elasticsearch

1

Introduction

- elasticsearch
  - NoSQL Search engine
  - Document-oriented NoSQL
  - JSON documents
  - Implemented in Java
  - Rely on:
- Lucene
  - Full-text indexing
  - Complex search queries on text

2

Applications with Elasticsearch

Companies:

- Uber
- Instacart
- Stack Overflow
- Shopify
- Udemy
- Expedia
- …

Integrated in:

- Datadog
- Couchbase
- Amazon
- Jaeger
- …

Evolutions

- V1.0 (2014)
  - Query/Get/Update APIs
- V2.0 (2015)
  - Custom config file, packaging, plugins
- V5.0 (2016)
  - Cluster enhancement, core evolutions, optimizations, mapping corrections
- V6.0 (2017)
  - Changes: mapping types, aggregations, cluster, indices, Java API, packaging, REST, Query DSL, scripting…
- V6.6 (2019)
  - Frozen indices, Index Lifecycle, BKD-backed Geoshapes

ELK Stack

- **Elasticsearch**
  - NoSQL search engine
- **Logstash**
  - Data collection pipeline tool
- **Kibana**
  - Data visualization tool

5

ELK Stack

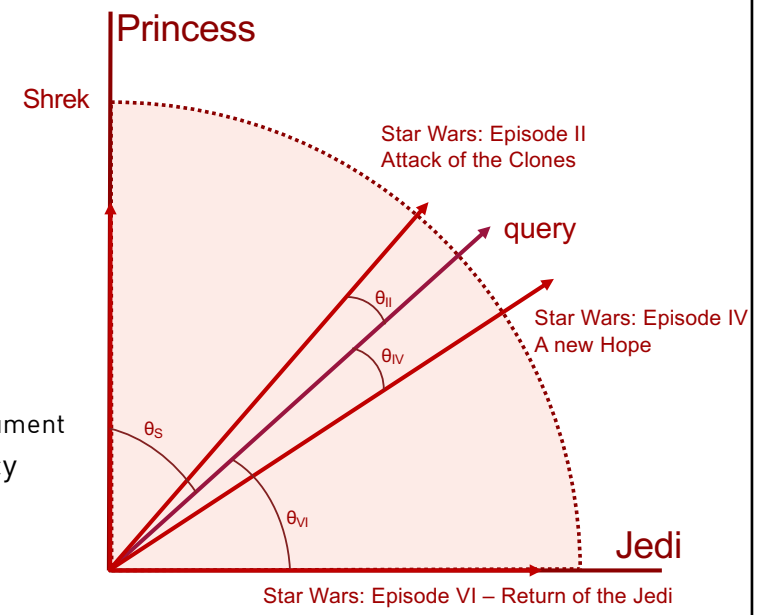
6

Elasticsearch RESTful API

- **cURL**[1] (executable for HTTP requests)
- Import data
  - curl -XPUT localhost:9200/_bulk -H"Content-Type: application/json" --data-binary @file.json
  - Dataset:
    - **Each** JSON document must be **prefixed** by a **header**
      - {"index": {"_index" : "INDEXNAME" "_id": X}}
      - Index = collection (table)
    - Each document must not contain an "id" key
- GET
  - Standard query: curl -XGET 'http://localhost:9200/INDEXNAME/_search?q=some+words'
  - Smart query (DSL[2]): curl -H"Content-Type: application/json" -XGET 'http://localhost:9200/INDEXNAME/_search' -d @queryFile
  - RESTful Integrated in Kibana (Dev Tools)
  - Suppose the index is "movies" and type "movie"

1 – https://curl.haxx.se/download.html
2 – DSL: Domain Specific Language

- Search Engine
  - Similarity between
    - The query: *q*
    - A textual document: *d*
  - Relevance score[1]: cos (q,d)
- The cosinus rely on
  - Term Frequency (**TF**)
    - Normalized per key or per document
  - Inverse Document Frequency (**IDF**)



1 – ranking : http://b3d.bdpedia.fr/ranking.html

DSL – Simple Queries

- Standard queries
  - Whole document: http://localhost:9200/movies/_search?q=**Star+Wars**
  - Within a key: http://localhost:9200/movies/_search?q=**title:**Star+Wars
  - Two keys: http://localhost:9200/movies/_search?q=**title:**Star+Wars AND actors:Harrison
- DSL
  - Document query: { "**query**": { "**match**": { "*title*":"Star Wars" }}}
  - Boolean queries:
    - should {"**query**":{ "**bool**": { "***should***": **[**
      { "match": { "*title*": "Star Wars" }},
      { "match": { "*actors*": "Harrison" }} **]** }}}
    - must/must_not {"**query**":{ "**bool**": {
      "***should***": { "match": { "*title*": "Star Wars" }},
      "***must***": { "match": { "*actors*": "Harrison" }} }}}
    - match_phrase {"**query**":{"***match_phrase***": {"*title*": "Star Wars" }}}
    - Range queries {"query": { "**bool**": { "must": { "**range**": { "*rank*": {"**lt**":1000 }}} }}}
      {"query": { "**bool**": { "must": { "**range**": { "*date*" : {"**from**": "2010-01-01", "**to**": "2015-12-31"}}}}}

DSL – Complex Queries

- Aggregate queries:
  - Simple group
    {"**aggs**": { "*produced_key*": { "**terms**": { "field": "*year*"}}}}
    {"**aggs**": { "*produced_key*": { "**terms**": { "field": "*actors*"}}}}
  - Group by range
    {"**aggs**": { "*produced_key*": { "**range**" : {
    "field": "*year*", "**ranges**":[{"from":..., "to":... }] }}}}}
  - Number of distinct values
    {"**aggs**": { "*produced_key*": { "**cardinality**": { "field": "*actors.keyword*"} }}}
  - Averages/Min/Max
    {"**aggs**": { "*produced_key*": { "**avg**": { "field": "*rating*"} }}}
  - Composition *(maybe "hard" queries)*
    {"**aggs**": { "*produced_key*": { "**terms**": { "field": "*year*"}, "**aggs**":{"avg" : ... } }}}}

DSL – Hard Queries with Mapping[1]

- In order to group keyword values
  - Query:

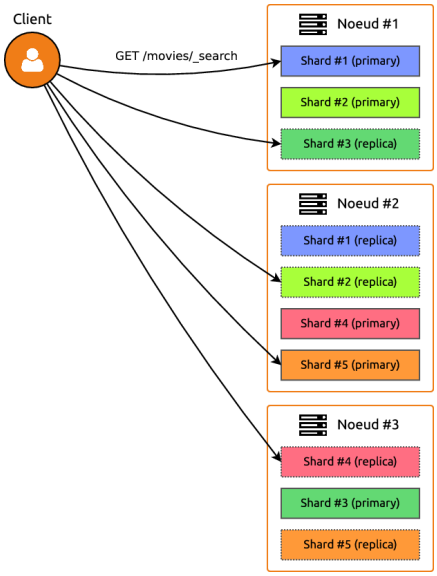
    { "query" :{"match" : {"title" : "Star Wars"}},
    "aggs" : { "top_keywords" : { "**significant_terms**" : {"field" : "*plot*"}}}}
  - Top keywords extraction
  - !!! This can consume a **lot** of memory
  - Need to map keys with type "fielddata"

    PUT /movies/movie/_mappings
    { "properties": { "plot": { "type": "text", "**fielddata**": true } } }

1 – Report on your dataset: Mapping can be used for "data model & import"

« Sharding » : Distribution & Replication

- Cluster
  - Must be **set** at the <u>beginning</u> of the index
  - **Static** hash function
  - Split the index in X fragments
  - Replicated on 2 other nodes



Kibana

- Dashboard to visualize
  - Differente chart types
- Intuitive interface
- Can handle:
  - Time-series
  - Geo-shapes (maps)
  - IP/Images/Dates
  - Tag Clouds