Elasticsearch

by
Irina Benediktovich

github.com/deffer

Prerequisites

elasticsearch.

JSONView

Simple REST Client

https://github.com/deffer/training/

After this training you ...

- Will understand how search engines work
- Know general search terminology
- Have experience creating elasticsearch index
- Have experience running different types of searches
- Know how to troubleshoot your problems
- Be able to implement searches that meet most common user demands
- Know what is available to you if user needs more

Content

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

In the world

Github

stackoverflow

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

SOUNDCLOUD

Kibana

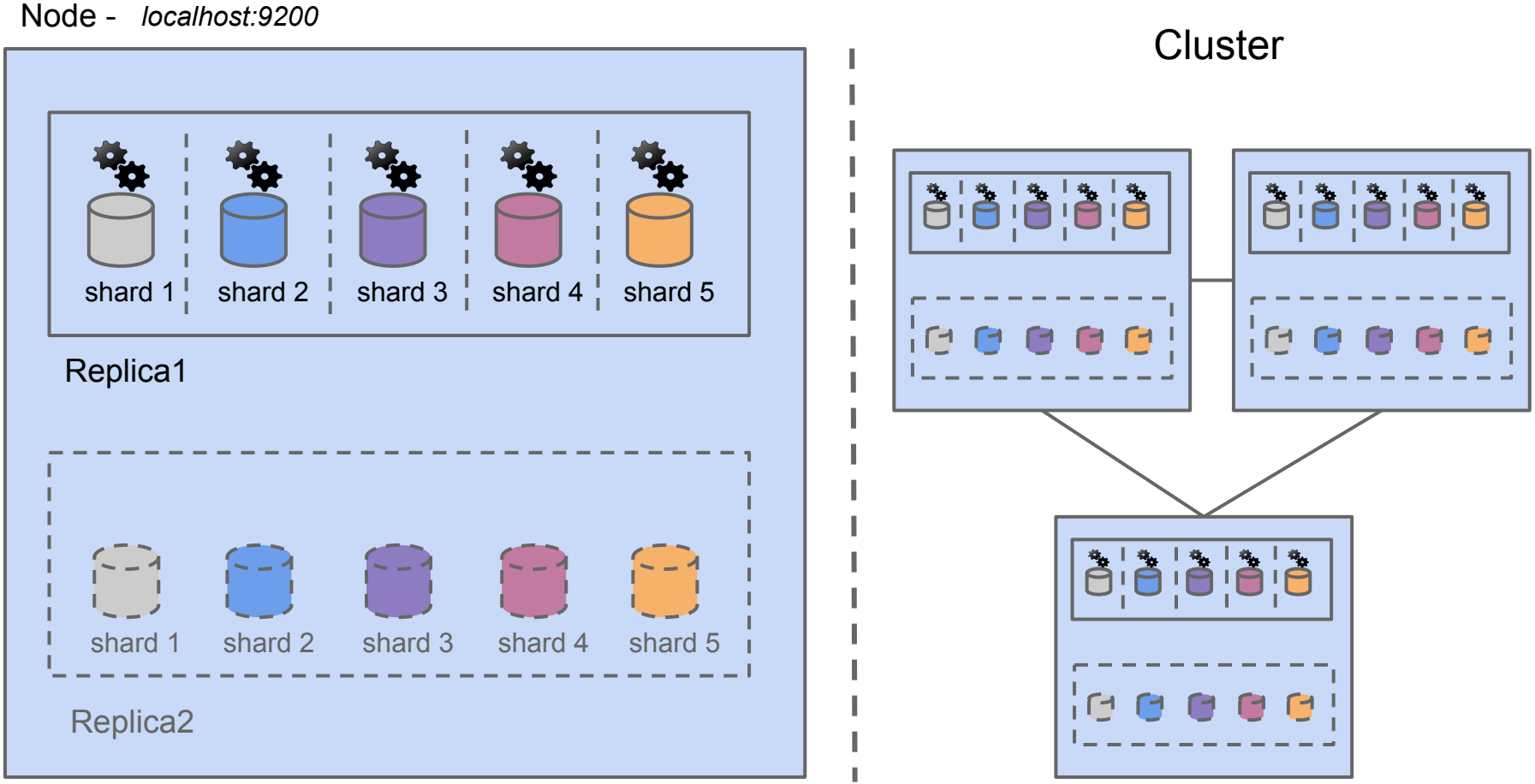
+

Logstash

next...

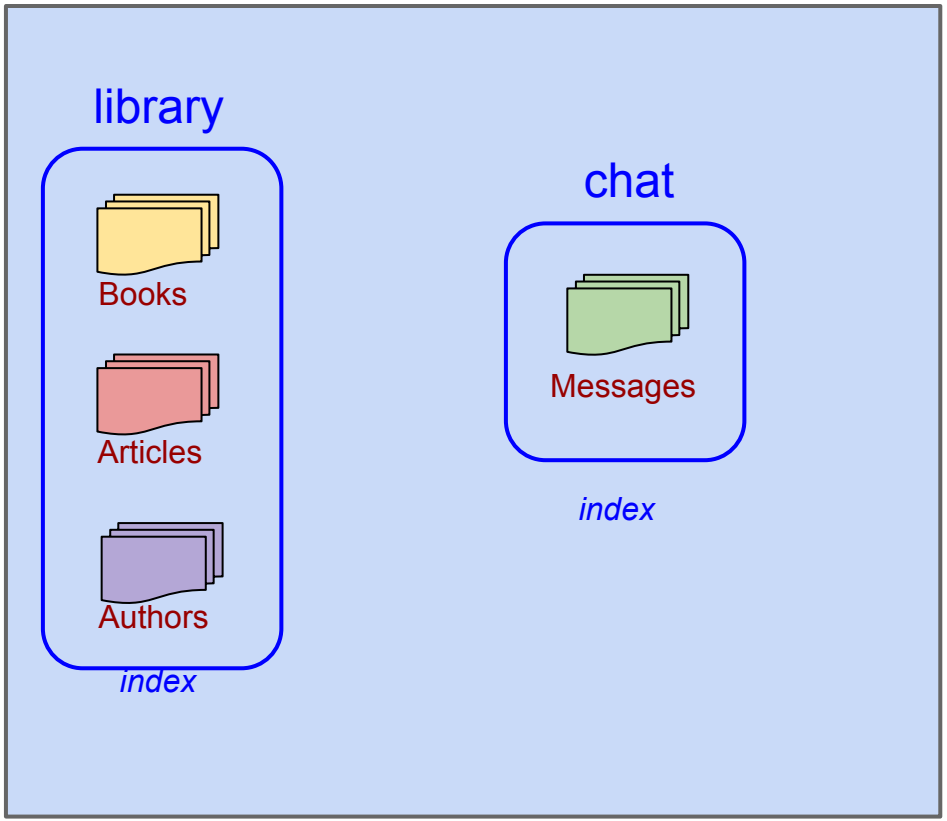
- General description
- Index/Document organisation
  cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  count, paging, highlights, explain
- Searching
  in fields, using wildcards, using analyzer
- Understanding index
  index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  term, boolean, etc...
- Other types of queries
  filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

Deployment view



Data organisation

Node



Every index has 1 or more document types.

Document type is equivalent to a table in the database.

Since elastic is NoSQL database, there are no relationships between those tables.

library

Authors	Articles	Books
name	title	title
dob	authorName	authorName
biography	pub_date	pub_date
pub_titles	abstract	revision
	body	intro
	references	chapters
		notes
		references


host
index
document
type
http://elasticsearch:9200/library/books

How many indices and document types I should have in my application?

Example

EPR: 1 index, 1 document (person).

FAT: 1 index, 1 doc (researchEntry).

PCF: 1 index, 2 docs (undergrad, postgrad).

Logstash: 100+ (1 per day of logs) indices.

Number of document types is same as number of message types.

Create index

```
$ POST http://localhost:9200/cat
```

View index

```
$ GET http://localhost:9200/cat/_status
```

Delete index

```
$ DELETE http://localhost:9200/cat
```

Create index with required settings

```
$ POST http://localhost:9200/cat

{"settings":{"index":{
    "analysis":{
      "analyzer":{
                "sb_analyzer":{"type":"snowball","language":"English"}
            }
     },
    "number_of_shards" : "1"
}}}
```

_status

default

```
{
  ok: true,
- _shards: {
    total: 10,
    successful: 5,
    failed: 0
  },
- indices: {
  - cat: {
      + index: {…},
      + translog: {…},
      - docs: {
          num_docs: 0,
          max_doc: 0,
          deleted_docs: 0
        },
      + merges: {…},
      + refresh: {…},
      + flush: {…},
      - shards: {
          + 0: […],
          + 1: […],
          + 2: […],
          + 3: […],
          + 4: […]
        }
    }
  }
}
```

5 shards

custom

```
{
  ok: true,
- _shards: {
    total: 2,
    successful: 1,
    failed: 0
  },
- indices: {
  - cat: {
      + index: {…},
      + translog: {…},
      - docs: {
          num_docs: 0,
          max_doc: 0,
          deleted_docs: 0
        },
      + merges: {…},
      + refresh: {…},
      + flush: {…},
      - shards: {
          + 0: […]
        }
    }
  }
}
```

1 shard

_settings

```
$ GET http://localhost:9200/cat/_settings
```


```
- cat: {
    - settings: {
          index.analysis.analyzer.sb_analyzer.type: "snowball",
          index.analysis.analyzer.sb_analyzer.language: "English",
          index.number_of_shards: "1",
          index.number_of_replicas: "1",
          index.version.created: "190899"
      }
  }
```


Compare with EPR


```
- epr: {
    - settings: {
          index.analysis.analyzer.phonetic_analyzer.filter.3: "stop",
          index.analysis.analyzer.phonetic_analyzer.filter.4: "unique",
          index.analysis.analyzer.phonetic_analyzer.filter.5: "custom_metaphone",
          index.analysis.analyzer.phonetic_analyzer.tokenizer: "whitespace",
          index.analysis.filter.custom_metaphone.replace: "false",
          index.analysis.analyzer.phonetic_analyzer.filter.0: "trim",
          index.analysis.filter.custom_metaphone.type: "phonetic",
          index.analysis.analyzer.phonetic_analyzer.filter.1: "lowercase",
          index.analysis.filter.custom_metaphone.encoder: "double_metaphone",
          index.analysis.analyzer.phonetic_analyzer.filter.2: "asciifolding",
          index.number_of_shards: "5",
          index.number_of_replicas: "1",
          index.version.created: "190899"
      }
  }
```


Compare with default


```
- library: {
    - settings: {
          index.number_of_shards: "5",
          index.number_of_replicas: "1",
          index.version.created: "190899"
      }
  }
```

to view all document types...

_mapping

```
$ GET http://localhost:9200/library/mapping
```

```
- library: {
    + author: {…},
    + article: {…},
    + book: {…}
  }
```

3 document types

EPR

```
- epr: {
    + person: {…}
  }
```

1 document type

Logstash

```
- logstash-2014.06.08: {
    - mappings: {
        + apache_access: {…},
        + _default_: {…},
        + apps: {…}
      }
  }
```

3 document types

to view all document types...

_mapping

to view document structure...

```
$ GET http://localhost:9200/library/mapping
```

```
- library: {
    + author: {…},
    + article: {…},
    + book: {…}
  }
```

3 document types

EPR

```
- epr: {
    + person: {…}
  }
```

1 document type

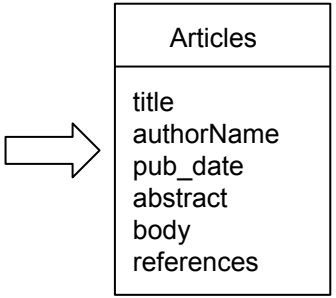
Logstash

```
- logstash-2014.06.08: {
    - mappings: {
        + apache_access: {…},
        + _default_: {…},
        + apps: {…}
      }
  }
```

3 document types

```
$ GET http://localhost:9200/library/article/mapping
```

```
- article: {
    - properties: {
        + abstract: {…},
        + author_name: {…},
        + body: {…},
        + publication_date: {…},
        + references: {…},
        + title: {…}
      }
  }
```



Logstash

```
- apache_access: {
    + dynamic_templates: […],
    - properties: {
        + @timestamp: {…},
        + @version: {…},
        + agent: {…},
        + bytes: {…},
        + client: {…},
        + cookies: {…},
```

next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

_mapping

To put a mapping, issue a POST request to `/index/document/_mapping` with body containing mapping JSON.

```
POST http://localhost:9200/library/article/_mapping
```

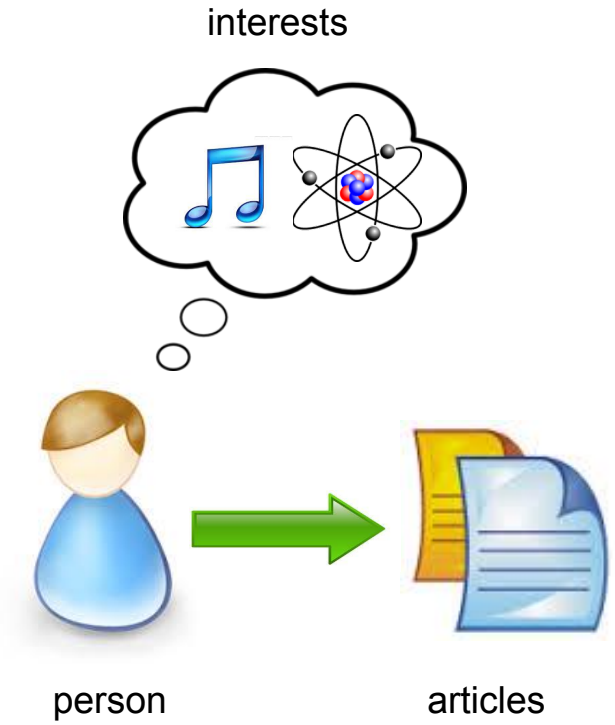
```
{"article":{
      "properties":{
            "title":{"type":"string", "index":"analyzed", "boost":"5.0"},
            "publication_date":{"type":"date", "format": "yyyy-MM-dd",
                                "store":"yes", "index":"not_analyzed"},
            "author_name":{"type":"string", "index":"analyzed", "boost":"5.0"},
            "abstract":{"type":"string", "index":"analyzed", "boost":"4.0"},
            "body":{"type":"string", "index":"analyzed"},
            "references":{"type":"string", "index":"analyzed"}
      }}
}
```

http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/mapping-core-types.html

Exercise

In the previously created index ‘cat’, create a ‘person’ mapping.

Example JSON from STEP 1 of attached document.



GET http://localhost:9200/cat/person/_mapping

```
- person: {
    - properties: {
        - articles: {
            - properties: {
                + body: {…},
                + title: {…},
                + topic: {…},
                + type: {…}
              }
        },
        - description: {
            type: "string",
            analyzer: "sb_analyzer"
        },
        - dob: {
            type: "date",
            format: "yyyy-MM-dd"
        },
        - group: {
            type: "long"
        },
        - interests: {
            type: "string",
            index: "not_analyzed"
        },
        - keywords: {
            type: "string",
            boost: 5,
            analyzer: "sb_analyzer"
        },
```

Inner object with its own properties

```
"articles":{
      "type" : "object",
      "properties":{
            "type":{"type":"string", …},
            "topic" :{"type":"string", … "index":"not_analyzed"},
            "title":{"type":"string",… "boost":"2.0"},
            "body":{"type":"string", …"analyzer":"sb_analyzer"}
      }
}
```

```
"dob":{"type":"date", "format": "yyyy-MM-dd", ...}
```

Saving document

To insert a document, issue a POST / PUT request to `/index/document/id` with the JSON body

Example:

```
POST http://localhost:9200/library/article/1
```


```
{
"title": "ON PROOF AND PROGRESS IN MATHEMATICS",
"publication_date": "2014-02-12",
"author_name":"WILLIAM P. THURSTON",
"abstract":"This essay on the nature of proof and progress in mathematics was stimulated by the article of Ja?e and Quinn, Theoretical Mathematics: Toward a cultural synthesis of mathematics and theoretical physics. Their article raises interesting issues that mathematicians should pay more attention to, but it also perpetuates...",
"body":"There are many issues buried in this question, which I have tried to phrase in a way that does not presuppose the nature of the answer. It would not be good to start, for example, with the question How do mathematicians prove theorems? This question introduces an interesting topic, ...",
"references": [
        "Simon, Julian . THE ULTIMATE RESOURCE (Princeton, N.J. : Princeton University Press, c1981)",
        "Simon, Julian (editor). THE STATE OF HUMANITY (Blackwell, Cambridge, MA and Oxford, UK) 1995."
        ]
}
```

retrieving document

GET http://localhost:9200/library/article/1


```
  _index: "library",
  _type: "article",
  _id: "1",
  _version: 1,
  exists: true,
- _source: {
      title: "ON PROOF AND PROGRESS IN MATHEMATICS",
      publication_date: "2014-02-12",
      author_name: "WILLIAM P. THURSTON",
      abstract: "This essay on the nature of proof and pro
      article raises interesting issues that mathematician
      body: "There are many issues buried in this question
      mathematicians prove theorems? This question introdu
      of mathematical proof, and (2) that progress made by
      not even How do mathematicians make progress in math
    - references: [
          "Simon, Julian . THE ULTIMATE RESOURCE (Princeto
          "Simon, Julian (editor). THE STATE OF HUMANITY (
      ]
  }
```


Exercise

Add 7 persons to the ‘cat’ index, with ids 1-7.

Example JSONs from STEP 2.1-2.7 of attached document.

_search

GET http://localhost:9200/library/article/_search?q=proof


```
took: 62,
timed_out: false,
- _shards: {
    total: 5,
    successful: 5,
    failed: 0
},
- hits: {
    total: 1,
    max_score: 0.13212296,
  - hits: [
    - {
        _index: "library",
        _type: "article",
        _id: "1",
        _score: 0.13212296,
      + _source: {…}
    }
  ]
}
```


1

Exercise

Run search for words:

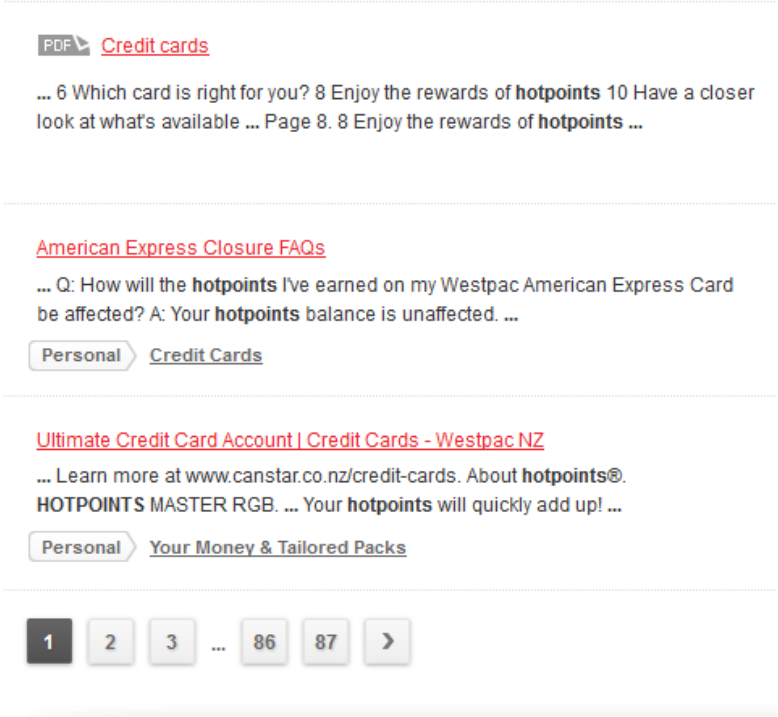
- anna
- physics
- duck
- rain

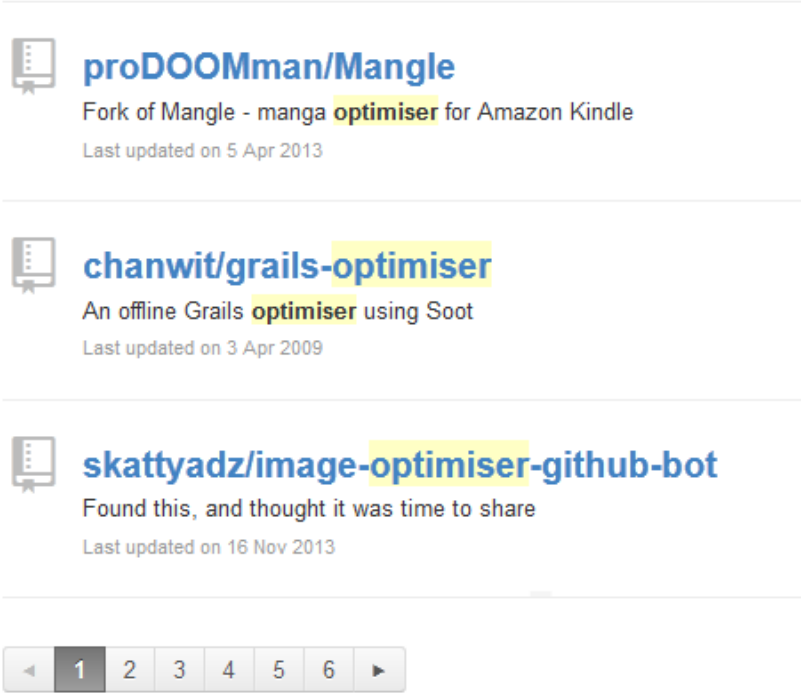
```
GET http://localhost:9200/cat/person/_search?q=anna
```

next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain (using logstash to demonstrate paging)
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

Paging search results





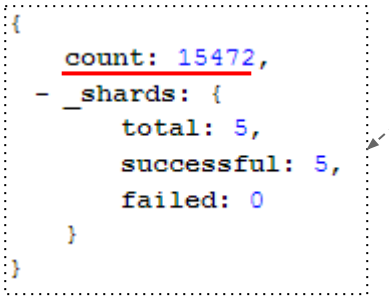
github.com/deffer

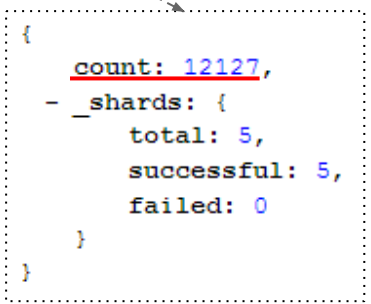
_count

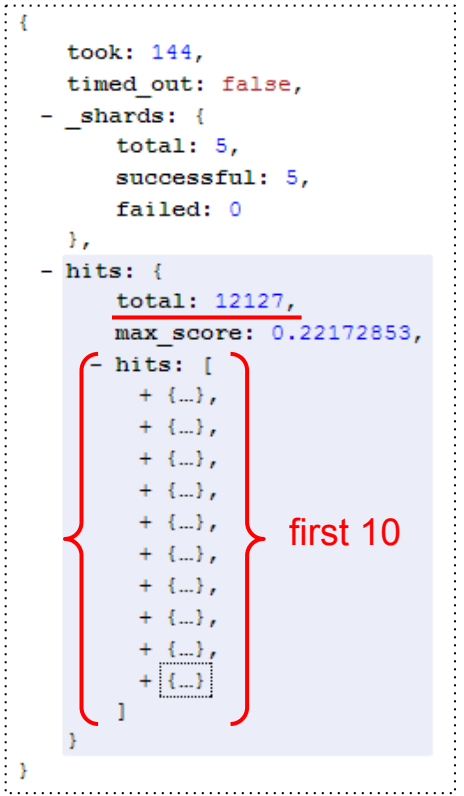
//*host*:9200/logstash-2014.06.08/**_search**?q=get

//*host*:9200/logstash-2014.06.08/**_count**?q=get

//*host*:9200/logstash-2014.06.08/**_count**

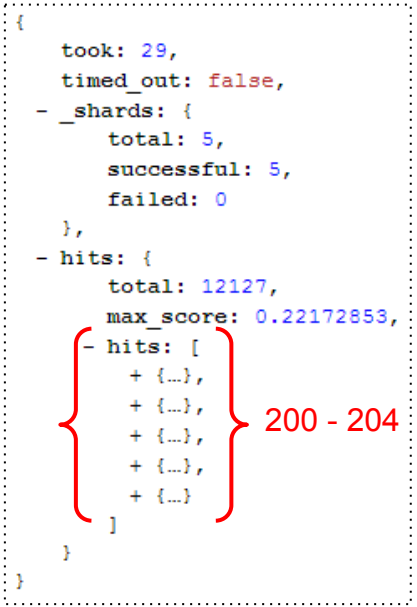


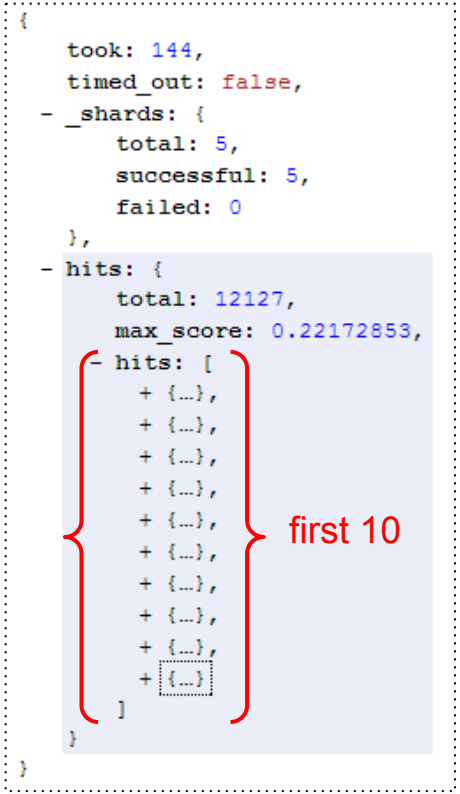




Paging

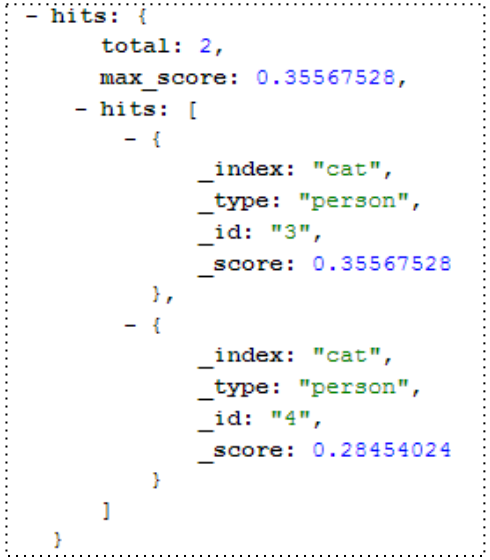
//*host*:9200/logstash-2014.06.08/**_search**?q=get&**from**=200&**size**=5





Specifying return fields

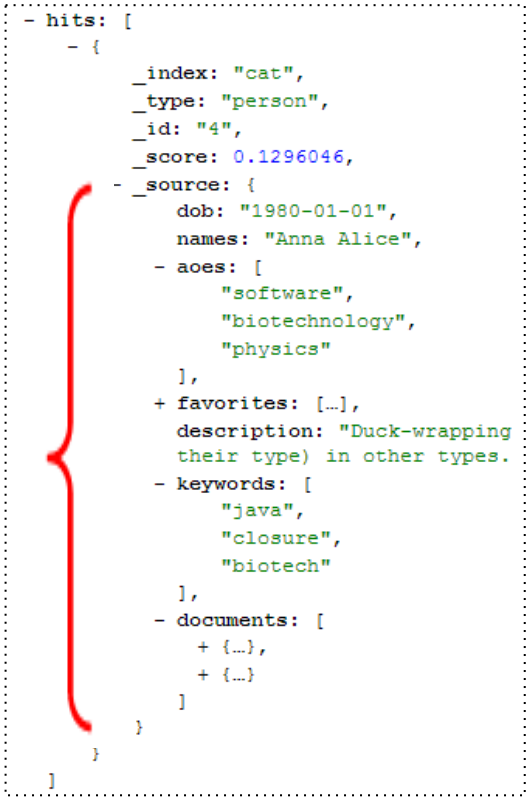
GET http://localhost:9200/cat/person/_search?q=anna&fields=id



id and score

id and score

whole entry



Highlights

query = duck

```
"highlight" : {
      "description" : [
         "<em>Duck</em>-wrapping (verb): If it doesn't quack like a <em>duck</em>, wrap it in a <em>duck</em>. JavaScript has"
         ]
   }
```

query = express

```
"highlight" : {
      "documents.body" : [
         "Each specialized cell type in an organism <em>expresses</em> a subset of all the genes that constitute",
         " <em>expression</em>. Cell differentiation is thus a transition of a cell from one cell type to another",
         " and it involves a switch from one pattern of gene <em>expression</em> to another. Cellular differentiation during" ]
   }
```

POST equivalent

POST http://localhost:9200/cat/person/_search

```
{
"query":  { "query_string" : {
     "query": "duck",
     "fields": ["names", "aoes", "description", "keywords", "documents.*" ]
    }
 },

 fields: [
     "dob", "names", "aoes", "favorites", "description", "keywords", "documents"
     ],

 highlight: {
    fields: {names:{}, aoes:{}, description:{}, keywords:{}, "documents.body":{}, "documents.title":{}}
 },

 from: 0,
 size: 100
}
```

next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

Searching inside specific field

http://localhost:9200/cat/person/_search?q=description:library

http://localhost:9200/cat/person/_search?q=articles.body:sun

Searching several terms

http://localhost:9200/cat/person/_search?q=sun → 1 result

http://localhost:9200/cat/person/_search?q=verb → 1 result

http://localhost:9200/cat/person/_search?q=sun verb → 2 results ‘sun’ OR ‘verb’

Changing default operator

../cat/person/_search?q=sun verb&default_operator=AND → No results

‘sun’ AND ‘verb’

Wildcards

http://localhost:9200/cat/person/_search?q=articles.*:physics

http://localhost:9200/cat/person/_search?q=description:re → No results

http://localhost:9200/cat/person/_search?q=description:re* → 3 results

```
"highlight" : {
      "description" : [ " <em>reason</em> about the code in question, and it can complicate some logic with extra boilerplate." ]
    }

"highlight" : {
      "description" : [
          " depending on the program mode. The routine to move the cursor on screen in <em>response</em> to mouse movement can",
          " be written for cursor, and polymorphism lets that cursor take whatever shape it <em>requires</em> at runtime. ..." ]
          }
```

Analyzers > wildcards

http://localhost:9200/cat/person/_search?q=description:security → secured


```
  _index: "cat",
  _type: "person",
  _id: "5",
  _score: 0.28159538,
- fields: {
    - description: [
          "We'll use SSH to create a pair of secured network tunnels that we can use to send
          and receive our unencrypted email. You might be able to create just one tunnel for
          retrieving email, but some providers won't let you send via SMTP unless you recently
          checked your email from the same machine. To keep them from thinking we're spammers,
          we'll create tunnels for both connections."
      ]
  }
```

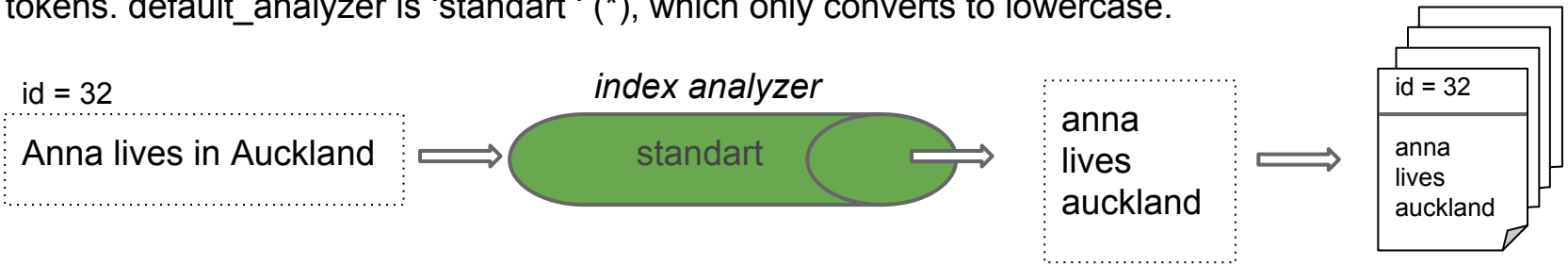
next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

Index analyzer

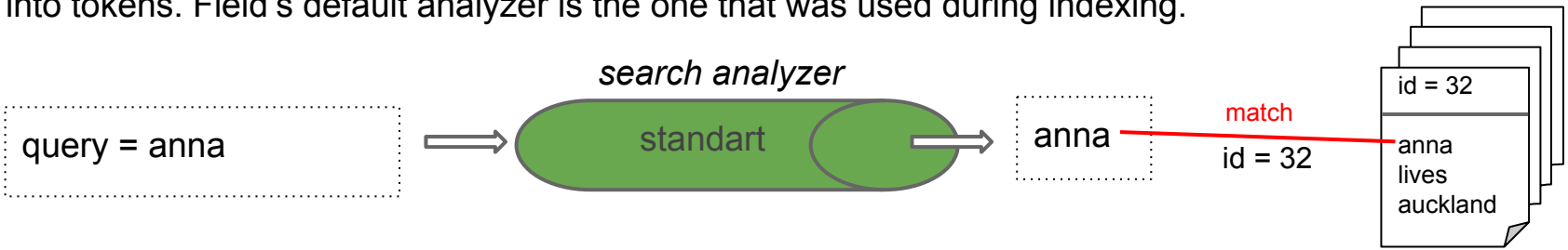
id: 32

If nothing is specified in the mapping, default_analyzer is used to convert all incoming words into tokens. default_analyzer is ‘standart ‘ (*), which only converts to lowercase.



Search analyzer

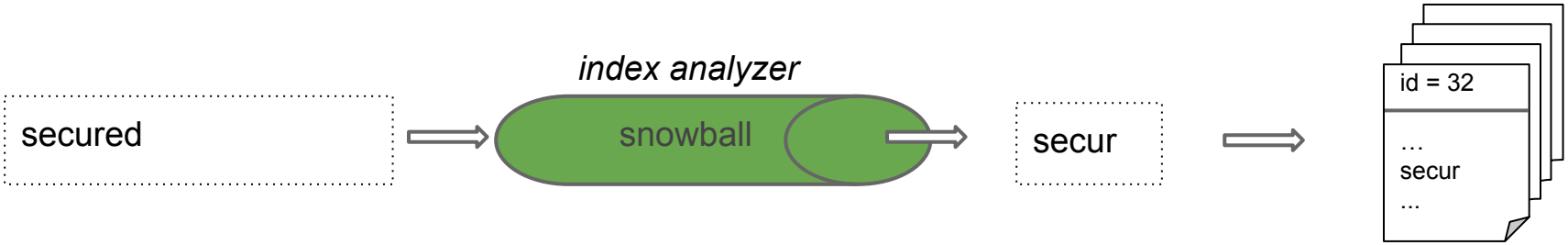
If nothing is specified the search body, the field’s default analyzer is used to convert search terms into tokens. Field’s default analyzer is the one that was used during indexing.



Snowball analyzer during indexing

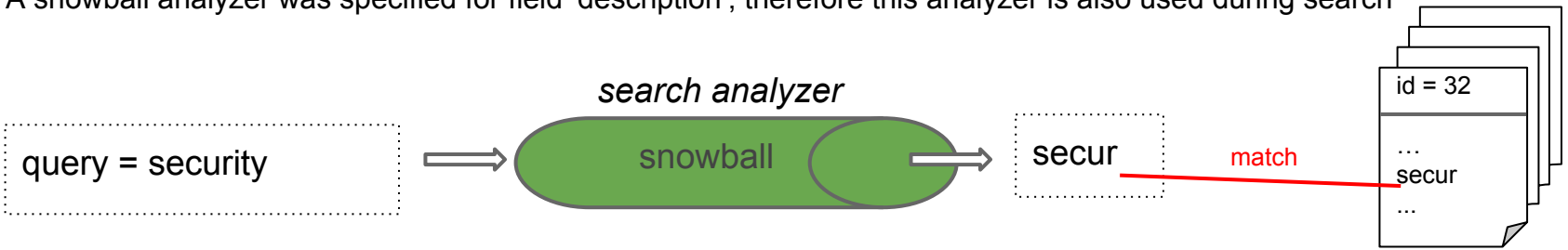
During mapping we assigned sb_analyzer to field ‘description’:

```
"description":{"type":"string", "analyzer":"sb_analyzer"}
```

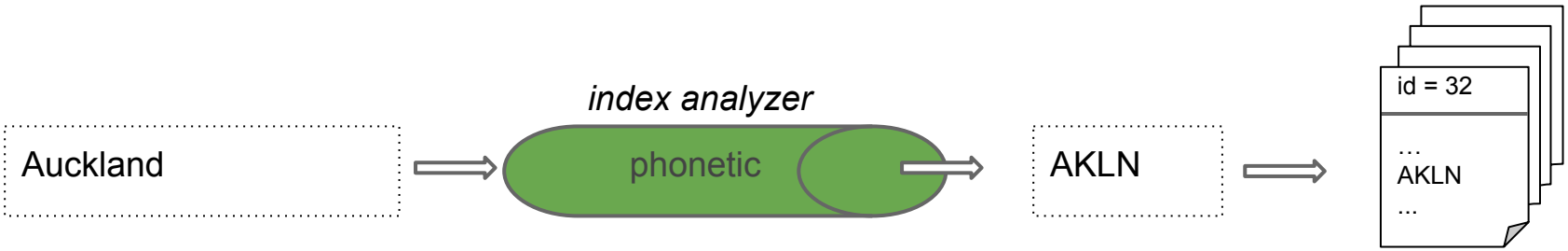


Snowball analyzer during search

A snowball analyzer was specified for field ‘description’, therefore this analyzer is also used during search

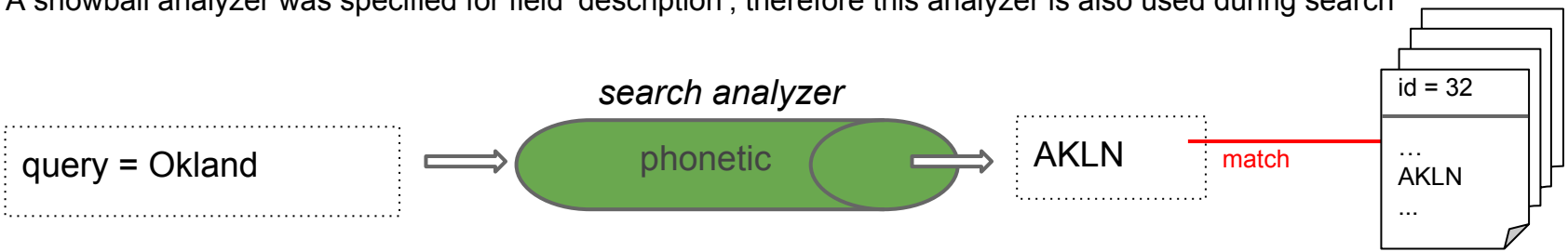


Phonetic analyzer during indexing



Phonetic analyzer during search

A snowball analyzer was specified for field ‘description’, therefore this analyzer is also used during search



specifying analyzer

http://localhost:9200/cat/person/_search?q=description:secured&**analyzer**=default

Nothing found. Because index contains term ‘secur’ (after passing word ‘secured’ through snowball analyzer during save). Now that we are searching with ‘default’ analyzer, our search term stays the same - ‘secured’, which doesn’t match what’s in the index.

http://localhost:9200/cat/person/_search?q=interests:Music

Nothing found. ‘interests’ field has no analyzer attached to it, therefore nothing is used during search. Music does not match what is in the index (music)

http://localhost:9200/cat/person/_search?q=interests:Music&**analyzer**=default

Have a result. Search analyzer converted Music into music and matched what was stored in the ‘interests’ field

_all field

By default, everything you save is also stored into _all field and analyzed with default analyzer (unless specified otherwise), or you can turn off _all at all on per field basis.

http://localhost:9200/cat/person/_search?q=Anna → 2 results.

This is equivalent to searching in the _all field.

http://localhost:9200/cat/person/_search?q=function → No results.

http://localhost:9200/cat/person/_search?q=description:function → 1 result.


```
  _id: "4",
  _score: 0.28159538,
- fields: {
    - description: [
          "Duck-wrapping (verb): If it doesn't quack like a duck, wrap it in a duck.
          code-smell: functions that wrap some values (based on their type) in other
          at which you can reason about the code in question, and it can complicate
      ]
  }
```


Exercise: explain why no results.

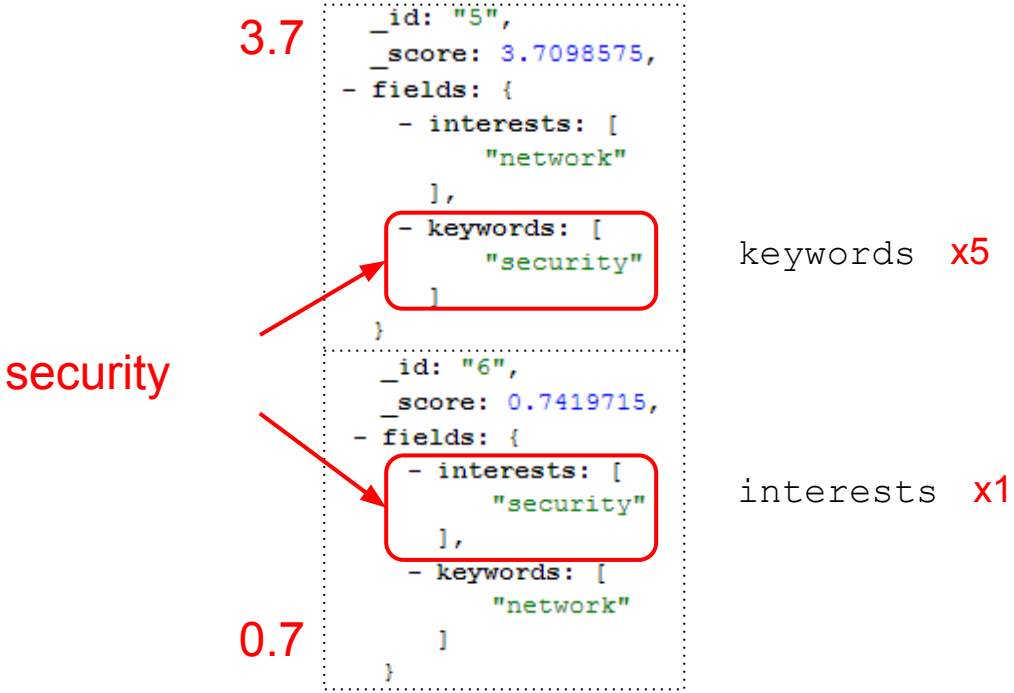
scoring

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t \text{ in } q} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d) \right)$$

- boost
- term freq (number of matches)
- field length
- number of terms matched
- idf

scoring - boost

_search?q=keywords:security OR interests:security



boost ^

During mapping

```
"names":{"type":"string", "store":"yes", "index":"analyzed", "boost":"5.0"}
```

During search (GET)

http://localhost:9200/cat/person/_search?q=description:duck^6

During search (POST)

```
"query_string" : {
    fields : ['title^5', 'descr^4', 'keywords^10', 'paragraphs'],
    query : 'Bachelor arts',
 }
```

scoring - term frequency

_search?q=description:some

0.24


```
  _id: "4",
  _score: 0.24845348,
- fields: {
    - description: [
        "Duck-wrapping (verb): If it doesn't quack like a duck, wrap
        functions that wrap some values (based on their type) in oth
        the code in question, and it can complicate some logic with
    ]
  }
```


'some' x2

0.17


```
  _id: "1",
  _score: 0.17568314,
- fields: {
    - description: [
        "While this did allow potential cool ideas like having library
        that shaders were compiled twice. Once in the compilation stag
        the compile twice. It didn't generate some kind of object code
        time."
    ]
```


'some' x1

scoring - field length

_search?q=description:setup

0.46

```
  _id: "7",
  _score: 0.46182448,
- fields: {
    - description: [
          "Here's our example network setup. The laptop is
          connected to the Internet through an untrusted wireless
          access point."
      ]
  }
```

0.28

```
  _id: "6",
  _score: 0.2886403,
- fields: {
    - description: [
          "Here's our example network setup. The laptop is
          connected to the Internet through an untrusted wireless
          access point. If we don't own it, we don't trust it.
          More importantly, we shouldn't trust any of the other
          wireless network users."
      ]
  }
```

scoring - terms matched

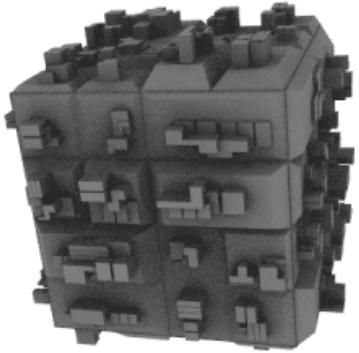
_search?q=description:some OR description:recently

0.34


```
_id: "5",
_score: 0.34249413,
- fields: {
  - description: [
      "We'll use SSH to create a pair of secured network tunnels that we can
      use to send and receive our unencrypted email. You might be able to
      create just one tunnel for retrieving email, but some providers won't
      let you send via SMTP unless you recently checked your email from the
      same machine. To keep them from thinking we're spammers, we'll create
      tunnels for both connections."
    ]
  }
```


0.07


```
_id: "4",
_score: 0.078467004,
- fields: {
  - description: [
      "Duck-wrapping (verb): If it doesn't quack like a duck, wrap it in a
      duck. JavaScript has an problematic idiom that I have come to consider
      a code-smell: functions that wrap some values (based on their type) in
      other types. I consider this to be an issue because it impairs the
      ease at which you can reason about the code in question, and it can
      complicate some logic with extra boilerplate."
    ]
  }
```



idf - inverted document frequency

‘plugin’ = 500 documents
‘extension’ = 100 documents
‘module’ = 50 documents
‘discombobulator’ = 1 document


```
query: ‘discombobulator plugin module extension’
```

...
Discombobulator plugin for Blender 3D Tutorial.

First, I will define the Protrusion settings. Any references in this section to the buttons shown in the above image will refer to those in the "Protrusion" box ...

...
There are five types of extensions for Joomla!: Components, Modules, Plugins, Templates, and Languages. Each of these extensions handle specific functionality
...

_search?q=description:type OR description:recently

0.108


```
  _id: "5",
  _score: 0.10887355,
- fields: {
   - description: [
        "We'll use SSH to create a pair of secured network tunnels that we
        can use to send and receive our unencrypted email. You might be able
        to create just one tunnel for retrieving email, but some providers
        won't let you send via SMTP unless you recently checked your email
        from the same machine. To keep them from thinking we're spammers,
        we'll create tunnels for both connections."
     ]
  }
```


0.103


```
  _id: "4",
  _score: 0.10353335,
- fields: {
   - description: [
        "Duck-wrapping (verb): If it doesn't quack like a duck, wrap it in a
        duck. JavaScript has an problematic idiom that I have come to
        consider a code-smell: functions that wrap some values (based on
        their type) in other types. I consider this to be an issue because it
        impairs the ease at which you can reason about the code in question,
        and it can complicate some logic with extra boilerplate."
     ]
  }
```

turning off idf

elasticsearch. ver. < 0.20

```
{"person":{
   "properties":{
            ...
             "description":{
                        "type":"string", "store":"yes",
                        "index":"analyzed", "analyzer":"sb_analyzer",
                        "omit_term_freq_and_positions" : "true"
            },
             "keywords":{
                        "type":"string", "store":"yes",
                        "index":"analyzed", "analyzer":"sb_analyzer",
                        boost: "5.0"
            },
            ...
   }}
}
```

making idf global

elasticsearch. ver. >= 0.20

```
../_search?search_type=dfs_query_then_fetch


{
  "query":  {
     "query_string" : { "query": "duck"  }
   },

  fields: ["dob", "name"],
  highlight: {
        fields: {names:{}, interests:{}, description:{},}
   },
 from: 0,
 size: 100
}
```

score

query = x1 x2 …

*score = weight(**x1**) * coord + weight(**x2**) * coord + ...*

*weight = (queryWeight) * (fieldWeight)*
*= (idf * queryNorm) * (tf * idf * fieldNorm)*

tf - how often term appears in a field

idf - how rare term is

fieldNorm - score, field length

next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

Query types

```
{
"query":  {
    "query_string" : {
       "query": "duck",
       "fields": ["name", "interests", "description", "keywords", "documents.*" ]
    }
 },

 fields: [
     "dob", "name", "interests", "favorites", "description", "keywords"
     ],

 highlight: {
    fields: {names:{}, aoes:{}, description:{}, keywords:{}, "documents.body":{}, "documents.title":{}}
 },

 from: 0,
 size: 100
}
```

Query types

http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/query-dsl-queries.html

...

- match all query
- more like this query
- more like this field query
- nested query
- prefix query
- query string query
- simple query string query
- range query
- regexp query
- span first query
- span multi term query
- span near query
- span not query

...

query string query

A query that uses a query parser in order to parse its content.

```
{
    "query_string" : {
        "default_field" : "content",
        "query" : "this AND that OR thus"
    }
}
```

query_string

Query_string is very powerful. The “query” part of it supports boolean operators, field names, wildcards, boosting, fuzzines, regular expressions, etc...

```
"query":  {
        "query_string" : {
                "query" : "name:(anna albert) AND _exists_:dob AND articles.body:each",
                "fields": ["name", "description", "keywords", "articles.body" ]
        }
  }
```

```
./cat/person/_search?q=name:(anna albert) AND _exists_:dob AND articles.body:each
```

http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#query-string-syntax

Term

```
"query":  {
        "term": {"articles.topic" : "economy"}
   }
```

Note: term is not-analyzed.

Combining

```
"query":  {
  "bool" : {

    "must": {
          "query_string" : {
                "query" : "anna",
                "fields": ["name", "description", "keywords", "documents.body" ]
          }
    },

    "must_not" :{
          "term": {"dob" : "1980-01-01"}
    }
  }
}
```

Typo

```
"query":  {
     "fuzzy_like_this" : {
          "like_text": "eleswhere",
          "fields": ["names", "description", "keywords", "documents.body" ]
   }
},
```

```
"highlight" : {
     "articles.body" : [
          "-regulatory modules are nodes in a gene regulatory network; they receive input
          and create output <em>elsewhere</em> in the network."
      ]
}
```

Ranges

```
"query":  {
        "range" : {
                "group" :{
                        "gte": 30,
                        "lte": "38"
                }
        }
 }
```

where group is a number in range [30, 38]

next...

- General description
- Index/Document organisation
  cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  count, paging, highlights, explain
- Searching
  in fields, using wildcards, using analyzer
- Understanding index
  index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  term, boolean, etc...
- Other types of queries
  filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

Proximity (exact) search

```
"query":  {
  "span_near" : {
        "clauses" : [
                { "span_term" : { "_all": "wrap" } },
                { "span_term" : { "_all": "duck" } }
        ],
        "slop" :0,
        "in_order" : true,
        "collect_payloads" : false
  }
}}
```

Slop : 0 - means words should be next to each other.
in_order - in combination with slop:0 will result in “exact search”

span_term: is a term query, its not analyzed. Need to run over not-analyzed field (ex. _all)

Filters

Filters are great for caching. They don’t produce score and run very fast.

```
"query":  {
        "constant_score" : {
                "filter": {"term": {"articles.topic":"economy"}}
        }
 }
```

http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/query-dsl-filters.html

Search + filter

```
"query":  {
        "filtered" : {

                "query" : {
                        "query_string" : {
                                “query" : "each",
                                "fields": ["description", "articles.body" ]
                        }
                },

                "filter" : {
                        "range" : {
                                "group" :{ "gte": 2, "lte": "4" }
                        }
                }
        }
}
```

Facets

When you need aggregated data

```
"query":  {"match_all" : {}  },
"facets" : {
        "range1" : {
                "range" : { "field" : "dob", "ranges" : [

                        { "to" : "1474-01-01" },

                        { "from" : "1474-01-01",
                                "to" : "1900-01-01" },

                        {"from" :"1900-01-01" }
                ]   }
        }
 },
```

http://www.elasticsearch.org/guide/reference/api/search/facets/

```
"facets" : {
   "range1" : {
     "  type" : "range",
     "ranges" : [ {
       "to" : -1.5652224E13,
       "to  str" : "1474-01-01",
       "count" : 1,
       "min" : -1.568376E13,
       "max" : -1.568376E13,
       "total  count" : 1,
       "total" : -1.568376E13,
       "mean" : -1.568376E13
        {
       "from  str" : "1474-01-01",
       "to  str" : "1900-01-01",
       "count" : 1,
       …
        {
       "from  str" : "1900-01-01",
       "count" : 2,

       ….
```

STEP 7

next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

javascript

```javascript
// elasticsearch.js adds the elasticsearch namespace to the window
var client = elasticsearch.Client({ ... });

// elasticsearch.jquery.js adds the es namespace to the jQuery object
var client = jQuery.es.Client({ ... });

// elasticsearch.angular.js creates an elasticsearch
// module, which provides an esFactory
var app = angular.module('app', ['elasticsearch']);
app.service('es', function (esFactory) {
  return esFactory({ ... });
});
```

```javascript
// search for documents (and also promises!!)
client.search({
  index: 'users',
  size: 50,
  body: {
    query: {
      match: {
        profile: 'elasticsearch'
      }
    }
  }
}).then(function (resp) {
  var hits = resp.body.hits;
});
```

Java

```java
Node node = nodeBuilder().client(true).node();
Client client = node.client();

// on shutdown

node.close();
```

OR

```java
// on startup

Client client = new TransportClient()
        .addTransportAddress(new InetSocketTransportAddress("host1", 9300))
        .addTransportAddress(new InetSocketTransportAddress("host2", 9300));

// on shutdown

client.close();
```

```java
SearchResponse response = client.prepareSearch("index1", "index2")
        .setTypes("type1", "type2")
        .setSearchType(SearchType.DFS_QUERY_THEN_FETCH)
        .setQuery(QueryBuilders.termQuery("multi", "test"))             // Query
        .setPostFilter(FilterBuilders.rangeFilter("age").from(12).to(18))   //
Filter
        .setFrom(0).setSize(60).setExplain(true)
        .execute()
        .actionGet();
```

next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

Plan

- Have “domain”/dto objects that can be used for serialization, deserialization.
- On start: create index
- On start: put mappings (may want to derive index structure/mapping from domain/dto object)
- Encapsulate indexing/marshalling logic into your Search Service - it will keep changing.
- Hook to database domain object’s change event to reindex entries
- Search Service should account for different search types/strategies (all words, any word, all words in given order, exact search, phonetic search) in combination with different boosting (ex. boost exact match higher than phonetic match)
- Use highlights for better user experience

Creating index - quick way

```
// number of shard 1 is recommended for our 'small' data
public static def indexSettings = [settings: [index: [
    analysis:
        [analyzer: [fat_analyzer: [
            type: 'snowball', language: 'English'
        ]]
        ],
    number_of_shards: '1'  // default value, overridden by system property elastic.shardsnumber.
]]]

protected static def documentMapping = [researchEntry: [properties: [
    title: [type: 'string', store: 'no', index: 'analyzed', analyzer: 'fat_analyzer', boost: '2.0'],
    organizationIds: [type: 'string', store: 'no', index: 'not_analyzed'],
    ourOrganization: [type: 'string', store: 'yes', index: 'no'],
    researchType: [type: 'string', store: 'yes', index: 'not_analyzed'],
    supervisorsNames: [type: 'string', store: 'no', index: 'analyzed', boost: '8.0', analyzer: 'fat_analyzer'],
    availableForMasters: [type: 'boolean', store: 'yes', index: 'not_analyzed'],
    availableForDoctorate: [type: 'boolean', store: 'yes', index: 'not_analyzed'],
    profileUrls: [type: 'string', store: 'no', index: 'analyzed', analyzer: 'fat_analyzer'],
    supervisors: [dynamic:false, type: 'object', store: 'yes', index: 'no'],
    keywords: [type: 'string', store: 'no', index: 'analyzed', analyzer: 'fat_analyzer', boost: '10.0'],
    contentSection1: [type: 'string', store: 'no', index: 'analyzed', analyzer: 'fat_analyzer'],
    contentSection2: [type: 'string', store: 'no', index: 'analyzed', analyzer: 'fat_analyzer'],
    contentSection3: [type: 'string', store: 'no', index: 'analyzed', analyzer: 'fat_analyzer']
]]]
```

Annotated entry

```groovy
@CompileStatic
@Entity
@Table(name = 'r_research_entry')
public class ResearchEntry extends BaseEntityBean {

    /**
     * This is the key that uniquely identifies the entry
     */
    @Column(name = 'entry_id', nullable = false)
    @IndexedField(indexFieldName = 'id')
    Long entryId

    @Column(name = 'r_title', length = 400)
    @IndexedField(indexFieldName = 'title')
    String title

    @Column(name = 'r_section1', nullable = false)
    @Lob
    @IndexedField(indexFieldName = 'contentSection1')
    String contentSection1

    @Transient
    @IndexedField(indexFieldName = 'organizationIds')
    public List<String> getOrganisationIds() {
        List<String> orgs = []
        Organization org = this.organization
        while (org != null) {
            orgs.add(org.id)
            org = org.parent
        }
        return orgs
    }
```

Saving entry

```groovy
protected void indexObject(Object object) {

    IndexedDocumentMetainfo info = new IndexedDocumentMetainfo(object.class, null)

    Map<String, Object> data = info.convertObject(object)

    http.request([body: data, method: "PUT", uri: url]) { HttpResponseDecorator resp ->
        if (resp.status in [200, 201]) {
            log.debug("Save document to elastic successful")
        } else {
            log.error("Failed : ${resp.status} - ${resp.statusLine.reasonPhrase}")
        }
    }

}
```

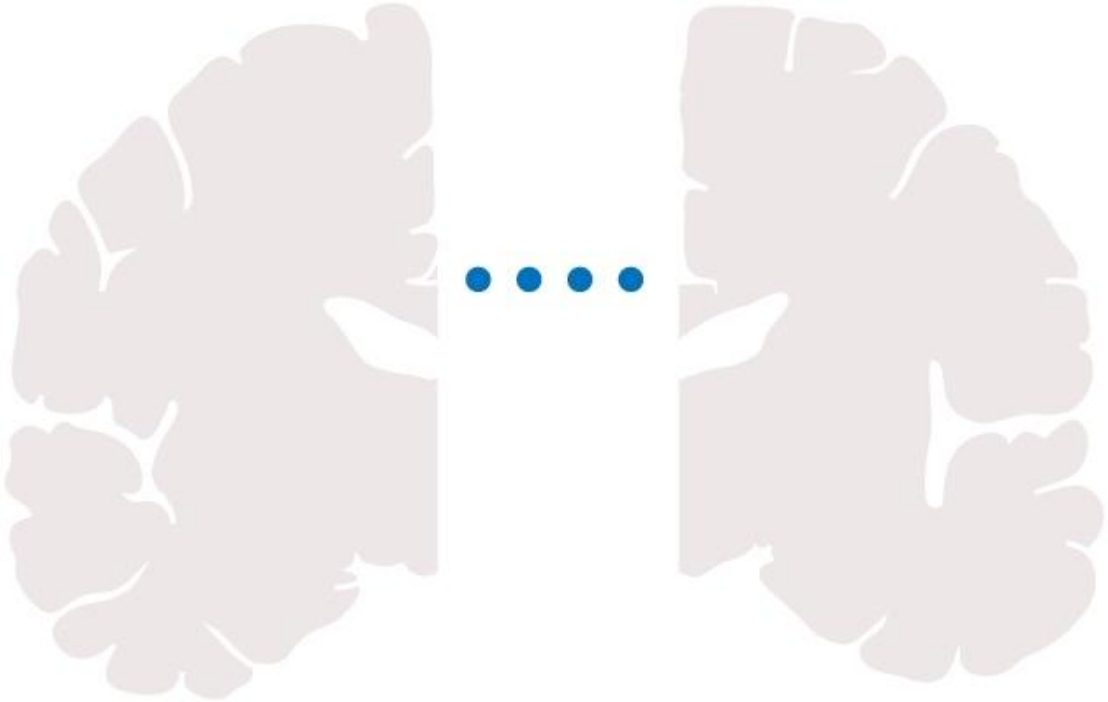
```xml
<dependency>
      <groupId>nz.ac.auckland.search</groupId>
      <artifactId>uoa-search-api</artifactId>
      <version>[1,2)</version>
</dependency>
```

github.com/deffer

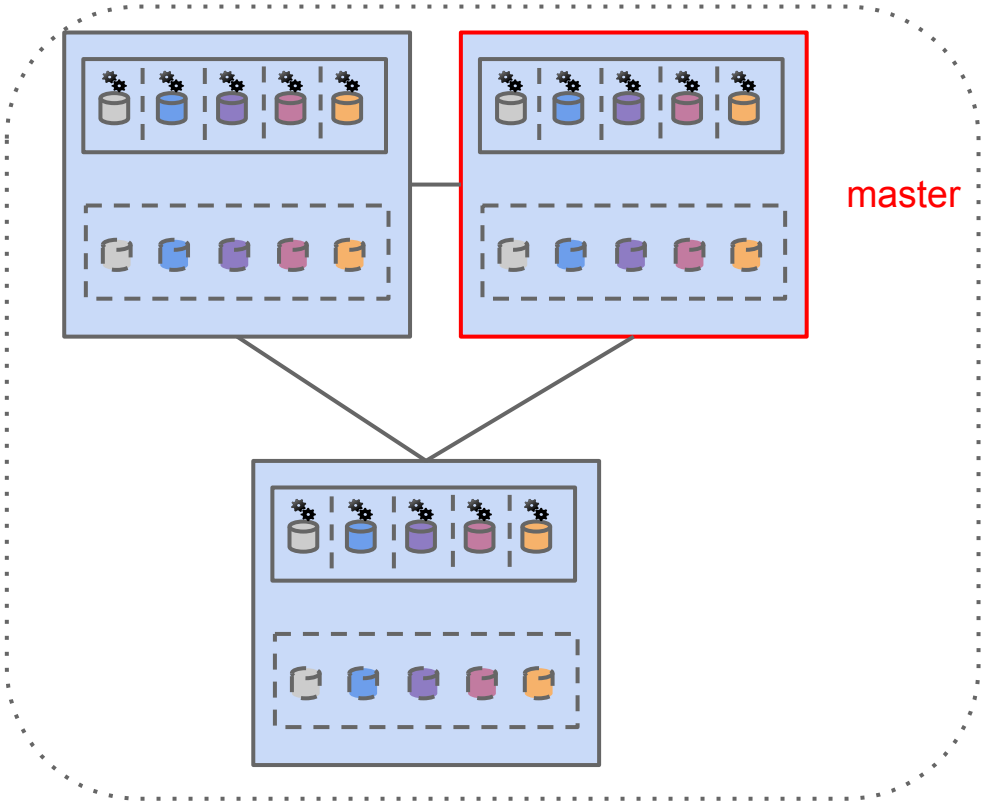
next...

- General description
- Index/Document organisation
  - cluster,replicas,shards,index per app, documents
- Put mapping, add data, execute simple search (using get)
- Search results
  - count, paging, highlights, explain
- Searching
  - in fields, using wildcards, using analyzer
- Understanding index
  - index vs. search analyzers, _all, boost, idf
- Using POST to run more complex queries
  - term, boolean, etc...
- Other types of queries
  - filters, facets, proximity
- Clients - java, javascript, ruby, python, etc...
- Using elastic in your app
- Notes on embedded elastic in distributed environment

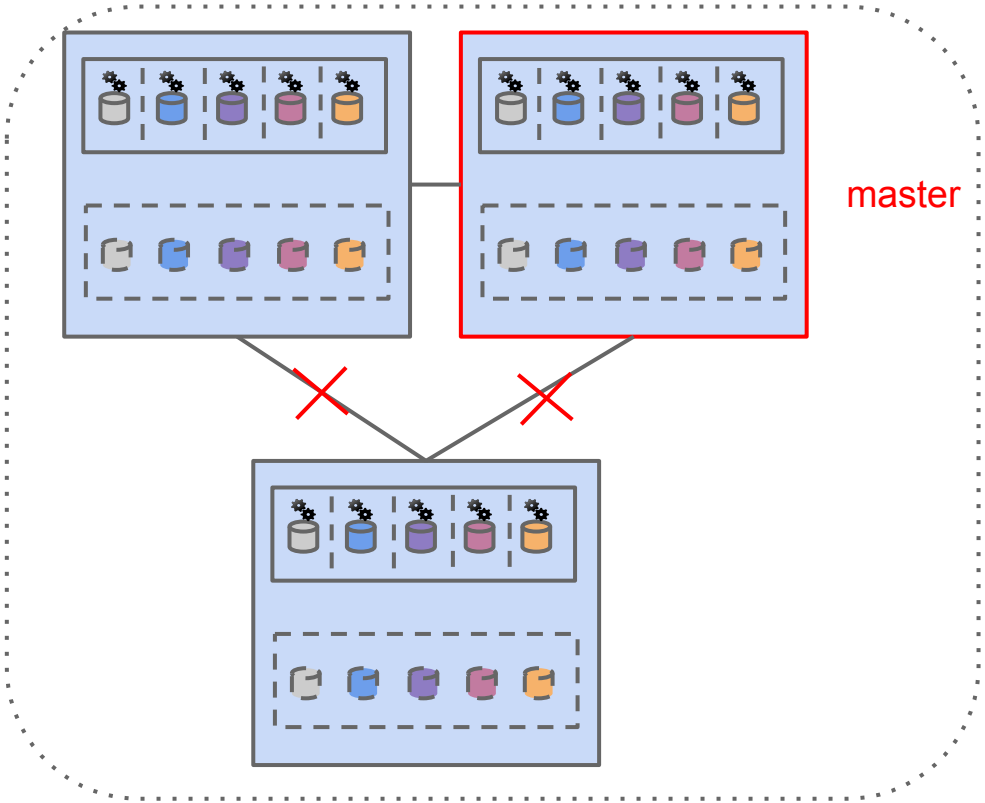
Splitbrain in cluster



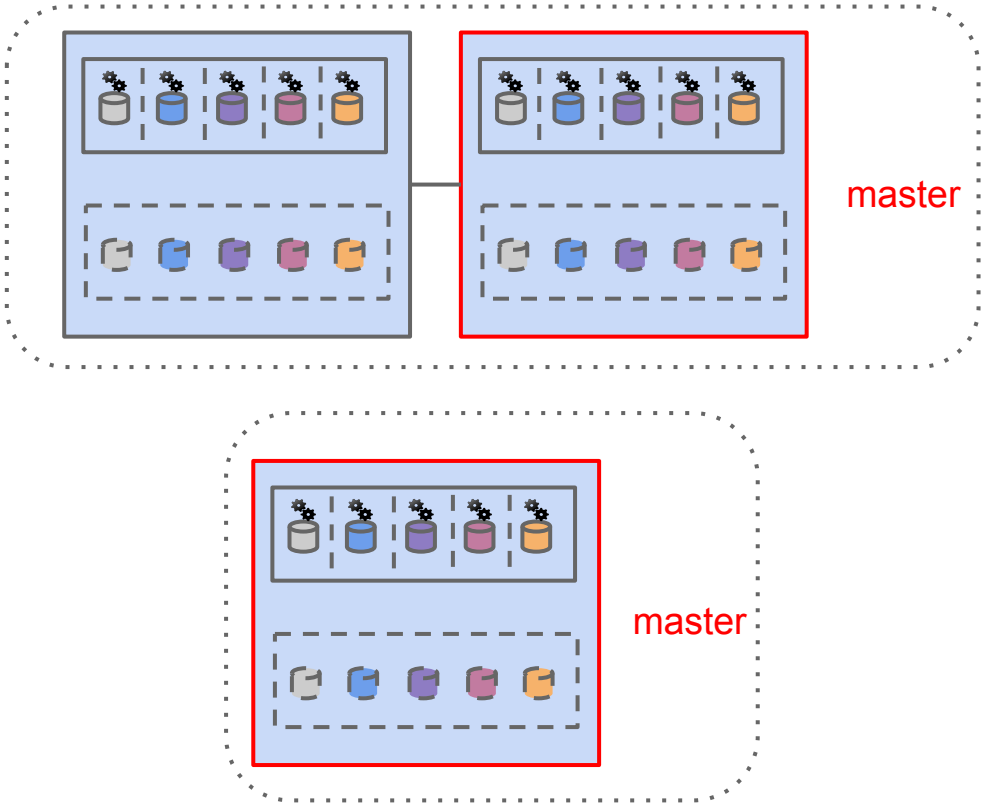
Splitbrain in cluster



Splitbrain in cluster

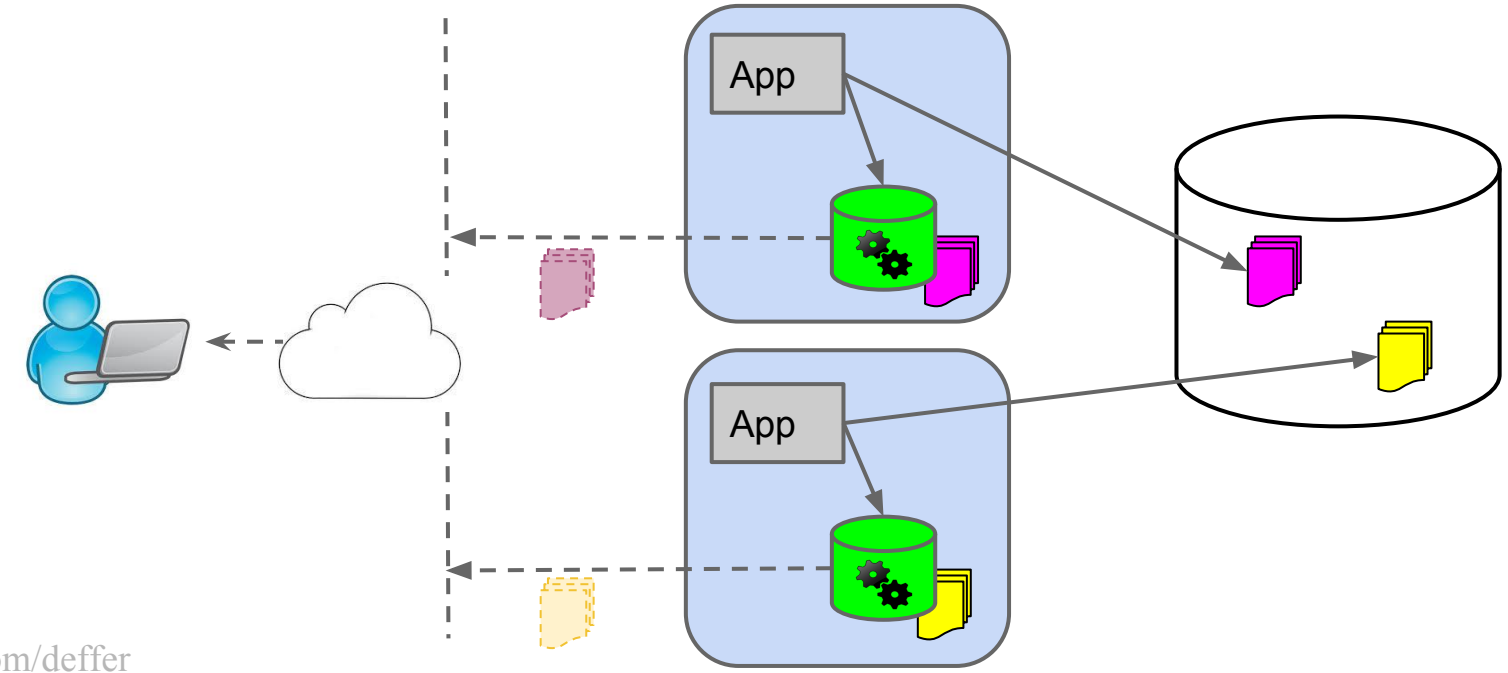


Splitbrain in cluster



Embedded

Since every application server has its own copy of index, changes done on one server do **not** become immediately available on the other servers. Reindexing job can be configured to run every 5 minutes to re-index everything that has changed since last re-indexing.



GET

uri search

A search request can be executed purely using a URI by providing request parameters. Not all search options are exposed when executing a search using this mode, but it can be handy for quick "curl tests". Here is an example:

query string query

edit

A query that uses a query parser in order to parse its content. Here is an example:

POST

request body search

The search request can be executed with a search DSL, which includes the Query DSL, within its body. Here is an example:

- match query
- multi match query
- bool query
- boosting query
- common terms query
- constant score query
- dis max query
- filtered query
- fuzzy like this query
- fuzzy like this field query
- function score query
- fuzzy query
- geoshape query
- has child query
- has parent query
- ids query
- indices query
- match all query
- more like this query
- more like this field query
- nested query
- prefix query
- query string query
- simple query string query
- range query
- regexp query
- span first query
- span multi term query

Thank you!

core types

edit

Each JSON field can be mapped to a specific core type. JSON itself already provides us with some typing, with its support for `string`, `integer`/`long`, `float`/`double`, `boolean`, and `null`.

github.com/deffer

analyze

edit

Performs the analysis process on a text and return the tokens breakdown of the text.

Can be used without specifying an index against one of the many built in analyzers:

indices apis

The indices APIs are used to manage individual indices, index settings, aliases, mappings, index templates and warmers.