ELASTICSEARCH

as a blackbox

ELASTICSEARCH

- Database built on top of Apache Lucene
  - A fulltext engine, built in 1999

RAW LUCENE QUERY

```
Directory directory =
FSDirectory.getDirectory(INDEX_DIRECTORY);
IndexSearcher indexSearcher = new IndexSearcher(directory);
Term term1 = new Term(FIELD_CONTENTS, "black");
Term term2 = new Term(FIELD_CONTENTS, "shoes");
PhraseQuery phraseQuery = new PhraseQuery();
phraseQuery.add(term1);
phraseQuery.add(term2);
phraseQuery.setSlop(2);
Hits hits = indexSearcher.search(phraseQuery);
```

ELASTICSEARCH

- Elasticsearch
  - Not only for fulltext
  - Provides nice REST API
  - Custom JSON-based "query language"
  - Distributed system: cluster, master instances, replication, failover, management

ELASTICSEARCH QUERY

```
GET /_search
{
    "query": {
        "match_phrase" : {
            "message" : "black shoes",
            "slop": 2
        }
    }
}
```

USE CASES

- Fulltext search & faceting
  - https://otvorenesudy.sk
- Data aggregation
  - https://demo.luigisbox.com/
- Centralized logging and analysis
  - ELK stack

THE GOOD PARTS

- Data replication
- Scaling reads and writes
- Fulltext search building blocks

"COMMUNICATION" CONCEPTS

- (mostly) JSON messages over HTTP on port 9200
- REST style
- not a binary format, you don't need a specialized client, like pgAdmin

HTTP

- Text-based protocol
- Request:
  - Method identification
  - Headers: request metadata
    - Content-Type
    - Accept-Encoding
    - ...
  - Body (sometimes)
- Response:
  - Headers: Response metadata
    - Status code
    - Max-age
    - ...
  - Body (sometimes)

```
telnet fiit.stuba.sk 80
Trying 147.175.154.48...
Connected to fiit.stuba.sk.
Escape character is '^]'.
GET /index.html HTTP/1.1
Host: fiit.stuba.sk

HTTP/1.1 302 Found
Date: Thu, 28 Nov 2019 22:27:48 GMT
Server: Apache/2.0.40 (Red Hat Linux)
Location: http://www.fiit.stuba.sk/index.html
Content-Length: 291
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.fiit.stuba.sk/index.html">here</a>.</p>
<hr />
<address>Apache/2.0.40 Server at fiit.stuba.sk Port 80</address>
</body></html>
```

HTTP STATUS CODES

- 200: OK
- 201: Created
- 302: Found
- 404: Not found
- 500: Internal server error
- 503: Service unavailable

HTTP METHODS / SEMANTICS

- GET - fetch resource, idempotent (safe to call repeatedly - does not modify data)
- POST - create a new resource (has request body)
- PUT/PATCH - replace/update resource (has request body)
- DELETE - delete resource
- HEAD, OPTIONS, ...

REST

- URLs = resources
- HTTP method - operations on resources
- For example, instead of:

```
GET /index.php?operation=delete&id=58936
```

do

```
DELETE /articles/58936
```

RESOURCES IN ELASTICSEARCH

- Indices
  - /pdt-test-data
- Documents
  - /pdt-test-data/1
- Mapping
  - /pdt-test-data/_mapping
- ...

- http://localhost:9200

- Requests via any HTTP compatible tool, e.g.
  - curl on command line
  - Your browser (for quick GET requests)
  - Postman - https://www.getpostman.com
  - Kibana

ELASTICSEARCH INDEX

- Basic unit of storage
- Somewhat like a table in SQL database
- Index has many "documents" - JSON documents, with "fields"

```
{
   "name": "Elasticsearch",
   "tagline": "For search"
}
```

INSERTING DOCUMENT

```
curl -XPOST 'localhost:9200/vi/lectures' \
     -H "Content-Type: application/json"\
     -d '{"title": "Ahoj"}'
```


```
{"_index":"vi","_type":"lectures","_id":"J5BStG4
BPgLDs8deVxjr","_version":1,"result":"created","
_shards":
{"total":2,"successful":1,"failed":0},"_seq_no":
0,"_primary_term":1}
```

FETCHING A SINGLE DOCUMENT

```
curl 'localhost:9200/vi/lectures/J5BStG4BPgLDs8deVxjr?pretty'
{
  "_index" : "vi",
  "_type" : "lectures",
  "_id" : "J5BStG4BPgLDs8deVxjr",
  "_version" : 1,
  "_seq_no" : 0,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "title" : "Ahoj"
  }
}
```

SEARCHING FOR A DOCUMENT (WITH NO QUERY)

```
curl 'localhost:9200/vi/_search?pretty'
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : { ... },
  "hits" : {
    "total" : { ... },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "vi",
        "_type" : "lectures",
        "_id" : "J5BStG4BPgLDs8deVxjr",
        "_score" : 1.0,
        "_source" : {
          "title" : "Ahoj"
        }
      }
    ]
  }
}
```

UPDATE & DELETE

- https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-update.html
- https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-delete.html
-

TYPES

- Schemaless
  - Marketing blahblah
- You need types:
  - For specialized queries (e.g. range query)
  - For specialized aggregations (e.g. date_histogram)
- You need analyzers:
  - To specify how (and if) the data should be indexed

- Schemaless = Elasticsearch will guess data type when the field first appears
- Useless for any real use-case

MAPPING (INFERRED)

```
curl 'localhost:9200/vi/_mapping?pretty'
{
  "vi" : {
    "mappings" : {
      "properties" : {
        "title" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        }
      }
    }
  }
}
```

SUBFIELDS

MAPPING

- Data type
- Analyzer
- Subfield definitions

ANALYZER

- Which tokens are stored in Lucene inverted index
- Analyzer:
  - char filter
  - tokenizer
  - token filter

- https://www.elastic.co/guide/en/elasticsearch/reference/7.4/analysis-analyzers.html

CUSTOM MAPPING: ANALYZER DEFINITION

```
curl -XPUT -H 'Content-Type: application/json' localhost:9200/
vi2 -d '{
  "settings": {
    "analysis": {
      "analyzer": {
         "whitespace_asciifolding": {
           "tokenizer": "whitespace",
           "filter": ["asciifolding"]
         }
       }
     }
   }
}'
{"acknowledged":true,"shards_acknowledged":true,"index":"vi2"}
```

ANALYSIS DEBUGGER

```
curl -H 'Content-Type: application/json' 'localhost:9200/_analyze?pretty' -d '{"analyzer": "standard", "text":
"George Formby Sr (1875–1921) was one of the greatest music hall performers of the early 20th century."}'
{
  "tokens" : [
    {
      "token" : "george",
      "start_offset" : 0,
      "end_offset" : 6,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "formby",
      "start_offset" : 7,
      "end_offset" : 13,
      "type" : "<ALPHANUM>",
      "position" : 1
    },
    {
      "token" : "sr",
      "start_offset" : 14,
      "end_offset" : 16,
      "type" : "<ALPHANUM>",
      "position" : 2
     }, ....
```

CUSTOM MAPPING

```
curl -XPUT -H 'Content-Type: application/json' localhost:9200/vi3 -d '{
   "settings": {
      "analysis": {
         "analyzer": {
            "whitespace_asciifolding": {
               "tokenizer": "whitespace",
               "filter": ["asciifolding"]
             }
           }
         }
    },
    "mappings": {
       "properties": {
          "title": {
             "type": "text",
             "analyzer": "whitespace_asciifolding"
           },
           "date_at": {
              "type": "date",
              "index": false
            }
        }
     }
}'
```

```
curl -XPOST -H 'Content-Type: application/json'
'localhost:9200/vi3/_doc' -d '{"title":
"Elasticsearch", "date_at": "ahoj"}'

{"error":{"root_cause":
[{"type":"mapper_parsing_exception","reason":"fa
iled to parse field [date_at] of type [date] in
document with id 'KpCWtG4BPgLDs8deXxhG'....
```

LET'S PREPARE SOME DATA

```
curl -XPUT -H 'Content-Type: application/json' localhost:9200/search-test -d '{
   "settings": {
     "analysis": {
       "analyzer": {
          "whitespace_asciifolding": {
             "tokenizer": "whitespace",
             "filter": ["lowercase", "asciifolding"]
           }
        }
      }
    },
  "mappings": {
    "properties": {
       "title": {
         "type": "text",
         "analyzer": "whitespace_asciifolding"
       },
       "length": {
         "type": "integer"
       },
       "date_at": {
         "type": "date"
       }
    }
}
```

```
curl -XPOST -H 'Content-Type: application/json'
'localhost:9200/search-test/_doc' -d '{
   "title": "Inverted indices",
   "date": "1.10.2015",
   "length": 120,
   "description": "Deep dive into inverted
indices",
   "tags": ["search", "index"]
}'
```

MATCH QUERY

```
curl -XPOST -H 'Content-Type: application/json' 'localhost:9200/search-test/_search?pretty' -d '{
                  "query": {
                    "match": {
                      "title": "Indices"
                    }
                  }
                }'
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {...},
  "hits" : {
    "total" : {...},
    "max_score" : 0.9808292,
    "hits" : [
      {
        "_index" : "search-test",
        "_type" : "_doc",
        "_id" : "K5CptG4BPgLDs8delBh2",
        "_score" : 0.9808292,
        "_source" : {
          "title" : "Inverted indices",
          "date" : "1.10.2015",
          "length" : 120,
          "description" : "Deep dive into inverted indices",
```

TERM QUERY

```
curl -XPOST -H 'Content-Type: application/json' 'localhost:9200/search-test/_search?pretty' -d '{
                  "query": {
                    "term": {
                      "title": "Indices"
                    }
                  }
                }'
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 0,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  }
}
```

TERM QUERY VS. MATCH QUERY

- Match query
  - Analyzes input query using the field analyzer
- Term query
  - Does not touch input query

TERM QUERY

```
curl -XPOST -H 'Content-Type: application/json' 'localhost:9200/search-test/_search?pretty' -d '{
                  "query": {
                    "term": {
                      "title": "indices"
                    }
                  }
                }'
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {...},
  "hits" : {...},
    "max_score" : 0.9808292,
    "hits" : [
      {
        "_index" : "search-test",
        "_type" : "_doc",
        "_id" : "K5CptG4BPgLDs8delBh2",
        "_score" : 0.9808292,
        "_source" : {
          "title" : "Inverted indices",
          "date" : "1.10.2015",
          "length" : 120,
          "description" : "Deep dive into inverted indices",
          "tags" : [
```

RANGE QUERY

```
curl -XPOST -H 'Content-Type: application/json'
'localhost:9200/search-test/_search?pretty' -d
'{
   "query": {
      "range": {
         "length": {
            "gte": 90
         }
      }
   }
}'
```

BOOL QUERY

```
curl -XPOST -H 'Content-Type: application/json' 'localhost:9200/search-test/_search?
pretty' -d '
{
  "query": {
    "bool": {
      "must": {
        "range": {
          "length": {
            "gte": 90
          }
        }
      },
      "should": [
        {"match": {"tag": "search"}},
        {"match": {"tag": "java"}}
      ]
    }
  }
}
```

AGGREGATIONS

```
{
  "query": {
    "match_all": ""
  },
  "aggregations": {
     "by_length": {
        "histogram": {
          "field": "length",
          "interval": 10
        }
      },
      "stats": {
        "stats": {
          "field": "length"
        }
      }
    }
}
```

RESPONSE

```
 "aggregations" : {
    "by_length" : {
      "buckets" : [
        {
          "key" : 110.0,
          "doc_count" : 2
        },
        {
          "key" : 120.0,
          "doc_count" : 1
        }
      ]
    },
    "stats" : {
      "count" : 3,
      "min" : 110.0,
      "max" : 120.0,
      "avg" : 113.33333333333333,
      "sum" : 340.0
    }
  }
```

AGGREGATIONS

- Cannot be built from "text" type, you have to use "keyword" type, or turn on fielddata
  - Fielddata is disabled on text fields by default. Set fielddata=true on [tags] in order to load fielddata in memory by uninverting the inverted index. Note that this can however use significant memory. Alternatively use a keyword field instead.
- Can be nested
  - Gets very unreable very fast

```
"aggregations": {
   "top_tags": {
      "terms": {
        "field": "tags"
       },
       "aggregations": {
          "by_length": {
             "histogram": {
               "field": "length",
               "interval": 100
             }
           }
        }
      }
    }
  }
}
```

WHAT YOU SHOULD NOW

- Understand mapping and analysis process
- Write a simple bool query using match/term/range query combinations
- Difference between match and term query
- Write a simple aggregation using terms/histogram aggregation