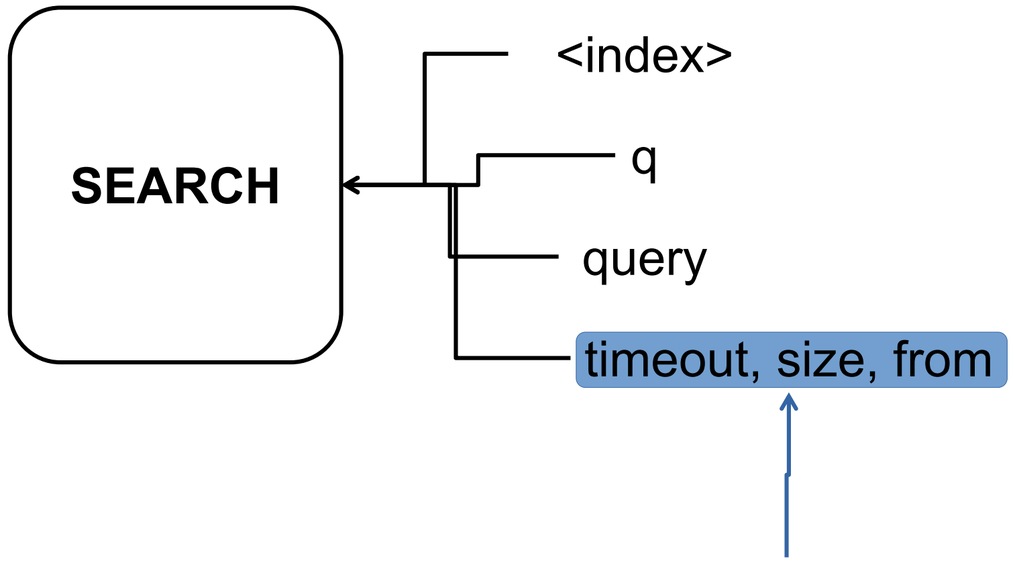


- Use it for complex, structured queries.

- Uses the **Query DSL** language.

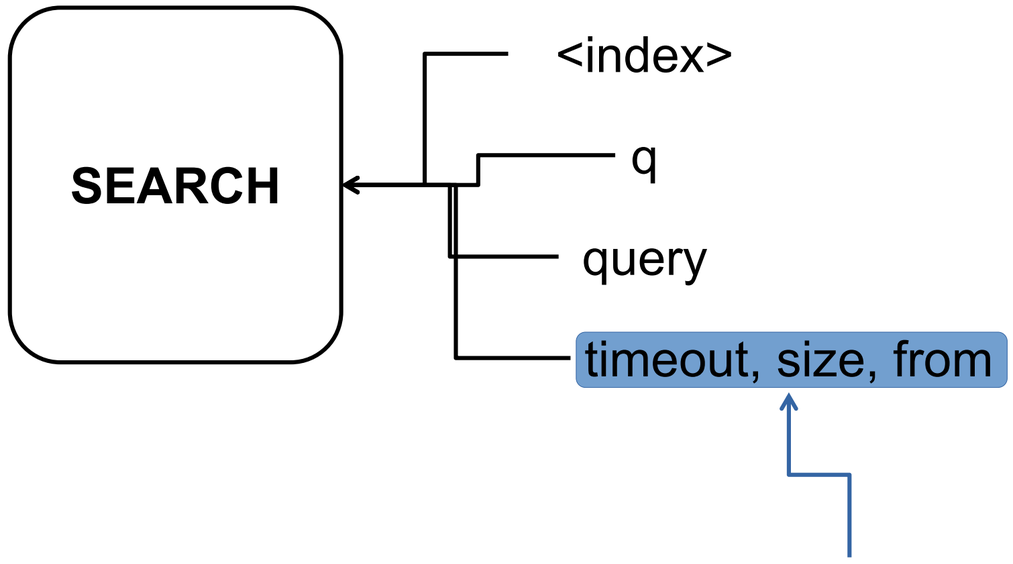
- Default value is **match all**.

The search API



- **Timeout**:
  - The maximum time to wait for a search request to complete.
  - Time units (seconds, milliseconds, days, etc) .
- **Size**:
  - Defines the number of hits (documents) to return.
  - Default value is 10. Max value is 10000.

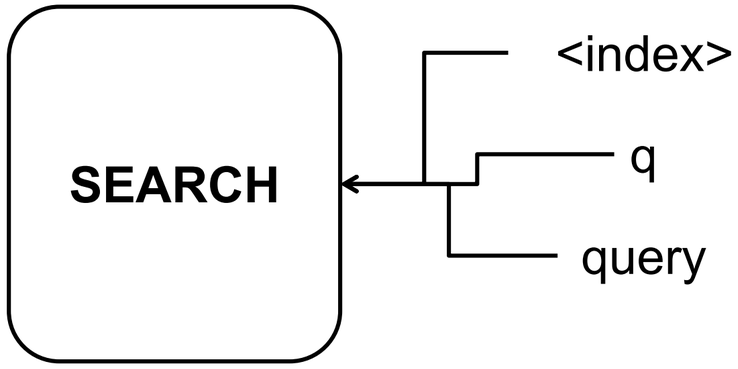
The search API



- **from**:
- Starting point from which to return search results (**pagination**) .
- Useful if you want to implement **skip** functionality.

13 The search API – Part 2

The search API – Query DSL



Query DSL is used to create complex, structured queries.

Query DSL consists of two types of clauses:

- Leaf clauses (**match, term, or range**)
- Compound clauses (**bool**)

ⓘ Query DSL means Query (Domain Specific Language)

The search API – Query DSL

Leaf clauses

1. match:

- Is used to perform **full-text search**.
- Returns documents that match a provided **text, number, date, or bool**.
- The field must be mapped to a **text** data type.

2. term:

- Returns documents that contain an **exact** term in a provided field.
- The field must be mapped to a **keyword** data type or a **numeric/date** type.
- Example usage : product ID, book ID, username, etc.

3. range:

- Returns documents that contain terms within a provided range.

The search API – Query DSL

.Example for match query:

```
response = es.search(
    index="my_index",
    body={
        "query": {
            "match": {
                "description": "A description."
            }
        }
    }
)
```

The search API – Query DSL

.Example for term query:

```
response = es.search(
    index="my_index",
    body={
        "query": {
            "term": {
                "product_id": "PRODUCT_12345"
            }
        }
    }
)
```

The search API – Query DSL

.Example for range query:

```
response = es.search(
    index="my_index",
    body={
        "query": {
            "range": {
                "publication_date":
                    {
                        "gte": "2023-01-01",
                        "lte": "2023-12-31"
                    }
                }
            }
        }
    }
)
```

The search API – Query DSL

Compound clauses

.bool:

- .Combines multiple queries using boolean logic:
- **.must, filter, should, must_not.**
- .The field must be mapped to a **text** data type.

The search API – Query DSL

.Example for bool query:

```
response = es.search(index="my_index", body={
    "query": {
        "bool": {
            "must": [
                {
                    "match": {
                        "title": "Elasticsearch"
                    }
                }
            ],
            "filter": [
                {
                    "term": {
                        "status": "published"
                    }
                }
            ],
            "should": [
                {
                    "match": {
                        "tags": "search"
                    }
                }
            ],
            "must_not": [
                {
                    "term": {
                        "deleted": True
                    }
                }
            ]
        }
    }
)
```

The search API – Query DSL

.Example for bool query:

```
response = es.search(index="my_index", body={
    "query": {
        "bool": {
            "must": [
                {
                    "match": {
                        "title": "Elasticsearch"
                    }
                }
            ],
            "filter": [
                {
                    "term": {
                        "status": "published"
                    }
                }
            ],
            "should": [
                {
                    "match": {
                        "tags": "search"
                    }
                }
            ],
            "must_not": [
                {
                    "term": {
                        "deleted": True
                    }
                }
            ]
        }
    }
})
```

Keeping documents where title is equal to **Elasticsearch**

The search API – Query DSL

.Example for bool query:

```
response = es.search(index="my_index", body={
    "query": {
        "bool": {
            "must": [
                {
                    "match": {
                        "title": "Elasticsearch"
                    }
                }
            ],
            "filter": [
                {
                    "term": {
                        "status": "published"
                    }
                }
            ],
            "should": [
                {
                    "match": {
                        "tags": "search"
                    }
                }
            ],
            "must_not": [
                {
                    "term": {
                        "deleted": True
                    }
                }
            ]
        }
    }
})
```

Filtering documents with a status of **published**

The search API – Query DSL

.Example for bool query:

```
response = es.search(index="my_index", body={
    "query": {
        "bool": {
            "must": [
                {
                    "match": {
                        "title": "Elasticsearch"
                    }
                }
            ],
            "filter": [
                {
                    "term": {
                        "status": "published"
                    }
                }
            ],
            "should": [
                {
                    "match": {
                        "tags": "search"
                    }
                }
            ],
            "must_not": [
                {
                    "term": {
                        "deleted": True
                    }
                }
            ]
        }
    }
})
```

This match is optional

The search API – Query DSL

.Example for bool query:

```
response = es.search(index="my_index", body={
    "query": {
        "bool": {
            "must": [
                {
                    "match": {
                        "title": "Elasticsearch"
                    }
                }
            ],
            "filter": [
                {
                    "term": {
                        "status": "published"
                    }
                }
            ],
            "should": [
                {
                    "match": {
                        "tags": "search"
                    }
                }
            ],
            "must_not": [
                {
                    "term": {
                        "deleted": True
                    }
                }
            ]
        }
    }
})
```

Excluding any document where the **deleted** field is set to true

14 The search API – Part 3

The search API

Timeout

- It sets the **maximum duration** for a query to execute.
- If the search **takes longer** than the specified time, Elasticsearch will **abort the search.**

The search API

Timeout

- Sets the **maximum duration** for a query to execute.
- If the search **takes longer** than the specified time, Elasticsearch will **abort the search.**

Size

- Controls how many search results are returned.
- Max value is **10000.**

The search API

Timeout

- Sets the **maximum duration** for a query to execute.
- If the search **takes longer** than the specified time, Elasticsearch will **abort the search.**

Size

- Controls how many search results are returned.
- Max value is **10000.**

From

- Used for pagination.
- It tells Elasticsearch how many documents to skip before starting to return results.

The search API

Aggregations

- Performs calculation on the data
- Average, max, min, count.

14.5 The search API – Part 4

MatchAllQuery

- Match all documents
- Baseline / debugging query
- Default when no query is provided

MatchQuery

- Full-text search
- Uses analyzer
- Relevance scoring (BM25)

MatchPhraseQuery & MatchPhrasePrefixQuery

- Phrase matching
- Word order matters
- Prefix useful for autocomplete

TermQuery & TermsQuery

- Exact matching
- No analysis
- Keyword / numeric fields

RangeQuery

- Numeric and date ranges
- gte / lte / gt / lt
- Very common in filters

IdsQuery

- Fetch by document IDs
- Very fast
- No scoring

ExistsQuery

- Checks if field exists
- Useful for data quality
- Often used in filters

PrefixQuery & WildcardQuery

- Prefix = starts with
- Wildcard = pattern matching
- Can be expensive

DisMaxQuery

- Best field wins
- Multi-field search
- Improves relevance

BoolQuery (The Backbone)

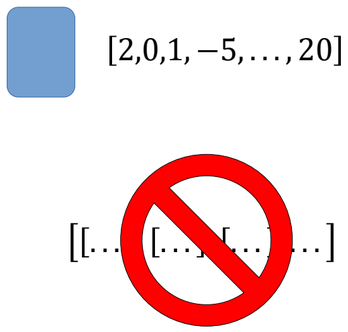
- must / filter / should / must_not
- Combines all queries
- Core production query

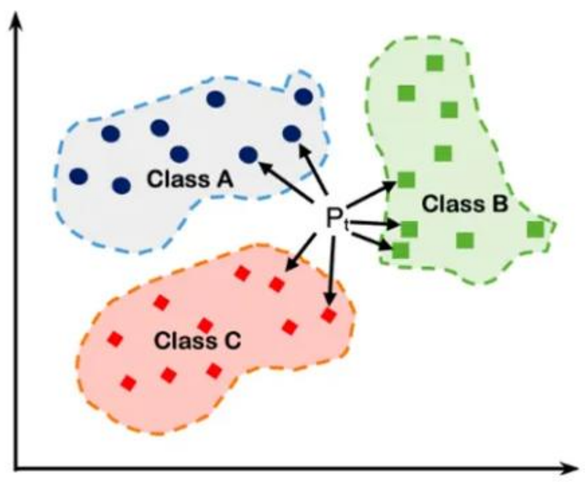
15 Dense vectors

Dense vector field type

- Stores dense vectors of numeric values.
- Use it if you have few or no **zero elements**.
- Does not support **aggregations** or **sorting**.
- It is not possible to store **multiple** values in one dense vector field.
- Use **kNN search** to retrieve the nearest vectors.
- Max size of a dense vector is **4096**.

Read more





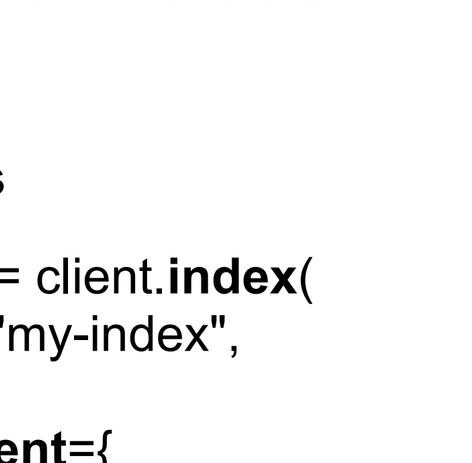
Medium article by Sachinsoni

Dense vector field type

Examples

```
response = client.index(
    index="my-index",
    id="1",
    document={
        "my_text": "example text",
        "my_vector": [0.5, 10, 6]
    }
)
```

```
response = client.index(
    index="my-index",
    id="2",
    document={
        "my_text": "example text",
        "my_vector": [[0.5, 10, 6], [1, 0, -2]]
    }
)
```



Dense vector field type

Important

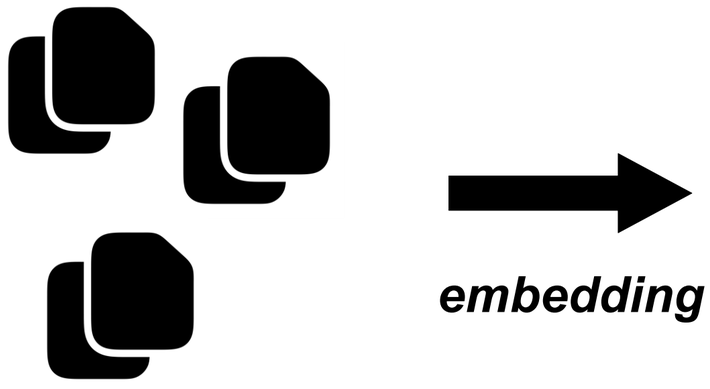
- You have to **manually** do the **mapping**.
- Elasticsearch **does not automatically** infer the mapping for dense vectors.
- It needs to know the **exact number of dimensions**.

```
response = es.indices.create(
    index="my_index",
    mappings={
        "properties": {
            "sides_length": {
                "type": "dense_vector",
                "dims": 4
            },
            "shape": {
                "type": "keyword"
            }
        }
    },
)
```

16 Embedding documents

Embedding documents

- **Embedding** transforms text into numerical vectors.
- **Deep learning models** are used to embed documents.
- These models preserve the **meaning** of the text.
- Use cases:
- Recommendation systems
- Retrieval-Augmented Generation (RAG)



$[2,0,1,-5,\ldots,20]$

$[0.54,-4.2,0,-0.6,\ldots,1]$

$[3,0.33,-.98,-1,\ldots,-1.1]$

Embedding documents

Deep learning models

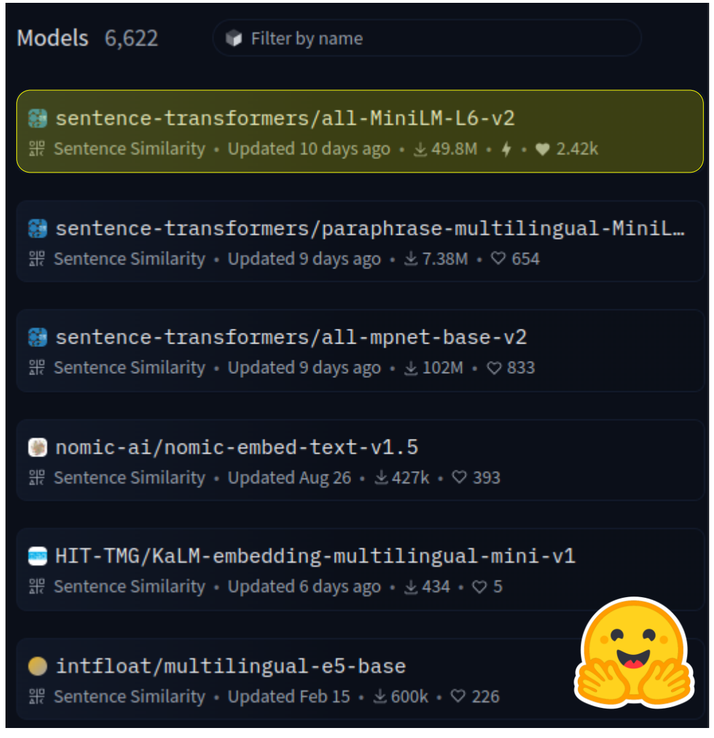
- **Closed** models.
- **Paid** (Pay what you use).
- No hardware required (**Cloud**).

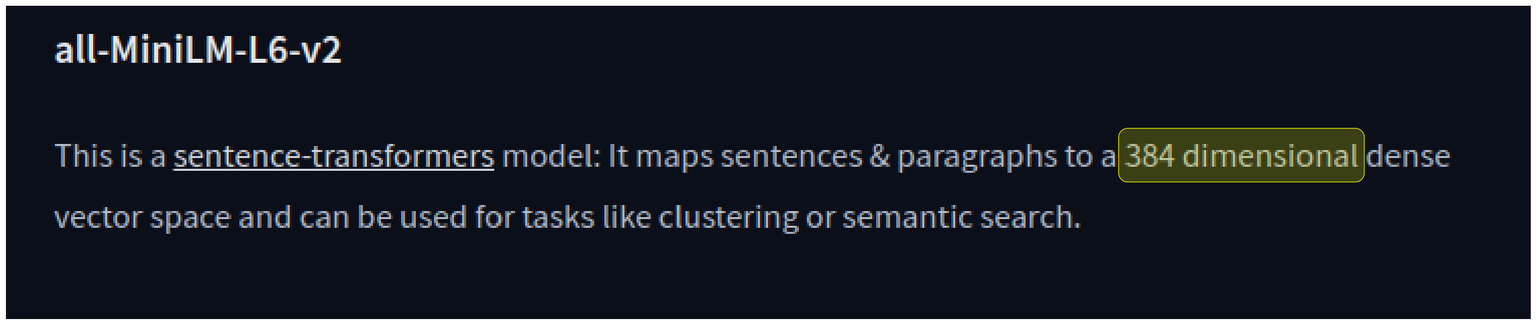
- **Open-source** models.
- **Free** to use.
- Hardware is required (**preferably GPU**).

Embedding documents

Embedding size

- Size of the dense vector.
- Larger sizes yield better embeddings.
- Common sizes include **384, 768, 1024**.

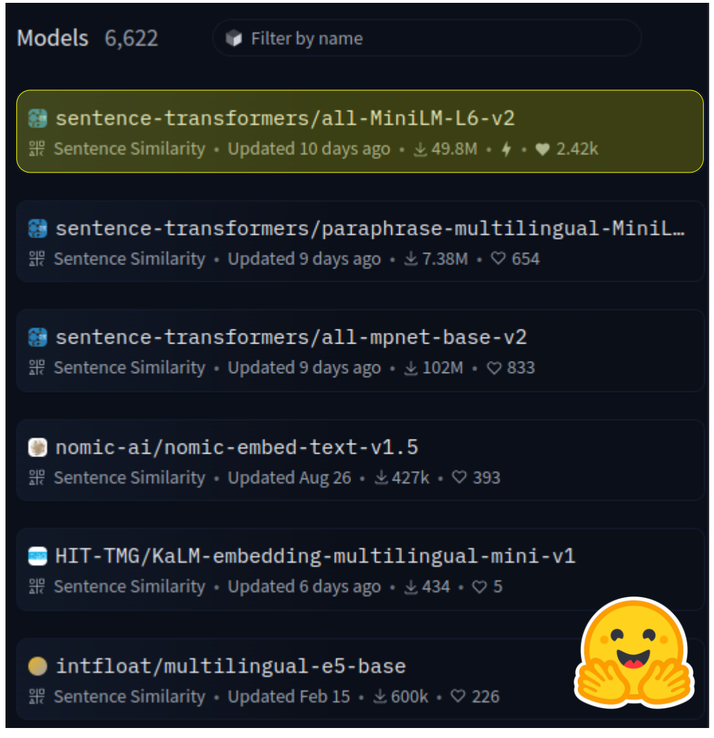




Embedding documents

Input size

- Size of the **input text** that the model can process.
- Text will be **truncated** if it exceeds the model's capacity.
- Common values include 256, 512 **tokens.**



Intended uses

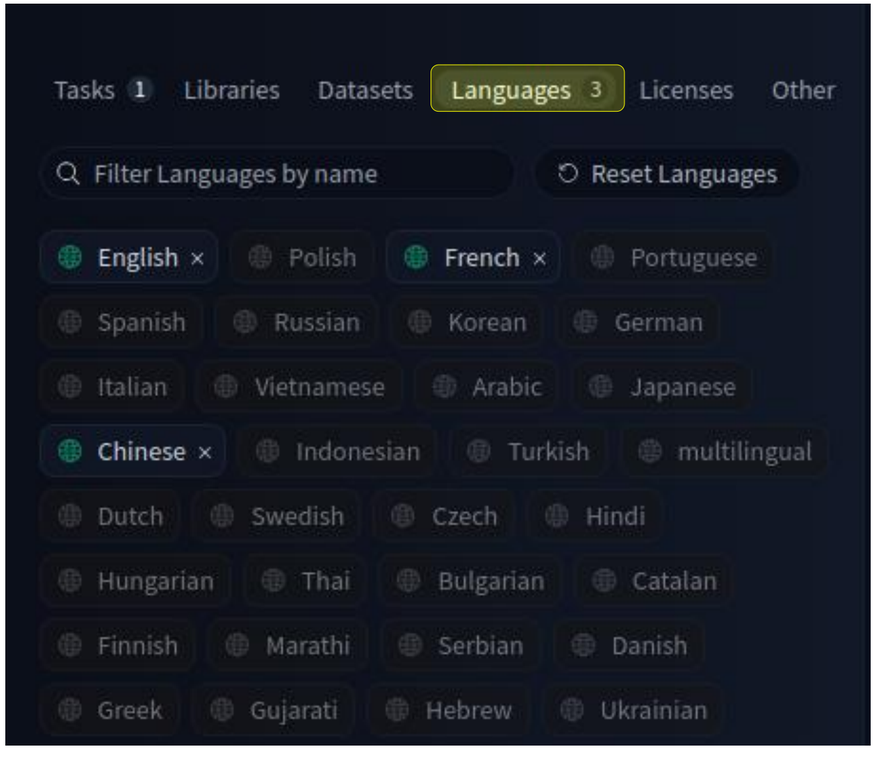
Our model is intended to be used as a sentence and short paragraph encoder. Given an input text, it outputs a vector which captures the semantic information. The sentence vector may be used for information retrieval, clustering or sentence similarity tasks.

By default, input text longer than 256 word pieces is truncated.

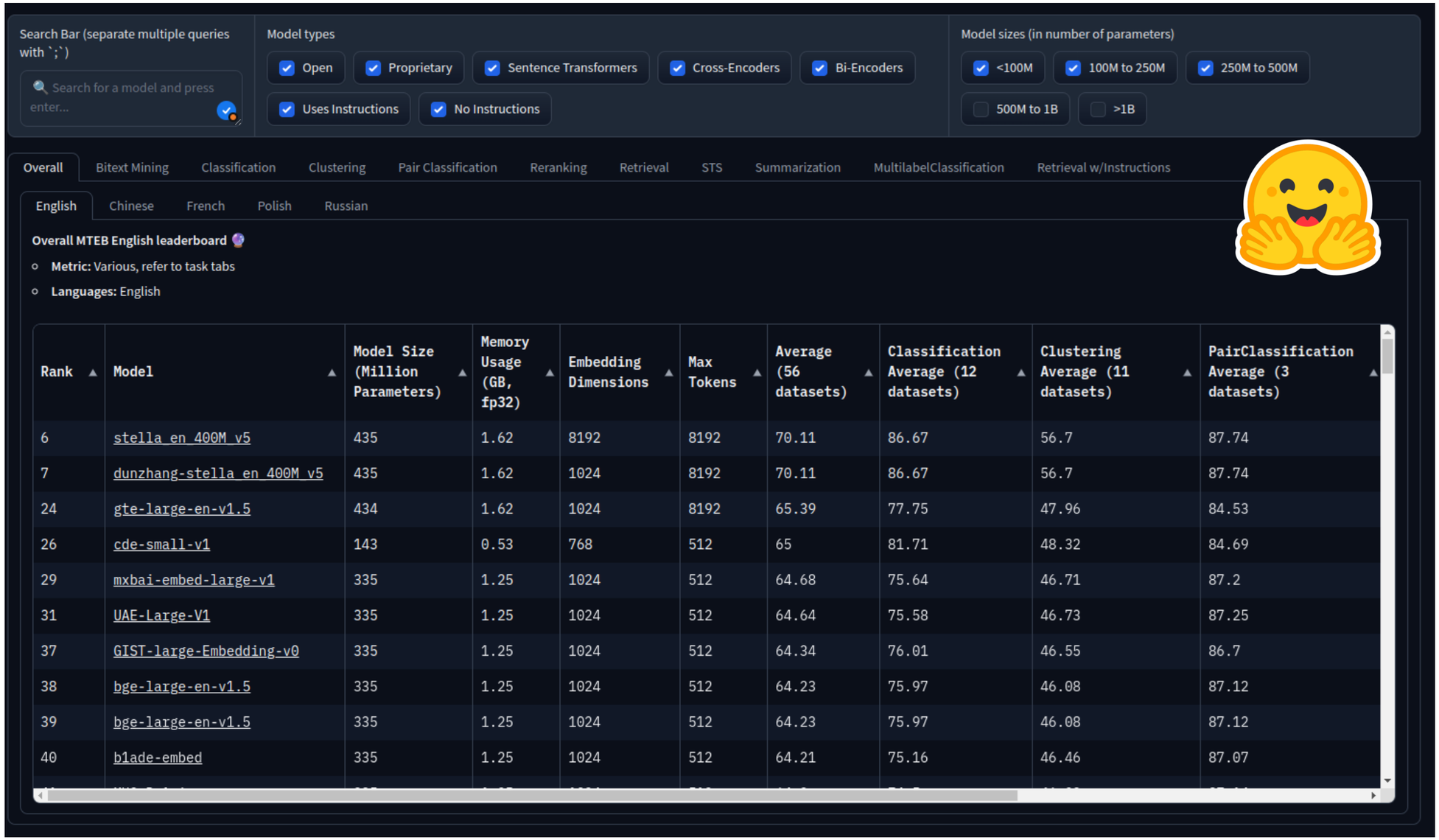
Embedding documents

Text language

- Some models can translate **specific languages**.
- Others support multiple languages and are **multilingual**.



Embedding documents



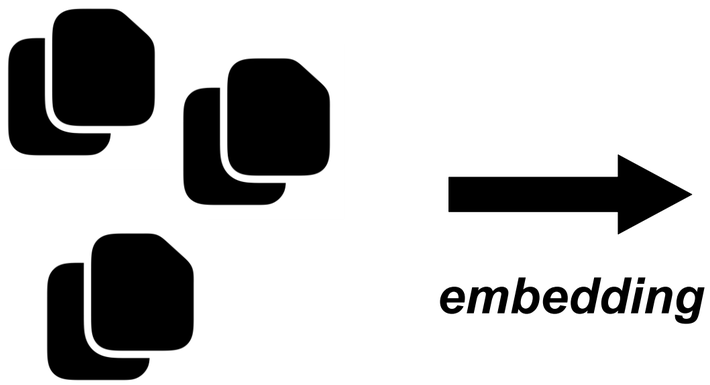
Rank	Model	Model Size (Million Parameters)	Memory Usage (GB, fp32)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	PairClassification Average (3 datasets)
6	stella_en_400M_v5	435	1.62	8192	8192	70.11	86.67	56.7	87.74
7	dunzhang-stella_en_400M_v5	435	1.62	1024	8192	70.11	86.67	56.7	87.74
24	gte-large-en-v1.5	434	1.62	1024	8192	65.39	77.75	47.96	84.53
26	cde-small-v1	143	0.53	768	512	65	81.71	48.32	84.69
29	mxbai-embed-large-v1	335	1.25	1024	512	64.68	75.64	46.71	87.2
31	UAE-Large-V1	335	1.25	1024	512	64.64	75.58	46.73	87.25
37	GIST-large-Embedding-v0	335	1.25	1024	512	64.34	76.01	46.55	86.7
38	bge-large-en-v1.5	335	1.25	1024	512	64.23	75.97	46.08	87.12
39	bge-large-en-v1.5	335	1.25	1024	512	64.23	75.97	46.08	87.12
40	b1ade-embed	335	1.25	1024	512	64.21	75.16	46.46	87.07

[Massive Text Embedding Benchmark (MTEB)]

17 KNN search

k-nearest neighbor (kNN) search

- How do we search for embedded documents?
- We use the **kNN** search for fields mapped as **dense vectors**.
- Important, you can't use the **query** parameter in this case.



$[2,0,1,-5,\ldots,20]$

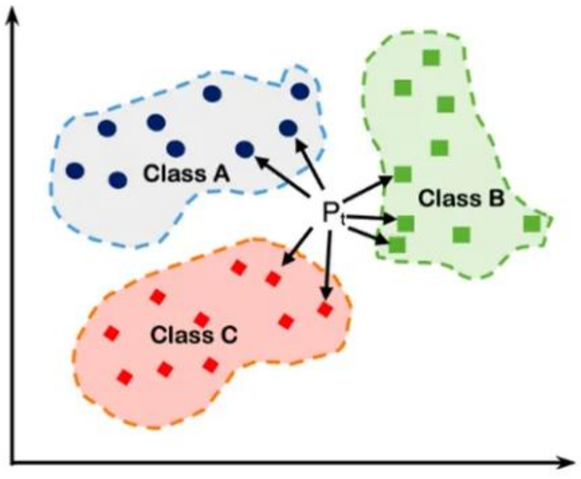
$[0.54,-4.2,0,-0.6,\ldots,1]$

$[3,0.33,-.98,-1,\ldots,-1.1]$

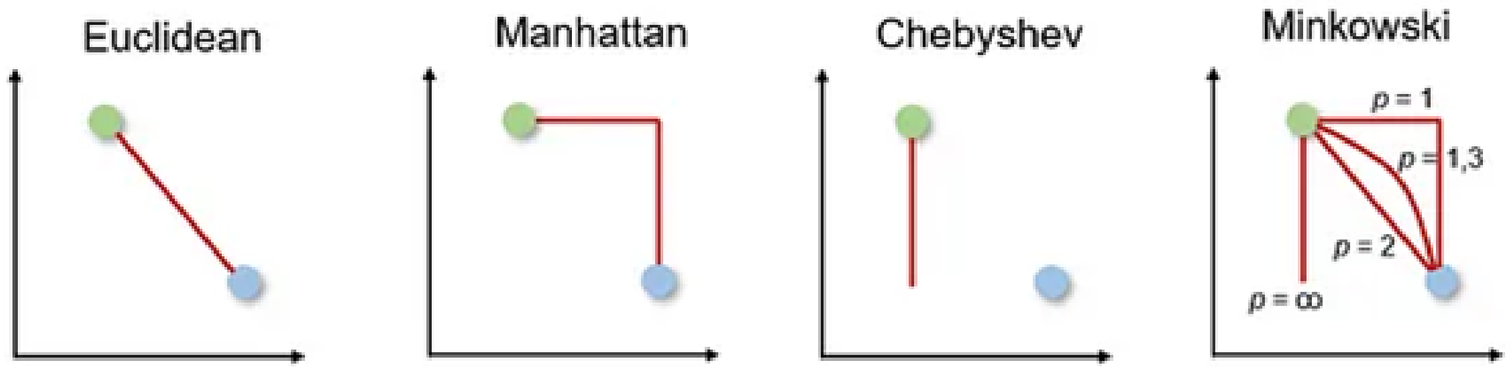
k-nearest neighbor (kNN) search

- The **kNN** algorithm is used for **classification** and **regression** tasks.
- It **classifies new data points** by comparing them with the **k nearest points** from the training data.
- Commonly distances metrics: **Euclidean**, **Manhattan** or **Minkowski**.
- This algorithm is **simple** and effective.

Read more

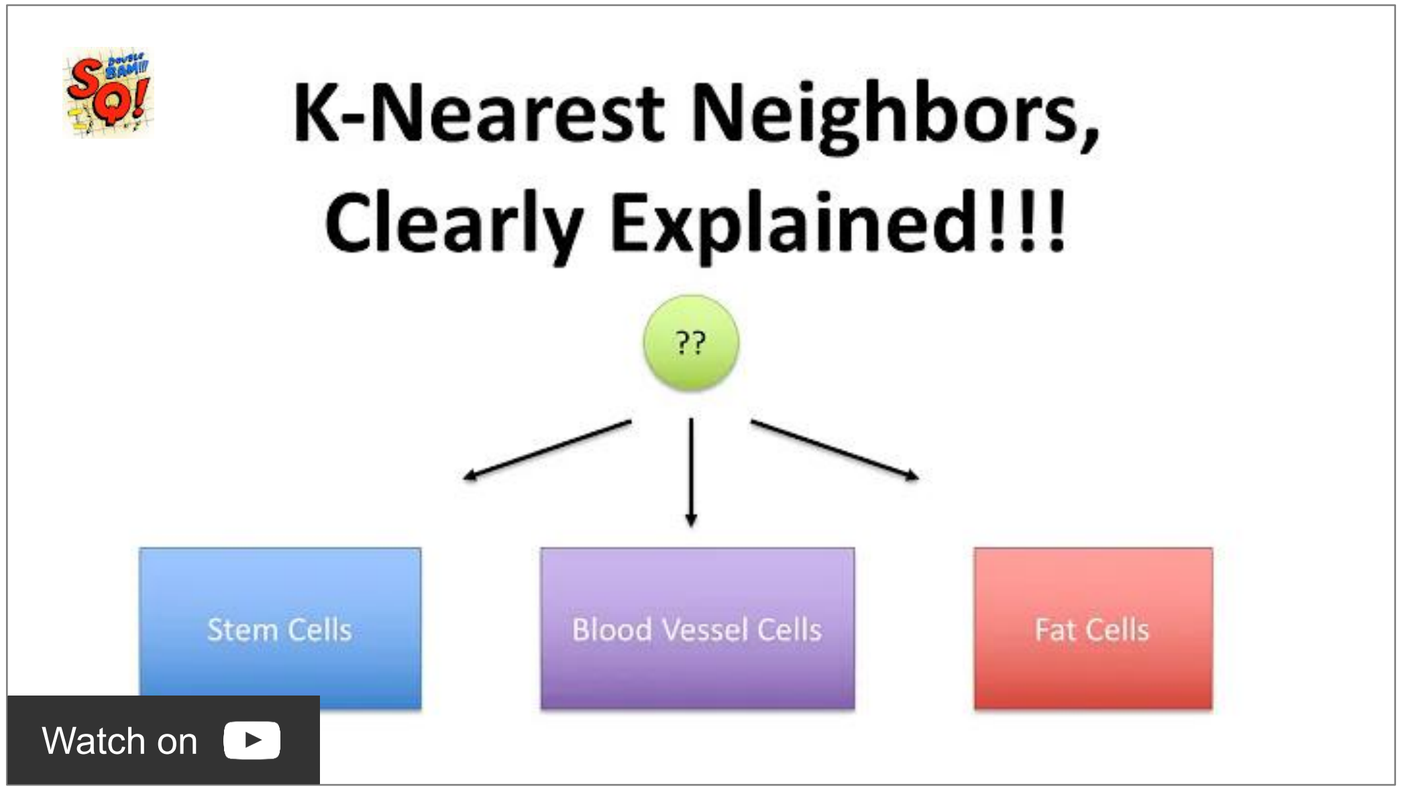


Medium article by Sachinsoni



Medium article by Dancker

k-nearest neighbor (kNN) search



k-nearest neighbor (kNN) search

Example

ⓘ Should be a **dense_vector**

```
results = es.search(
        knn={
            'field': 'embedding',
            'query_vector': es.get_embedding(parsed_query),
            'num_candidates': 50,
            'k': 10,
        }
    )
```

k-nearest neighbor (kNN) search

Example

```
results = es.search(
        knn={
            'field': 'embedding',
            'query_vector': es.get_embedding(parsed_query),
            'num_candidates': 50,
            'k': 10,
        }
)
```

ⓘ The query should be a vector.

k-nearest neighbor (kNN) search

Example

```
results = es.search(
        knn={
            'field': 'embedding',
            'query_vector': es.get_embedding(parsed_query),
            'num_candidates': 50,
            'k': 10,
        }
)
```

ⓘ Retrieves 50 **potentially relevant** documents **before applying distance calculations** to select the k best documents.

k-nearest neighbor (kNN) search

Example

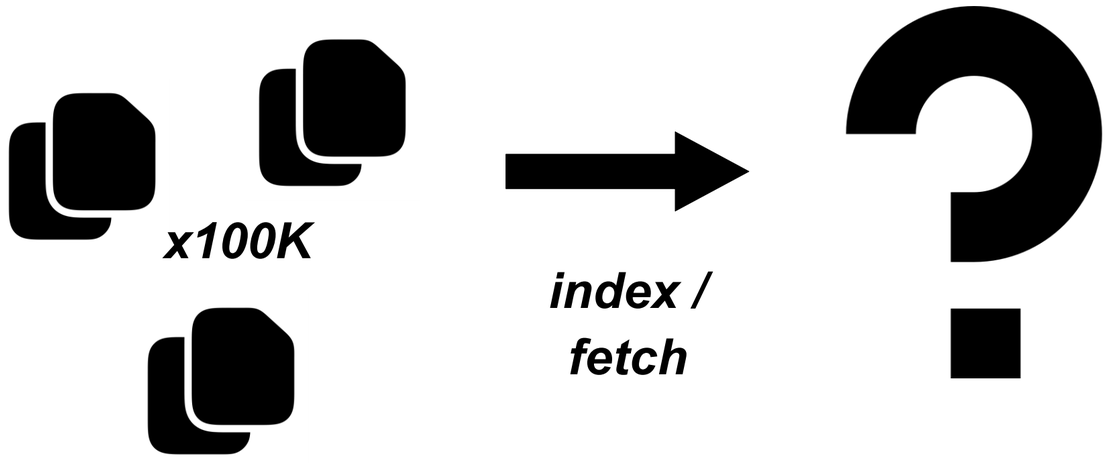
```
results = es.search(
        knn={
            'field': 'embedding',
            'query_vector': es.get_embedding(parsed_query),
            'num_candidates': 50,
            'k': 10,
        }
)
```

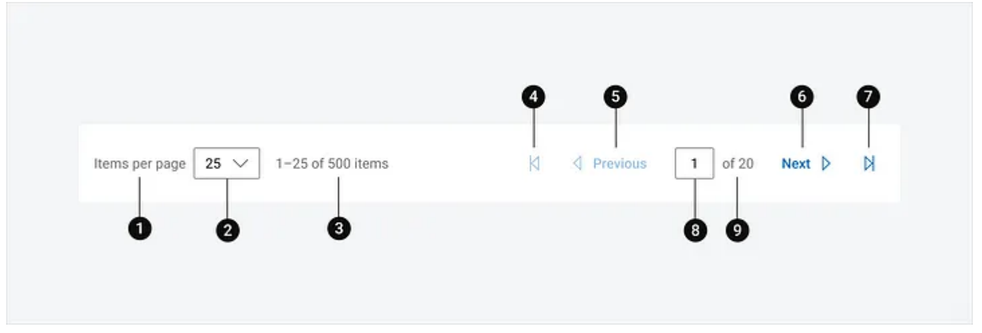
ⓘ Returns up to 10 documents that match your query.

18 Deep pagination

Deep pagination

- **Indexing / fetching** all documents at once is **inefficient** and **slow**.
- **Pagination:**
  - Retrieves data in small chunks from large indexes.
  - Improves performance and efficiency.
  - Fast search experience.
  - Cost effective.





Medium article by Dayanand

Deep pagination

Pagination methods

.from/size: commonly used for paginating **smaller datasets**.

.search_after**:** offers more efficient deep pagination for **large datasets**.

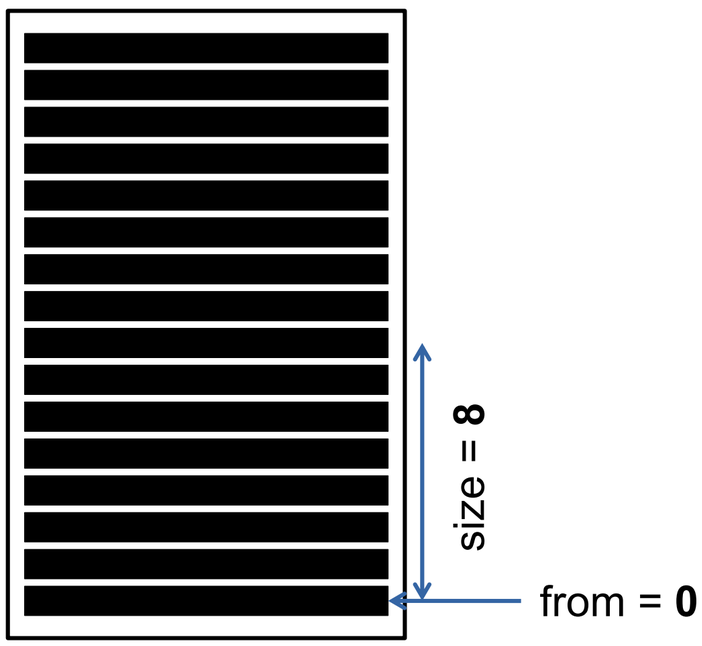
Deep pagination

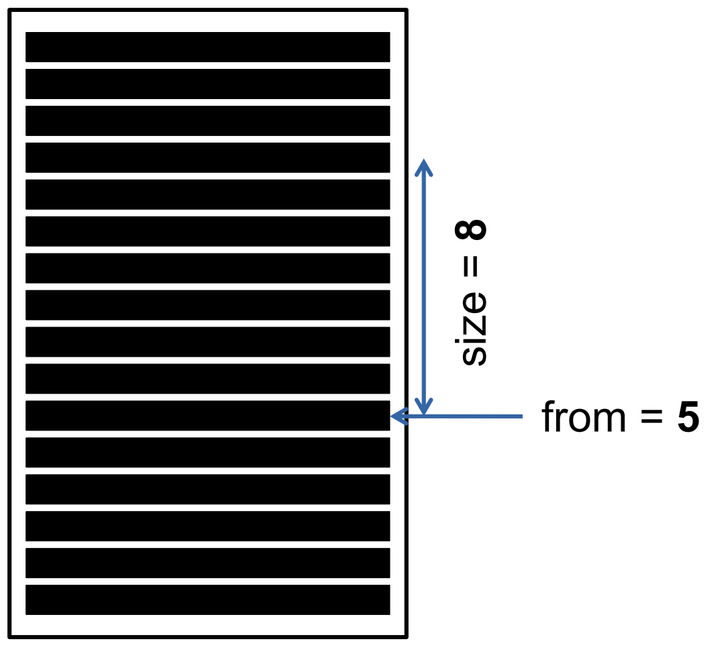
Pagination methods

.from/size: commonly used for paginating **smaller datasets**.

***.search_after*: offers more efficient deep pagination for **large datasets**.**

Read more





ⓘ Note:

.from/size is limited to **10k** results.

.Requires a lot of memory for deep pages.

.Not suitable for larger indexes.

Deep pagination

Pagination methods

.from/size: commonly used for paginating **smaller datasets**.

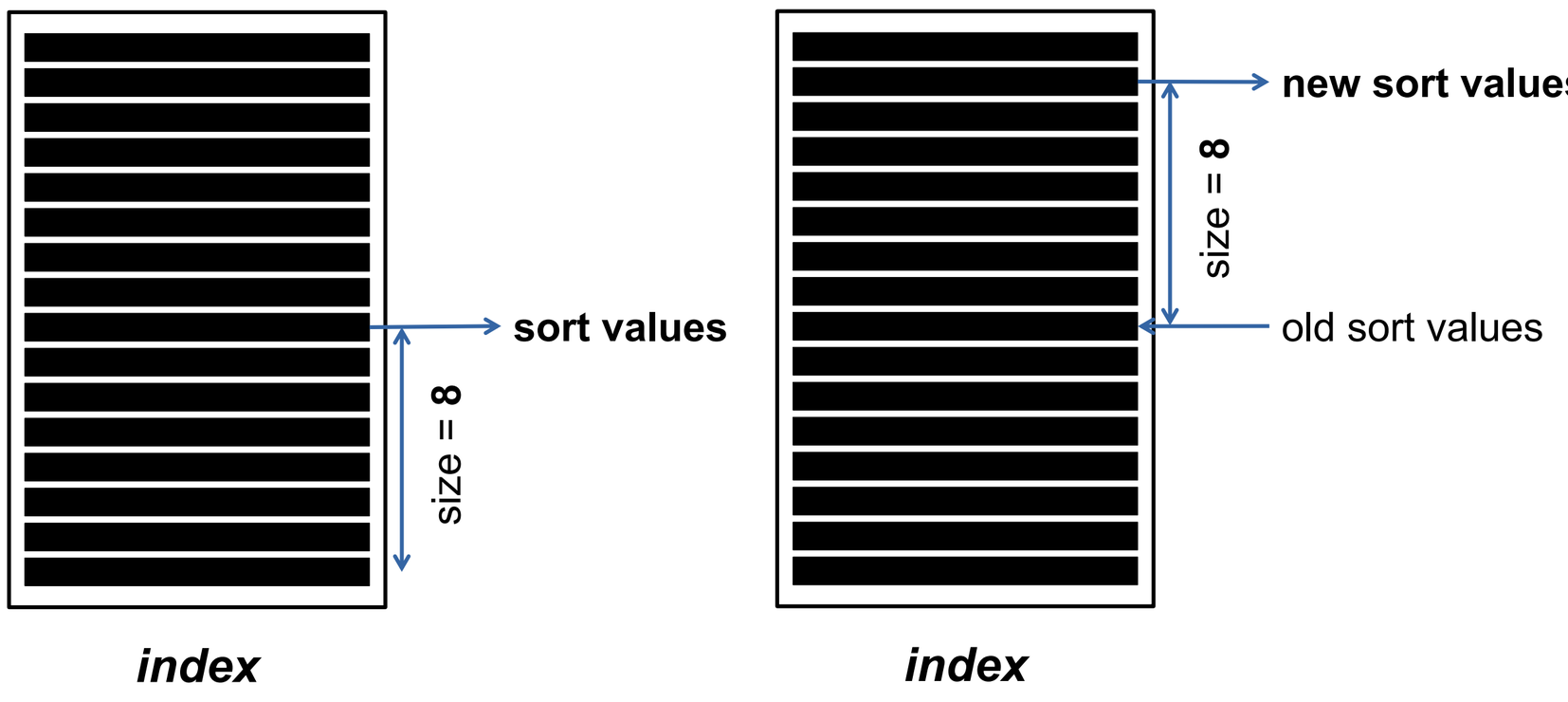
.*search_after*: offers more efficient deep pagination for **large datasets**.

Read more

Sortable fields

.timestamp

.id



Deep pagination

Pagination methods

.from/size: commonly used for paginating **smaller datasets**.

.*search_after*: offers more efficient deep pagination for **large datasets**.

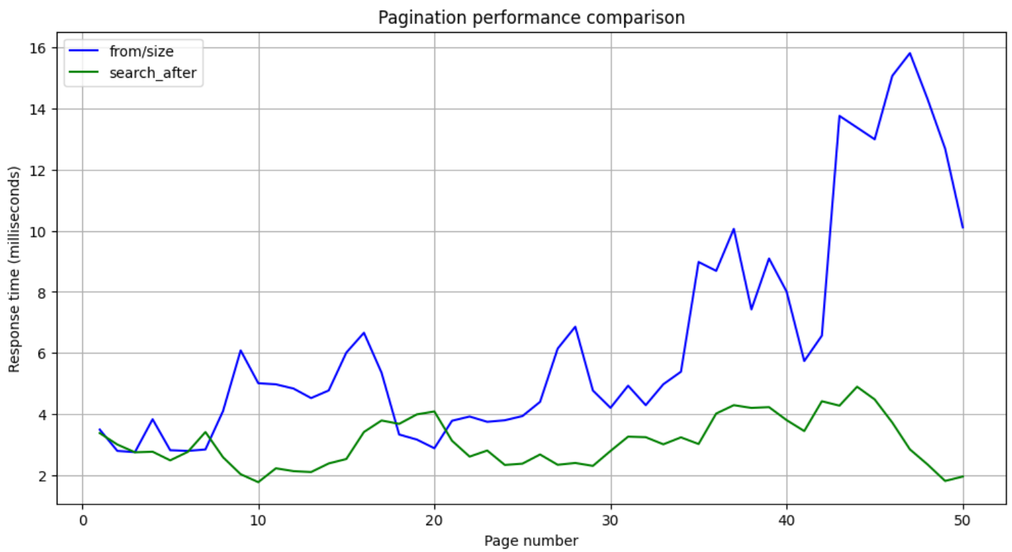
ⓘ Note:

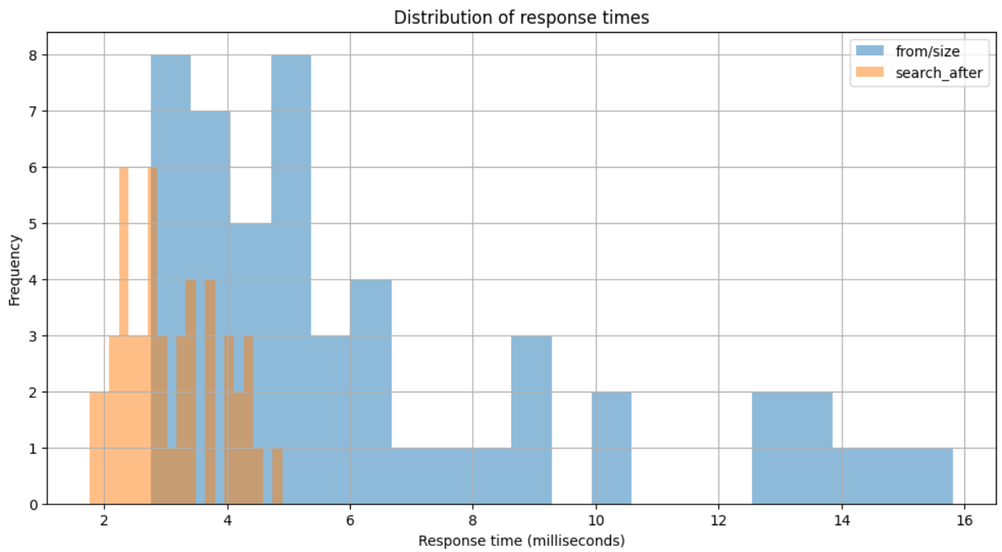
- The ***search_after*** method is not constrained by the **10k** result limit.
- It does not utilize an offset (i.e., the **from** parameter).
- Results **must be sorted** by fields such as ID or timestamp.
- Uses a **pointer** derived from the sort values of the last document on the previous page.
- This approach prevents the **skipping** of documents.
- It is particularly beneficial for handling **larger indexes**.

Deep pagination

Comparison

Below are the results obtained after attempting to retrieve **10,000** documents using the following parameters (**size = 200, iterations = 50**)





Deep pagination

Comparison

	from/size	search_after
Average time (ms)	6.417	**3.072**
Maximum time (ms)	15.812	**4.896**
Minimum time (ms)	2.757	**1.772**

	from/size	search_after
Performance degradation	2.9x	**0.58x**

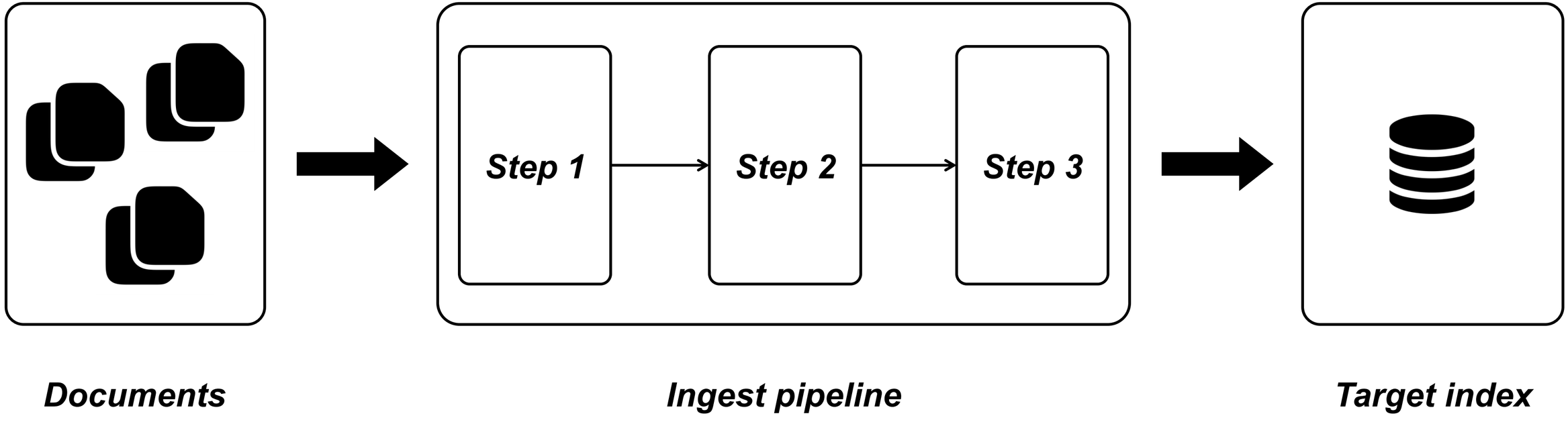
ⓘ Performance degradation is calculated like this (last page time / first page time)

19 Ingest pipelines

Ingest pipelines

- You can perform **transformations** on data **before indexing**.
- Common transformations: remove fields, lowercase text, remove HTML tags, and more.

Read more



Ingest pipelines

- We use the **ingest API** to:
- Create or update a pipeline.


```
response = client.ingest.put_pipeline(
    id="my-pipeline",
    description="A description",
    processors=[
        {
            "set": {
                "description": "A description",
                "field": "field",
                "value": “value”
            }
        },
        {
            "lowercase": {
                "field": "field"
            }
        }
    ],
)
```

Ingest pipelines

- We use the **ingest API** to:
- Simulate a pipeline.

```
response = client.ingest.simulate(
    id="my-pipeline",
    docs=[
        {
            "_index": "index",
            "_id": "id",
            "_source": {
                "foo": "bar"
            }
        },
        {
            "_index": "index",
            "_id": "id",
            "_source": {
                "foo": "rab"
            }
        }
    ],
)
```

OR

```
response = client.ingest.simulate(
    pipeline={
        "processors": [
            {
                "lowercase": {
                    "field": "field"
                }
            }
        ]
    },
    ...
)
```

Ingest pipelines

- We use the **ingest API** to:
- Delete a pipeline.

```
response = client.ingest.delete_pipeline(
    id="my-pipeline",
)
```

Ingest pipelines

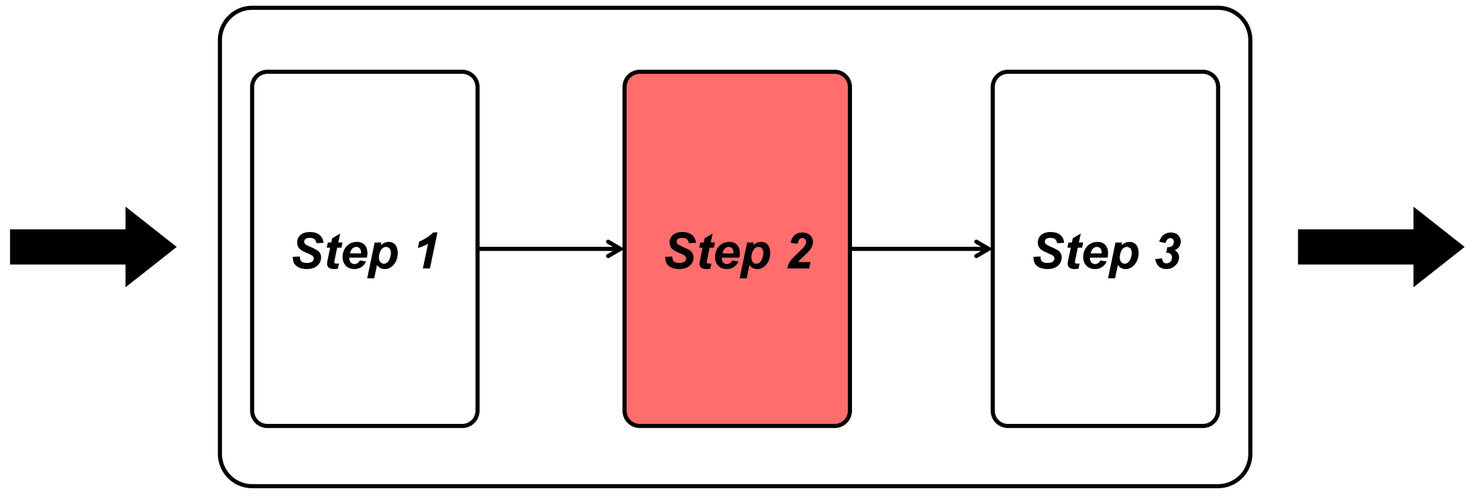
- We use the **ingest API** to:
- Get a pipeline.

```
response = client.ingest.get_pipeline(
    id="my-pipeline",
)
```

Ingest pipelines

- Pipelines **can fail**. You can either **ignore** the failure or **handle** it.
- If you ignore the failure, the pipeline **will skip over the failed steps**.

```
response = client.ingest.put_pipeline(
    id="my-pipeline",
    processors=[
        {
            "rename": {
                "description": "A description",
                "field": "field",
                "ignore_failure": True
            }
        }
    ],
)
```

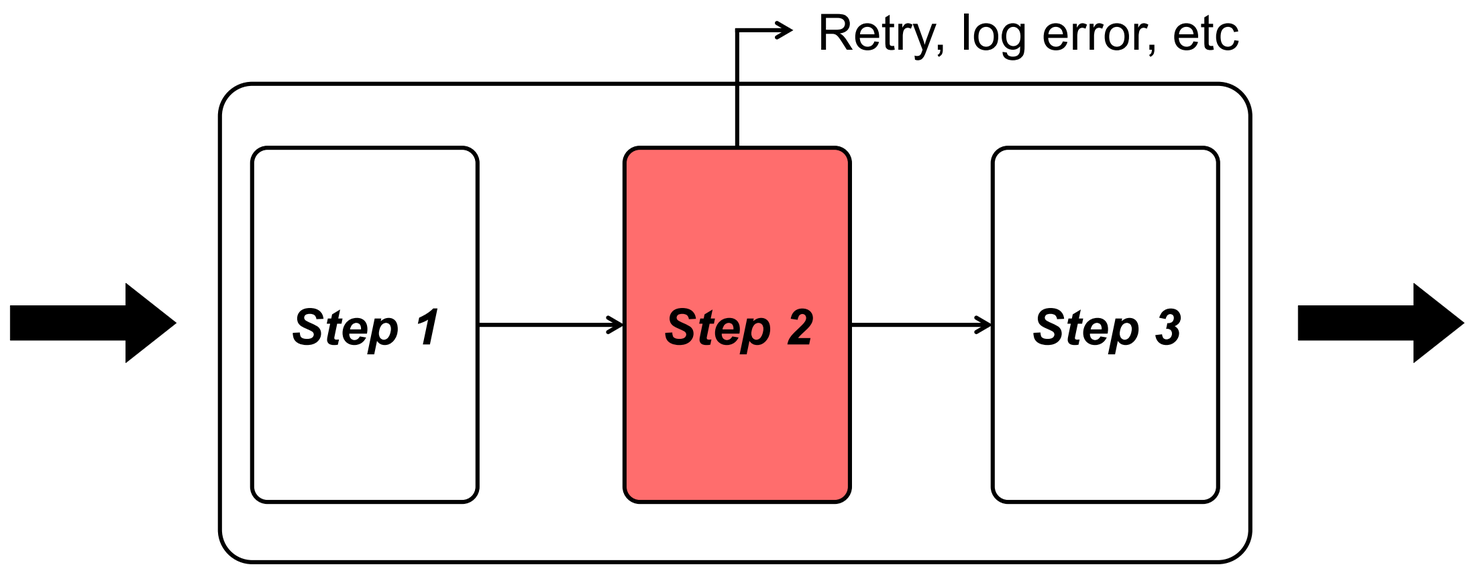


Ingest pipeline

Ingest pipelines

- Pipelines **can fail**. You can either **ignore** the failure or **handle** it.
- Specify custom error-handling steps with ***on_failure***.

```
response = client.ingest.put_pipeline(
    id="my-pipeline",
    processors=[
        {
            "rename": {
                "description": "A description",
                "field": "field",
                "on_failure": [...]
            }
        }
    ],
)
```

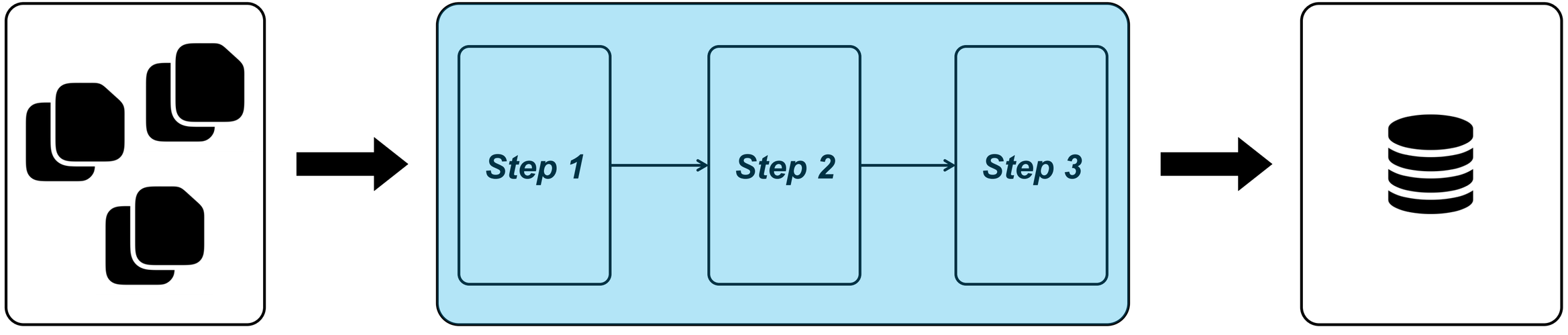


Ingest pipeline

20 Ingest processors

Ingest processors

- Common transformations: remove fields, lowercase text, remove HTML tags, and more.



Ingest processors

Ingest processors by category

- Ingest processors are organized into 5 categories.

Data enrichment

- Append
- Inference
- Attachment
- ...

Array/JSON handling

- For each
- JSON
- Sort

Data transformation

- Convert
- Rename
- Set
- HTML strip
- Lowercase / Uppercase
- Trim
- Split
- ...

Data Filtering

- Drop
- Remove

Pipeline handling

- Fail
- Pipeline

21 Filters in depth

Filters in depth

- When searching in Elasticsearch, you can use either **query context or filter context.**

Query → **Query context** → Score

How well does this document match this query clause ?

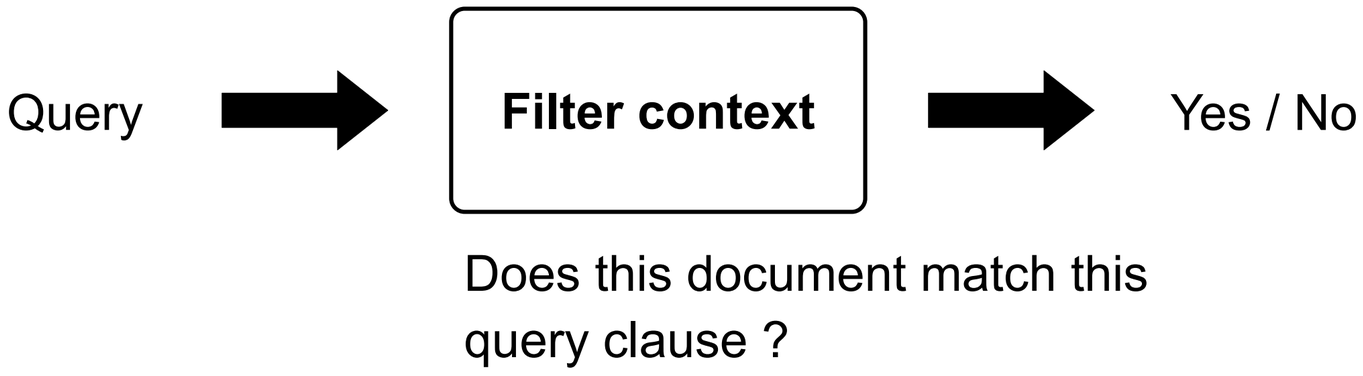
Query → **Filter context** → Yes / No

Does this document match this query clause ?

Filters in depth

Why use the filter context ?

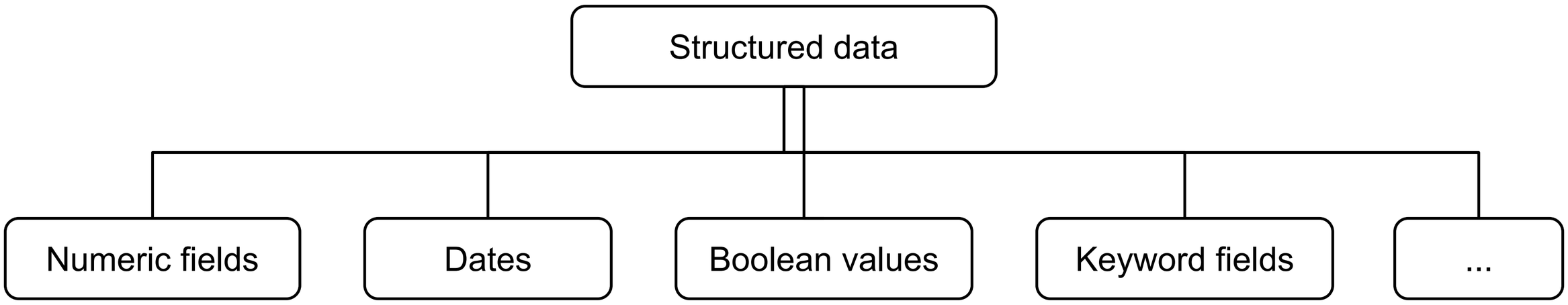
- **Binary matching**.
- **No score** is needed.
- Filters **execute faster** than queries (no score is computed).
- Filters consume **less CPU resources**.



Filters in depth

Use cases

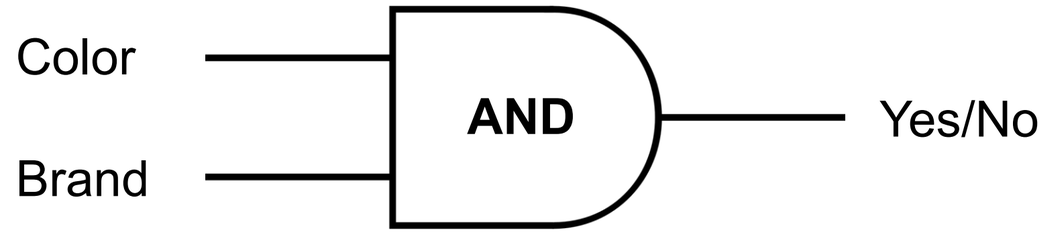
Filters are effective for querying structured data.



Filters in depth

Example 1

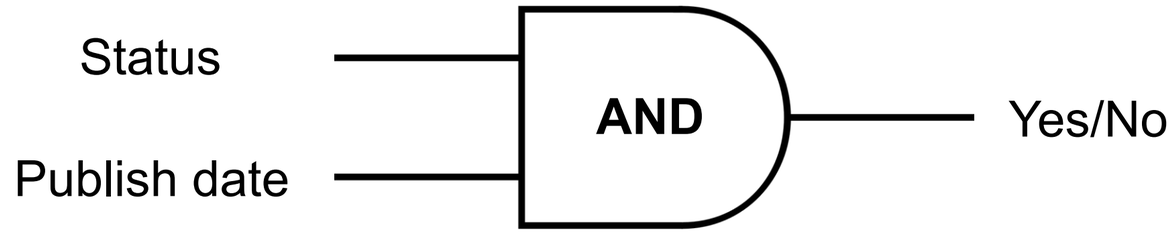
```
response = client.search(
    index="phones",
    query={
        "bool": {
            "filter": [
                {
                    "term": {
                        "color": "black"
                    }
                },
                {
                    "term": {
                        "brand": "samsung"
                    }
                }
            ]
        }
    },
)
```



Filters in depth

Example 2

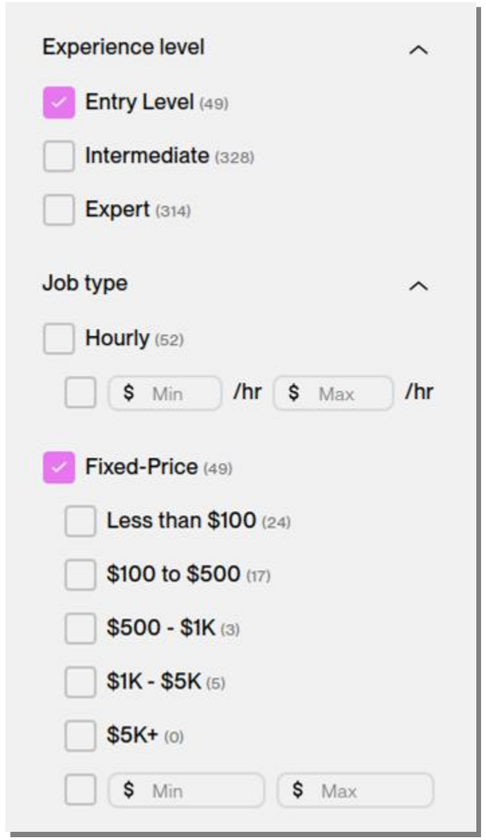
```
response = client.search(
    query={
        "bool": {
            "filter": [
                {
                    "term": {
                        "status": "published"
                    }
                },
                {
                    "range": {
                        "publish_date": {
                            "gte": "2015-01-01",
                            "lte": "2015-02-01"
                        }
                    }
                }
            ]
        }
    },
)
```



Filters in depth

Post filters

- Applies filters after **aggregations** are calculated.
- Does not affect aggregations.
- Only filters the search results.
- ***Let you narrow down what users see without limiting what they can choose from.***



Filters in depth

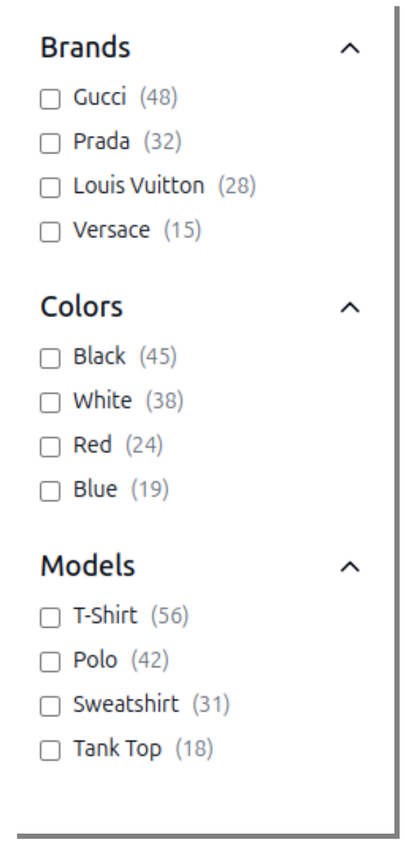
Example

```
response = client.search(
    index="shirts",
    query={
        "bool": {
            "filter": {
                "term": {
                    "brand": "gucci"
                }
            }
        }
    },
    ...
)
```

```
response = client.search(
    aggs={
        "colors": {
            "terms": {
                "field": "color"
            }
        },
        "color_red": {
            "filter": {
                "term": {
                    "color": "red"
                }
            },
            "aggs": {
                "models": {
                    "terms": {
                        "field": "model"
                    }
                }
            }
        },
    ...
)
```

```
response = client.search(
    post_filter={
        "term": {
            "color": "red"
        }
    },
)
```

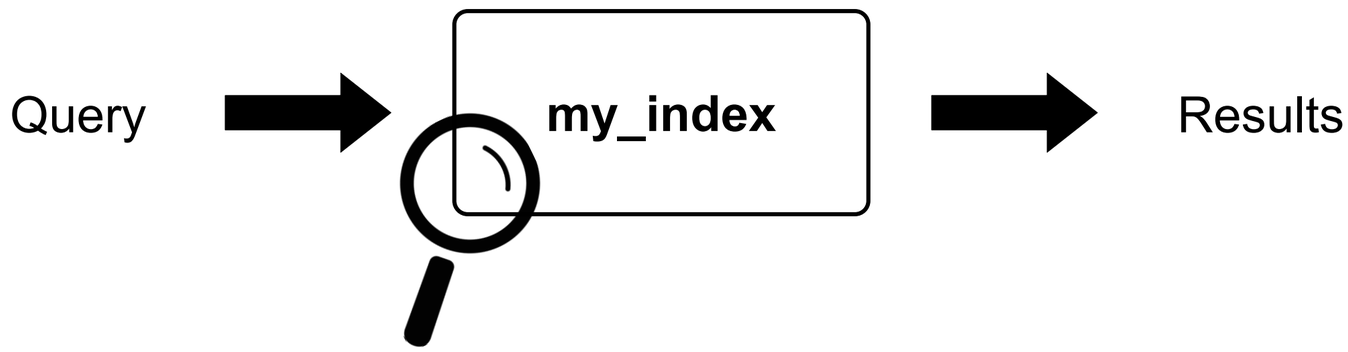
Read more



22 SQL search API

SQL Search API

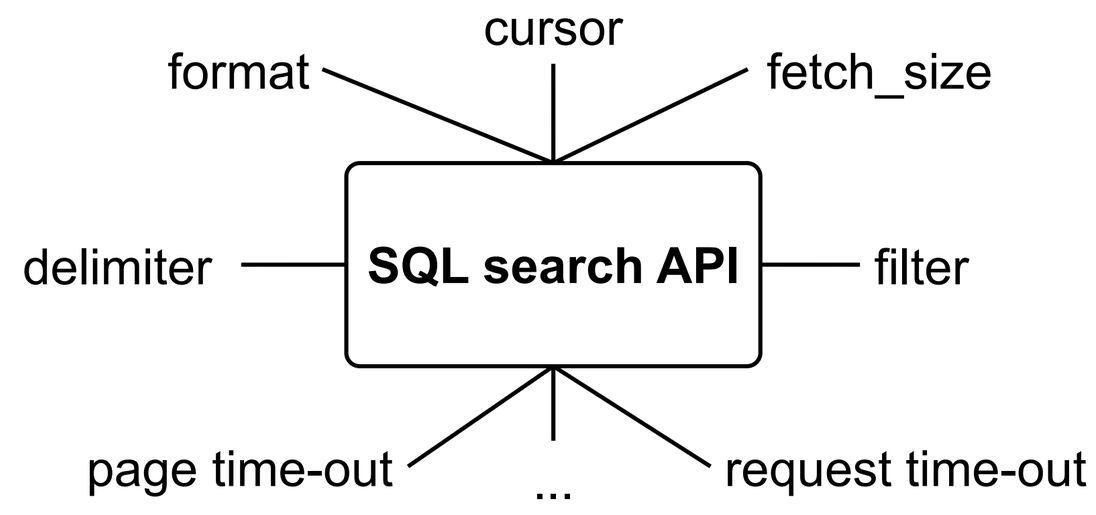
- We used **Query DSL** to search for documents.
- An alternative method for searching documents is the **SQL Search API**.



SQL Search API

The **SQL search API** supports numerous parameters.

Read more



SQL Search API

Example

```
response = client.sql.query(
    format="txt",
    query="SELECT * FROM library ORDER BY page_count DESC LIMIT 5",
)
```

field

Index

size

SQL Search API

The available response formats are:

- .CSV
- .JSON
- .TSV
- .TXT
- .YAML
- .CBOR
- .SMILE
- .Binary formats

```
response = client.sql.query(
    format="txt",
    query="SELECT * FROM library ORDER BY page_count DESC LIMIT 5",
)
```

SQL Search API

Pagination

```
response = client.sql.query(
    format="txt",
    cursor="sDHOSBDISBXMLK…",     ?
)
```


```
{
  "rows" : [
    ["Dan Simmons",       "Hyperion",            482,  "1989-05-26T00:00:00.000Z"],
    ["Iain M. Banks",     "Consider Phlebas",    471,  "1987-04-23T00:00:00.000Z"],
    ["Neal Stephenson",   "Snow Crash",          470,  "1992-06-01T00:00:00.000Z"],
    ["Frank Herbert",     "God Emperor of Dune", 454,  "1981-05-28T00:00:00.000Z"],
    ["Frank Herbert",     "Children of Dune",    408,  "1976-04-21T00:00:00.000Z"]
  ],
  "cursor" : "sDXF1ZXJ5QW5kRmV0Y2gBAAAAAAAAAAEWODRMaXBUaVlRN21iTlRyWHZWYUdrdw==:BAFmBm
}
```

Original example

SQL Search API

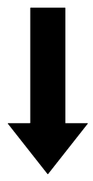
Filtering

```
response = client.sql.query(
    format="txt",
    query="SELECT * FROM library ORDER BY page_count DESC",
    filter={
        "range": {
            "page_count": {
                "gte": 100,
                "lte": 200
            }
        }
    },
    fetch_size=5,
)
```

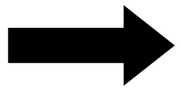
SQL Search API

SQL Translate API

```
response = client.sql.translate(
    query="SELECT * FROM library ORDER BY page_count
DESC",
    fetch_size=10,
)
```



SQL Translate API



```
{
  "size": 10,
  "_source": false,
  "fields": [{"field": "author"}, ...],
  "sort": [
    {
      "page_count": {
        "order": "desc",
      }
    }
  ],
  "track_total_hits": -1
}
```

SQL Search API

SQL Limitations

SQL Limitations

Large queries may throw `ParsingException`

edit

Extremely large queries can consume too much memory during the parsing phase, in which case the Elasticsearch SQL engine will abort parsing and throw an error. In such cases, consider reducing the query to a smaller size by potentially simplifying it or splitting it into smaller queries.

Nested fields in `SYS COLUMNS` and `DESCRIBE TABLE`

edit

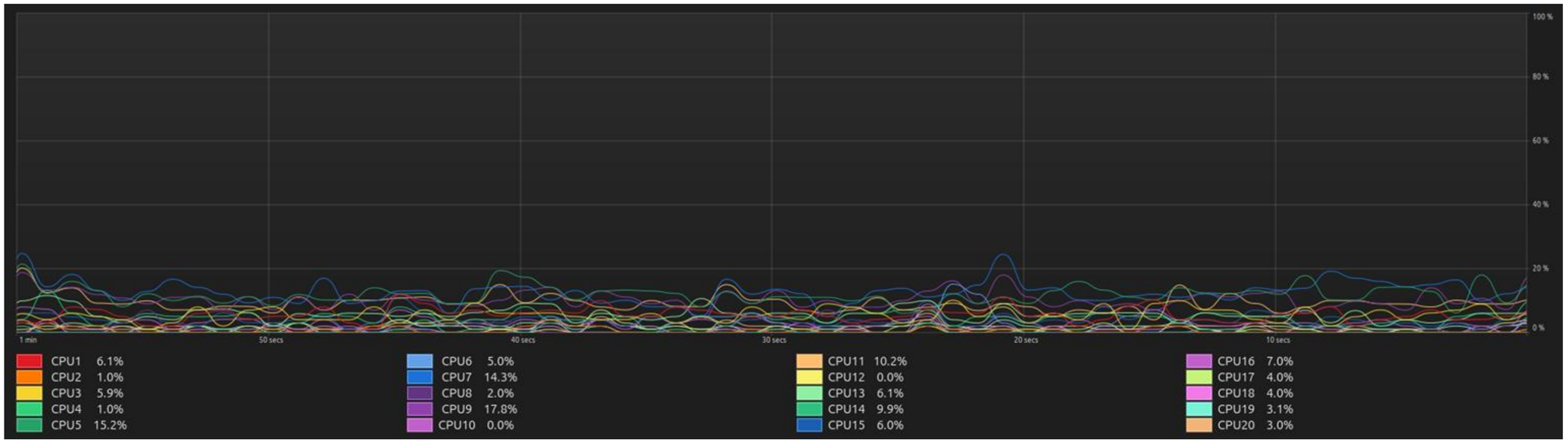
Elasticsearch has a special type of relationship fields called `nested` fields. In Elasticsearch SQL they can be used by referencing their inner sub-fields. Even though `SYS COLUMNS` in non-driver mode (in the CLI and in REST calls) and `DESCRIBE TABLE` will still display them as having the type `NESTED`, they cannot be used in a query. One can only reference its sub-fields in the form:

---

23 Time Series Data Stream

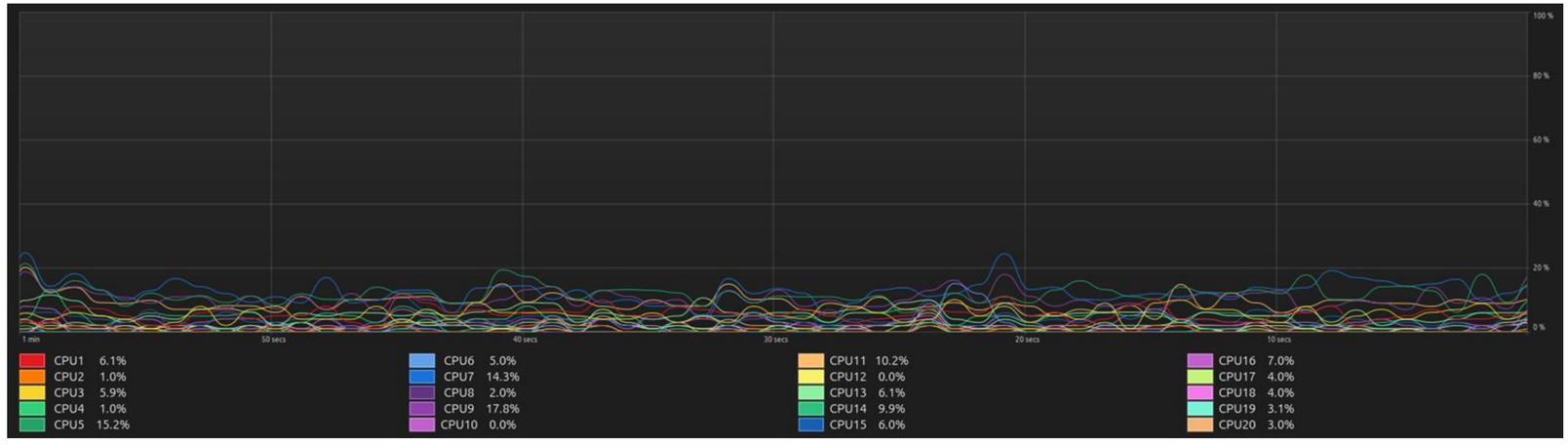
Time Series Data Stream

- Time series data refers to data points **ordered by time**.
- Data is collected at **regular intervals**.
- Example: CPU usage over time.



Time Series Data Stream

- **Managing** time series data is challenging.
- The data can **grow rapidly** (high frequency measurements)
- How to store this large volume efficiently?
- Deciding which old data to **keep** and when to **delete** it.



Time Series Data Stream

Why use Elasticsearch for time series data?

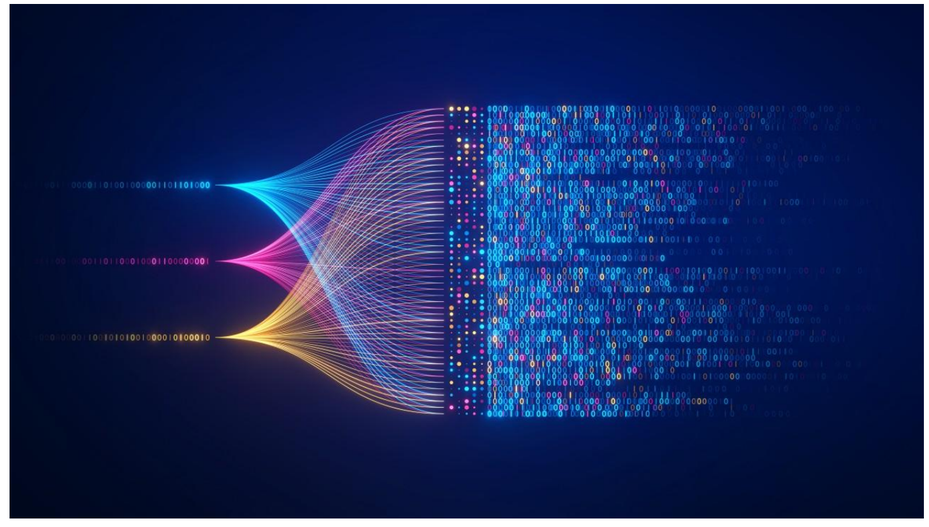
.Elasticsearch can handle **massive volumes of data**.

.Supports **real-time** data ingestion and querying.

.**Analyze** time-series.

Read more

Original post



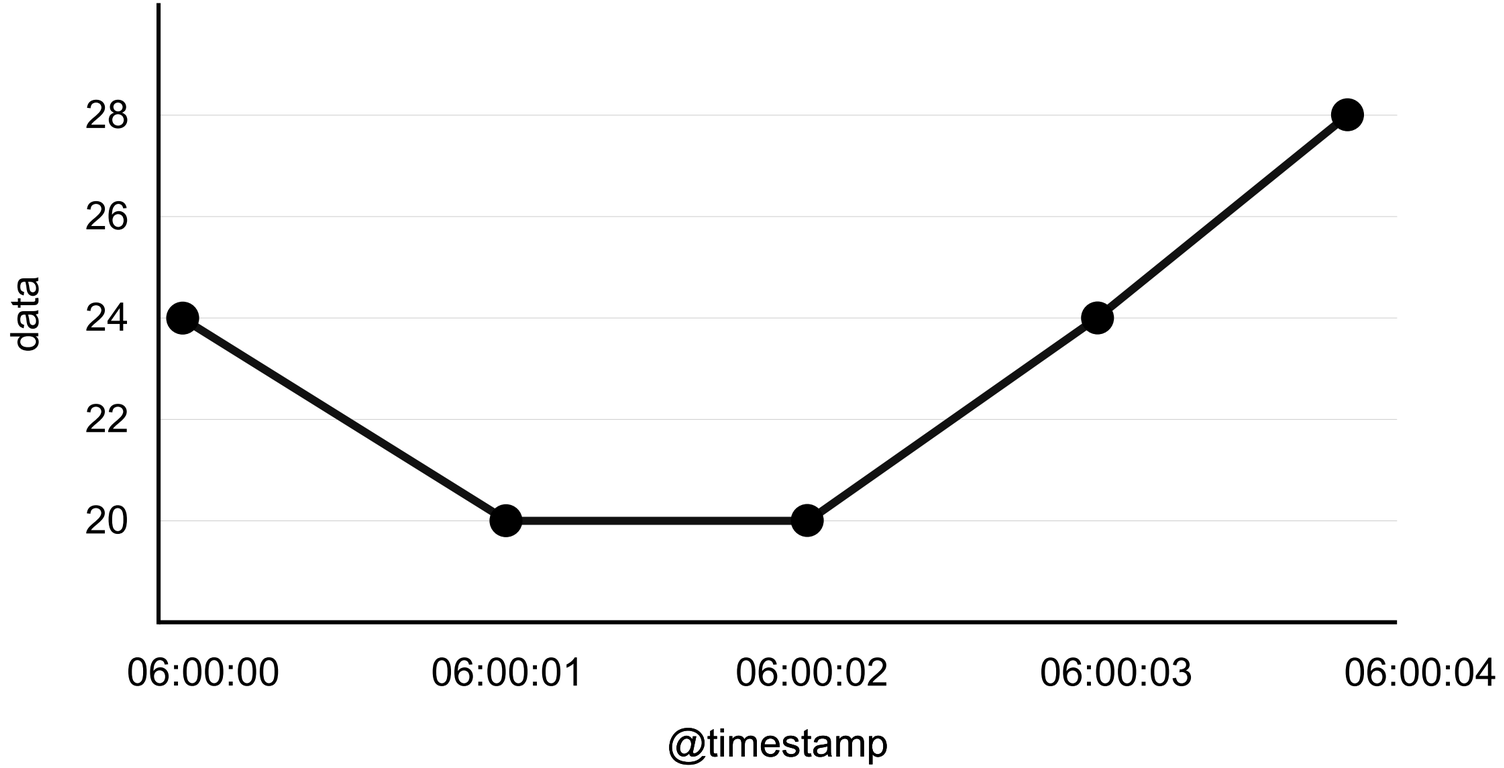
Original post

Time Series Data Stream

Time series data structure

.Each **data point** is a document

.Each **document** contains the **timestamp** field and the **data**.



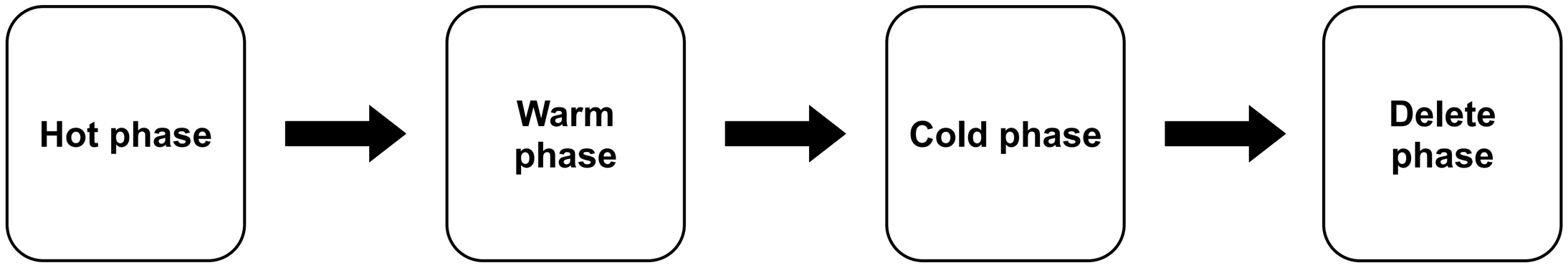
Time Series Data Stream

Index Lifecycle Management (ILM)

.ILM automates the **rollover** and **management** of indices.

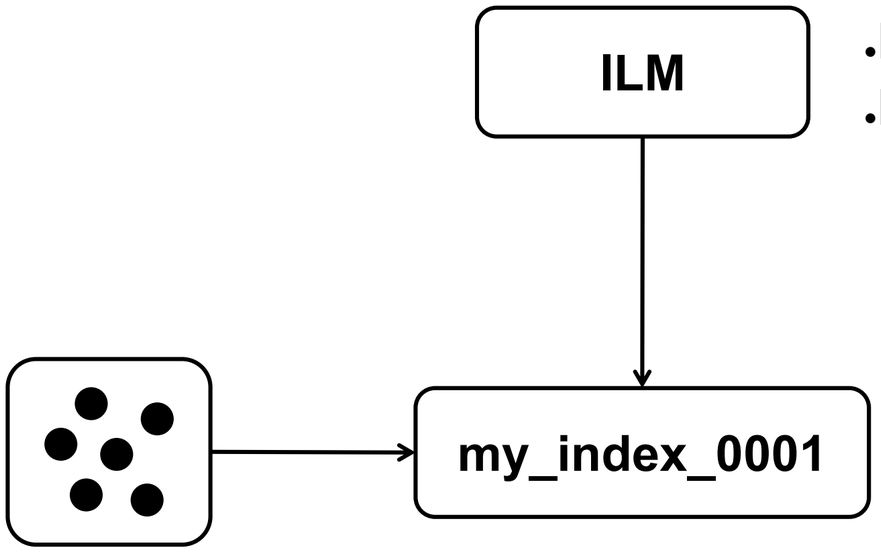
.**Benefits**: Storage optmization, automated data retention, efficient management of index size.

.Phases of ILM:



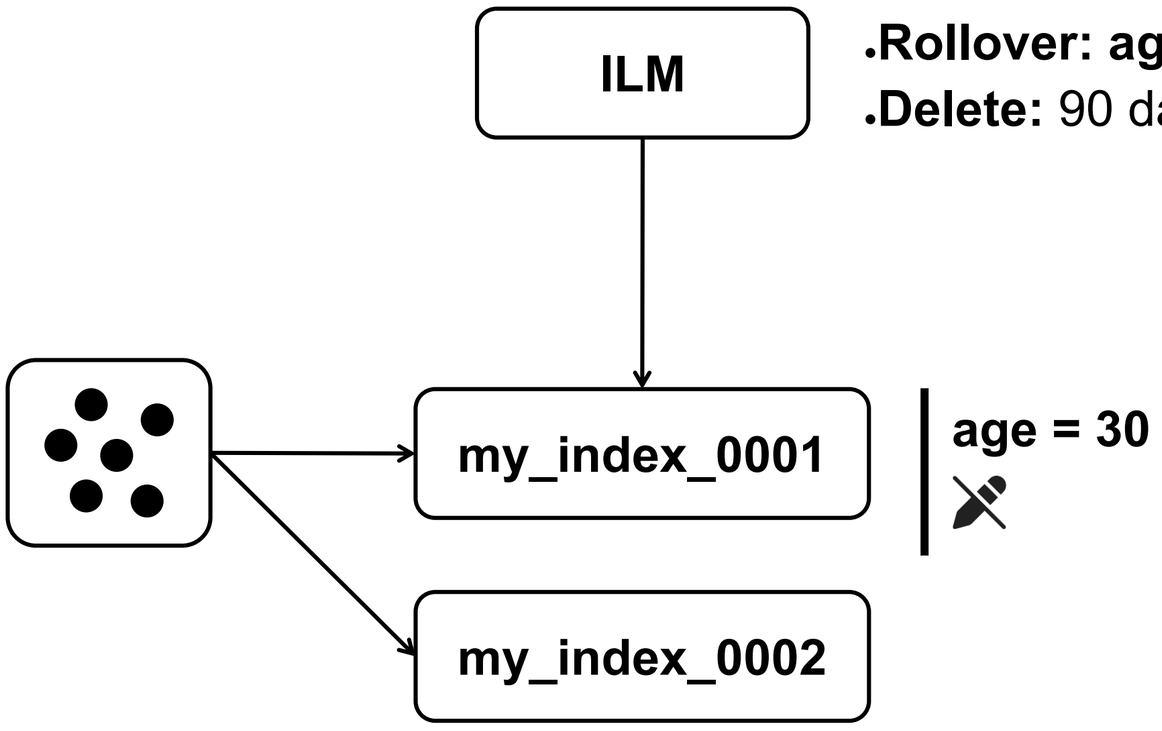
Time Series Data Stream

ILM visualized



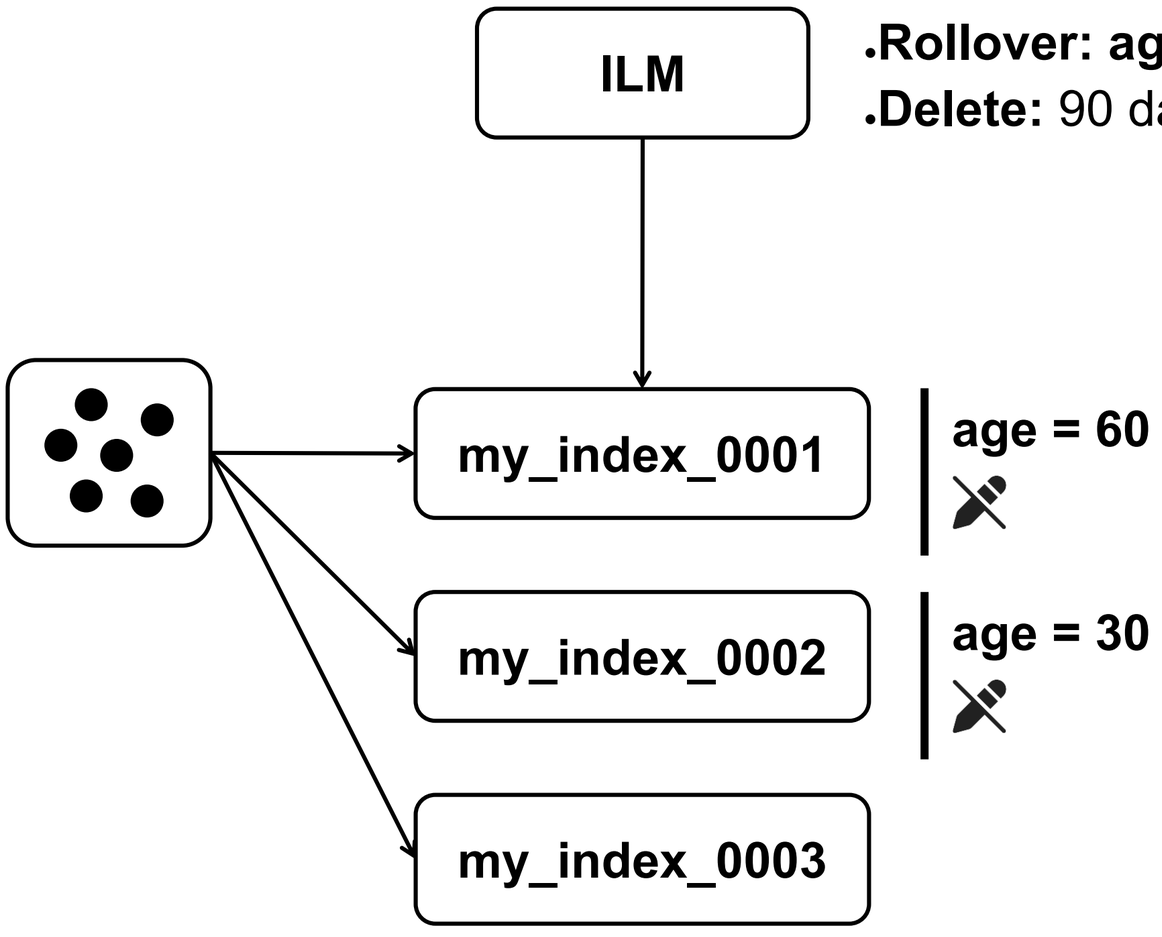
Time Series Data Stream

ILM visualized



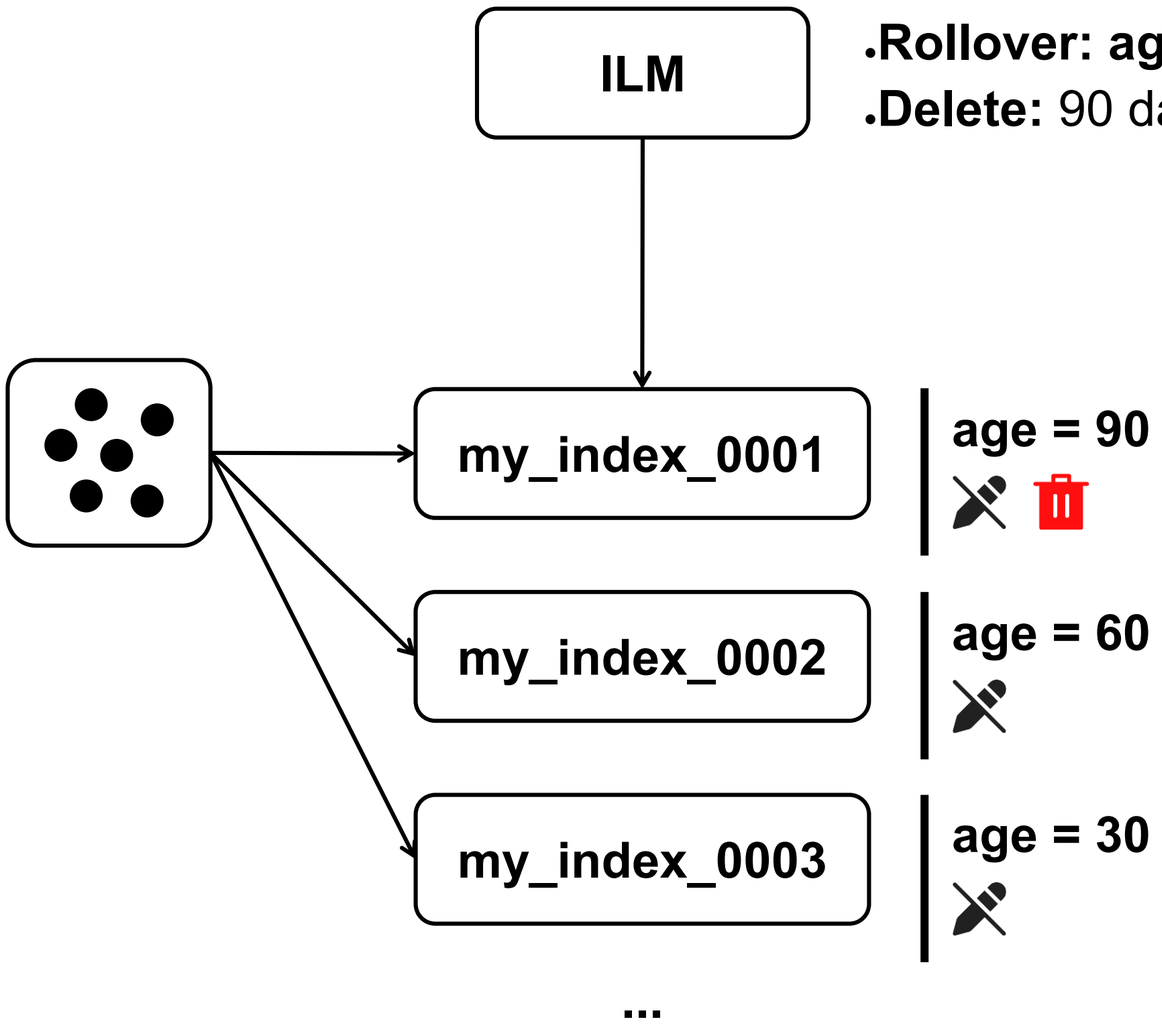
Time Series Data Stream

ILM visualized



Time Series Data Stream

ILM visualized



Time Series Data Stream

Querying time series data

```
{
  "query": {
    "range": {
      "@timestamp": {
        "gte": "2024-11-01T00:00:00",
        "lte": "2024-11-07T23:59:59"
      }
    }
  }
}
```

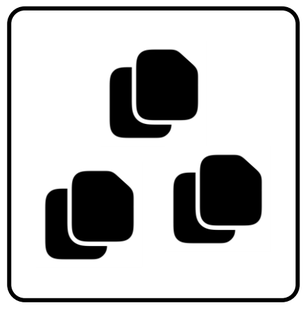
```
{
  "aggs": {
    "avg_cpu_usage": {
      "avg": {
        "field": "cpu_usage"
      }
    }
  }
}
```

24 Analyzers

Analyzers

- Analyzers **process** text during **indexing** and **searching**.
- They transform text into **tokens**.
- They make the **search** process **efficient** and **accurate**.

Read more



Documents

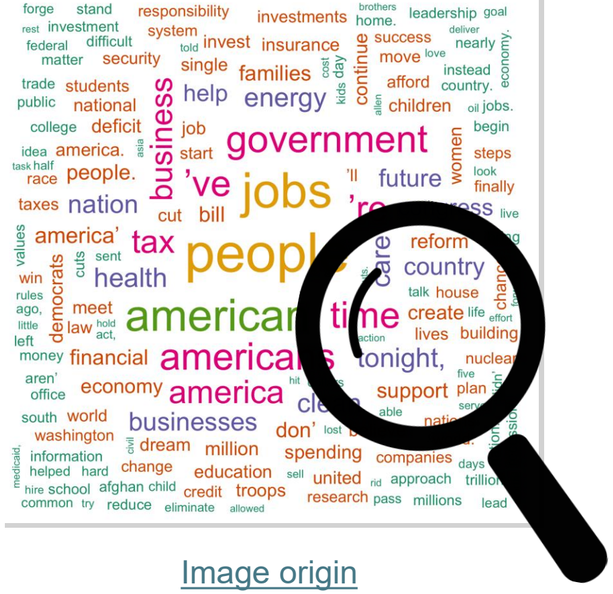


Image origin

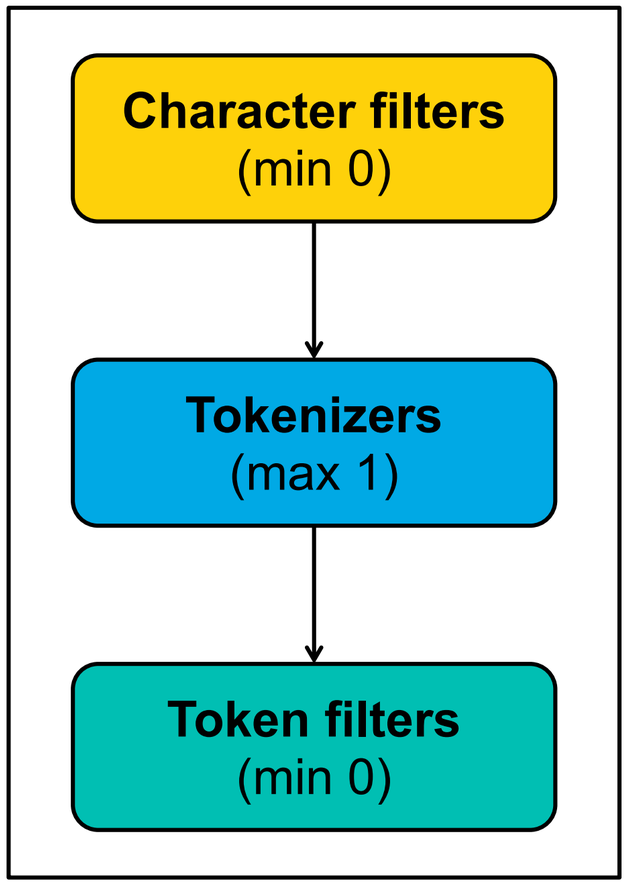
Term	Document
hello	Document 1
world	Document 1
imad	Document 2
saddik	Document 2

Analyzers

Analyzer components

.An analyzer is a **combination** of 3 components:

- **.Character filter**
- **.Tokenizer**
- **.Token filter**



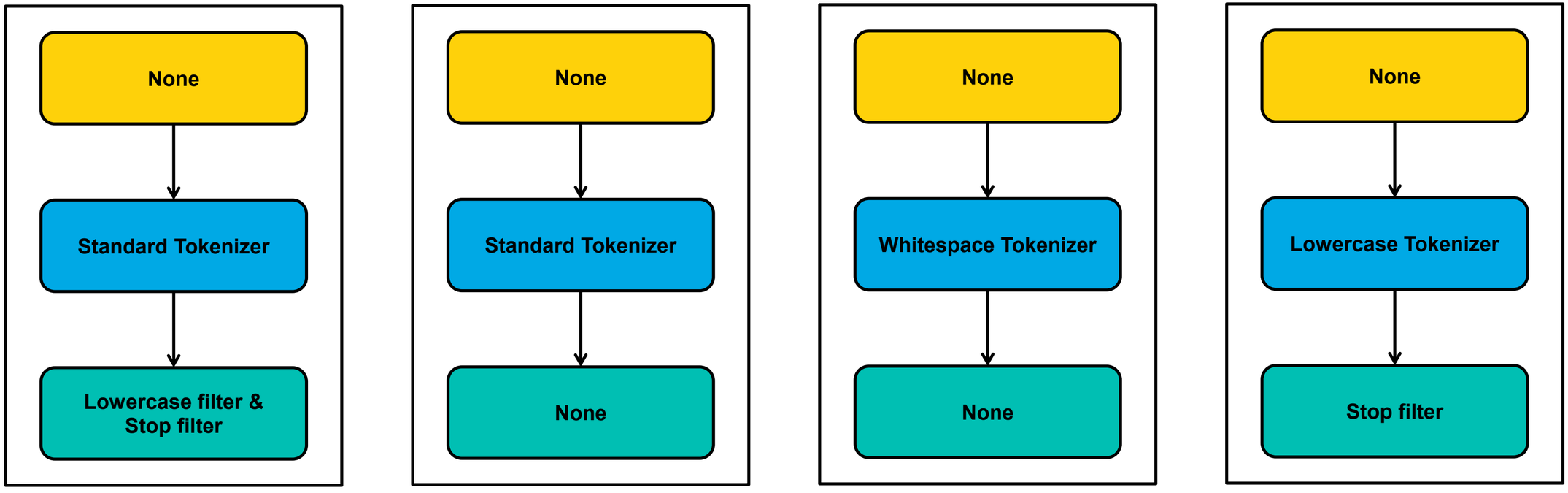
Analyzer

Analyzers

Built-in analyzers

- Provide **ready-made** options for processing text in various ways.
- Each built-in analyzer is designed for specific types of data.
- Common analyzers:

Read more

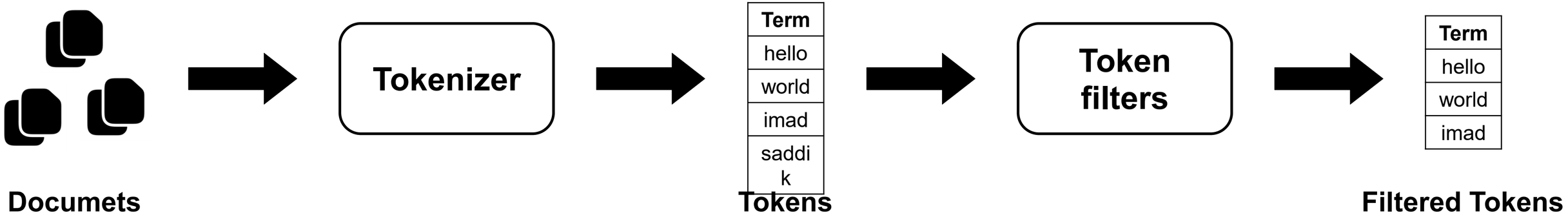


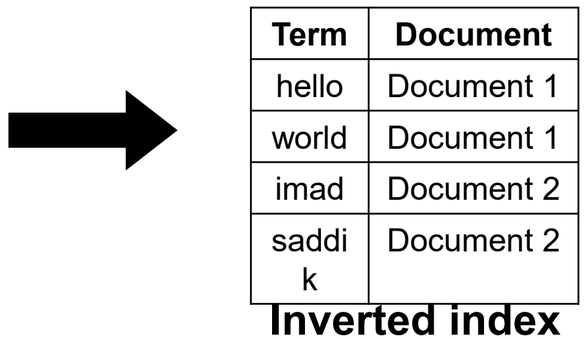
Standard Analyzer | Simple Analyzer | Whitespace Analyzer | Stop Analyzer

Analyzers

Phases of analysis

- Index time analysis





Term	Document
hello	Document 1
world	Document 1
imad	Document 2
saddi k	Document 2

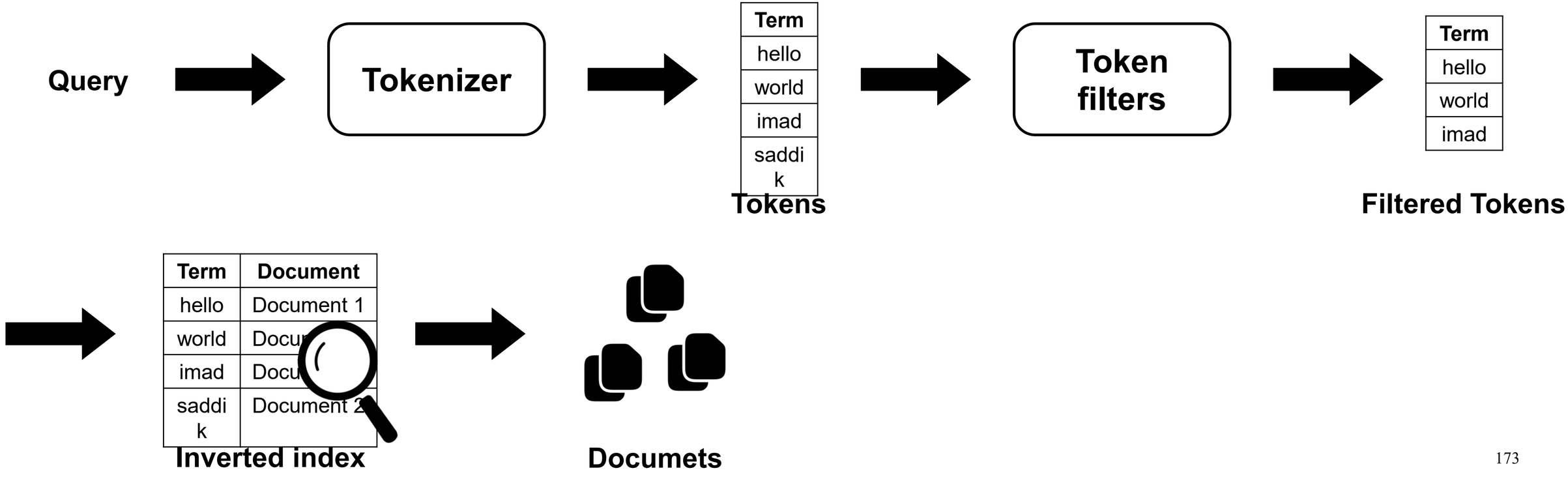
Inverted index

Analyzers

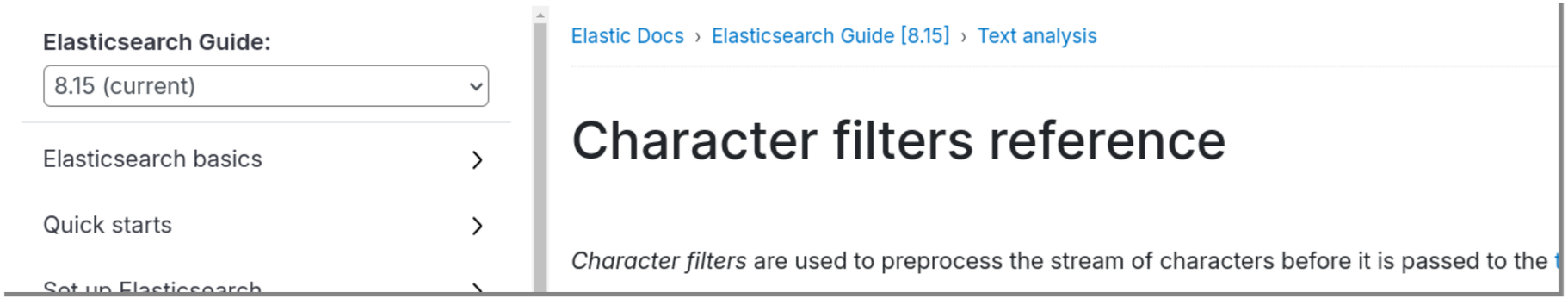
Phases of analysis

- Index time analysis
- Search time analysis

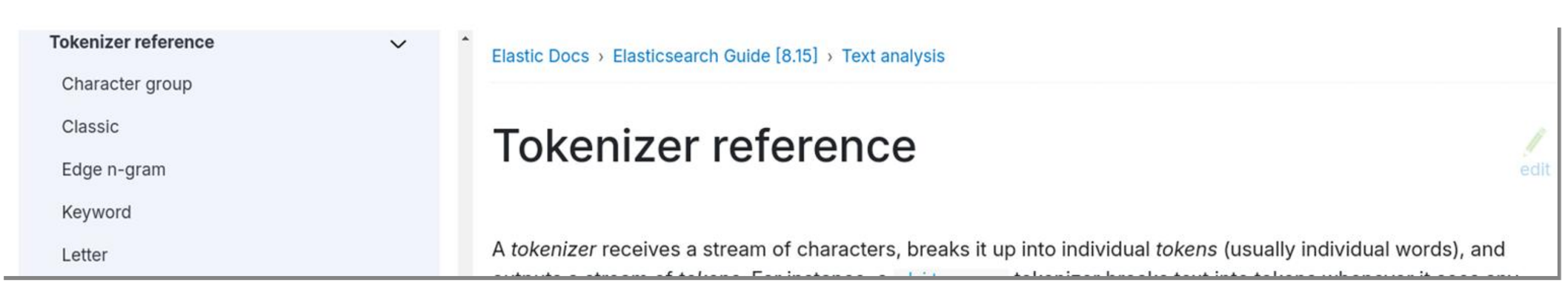
Read more



Analyzers


Elasticsearch Guide:
8.15 (current)
Elasticsearch basics
Quick starts
Set up Elasticsearch
Elastic Docs › Elasticsearch Guide [8.15] › Text analysis
Character filters reference
Character filters are used to preprocess the stream of characters before it is passed to the


Read more


Tokenizer reference
Character group
Classic
Edge n-gram
Keyword
Letter
Elastic Docs › Elasticsearch Guide [8.15] › Text analysis
Tokenizer reference
edit
A tokenizer receives a stream of characters, breaks it up into individual tokens (usually individual words), and


Token filter reference
Apostrophe
ASCII folding
CJK bigram
CJK width
Classic
Elastic Docs › Elasticsearch Guide [8.15] › Text analysis
Token filter reference
edit
Token filters accept a stream of tokens from a tokenizer and can modify tokens (eg lowercasing), delete tokens

Analyzers

Introduction to Analysis and analyzers in Elasticsearch

Phase 02 — indexing, mapping and analysis — Blog 08

Arun Mohan · Follow

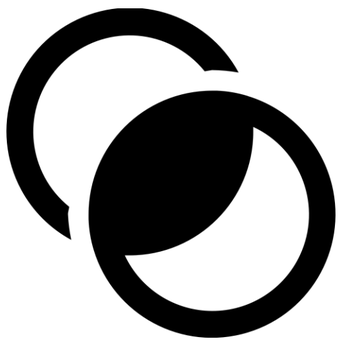
Published in elasticsearch · 8 min read · Dec 9, 2017

450 2

25 Synonyms

Synonyms

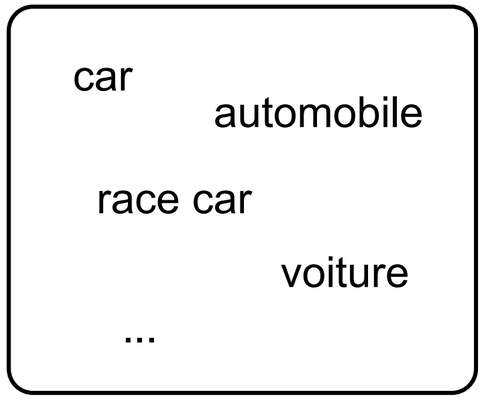
- Synonyms help **enhance search** accuracy.
- Useful for **matching variations or related terms**.
- Synonyms are defined using the **Solr format**.



Synonyms

Solr format

- This is a **flexible syntax** for **defining** synonyms.
- It uses **two** different definitions:
- **Equivalent synonyms:** “term 1, term 2, term 3”
- **Explicit synonyms:** “term 1, term 2 => term 3”



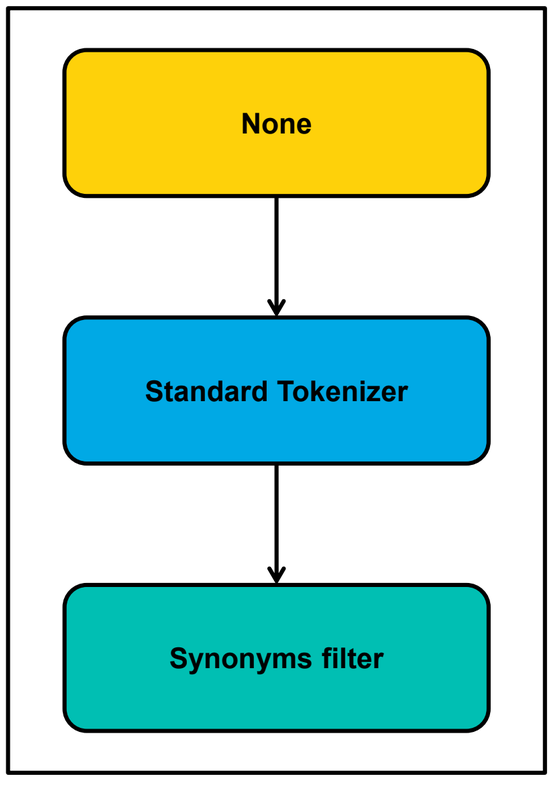
Equivalent terms

Personal computer → **PC**

i-pod, i pod → **ipod**

Synonyms

- Synonyms are used within **analyzers**.
- You can use synonyms **in index and search time**.
- Synonyms are a custom **token filter**.

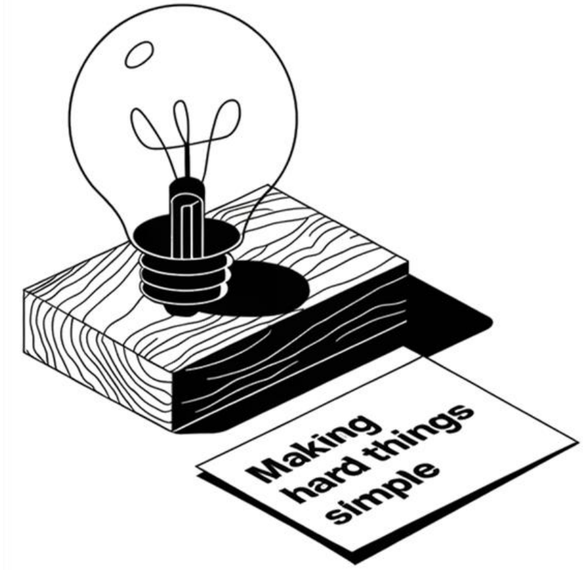


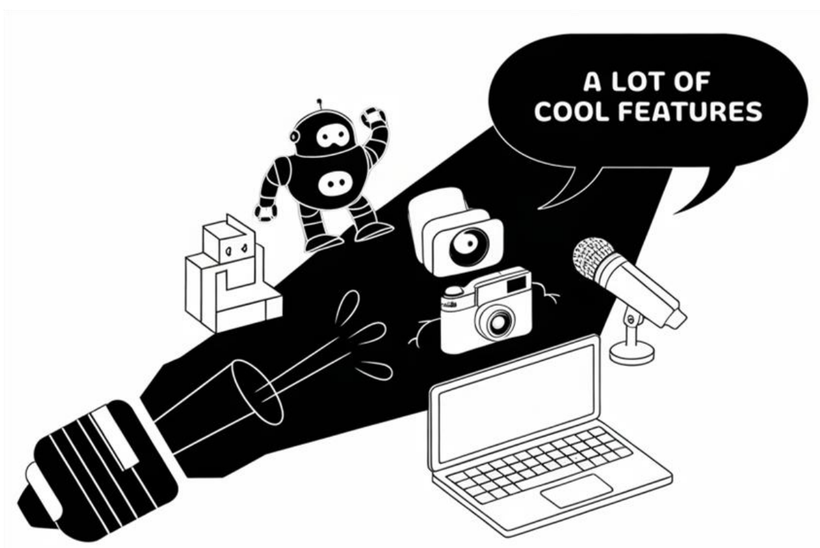
Custom Analyzer

26 Common options

Common options

- **Simplify** Elasticsearch management.
- Provide features like **human-readable output, date math, and filtering**.
- All Elasticsearch **REST APIs** support these common options.





Common options

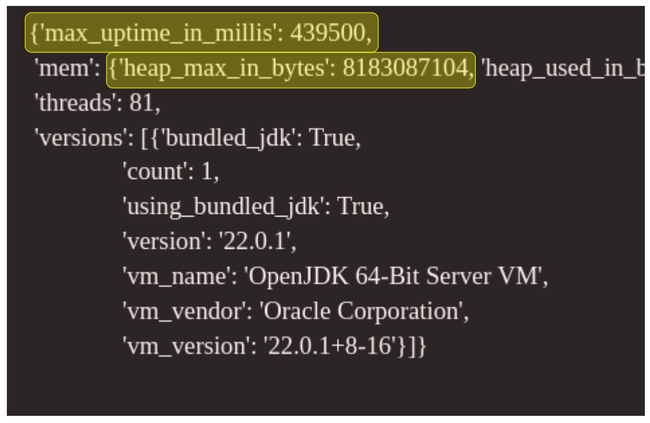
Human-readable output

.Make **statistics** in a format that **humans can understand**.

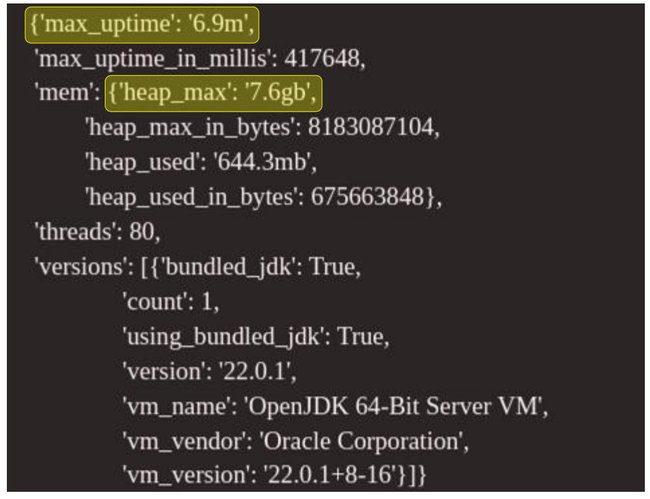
.It applies to **disk space, memory, time**, and other metrics.

Example

response = es.cluster.stats(**human=True**)
pprint(response["nodes"]["jvm"])



Before



After

Common options

Date math

- Perform **math operations** on dates.
- Operations include: **Add, Subtract, Round down to nearest day**.
- Supported time units: **y (years), M (moths)**, etc.
- The expression starts with an **anchor date** (“**now**” or a string ending with **||**).

Examples

- **now** := 2024-11-16 11:55:00
- **now+1h** := 2024-11-16 12:55:00
- **now-1h** := 2024-11-16 10:55:00
- **now-1h/d** := 2024-11-16 00:00:00
- **2024.11.16||+1M/d** := 2024-12-16 00:00:00

Common options

Response filtering

.Inclusive filtering: Specify fields to include.

.Exclusive filtering: Remove unnecessary fields.

.Combined filtering

Example

```
response = es.search(
    index=index_name,
    body={
        "query": {
            "match_all": {}
        }
    },
    filter_path="hits.hits._id,hits.hits._source"
)
pprint(response.body)
```

```
{'_shards': {'failed': 0, 'skipped': 0, 'successful': 1, 'total': 1},
'hits': {'hits': [{'_id': '3Eu8ApMB770u3lhz-QPW',
                   '_index': 'my_index',
                   '_score': 1.0,
                   '_source': {'content': 'The Solar System consists of the '
                                          'Sun and the objects that orbit it, '
                                          'including eight planets, their '
                                          'moons, dwarf planets, and '
                                          'countless small bodies like '
                                          'asteroids and comets.',
                               'id': 1,
                               'title': 'The Solar System'}},
```

Before

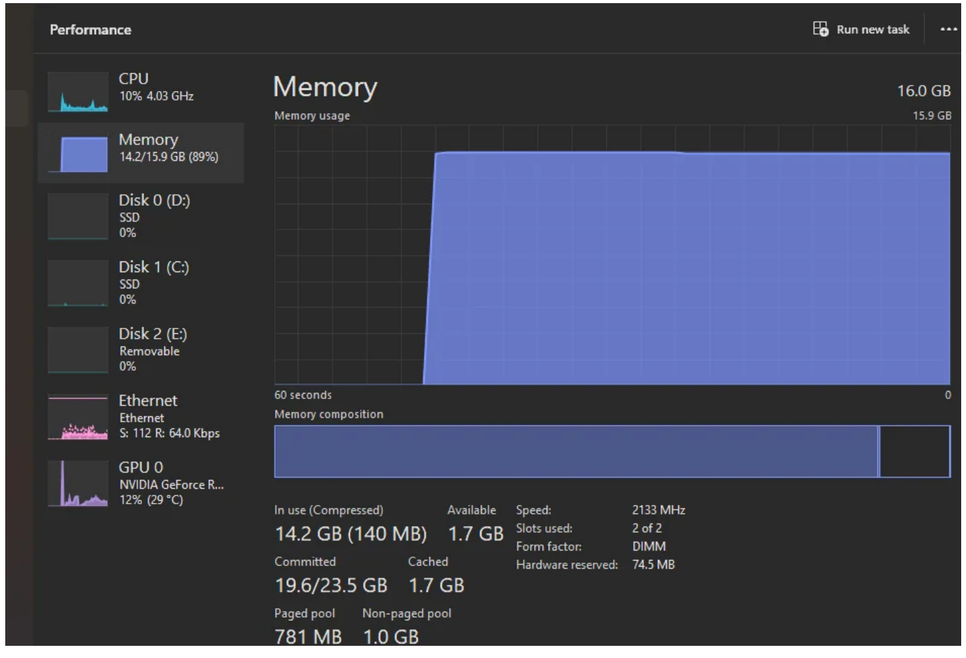
```
{'hits': {'hits': [{'_id': '3Eu8ApMB770u3lhz-QPW',
                    '_source': {'content': 'The Solar System consists of the '
                                           'Sun and the objects that orbit it, '
                                           'including eight planets, their '
                                           'moons, dwarf planets, and '
                                           'countless small bodies like '
                                           'asteroids and comets.',
                                'id': 1,
                                'title': 'The Solar System'}},
```

After

27 Change heap size

Change heap size

- By default Elasticsearch uses **50% of the available RAM**.
- This can **slow down** you PC.
- You only need **1 or 2GB** when dealing with small indices.



Change heap size

Steps to change the heap size

1) Start the container.

sudo docker start elasticsearch

Change heap size

Steps to change the heap size

1) Start the container.

2) Go inside the container.

sudo docker exec -u 0 -it elasticsearch bash

Change heap size

Steps to change the heap size

1) Start the container.

2) Go inside the container.

3) Create the **heap.options** file inside **jvm.options.d** folder and add this.

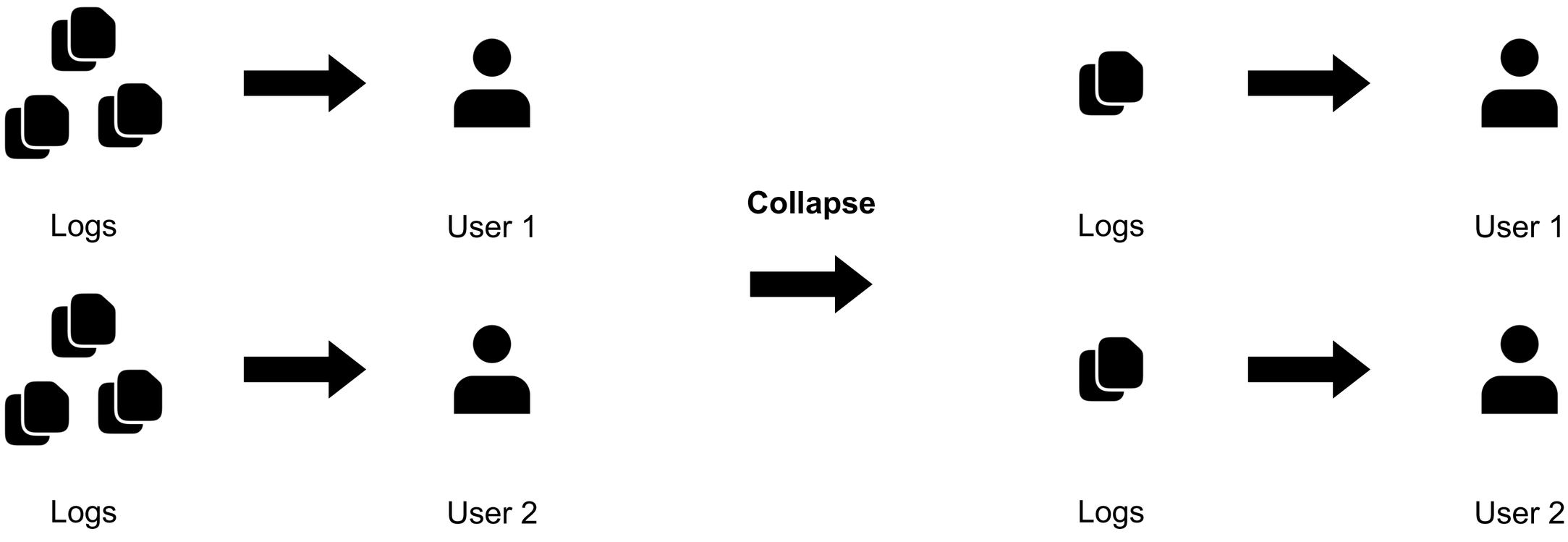
echo "-Xms2g" > /usr/share/elasticsearch/config/jvm.options.d/heap.options
echo "-Xmx2g" >> /usr/share/elasticsearch/config/jvm.options.d/heap.options

cat /usr/share/elasticsearch/config/jvm.options.d/heap.options

28 Collapse search results

Collapse search results

- Collapse is used to show only the **best document for each group**.
- It is **useful when many documents are similar.**



Collapse search results

.Example:

```
response = es.search(
    index="my_index",
    body={
        "query": {
            "match": {
                "message": "My Universe Hub"
            }
        },
        "collapse": {
            "field": "user_id"
        },
    }
)
```

ⓘ Important

- .By default, collapse returns **one document for each user**.
- .The returned document has the **highest score**.
- .Collapse works with **keyword** and **numeric** fields.

Collapse search results

- After collapsing the search results, you can use **inner hits** to **show related documents within each group**.
- This **changes the default behavior** by giving you a **deeper view into each collapsed group**.
- You can set how many related documents to get per group (using ***size***).
- You can also control how these related documents are sorted (using ***sort***).

Collapse search results

.Example:

```
response = es.search(
    index="my_index",
    body={
        "query": {
            "match": {
                "message": "My Universe Hub"
            }
        },
        "collapse": {
            "field": "user_id",
            "inner_hits": {
                "name": "most_recent",
                "size": 5,
                "sort": [ { "@timestamp": "desc" } ]
            },
        },
    }
)
```

Name given to the inner hit section in the search reponse

Number of hits (documents) to retrieve per group.

How to sort the document inside the group.

Collapse search results

.You can combine **collapse** with **search_after** but not **scroll**.

.**Example**:

```
response = es.search(
    index="my_index",
    body={
        "query": {
            "match": {
                "message": "My Universe Hub"
            }
        },
        "collapse": {
            "field": "user_id"
        },
        "search_after": ["idid3647"]
    }
)
```

29 Pre-filtering + kNN search

Pre-filtering + kNN search

.**Pre-filtering** means filtering documents before performing the kNN search.

.It is supported through the **filter** parameter of the **kNN query**.

.**Example:**

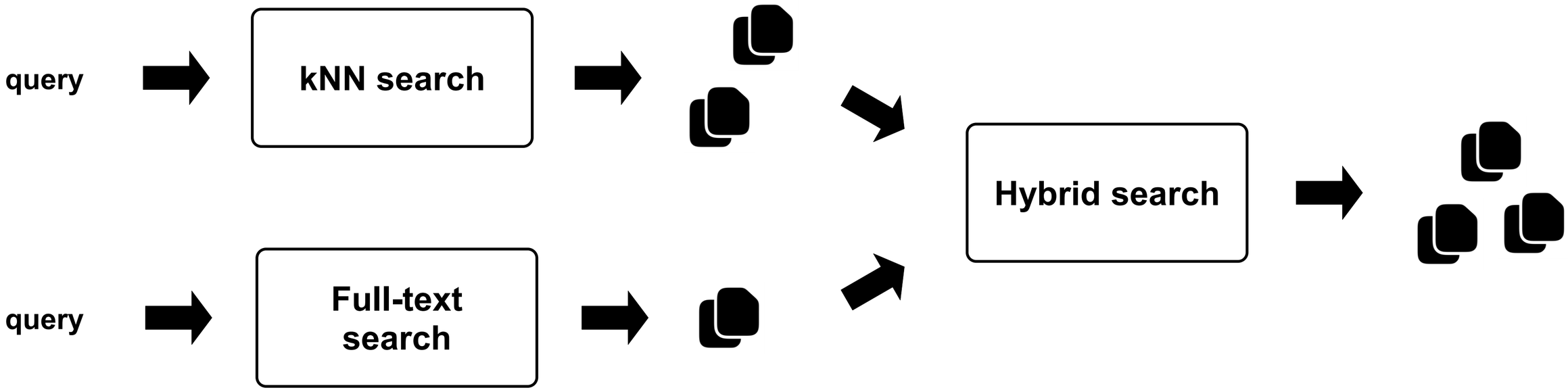
```
result = es.search(
    index="apod",
    knn={
        "field": "embedding",
        "query_vector": embedded_query,
        "num_candidates": 20,
        "k": 10,
        "filter": {"term": {"year": 2024}},
    },
)
```

30 Hybrid search

Hybrid search

.**Hybrid search** is a combination of **kNN** and **full-text** search.

Read more



Hybrid search

- **How do we combine** the results from kNN and full-text search?
- With a fusion algorithm like RRF (Reciprocal Rank Fusion).
- Using RRF in Elasticsearch requires a **paid license**.
- Possible to use the ranx library to perform RRF because it is **free, open source** and **supports a lot of fusion algorithms**.

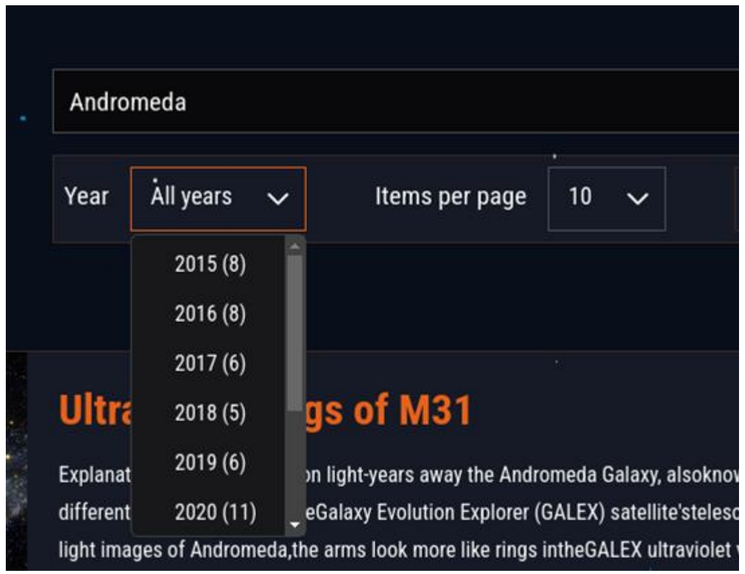
Read more

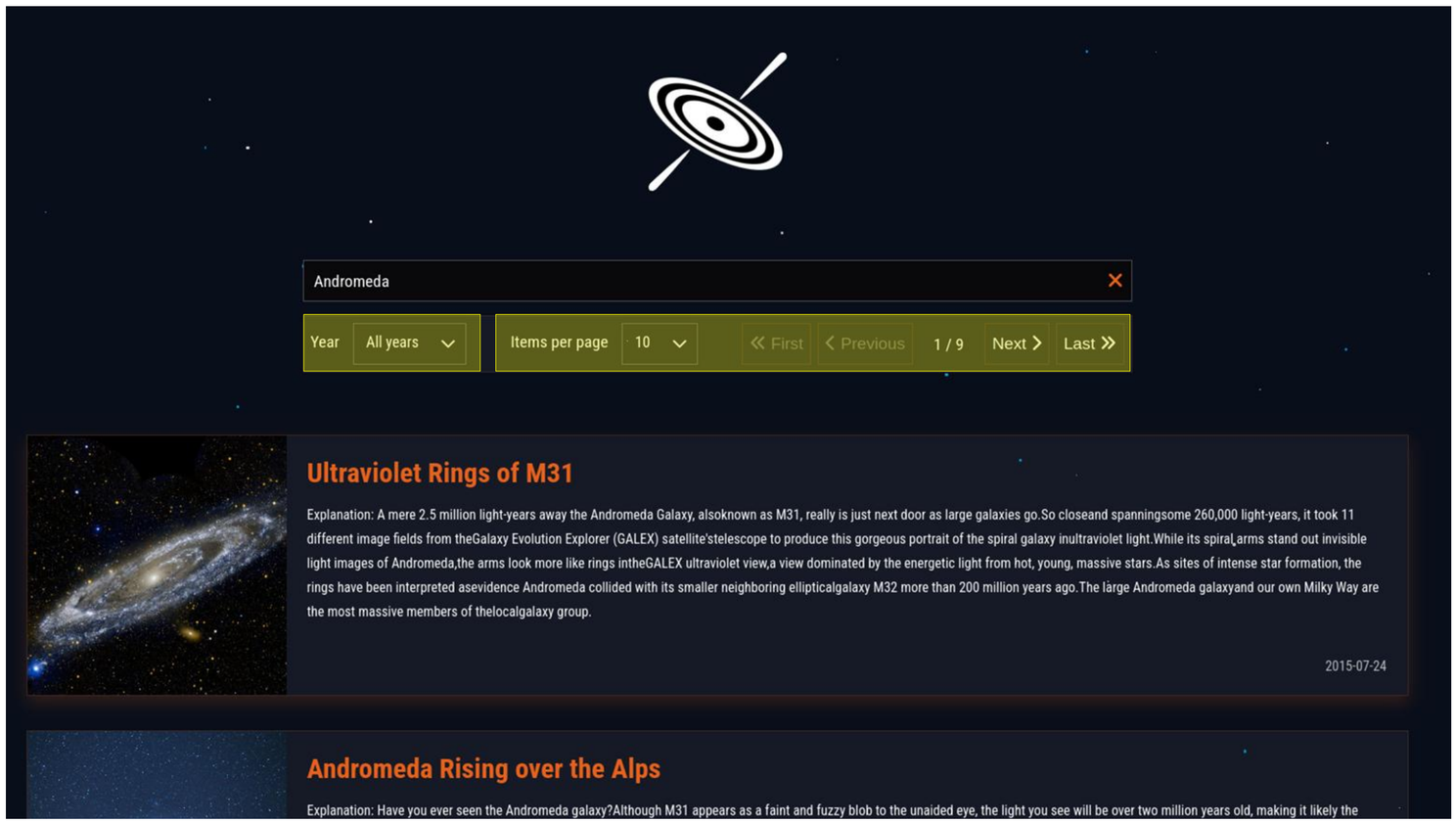
	Free and open - Basic [1, 2]	Platinum	Enterprise	Gold (Discontinued)[9]
SEARCH & ANALYSIS				
Similarity functions for vector fields	✓	✓	✓	✓
Vector search	✓	✓	✓	✓
Reciprocal Rank Fusion (RRF) for hybrid search	—	—	✓	—
Doc-value-only fields	✓	✓	✓	✓
Synonym management	✓	✓	✓	—
Retrievers: Standard, kNN, pinned, rescorer	✓	✓	✓	—
Retrievers: linear, rule, RRF, text similarity re-ranker	—	—	✓	—
Reciprocal Rank Fusion (RRF) for hybrid search	—	—	✓	—

31 Examples

Examples

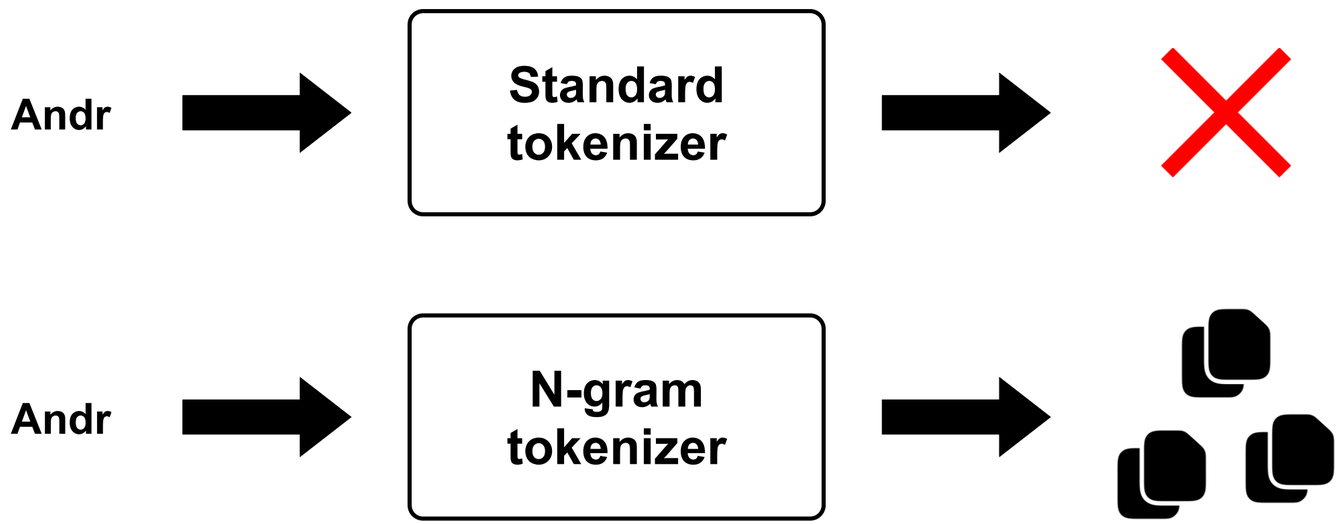
- Add the pagination controls.
- Filter by year.
- Use aggregations.





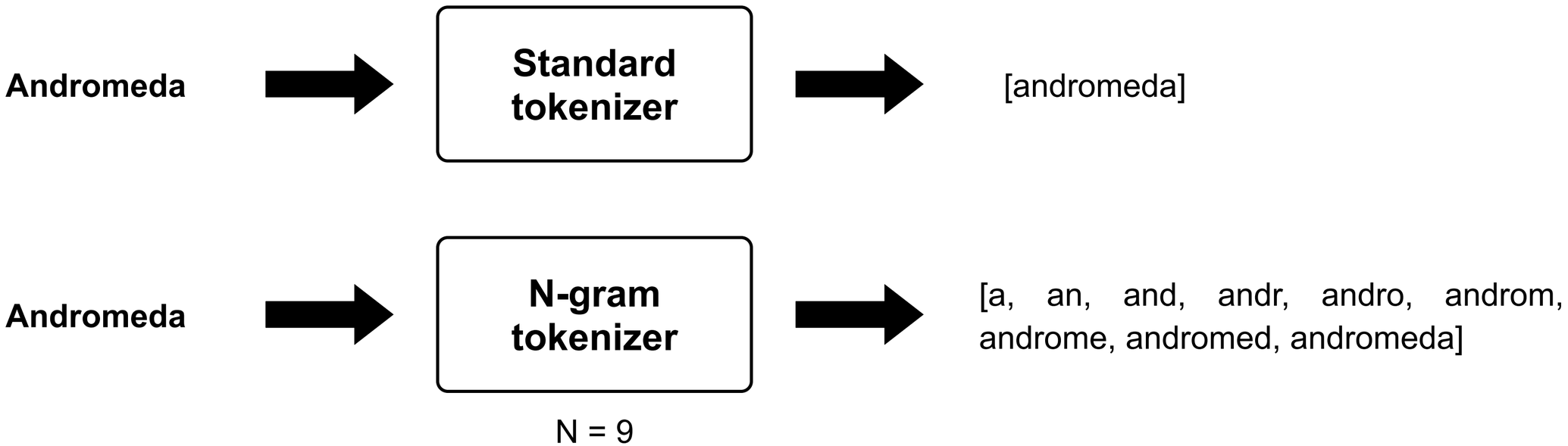
Examples

- Implement a ***search as you type*** feature.
- Utilize the N-gram tokenizer.



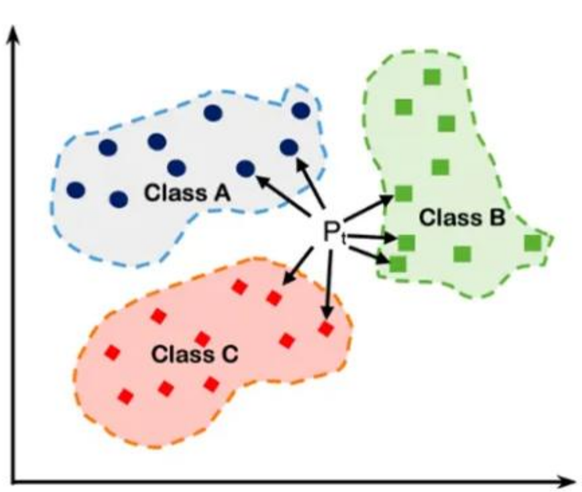
Examples

- Why use the N-gram tokenizer?



Examples

- Implement **semantic search**.
- Use an **embedding model** from **HuggingFace**.
- Use **kNN search** to find documents.



Medium article by Sachinsoni

Examples

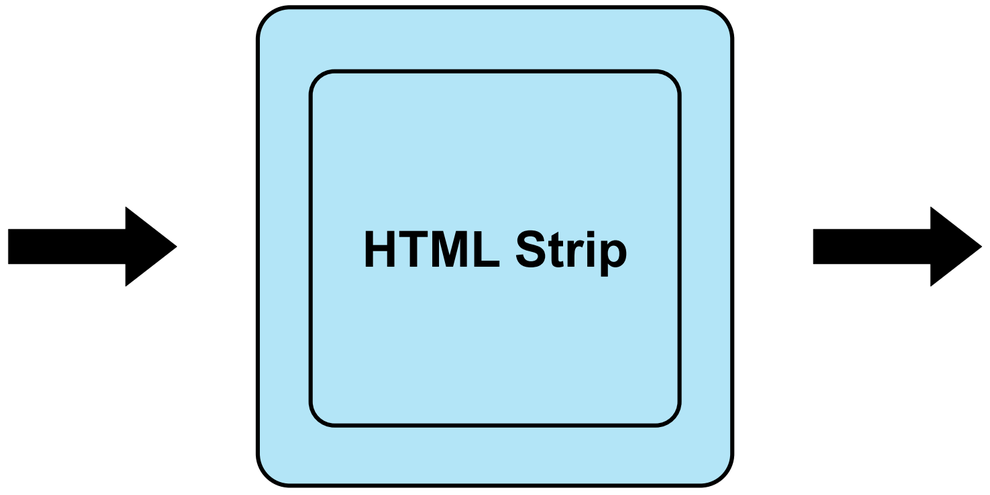
- Add the raw data.
- Contains HTML tags.

{
  "date": "2024-11-30",
  "title": "Winter and Summer on a Little Planet",
  "explanation": "<p>\n Explanation: \n\nWinter and summer appear to come on a single night to this\nstunning little planet.\n\nIt's planet Earth of course.\n\nThe\ndigitally mapped,\nnadir centered panorama covers 360x180\ndegrees and is\ncomposed of frames recorded during January and July from the\nCol du Galibier ...
}

Examples

- Contains **HTML tags**.
- Use **pipelines** to remove the **HTML tags**.


```
{
  "date": "2024-11-30",
  "title": "<a
href=\"ap241130.html\">Winter and
Summer on a Little Planet</a>",
  "explanation": "<p>\n<b> Explanation:
</b> \n\nWinter and summer appear to
come on a single night to this\n<a
href=\"https://www.instagram.com/c
amille.niel_photography/p/C270AVzr
Kcp/?img_index=1\">stunning little
planet</a>.\n\nIt's planet Earth of
course.\n\nThe\n<a
href=\"http://srcematematike.si/2014/
03/09/math-behind-tiny-
planets/\">digitally mapped...
```


```
{
  "date": "2024-11-30",
  "title": "Winter and Summer on a Little
Planet",
  "explanation": "\n Explanation:
\n\nWinter and summer appear to
come on a single night to
this\nstunning little planet.\n\nIt's planet
Earth of course.\n\nThe\ndigitally
mapped...
```


Ingest pipeline

Thank you for your attention!