ELASTICSEARCH

dla początkujących

Maciej Nowak

15-05-2018

ABOUT

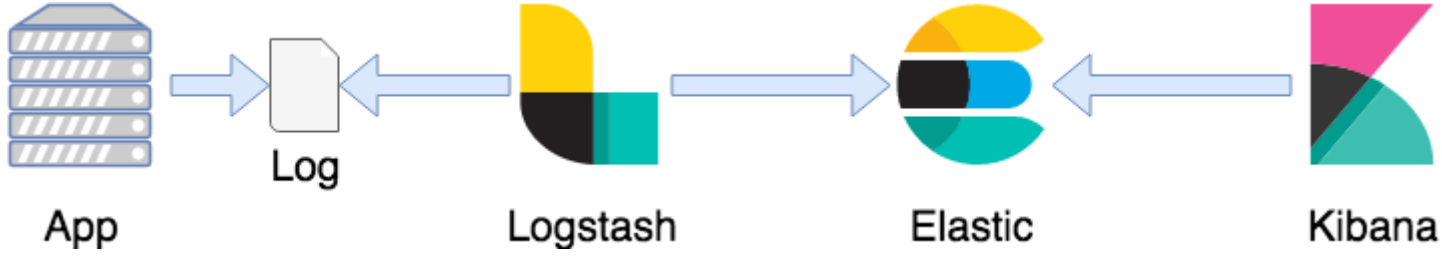
Java developer @ Decerto

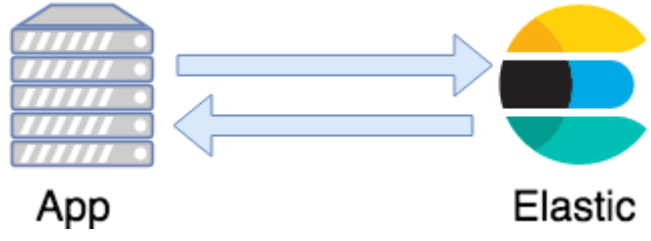
https://github.com/macnowak

maciej.nowak@outlook.com

WHAT WILL YOU LEARN

- What is Elastic?
- Installation
- API
- Indexing
- Searching
- Mapping
- Tools


App
Log
Logstash
Elastic
Kibana


App
Elastic

HISTORY

Shay Banon

Abstraction layer for Apache Lucene

First version was called Compass

First public release came out in February 2010

Open source from the beginning

1K contributors now

ELASTICSEARCH IS...

- Open source
- Aims to make full-text search easy
- NoSql DB
- Analytics engine
- Java
- Apache Lucene
- Inverted indexes
- Distributed
- Scalable
- REST api
- Schema Free*!
- High Availability
- Near Real Time

INSTALATION

```
//download
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.6.9.zip

//unzip

//run
$ bin/elasticsearch
```

IT'S ALIVE!!!

```
$ curl -XGET http://localhost:9200
```

```
{
  "name" : "elasticsearch",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "gSjeSDoXS62BTssMMj9OBg",
  "version" : {
    "number" : "5.6.9",
    "build_hash" : "877a590",
    "build_date" : "2018-04-12T16:25:14.838Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.1"
  },
  "tagline" : "You Know, for Search"
}
```

HOW DO WE TALK TO ELASTIC

```
$ curl -X <VERB><PROTOCOL>://<HOST>:<PORT>/<PATH>?<QUERY_STRING> -d <BODY>
```

```
VERB -> GET, POST, PUT, DELETE...
PROTOCOL -> http / https
HOST -> node host
PORT -> 9200 http api
PATH -> api endpoint eg. /_cat /_plugin
QUERY_STRING -> optional ?pretty
BODY -> json
```

BASICS

WHAT IS DATA?

```
{
    "partyId": 1,
    "firstName": "Jan",
    "name": "Kowalski",
    "label": "Jan Kowalski",
    "birthDate": "2015-01-04"
}
```

WRITE SOME DATA

```
$ curl -XPUT localhost:9200/party/party/1?pretty -d '{
  "partyId": 1,
  "firstName": "Jan",
  "name": "Kowalski",
  "label": "Jan Kowalski",
  "birthDate": "2015-01-04"
}'
```

```
{
  "_index": "party",
  "_type": "party",
  "_id": "1",
  "_version": 1,
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "created": true
}
```

READ DATA

```
$ curl -XGET localhost:9200/party/party/1?pretty
```

```
{
  "_index": "party",
  "_type": "party",
  "_id": "1",
  "_score": 1,
  "_source": {
    "partyId": 1,
    "firstName": "Jan",
    "name": "Kowalski",
    "label": "Jan Kowalski",
    "birthDate": "2015-01-04"
  }
}
```

CHANGE DATA

```
$ curl -XPUT localhost:9200/party/party/1?pretty -d '{
  "partyId": 1,
  "firstName": "Jan",
  "name": "Nowak",
  "label": "Jan Nowak",
  "birthDate": "2015-01-04"
}'
```

```
{
  "_index": "party",
  "_type": "party",
  "_id": "1",
  "_version": 2,
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": false
}
```

UPDATE = atomic DELETE + PUT

CHANGE SOMETHING MORE

```
$ curl -XPUT localhost:9200/party/party/1?version=1?pretty -d '{
  "partyId": 1,
  "firstName": "Jan",
  "name": "Zieliński",
  "label": "Jan Zieliński",
  "birthDate": "2015-01-04"
}'
```

```
{
  "error": {
    "root_cause": [...],
    "type": "version_conflict_engine_exception",
    "reason": "[party][1]: version conflict, current [2], provided [1]",
    "index": "party",
    "shard": "3"
  },
  "status": 409   ← http 409 conflict
}
```

Version → optimistic locking

CLEAN IT UP...

Document

```
$ curl -XDELETE localhost:9200/party/party/1?pretty
```

Type

```
$ curl -XDELETE localhost:9200/party/party?pretty
```

Index

```
$ curl -XDELETE localhost:9200/party?pretty
```

Delete → marks document as deleted → delete in background

WHAT IS INDEX ?

SOME SAY...

Elastic	RDMS
Index	DB
Type	Table
Document	Row

BUT...

*"**In the past** we tried to make elasticsearch **easier to understand** by building an analogy with relational databases: indices would be like a database, and types like a table in a database. **This was a mistake:** the way data is stored is so different that any comparisons can hardly make sense, and this ultimately led to an overuse of types in cases where they were more harmful than helpful."*

https://www.elastic.co/blog/index-vs-type

INDEX

An index is a collection of documents that have somewhat similar characteristics.

An index is stored in a set of shards, which are themselves Lucene indices.

Each shard is searched independently.

Each shard may have it's replica

Elasticsearch eventually needs to merge results from all the searched shards.

10 indices that have 5 shards each, 50 results need to be merged.

WHAT IS SHARD?

Shard is a box…

Each shard is in itself a fully-functional and independent "index" that can be hosted on any node in the cluster.

SHARDING

Allows to horizontally split/scale your content volume

Allows you to distribute and parallelize operations across shards / nodes → increasing performance

WHAT IS REPLICA?

It's a copy of shard

Replica shard is never allocated on the same node as the original/primary shard.

REPLICATION

Provides high availability in case a shard/node fails.

Allows to scale out your search volume/throughput since searches can be executed on all replicas in parallel.

Index = Shards + Replicas

INDEX CONFIGURATION

```
$ curl -XPUT localhost:9200/party -d '{
  "settings": {
        "index": {
          "number_of_replicas": "1",
          "number_of_shards": "5"
        }
  }
}'
```

TYPE

Types are a convenient way to store several types of data in the same index.

Lower total number of indices

“_type” field on every document

One index with many types ?

OR

Many indices with one type ?

IT DEPENDS...

- Documentations says : Do your documents have similar mappings? If no, use different indices.
- Stackoverflow says : http://stackoverflow.com/questions/14465668/elastic-search-multiple-indexes-vs-one-index-and-types-for-different-data-sets

PROBLEM SOLVED

Since elastic 5.6: Type is deprecated...

https://www.elastic.co/guide/en/elasticsearch/reference/master/removal-of-types.html

Elastic **Search**

BASIC TOOLS

- Kibana
- Dev Tools
  (Sense)

HOW TO SEARCH

```
$ curl -XGET localhost:9200/party/party/_search
```

```
$ curl -XGET localhost:9200/party/_search
```

```
$ curl -XGET localhost:9200/p*/_search
```

```
$ curl -XGET localhost:9200/p*/type1,type2/_search
```

```
$ curl -XGET localhost:9200/party/party/_search
```

```
{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "party",
      "_type" : "party",
      "_id" : "1",
      "_score" : 1.0,
      "_source" : {
        "partyId" : 1,
        "firstName" : "Jan",
        "name" : "Nowak",
        "label" : "Jan Nowak",
        "birthDate" : "2015-01-04"
      }
    } ]
  }
}
```

WHAT IS SCORE?

_score → how returned document is relevant to query

_score is based on term frequency, inverse document frequency, and field-length norm

https://www.elastic.co/guide/en/elasticsearch/guide/current/relevance-intro.html

QUERY SYNTAX

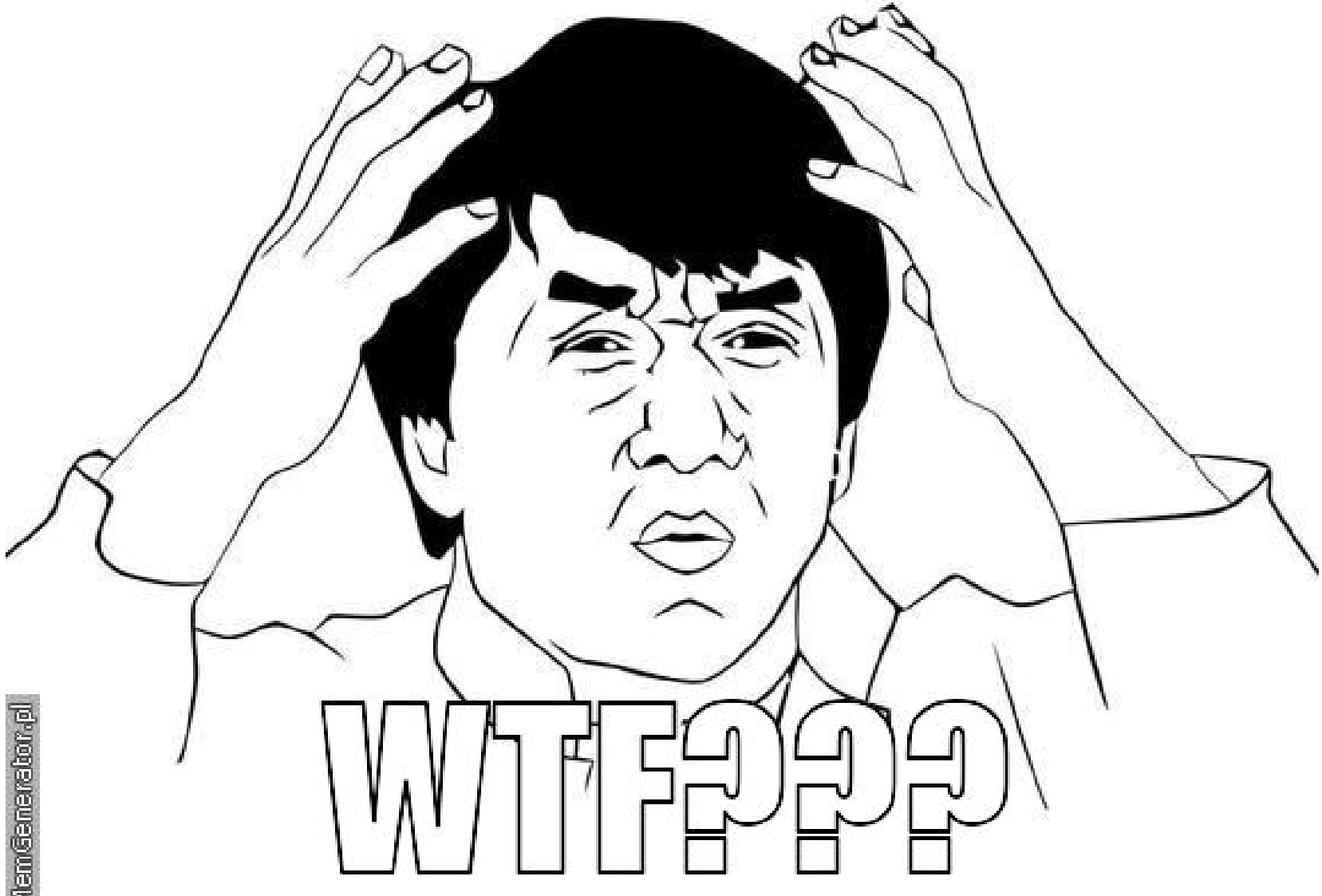
URI Search

```
$ curl -XGET "http://localhost:9200/party/party/_search?q=name:Kowalski&pretty"
```

Request Body Search

```
$ curl -XPOST "http://localhost:9200/party/party/_search" -d
'{
 "query": {
    "match": {
      "name": "Kowalski"
    }
  }
}'
```

DEMO 1



HOW IT WORKS?

Full text search by default !

“The quick brown fox jumped over the lazy dog”

“Quick brown foxes leap over lazy dogs in summer”

SEPARATE WORDS / TERMS

The,quick,brown,fox,jumped,over,the,lazy,dog

Quick,brown,foxes,leap,over,lazy,dogs,in,summer

SORT TERMS

The,brown,dog,fox,jumped,lazy,over,quick,the

Quick,brown,dogs,foxes,in,lazy,leap,over,summer

Term	Document 1	Document 2
Quick	X	
The		X
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick		X
summer		X
the	X	

Term	Document 1	Document 2
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	X
summer		X
the	X	X

Term	Document 1	Document 2
brown	X	X
dog	X	X
fox	X	X
in		X
jumped	X	X
lazy	X	X
over	X	X
quick	X	X
summer		X
the	X	X

"ANALYSIS"

tokenization + normalization

"ANALYZERS"

tokenizer + token filters

STANDARD ANALYZER

standard tokenizer → The,Quick,Brown,Fox,jumped,over,the,Lazy,Dog

lowercase filter → the,quick,brown,fox,jumped,over,the,lazy,dog

stopwords filter → ,quick,brown,fox,jumped,over, ,lazy,dog

SCHEMA FREE??

NOT EXACTLY...

MAPPINGS

```
$ curl -XGET localhost:9200/party/party/_mapping?pretty
```

```
{
    "mappings": {
      "party": {
        "properties": {
          "birthDate": {
            "type": "date",
            "format": "strict_date_optional_time||epoch_millis"
          },
          "name": {
            "type": "text"
          },
          "partyId": {
            "type": "long"
          }
        }
      }
    }
}
```

DEFINED TYPES

Strings	string (pre v5.0), text, keyword
Datetimes	date
Whole numbers	byte, short, integer, long
Floats	float, double
Booleans	boolean
Objects	object
Also	multi_field, ip, geo_point, geo_shape,

	v2.x	v5.x	v6.x
Full text (default)	{"type":"string", "index":"**analyzed**"}	{"type":"string", "index":"**analyzed**"} {"type":"text"}	{"type":"text"}
Exact String	{"type":"string", "index":"**not_analyzed**"}	{"type":"string", "index":"**not_analyzed**"} {"type":"keyword"}	{"type":"keyword"}
Not searchable	{"type":"string", "index":"**no**"}	{"type":"string", "index":"**no**"} {"type":"text | keyword", "index":"**false**"}	{"type":"text | keyword", "index":"**false**"}

ADD MAPPING

```
PUT /party/_mapping
{
    "mappings": {
      "party": {
        "properties": {
          "birthDate": {
            "type": "date",
            "format": "strict_date_optional_time||epoch_millis"
          },
          "name": {
            "type": "keyword"
                  },
          "partyId": {
            "type": "long"
          }
        }
      }
    }
}
```

Can’t change the mapping !!

Can add new fields

DEMO 2

Real life example?

AUTOCOMPLETE

N-grams == window-on-a-word:

Jan Kowalski

Length 1: j,a,n,k,o,w,a,l,s,k,i

Length 2: ja,an,nk,ko,ow,wa,al,ls,ki

Length 3: jan,ank,nko,kow,owa,wal,als,lsk,ski

Length 4: jank,anko,nkow,kowa,owal,wals,alsk,lski

...

FIRST DEFINE FILTER

```
"filter": {
          "nGramFilter": {
            "type": "nGram",
            "min_gram": 3,
            "max_gram": 40
          }
        }
```

SECOND DEFINE ANALYZER

```
"analyzer": {
         "search_analyzer": {
           "type": "custom",
           "tokenizer": "keyword",
           "filter": [
             "lowercase"
           ]
         },
         "autocomplete_analyzer": {
           "type": "custom",
           "tokenizer": "whitespace",
           "filter": [
             "lowercase",
             "nGramFilter"
           ]
         }
       }
```

THIRD DEFINE FIELD

```
"name": {
          "type": "text",
          "analyzer": "standard_analyzer",
          "fields": {
            "autocomplete": {
              "type": "text",
              "analyzer": "autocomplete_analyzer", <-- handles data
              "search_analyzer": "search_analyzer" <-- handles query
            }
          }
        }
```

DEMO 3

WHEN SOMETHING IS NOT WORKING...

```
GET /party/party/25/_termvector?fields=name.autocomplete
```

```
GET /_analyze
{
"tokenizer": "keyword",
"filter": ["lowercase"],
"text" : "Nowak-kowa"
}
```

DEMO 4

ARRAYS

"addresses": { "type":"object", …}

By default JSON document is flattened into a simple key-value format :

```
addresses.street : [“Marszałkowska”,”Elektoralna” ]
addresses.city : [“Gdynia”,”Solec-Kujawski”]
```

SOLUTION:

"addresses": { "type":"nested", …}

DEMO 5

MONITORING

- Hq - https://github.com/ElasticHQ/elasticsearch-HQ
- Cerebro (Kopf) - https://github.com/lmenezes/cerebro

JUST THE TIP OF THE ICEBERG...

Documentation : https://www.elastic.co/guide/index.html

QUESTIONS?

FEEDBACK

maciej.nowak@outlook.com