

# A Search Engine: Elasticsearch

Chapter-content:

2022-2023

- Information retrieval, reminder
- Lucene, the cornerstone for open-source search engines
- REST APIs and HTTP requests
- Elasticsearch: a distributed search engine

# Table of content

---

2022-2023

## A Search Engine: Elasticsearch

- Information retrieval, reminder
- Lucene, the cornerstone for open-source search engines
- REST APIs and HTTP requests
- Elasticsearch: a distributed search engine

# Information retrieval, terminology

- *Preprocess*: corpus of documents.
- *Queries*: a term.
- *Expected answer*: list of documents containing the terms (possibly location(s) of term within document, too).

Relevant documents = true positives ( $t_p$ ) + false negatives ( $f_n$ ).


**Precision** (prop. of  $t_p$  in result) =  $\frac{t_p}{t_p + f_p}$

**Recall** (prop. of  $t_p$  in the expected result) =  $\frac{t_p}{t_p + f_n}$


# Inverted index

## "Forward" Index

**documents**    **content**

d1  : apple, pear

d2  : cherry, pear, walnut

d3  : cherry, pear

## Inverted Index = postings list

apple : d1

cherry : d2 d3

pear : d1, d2, d3

walnut : d2

# Text processing

## Tokenization

```
import nltk
tokens = nltk.word_tokenize("""At eight o'clock on""")
['At', 'eight', "o'clock", 'on']
#other NLP library: spacy...
```

Stemming : removes prefix/suffix to return the stem of a word, based on rules Ex: in english, removes "ement" suffix in long words ; replacement → replac

Lemmatization : replaces a word with its lemma (base form, which is a valid word), based on dictionary Ex: in english, maps the words "are", "am", "is" ...to "be"

# Table of content

---

2022-2023








## A Search Engine: Elasticsearch

- Information retrieval, reminder
- Lucene, the cornerstone for open-source search engines
- REST APIs and HTTP requests
- Elasticsearch: a distributed search engine

A text search engine library.

- written in 1999 by Doug Cutting, in Java
- free and open source

Many projects have been based on Lucene :

- projects that were originally sub-projects of Lucene:
  -  **Solr** search engine
  -  **hadoop** distributed filesystem and batch processing
  -  **Tika** detects and extracts metadata (file type, language identification, OCR with tesseract)
  -  **nutch** crawler (and html parser)
- projects built on top of Apache Lucene:
  -  **elasticsearch** search engine
  -  **mongoDB**, **Atlas** **Mongodb + Lucene for full-text search.**
  -  **CrateDB** document store

# Lucene : features

- Builds an index by parsing files:
  - Inverted index (input: a term  $t$ , finds docs that contain  $t$ ),
  - Field indexes (input:  $docid$ ,  $field$ ; return value of field in doc)

Transformation pipelines (**Analyzer**) support: tokenization, simple conversions (lowercase), stop word removal, synonyms.

- Searching the indexes:
  - phrase/wildcard/proximity/range queries
  - can search/sort based on fields
  - ranking of results
  - possibility of highlighting search terms.

<https://lucene.apache.org/core/index.html>

<https://lucene.apache.org/core/>



# Lucene: fundamental concepts

- An index is a sequence of documents.
- A document is a sequence of fields.
- A field is a named sequence of terms.
- A term is a sequence of bytes, or rather a pair name, sequence of byte)

An *index* is a collection of documents. An index is split in segments that group together some of the documents from the index. Lucene segments are *immutable*. When a doc is updated, a new document is created (in another segment) and the old one is marked as deleted. The writer may eventually merge segments and reclaim space left by deleted documents.

[https://lucene.apache.org/core/8\\_0\\_0/core/org/apache/lucene/codecs/lucene80/package-summary.html](https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/codecs/lucene80/package-summary.html)

# Lucene: main classes

## Creating the index:

- **Document** : a set of **Fields** =(name, value, type, and tokenStream)
- **IndexWriter** : to create an index, add or delete documents to an index.

## Searching the index:

- **QueryParser.parse()** parses a string into a **Query**
- **IndexReader** : to read from an index. Guarantees consistency (reads a consistent snapshot of the index).
- **IndexSearcher.search(query, int)** : implements a search over an IndexReader (hence consistency guaranteed).

[https://lucene.apache.org/core/8\\_8\\_1/core/overview-summary.html](https://lucene.apache.org/core/8_8_1/core/overview-summary.html)

[https://lucene.apache.org/core/8\\_1\\_0/core/org/apache/lucene/index/package-summary.html](https://lucene.apache.org/core/8_1_0/core/org/apache/lucene/index/package-summary.html)

<https://fr.wikipedia.org/wiki/Lucene>

## Lucene : creating an index (sketch)

```
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.IndexWriter;
...
IndexWriter writer = new IndexWriter(dir, iwconfig)
Document doc = new Document();
doc.add(new TextField("booktitle", "The Art of Computer Programming",
    Field.Store.YES));
Field pathField = new StringField("path", file.toString(), Field.Store.YES)
doc.add(pathField);
long lastModified = Files.getLastModifiedTime(path).toMillis()
doc.add(new LongPoint("modified", lastModified));

writer.addDocument(doc);

writer.close();
```

[https://lucene.apache.org/core/8\\_8\\_1/demo/src-html/org/apache/lucene/demo/IndexFiles.html](https://lucene.apache.org/core/8_8_1/demo/src-html/org/apache/lucene/demo/IndexFiles.html)

<http://www.lucenetutorial.com/lucene-in-5-minutes.html>

# Lucene : searching an index (sketch)

```
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
...
IndexReader reader = DirectoryReader.open(...);
IndexSearcher searcher = new IndexSearcher(reader);
Analyzer analyzer = new StandardAnalyzer();
QueryParser parser = new QueryParser("contents", analyzer);
Query query = parser.parse(query_str);

searcher.search(query, max_number_of_hits_returned);
```

[https://lucene.apache.org/core/8\\_8\\_1/demo/src-html/org/apache/lucene/demo/SearchFiles.html](https://lucene.apache.org/core/8_8_1/demo/src-html/org/apache/lucene/demo/SearchFiles.html)

<http://www.lucenetutorial.com/lucene-in-5-minutes.html>

# Lucene Query Syntax (Classic QueryParser)

Mostly a facility for human users. Otherwise, build query using Query API.

- title:"The Right Way" AND text:go
- "jakarta apache" jakarta    same as "jakarta apache" OR jakarta
- (jakarta OR apache) NOT "Apache Lucene"
- te\*t OR t?st    wildcards
- /[mb]oat/    regex
- "jakarta apache"~10    jakarta and apache within 10 words of each other
- title:{Aida TO Carmen}    range query
- roam~1    fuzzy search with edit distance up to 1
- jakarta^4 apache    jakarta is boosted (factor can be any float >0)

[https://lucene.apache.org/core/8\\_8\\_1/queryparser/org/apache/lucene/queryparser/classic/package-summary.html](https://lucene.apache.org/core/8_8_1/queryparser/org/apache/lucene/queryparser/classic/package-summary.html)

```
BooleanQuery myQuery = new BooleanQuery();
myQuery.add(new TermQuery(new Term("contents", "gold")), true, false);
myQuery.add(new TermQuery(new Term("contents", "iron")), true, false);
myQuery.add(new TermQuery(new Term("contents", "sport")), false, true);
//same : Query Query = QueryParser.parse("gold AND iron NOT sport", "contents", new StandardAnalyzer());
```

We thus have: **TermQuery, PhraseQuery, BooleanQuery, FuzzyQuery, WildcardQuery, TermRangeQuery, RegexpQuery PrefixQuery...**

[https://lucene.apache.org/core/8\\_0\\_0/core/org/apache/lucene/search/Query.html](https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/search/Query.html)

# Lucene : inverted indexes = postings

Documents identified by 32bits *docid* modified on merge

Term  $\rightarrow$  ordered list of docs containing the term

and for each of those docs  $\rightarrow$  (frequency of term, list of positions)

A *posting* is a pair (term, document) appearing this index.

Index gathers a lot of statistics to speedup retrieval:

Term	TermsEnum.docFreq()	#docs that contain at least once the term
	TermsEnum.totalTermFreq()	#occ of the term in index
Field	Terms.size()	#distinct terms for the field
	Terms.getSumDocFreq()	#postings for the field: $\sum_{\text{term in Field}} \text{docFreq}()$
	Terms.getDocCount()	#docs containing $\geq 1$ term for the field
Index	IndexReader.numDocs()	#live docs in index
	IndexReader.maxDocs()	#docs including deleted

[https://lucene.apache.org/core/8\\_1\\_0/core/org/apache/lucene/index/package-summary.html](https://lucene.apache.org/core/8_1_0/core/org/apache/lucene/index/package-summary.html)

## Lucene : inverted indexes (2)

The way the index is organized depends on field type:

**TextField** Reader or String indexed for full-text search

**StringField** String indexed verbatim as a single token

**IntPoint** int indexed for exact/range queries.

**StoredField** are not indexed in the (inverted) index and therefore not searchable, but are stored so are returned on search hits.

Storing or not a field determines whether it is returned in the result.

[https://lucene.apache.org/core/8\\_0\\_0/core/org/apache/lucene/document/Field.html](https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/document/Field.html)

# Lucene Scoring

## Lucene

1. first filters document according to *Boolean Model of Information Retrieval* (doc must satisfy query)
2. then scores results with a retrieval model. Ex: the *Vector Space Model*:

Simplified formula:

$$\text{score}(q,d) = \text{query-boost}(q) \cdot \underbrace{\frac{V(q) \cdot V(d)}{\|V(q)\| \|V(d)\|}}_{\text{cosine similarity of weighted vectors } V(q) \text{ and } V(d)} \cdot \underbrace{\text{doc-len-norm}(d)}_{\text{Normalize } V(d)} \cdot \text{doc-boost}(d)$$

Lucene's actual formula:

$$\text{score}(q,d) = \sum_{t \text{ in } q} \text{boost}(t,q) \cdot \underbrace{\text{tf}(t \text{ in } d)}_{(\text{frequency of } t \text{ in } d)^{1/2}} \cdot \text{idf}(t) \cdot \underbrace{\text{norm}(t,d)}_{1 + \log \frac{1 + \text{Nb Docs}}{1 + \text{Nb Docs containing } t}}$$

[https://lucene.apache.org/core/8\\_8\\_1/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html](https://lucene.apache.org/core/8_8_1/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html)



## Lucene: highlighting

To provide "keyword in context", choose among 3 highlighter classes:

- **UnifiedHighlighter**: scores each passage in the doc as if they were individual docs.
- **Highlighter** : the original Highlighter
- **FastVectorHighlighter**

Customize:

- how the text is divided into passages.
- how passages are ranked.
- how snippets are formatted.

[https://lucene.apache.org/core/8\\_8\\_1/highlighter/index.html](https://lucene.apache.org/core/8_8_1/highlighter/index.html)

# Lucene : performance

- ✓ Index size roughly 20% of input raw text.
- ✓ Indexes  $\simeq$  500GB/h on high-end PC (for 2011-2016:  $\simeq$  150GB/h)
  - AMD Ryzen Threadripper 3990X : 4GHz, 64 cores, 256 GB RAM, 256MB L3.
  - 256 GB RAM
  - Intel 9050 Optane : 960GB SSD, 2GB/s sequential Read/Write (latency  $10\mu$ s), PCIe 3.0 x4, NVMe
- ✓ Can run with small RAM (1MB)

<https://lucene.apache.org/core/>

# Table of content

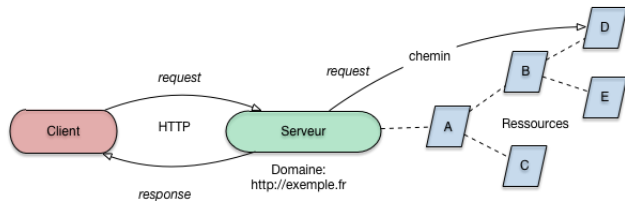
---

2022-2023

## A Search Engine: Elasticsearch

- Information retrieval, reminder
- Lucene, the cornerstone for open-source search engines
- **REST APIs and HTTP requests**
- Elasticsearch: a distributed search engine

# HTTP and REST API: reminder

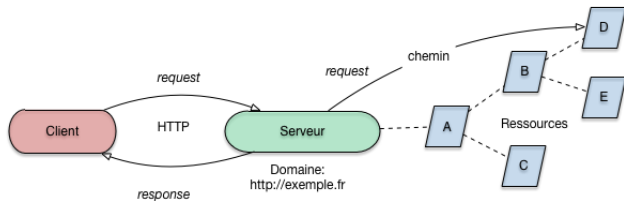


HTTP method	Description
GET	Get a representation of the target resource's state.
POST	Let target resource process the representation enclosed in request.
PUT	Set target resource's state to the state defined by representation enclosed in request.
DELETE	Delete the target resource's state.

Query defined through base URI (`http://api.example.com/`), HTTP method (GET, POST, PUT, and DELETE), media type.

- media type = type/subtype (structured as a tree, levels separated with `.`) +suffix ; parameter1 ; parameter2
- only type/subtype is mandatory.
- Standard trees don't have prefix (application/json), but vendors have .vnd:  
`application/vnd.openxmlformats-officedocument.wordprocessingml.document = docx`

## HTTP and REST API: reminder (2)



HTTP method	Idempotent	Cachable	safe
GET	✓	✓	✓
POST		✓	
PUT	✓		
DELETE	✓		

6 REST constraints :

- Client-server architecture
- Statelessness
- Cacheability
- Layered system
- Uniform interface
- (Code on demand).

# HTTP request: curl

*# GET request:*

```
curl 'http://www.7timer.info/bin/astro.php?lon=3.3&lat=50.6&ac=0&unit=metric&output=json'
```

*# POST request:*

```
curl -X POST "localhost:9200/my-index-000001/_doc/?pretty" -H 'Content-Type: application/json' -d '{
  "@timestamp": "2099-11-15T13:12:00",
  "message": "GET /search HTTP/1.1 200 1070000",
  "user": {
    "id": "kimchy"
  }
}'
```

```
-X request-method : -X GET (default), -X POST
-d data-url
-d jsonstring -H "Content-Type: application/json"
-d "@my-file.json"
--data-binary "@my-file.txt"
-i include response header
-o myfilename saves result as named file
-O saves result with original name
-v verbose
-u user:passwd
-c, --cookie-jar cookiefile (to save)
-b, --cookie cookiefile (to read) or "name1=value1;name2=v2"
-L follow redirect
```

# HTTP requests: curl

```
# GET request to list indexes in Elastic. All headers are returned (-v).
```

```
benoit.groz@a-140395 ~ $ curl -v -X GET "localhost:9200/_cat/indices"
```

```
Note: Unnecessary use of -X or --request, GET is already inferred.
```

```
* Trying 127.0.0.1...
```

```
* TCP_NODELAY set
```

```
* Connected to localhost (127.0.0.1) port 9200 (#0)
```

curl Info

```
> GET /_cat/indices HTTP/1.1
```

```
> Host: localhost:9200
```

```
> User-Agent: curl/7.58.0
```

```
> Accept: */*
```

```
>
```

HTTP request

```
< HTTP/1.1 200 OK
```

```
< content-type: text/plain; charset=UTF-8
```

```
< content-length: 142
```

```
<
```

HTTP response header

```
yellow open bank Com9KAS9Tw28eSzv9l0XMA 1 1 1000 0 381.9kb 381.9kb
```

```
yellow open customer sQEmw-8ERS29xVEW56an8g 1 1 1 0 3.5kb 3.5kb
```

HTTP  
response  
body

```
* Connection #0 to host localhost left intact
```

curl Info

# HTTP requests: telnet (establish a TCP connection)

```
benoit.groz@a-140395 ~ $ telnet localhost 9200
```

```
Trying 127.0.0.1...
```

```
Connected to localhost.
```

```
Escape character is '^['.
```

telnet Info

```
GET _cat/indices HTTP/1.0
```

user's HTTP request

```
HTTP/1.0 200 OK
```

```
content-type: text/plain; charset=UTF-8
```

```
content-length: 142
```

HTTP response header

```
yellow open bank      Com9KAS9Tw28eSzv910XMA 1 1 1000 0 381.9kb 381.9kb
```

```
yellow open customer sQEmw-8ERS29xVEW56an8g 1 1      1 0    3.5kb   3.5kb
```

HTTP  
response  
body

```
Connection closed by foreign host.
```



# HTTP request:Python

```
import requests
r = requests.get('https://api.github.com/user', auth=('user', 'pass'))

r.status_code # 200
r.headers['content-type'] # application/json; charset=utf8'
r.encoding # 'utf-8' educated guess. Can be modified
r.text # returns response content, based on r.encoding
r.content # for binary response
r.json() # for json response

r = requests.post("https://httpbin.org/post", data={'key1': 'v1', 'k2': 'v2'})
```

<https://requests.readthedocs.io/en/master/>

# Table of content

---

2022-2023

## A Search Engine: Elasticsearch

- Information retrieval, reminder
- Lucene, the cornerstone for open-source search engines
- REST APIs and HTTP requests
- **ElasticSearch: a distributed search engine**

Distributed search engine based on REST API.

- for full-text search, but also analyzing logs, real-time monitoring.
- part of ELK stack (Elastic, Logstash, Kibana).
- typical latency: a few ms.
- built on top of Apache Lucene, coded in Java.
- first released in 2010.
- used to be Open Core (ML and monitoring bricks were in a proprietary package), now dual license: Elastic and SSPL.
- Elastic NV: 1000+ employees, US (and dutch), market cap: 10B(03/2021).

General idea of SSPL : force SaaS providers to "contribute back to the community" (= form partnership with Elastic NV).

# ELK Stack

 **Elasticsearch** Search engine

 **Kibana** Visualization

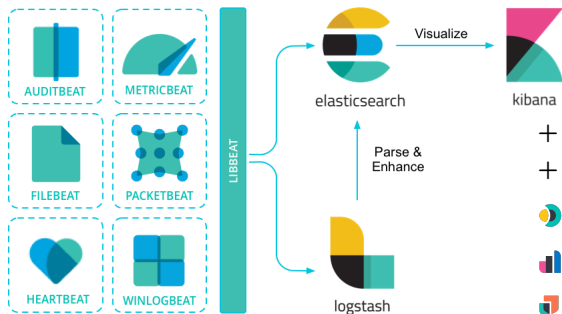
 **Beats**

 **Logstash**

} Ingestion

Beats is a collection of lightweight agents you install on your servers to send data to Elasticsearch.

Logstash is heavier, but has more options to transform/enhance data.



+ community beats: nvidiagpubeat.

+ Elastic's package solutions:

 Elastic Enterprise Search

 Elastic Observability

 Elastic Security

# ELK Stack: commercial offers

<https://www.elastic.co/subscriptions>

A sample (as of 03/2021):

	SSPL	Elastic	Gold	Plat.	Entrpr.
Inverted Index, BKD-tree, columnar	✓	✓	✓	✓	✓
Shape, Wildcard field, RB access control, authentication		✓	✓	✓	✓
Centralized Beats/Logstash management, Geoshape aggregations, Jira, IP filter, LDAP			✓	✓	✓
Cross-cluster replication, attribute-based access control, encryption, Tableau connector, JDBC, ML: classification...				✓	✓
Elastic endgame (cyberthreats), Enterprise deployment/update features					✓

# Indexes in Elasticsearch : index mappings

Based on Lucene indexes; collection of documents.

Each field stored in a dedicated structure:

- inverted index for text fields
- numeric (and geo.) fields in BKD tree

Fields can be inferred automatically **dynamic mappings**, or specified manually **explicit mappings**. Defining schema manually helps:

- Distinguish between full-text string fields and exact value string fields
- Perform language-specific text analysis
- Optimize fields for partial matching
- Use custom date formats
- Use data types such as `geo_point` and `geo_shape` that cannot be automatically detected

<https://www.elastic.co/guide/en/elasticsearch/reference/current/documents-indices.html>

## Indexes in Elasticsearch (2) : index settings

- Indexes are sharded and replicated (document stored in 1 primary shard, then multiple replicas)
  - Master (-eligible) nodes should have at most 3000 indexes per GB of heap.
  - Each node should have .5GB of heap space to manage workloads (indexing, searches, aggregations...), plus an extra 1KB of heap space per field per index
  - Size of a shard typically btw a few GB to a few tens of GB.
  - Nb of primary shards, first set at index creation. Default 1, up to 1024 shards per index
  - Nb of replicas can evolve
- To mitigate datacenter failure? Cross-cluster replication. Active-passive: primary cluster is active, other read-only

<https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/size-your-shards.html>

# Sharding and Replication in Elasticsearch

ES uses consistent hashing:

- Number of routing shards (hash table slots) is fixed at index creation.  
Can't be altered unless recreating new index : no incremental resharding.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-split-index.html>

- Nb of primary shards always a multiple of routing shards nb.
- An index can be shrunk or split (nb of shard is multiplied/divided. Must remain multiple of routing shard nb).
- In ES, the primary/replica are defined at shard level, not at node level.
- ES implements three functions to allocate shards :
  - allocating unassigned shards : (i) allocates primary shards of all indexes first, then 1st replica, then 2nd replica, etc. (ii) to allocate a shard, chooses a "best" eligible node.
  - moving shards when needed (ex: to free space on a node, when nodes are removed, or when rebalancing)
  - rebalancing shards across nodes.
- Routing table is replicated on every node.

<https://github.com/elastic/elasticsearch/blob/main/server/src/main/java/org/elasticsearch/cluster/routing/>



# Searching in Elastic

- Full-text queries : use a string (ex: keyword), sort results by relevance
- Structured queries : SQL-like combinations of filters, orders...
- aggregates
- there is also an SQL API (X-pack).

```
curl -X POST "localhost:9200/_sql?format=txt&pretty"\  
-H 'Content-Type: application/json' -d'  
{  
  "query": "SELECT * FROM library ORDER BY page_count DESC LIMIT 5"  
}'
```

# ETL

ETL pipelines can be defined through processors:

- *ingest* processors mostly modify existing data in docs.  
Ex: lowercase, trim, remove, split, gsub (regexp replace), set (a new field), append, join
- *enrich* processors add new data to docs using source indexes.

Exemples:

allow to transform documents before indexing.

## Elastic : advanced features

- Aliases : secondary names for one or more index (or for one or more data stream)
- Index templates : define settings, mappings, and aliases to be applied automatically when an index is created whose name matches the template pattern (templates have priority)
- Data streams: Allow to analyze a time serie as a rolling window.
  - like an alias for a collection of indices : queries cover all indexes
  - has a backing index template : write on latest index (append-only), previous ones are read-only.
  - every document in the data stream must have an `@timestamp` field
  - update possible on any index using `update_by_query`, `delete_by_query`,