

slithering into

A photograph of two hand-knotted paracord bracelets resting on a dark, weathered wooden surface. One bracelet is primarily yellow with black and orange accents, while the other is primarily blue with red and white accents. Both bracelets feature a central knot and several smaller knots along their length.

elastic search

setup:

<https://github.com/erikrose/elasticsearch-tutorial/wiki>

Erik Rose (@ErikRose)

Mozilla

slithering into

A photograph of two hand-knotted paracord bracelets resting on a dark, weathered wooden surface. One bracelet is primarily yellow with black and orange accents, while the other is primarily blue with red and white accents. Both bracelets feature a central knot and several smaller knots along their length.

elastic search

setup:

<https://github.com/erikrose/elasticsearch-tutorial/wiki>

Erik Rose (@ErikRose)

Mozilla

housekeeping

<https://github.com/erikrose/elasticsearch-tutorial/wiki>

housekeeping

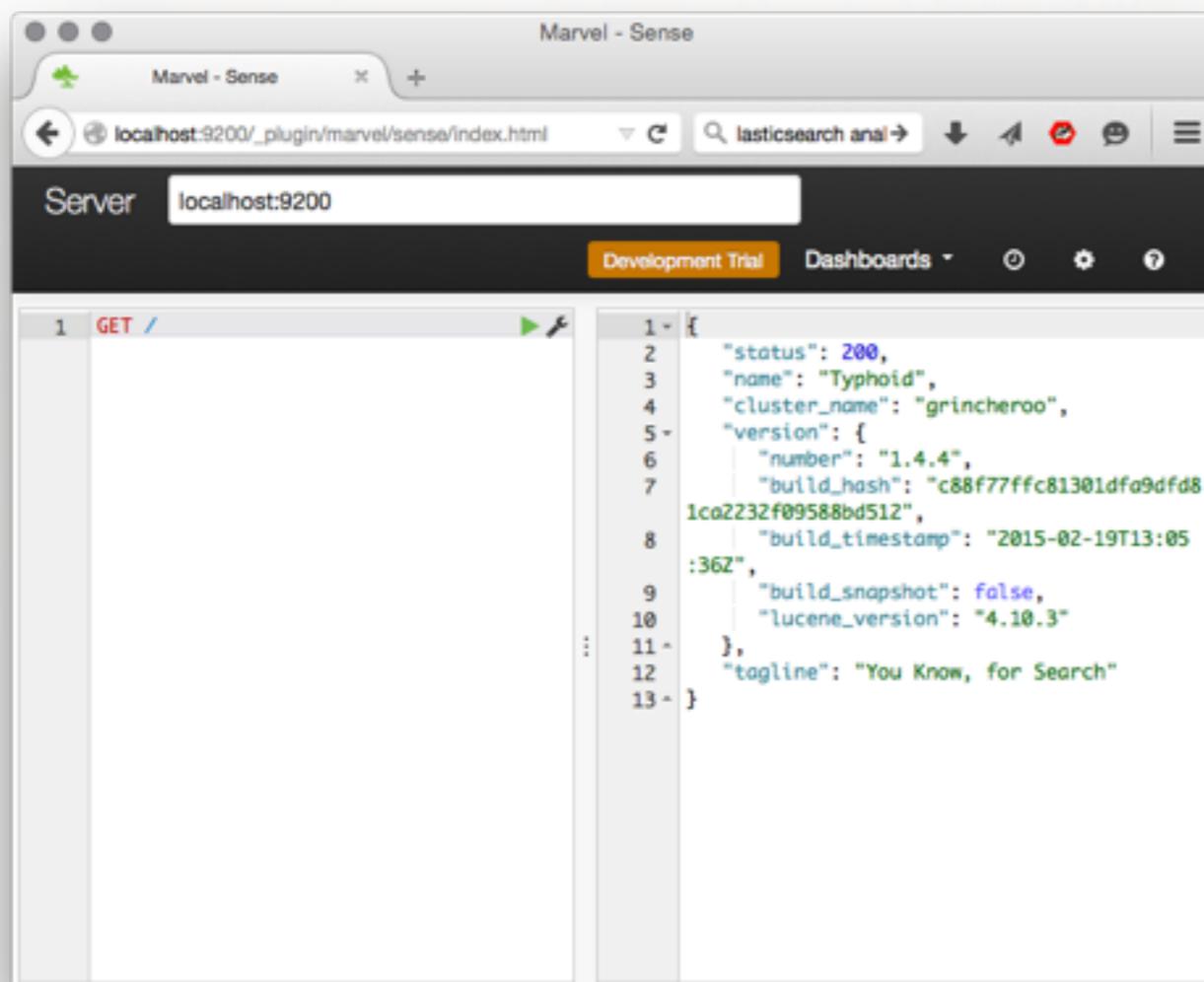
<https://github.com/erikrose/elasticsearch-tutorial/wiki>

- elasticsearch: I'm using 1.4. 1.5 is fine, too.

housekeeping

<https://github.com/erikrose/elasticsearch-tutorial/wiki>

- elasticsearch: I'm using 1.4. 1.5 is fine, too.
- Marvel



The screenshot shows a browser window titled "Marvel - Sense" with the URL "localhost:9200/_plugin/marvel/sense/index.html". The server dropdown is set to "localhost:9200". The main area displays a JSON response from a "GET /" request:

```
1 1 {  
2   "status": 200,  
3   "name": "Typhoid",  
4   "cluster_name": "grincheroo",  
5   "version": {  
6     "number": "1.4.4",  
7     "build_hash": "c88f77ffc81301dfa9df8  
1ca2232f09588bd512",  
8     "build_timestamp": "2015-02-19T13:05  
:36Z",  
9     "build_snapshot": false,  
10    "lucene_version": "4.10.3"  
11  },  
12  "tagline": "You Know, for Search"  
13}
```

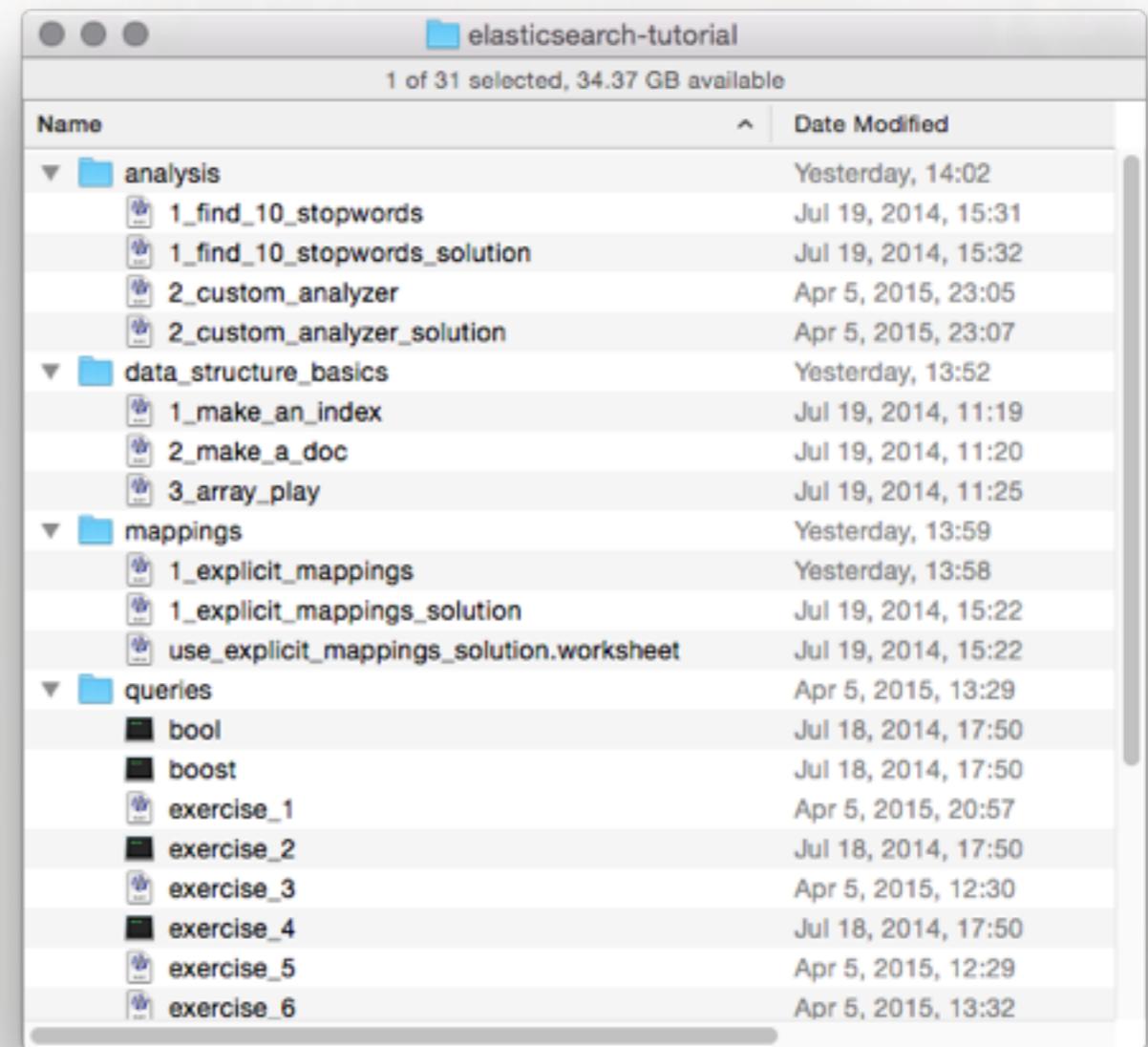
housekeeping

<https://github.com/erikrose/elasticsearch-tutorial/wiki>

- elasticsearch: I'm using 1.4. 1.5 is fine, too.
 - Marvel
 - Exercises

The screenshot shows a browser window titled "Marvel - Sense". The address bar displays "localhost:9200/_plugin/marvel/sense/index.html". The main content area shows a JSON response for a "GET /" request. The JSON object contains the following fields:

```
1  {  
2      "status": 200,  
3      "name": "Typhoid",  
4      "cluster_name": "grincheroo",  
5      "version": {  
6          "number": "1.4.4",  
7          "build_hash": "c88f77ffc81301dfa9dfd81c02232f09588bd512",  
8          "build_timestamp": "2015-02-19T13:05:36Z",  
9          "build_snapshot": false,  
10         "lucene_version": "4.10.3"  
11     },  
12     "tagline": "You Know, for Search"  
13 }
```

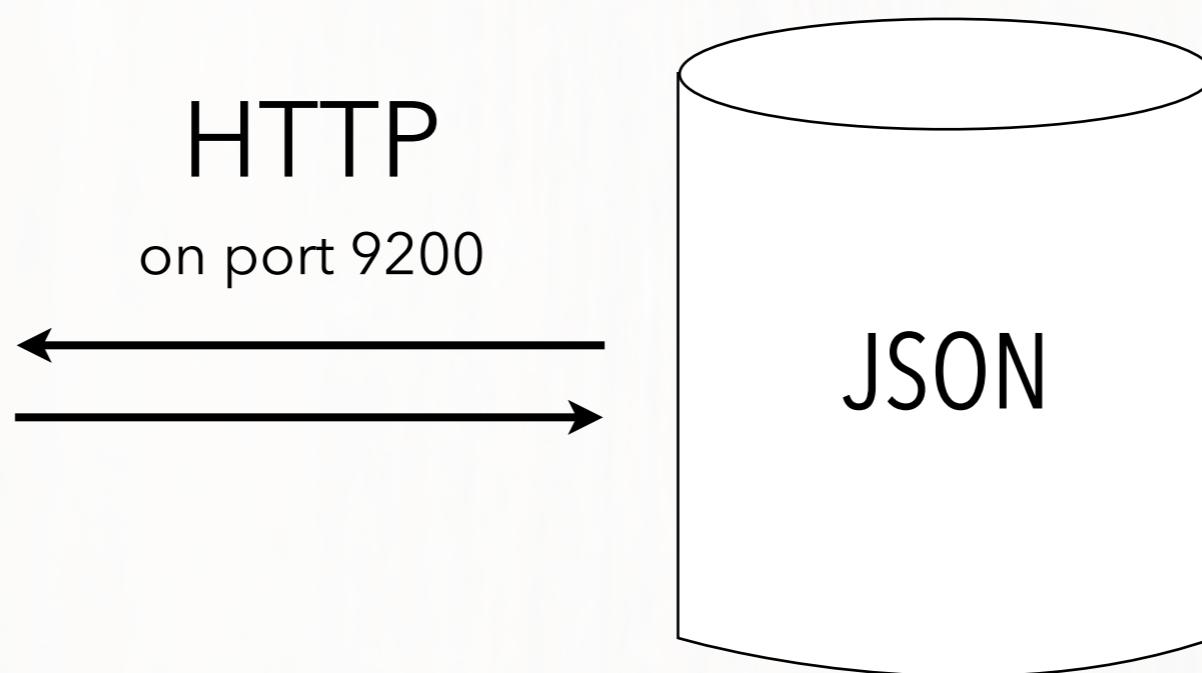




what is this thing?

Lucene and then some

- Elasticsearch wraps Lucene.
- Read/write/admin via REST
- Native format is JSON.



what it's good for

what it's good for

- Full-text search

what it's good for

- Full-text search
- Scoring

what it's good for

- Full-text search
- Scoring
- Big data

what it's good for

- Full-text search
- Scoring
- Big data
- Faceting

what it's good for

- Full-text search
- Scoring
- Big data
- Faceting
- Geographical queries

what it's bad for

what it's bad for

- Atomicity

what it's bad for

- Atomicity
- Exact answers (in some cases)

what it's bad for

- Atomicity
- Exact answers (in some cases)
- Durability

what it's bad for

- Atomicity
- Exact answers (in some cases)
- Durability
- Joins

CAP theorem

CAP theorem

Consistency

Availability

Partition-tolerance

CAP theorem

Consistency

Availability

Partition-tolerance

pick any 2.

CAP theorem

~~Consistency~~

Availability

Partition-tolerance

pick any 2.

CAP theorem

~~Consistency~~

Availability

Partition-tolerance

pick any 2.

Read <http://aphyr.com/posts/317-call-me-maybe-elasticsearch> (and despair).

why it's compelling

why it's compelling

- Simple things are simple.

why it's compelling

- Simple things are simple.
- Easy to integrate with existing apps

why it's compelling

- Simple things are simple.
- Easy to integrate with existing apps
- Easy to configure not-too-terribly

why it's compelling

- Simple things are simple.
- Easy to integrate with existing apps
- Easy to configure not-too-terribly
- Wicked-fast

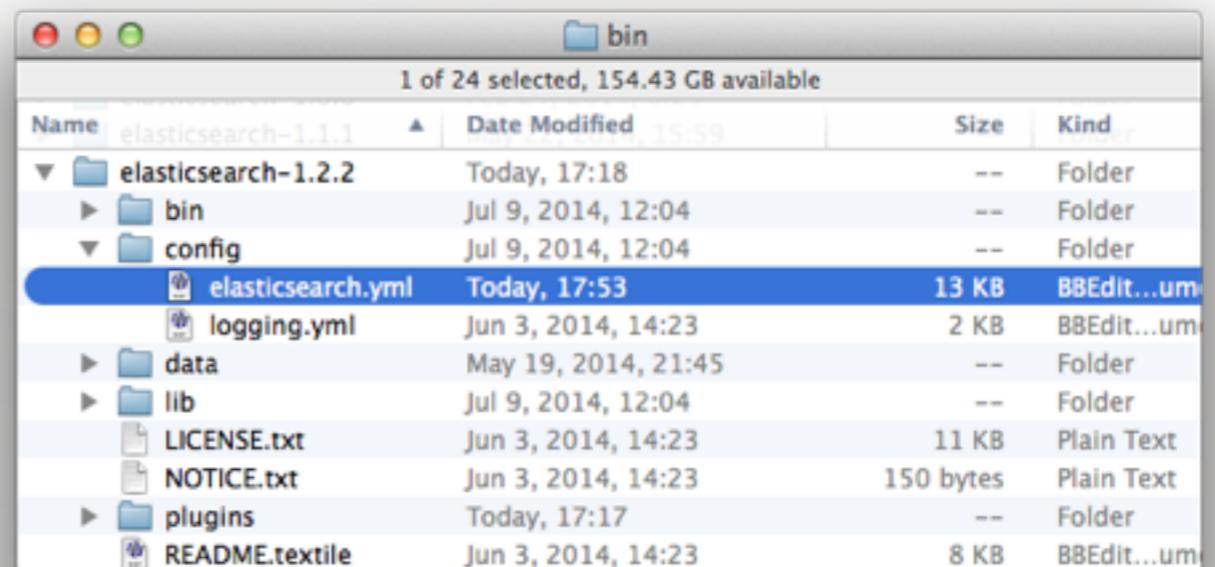
The Problem
&
The Solution



set up

exercise 1: fix networking

```
# Unicast discovery allows to explicitly control which nodes will be used  
# to discover the cluster. It can be used when multicast is not present,  
# or to restrict the cluster communication-wise.  
#  
# 1. Disable multicast discovery (enabled by default):  
# discovery.zen.ping.multicast.enabled: false  
  
# Elasticsearch, by default, binds itself to the 0.0.0.0 address, and listens  
# on port [9200–9300] for HTTP traffic and on port [9300–9400] for node-to-node  
# communication. (the range means that if the port is busy, it will automatically  
# try the next port).  
  
# Set the bind address specifically (IPv4 or IPv6):  
#  
network.host: 127.0.0.1
```



exercise 2: start up

```
% cd elasticsearch-1.4.4  
  
# On Macs and Unix-alikes:  
% bin/elasticsearch  
  
# On Windows:  
% bin\elasticsearch.bat
```

exercise 3: open Sense

http://localhost:9200/_plugin/marvel/sense/index.html

The screenshot shows the Marvel - Overview dashboard. At the top, there's a status bar with "Development Trial" and a refresh interval of "an hour ago to a few seconds ago refreshed every 10s". Below the status bar, there are two tabs: "QUERY" (selected) and "FILTERING".

CLUSTER SUMMARY

Name: grinchertoo Status: red ⓘ Nodes: 1 Indices: 10 Shards: 26 Data: 3.02 GB CPU: 3% Mem: 25 MB
Up time: 33.8 m Version: 1.1.1

DOCUMENT COUNT

20 Mil

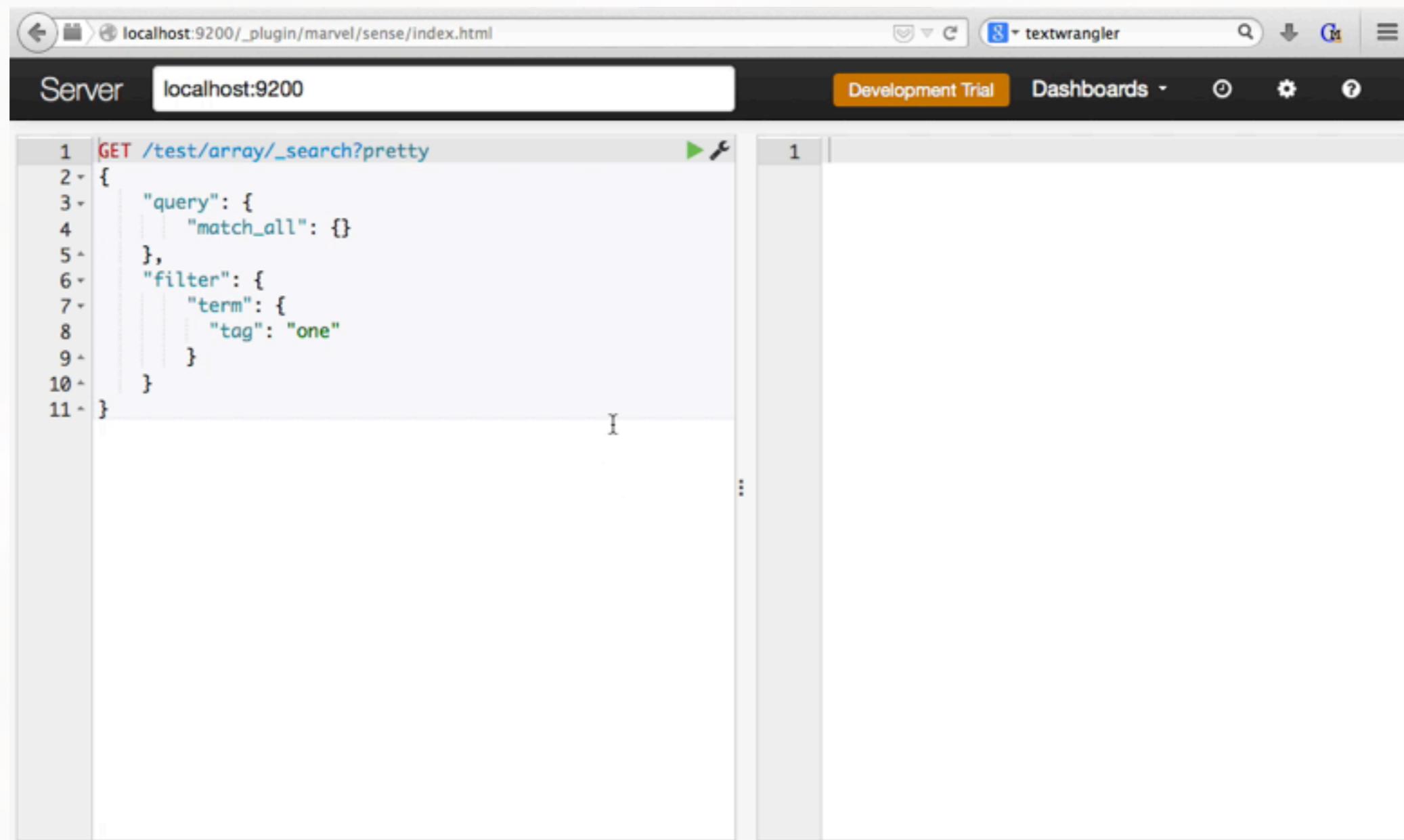
SEARCH REQUEST RATE

INDEXING REQUEST RATE

A context menu is open over the "Sense" section in the cluster summary. The menu items are: Cluster Pulse, Shard Allocation, Sense (selected), Node Statistics, and Index Statistics.

exercise 3: open Sense

http://localhost:9200/_plugin/marvel/sense/index.html



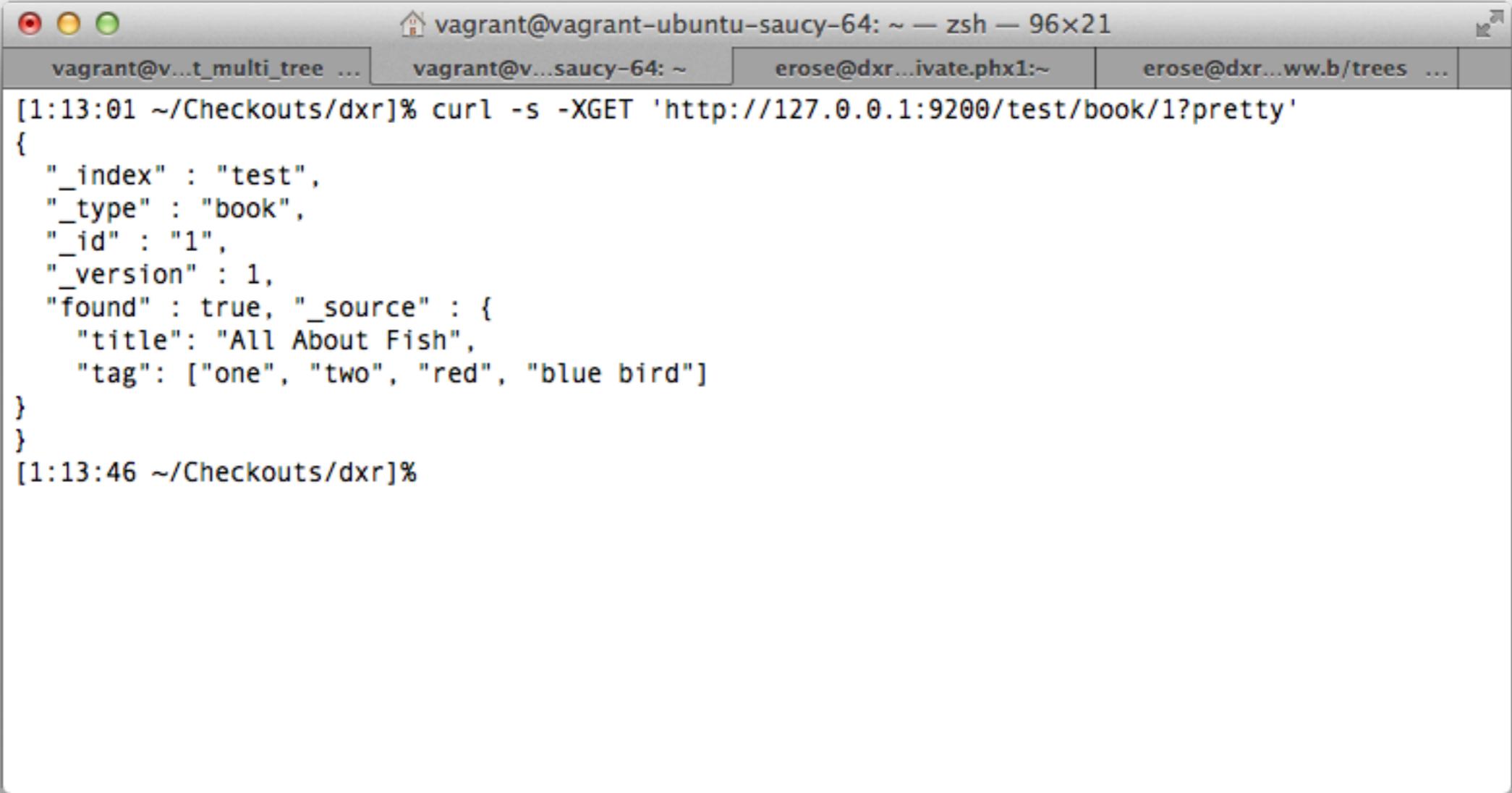
The screenshot shows the TextWrangler application window. The title bar reads "localhost:9200/_plugin/marvel/sense/index.html". The menu bar includes "File", "Edit", "Server", "localhost:9200", "Development Trial", "Dashboards", and "Help". The main area has two panels. The left panel contains a code editor with the following Elasticsearch search query:

```
1 GET /test/array/_search?pretty
2 {
3     "query": {
4         "match_all": {}
5     },
6     "filter": {
7         "term": {
8             "tag": "one"
9         }
10    }
11 }
```

The right panel displays the results of the search query, showing a single document with the ID "1".

exercise 3: open Sense

http://localhost:9200/_plugin/marvel/sense/index.html



A screenshot of a terminal window with a light gray background and a dark gray header bar. The header bar contains four tabs: "vagrant@v...t_multi_tree ...", "vagrant@v...saucy-64: ~", "erose@dxr...ivate.phx1:~", and "erose@dxr...ww.b/trees ...". The main window area shows the following command and its JSON response:

```
[1:13:01 ~/Checkouts/dxr]% curl -s -XGET 'http://127.0.0.1:9200/test/book/1?pretty'
{
  "_index" : "test",
  "_type" : "book",
  "_id" : "1",
  "_version" : 1,
  "found" : true, "_source" : {
    "title": "All About Fish",
    "tag": ["one", "two", "red", "blue bird"]
  }
}
[1:13:46 ~/Checkouts/dxr]%
```

exercise 4: check the cluster

http://localhost:9200/_plugin/marvel/sense/index.html

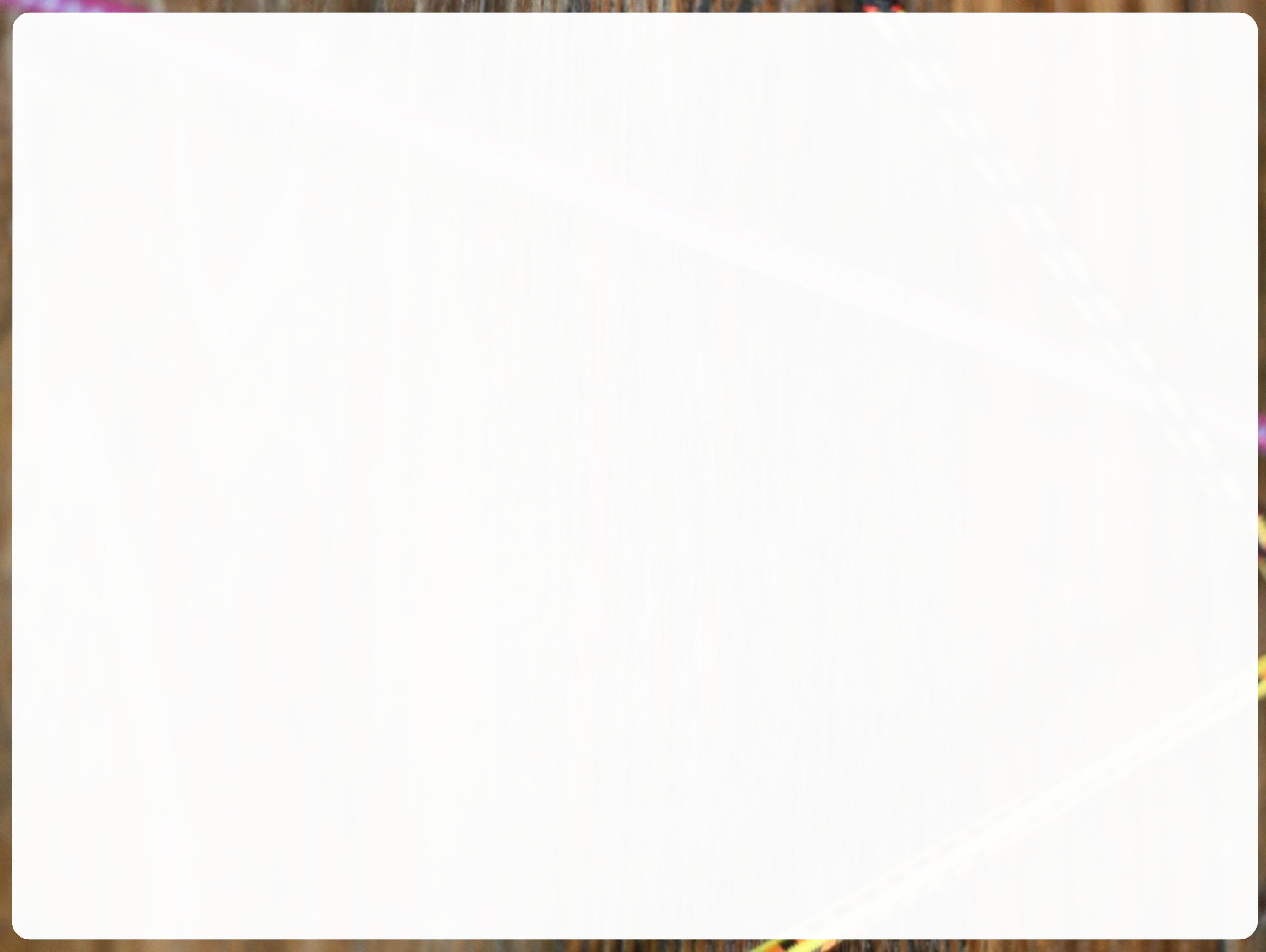
GET /_cluster/health

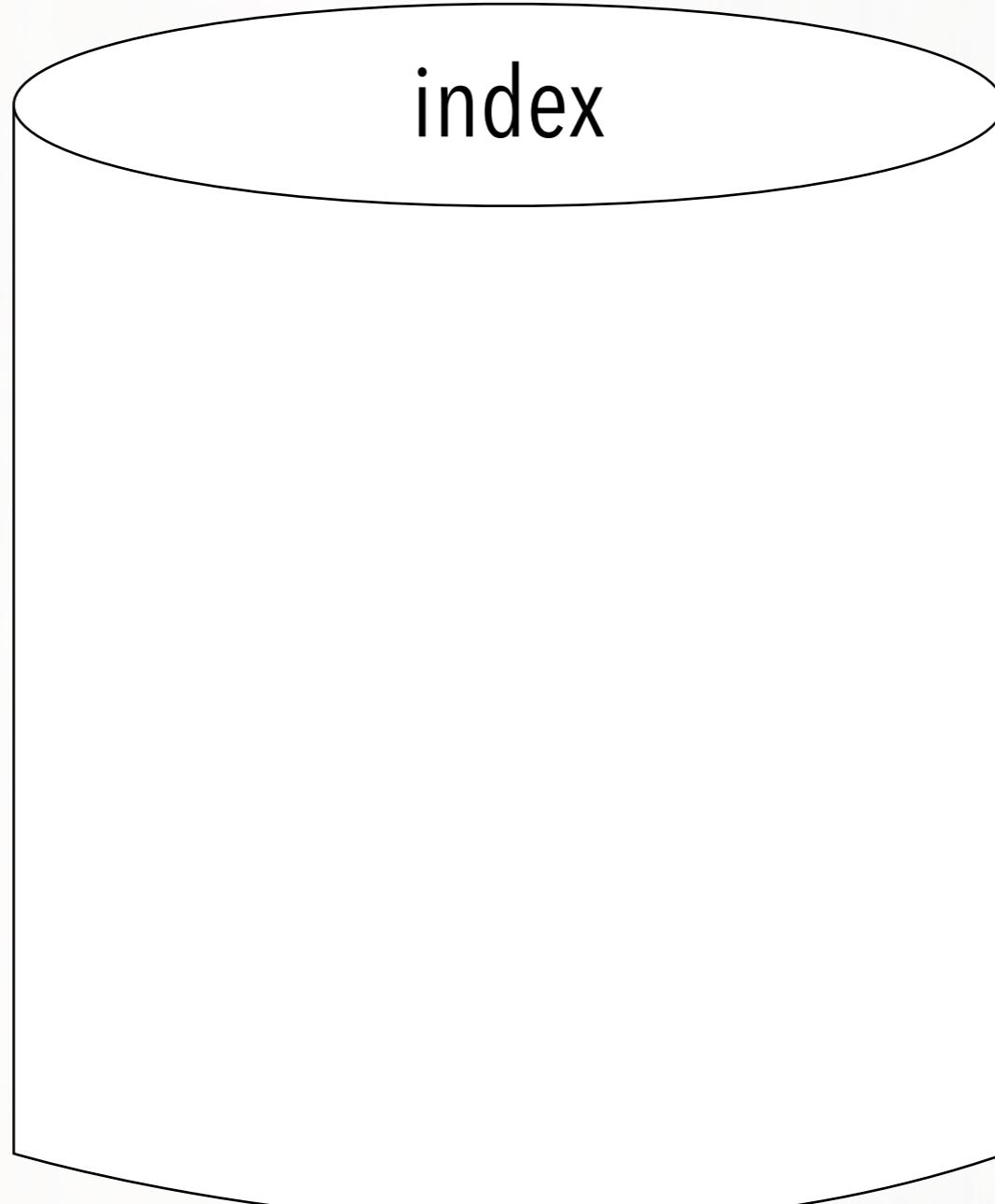


```
{  
  "cluster_name" : "grinchertoo",  
  "status" : "yellow",  
  "timed_out" : false,  
  "number_of_nodes" : 1,  
  "number_of_data_nodes" : 1,  
  "active_primary_shards" : 19,  
  "active_shards" : 19,  
  "relocating_shards" : 0,  
  "initializing_shards" : 0,  
  "unassigned_shards" : 13  
}
```

data structure basics







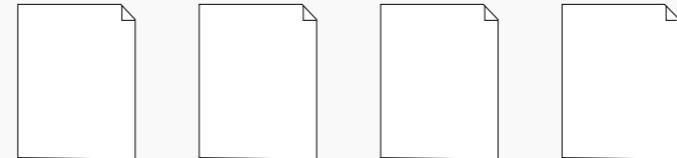
index

index

doctype

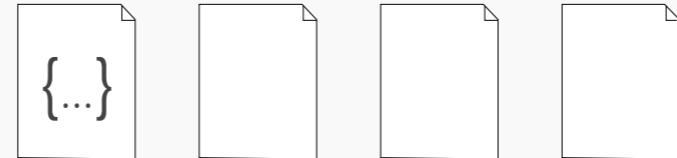
index

doctype



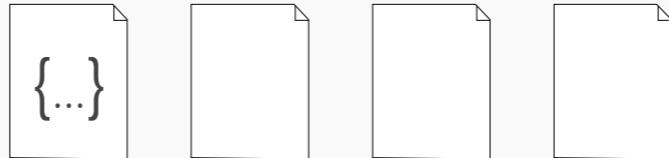
index

doctype

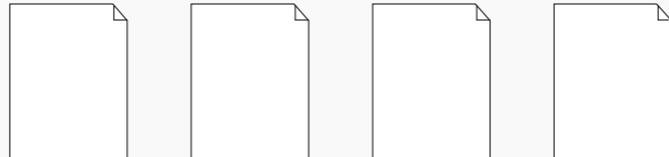


index

doctype

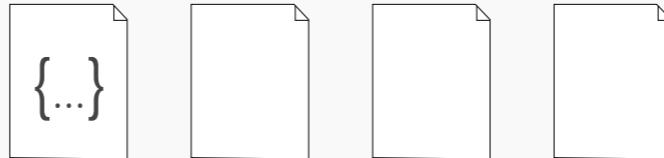


another doctype

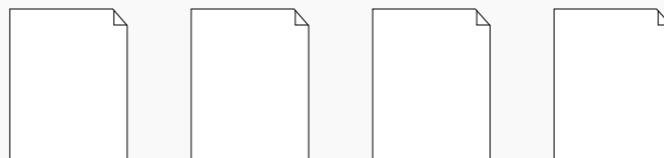


index

doctype



another doctype



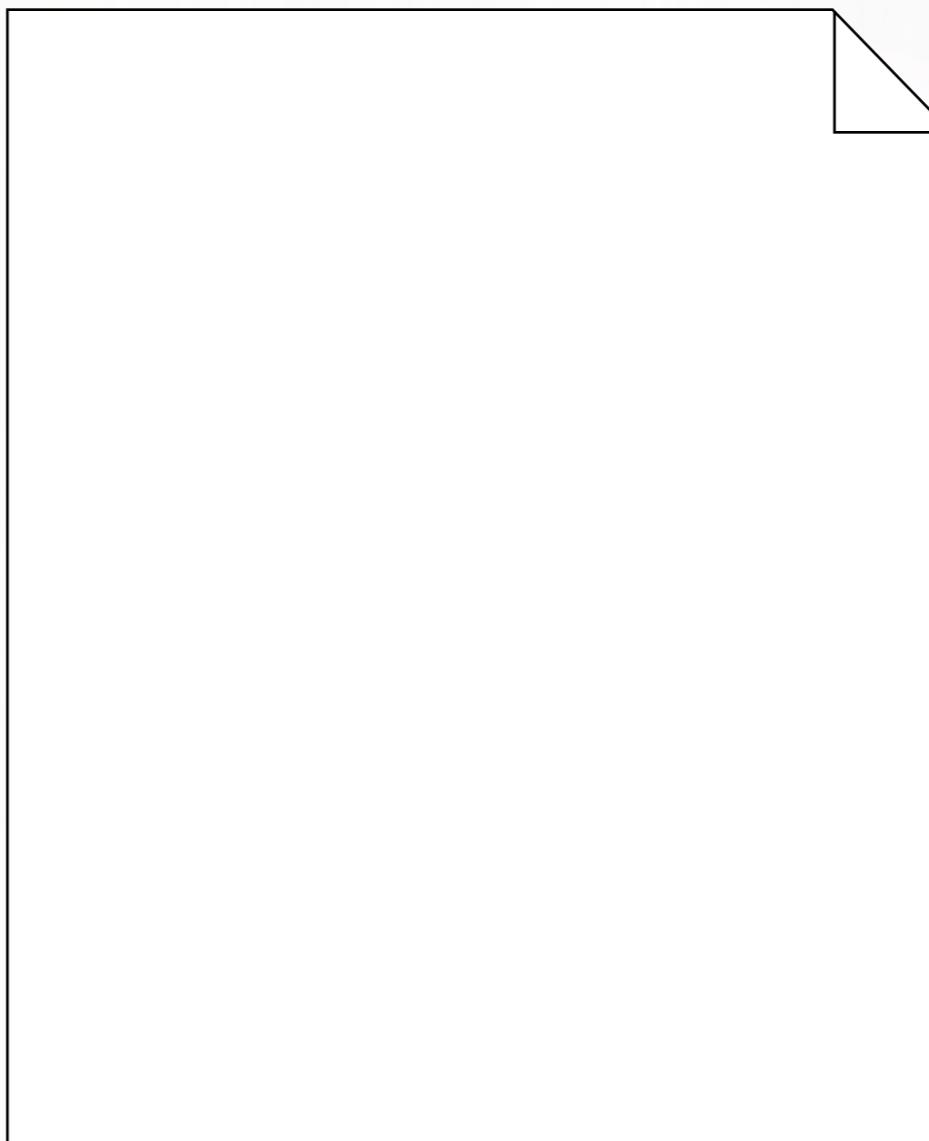
exercise 1: make an index

PUT /test

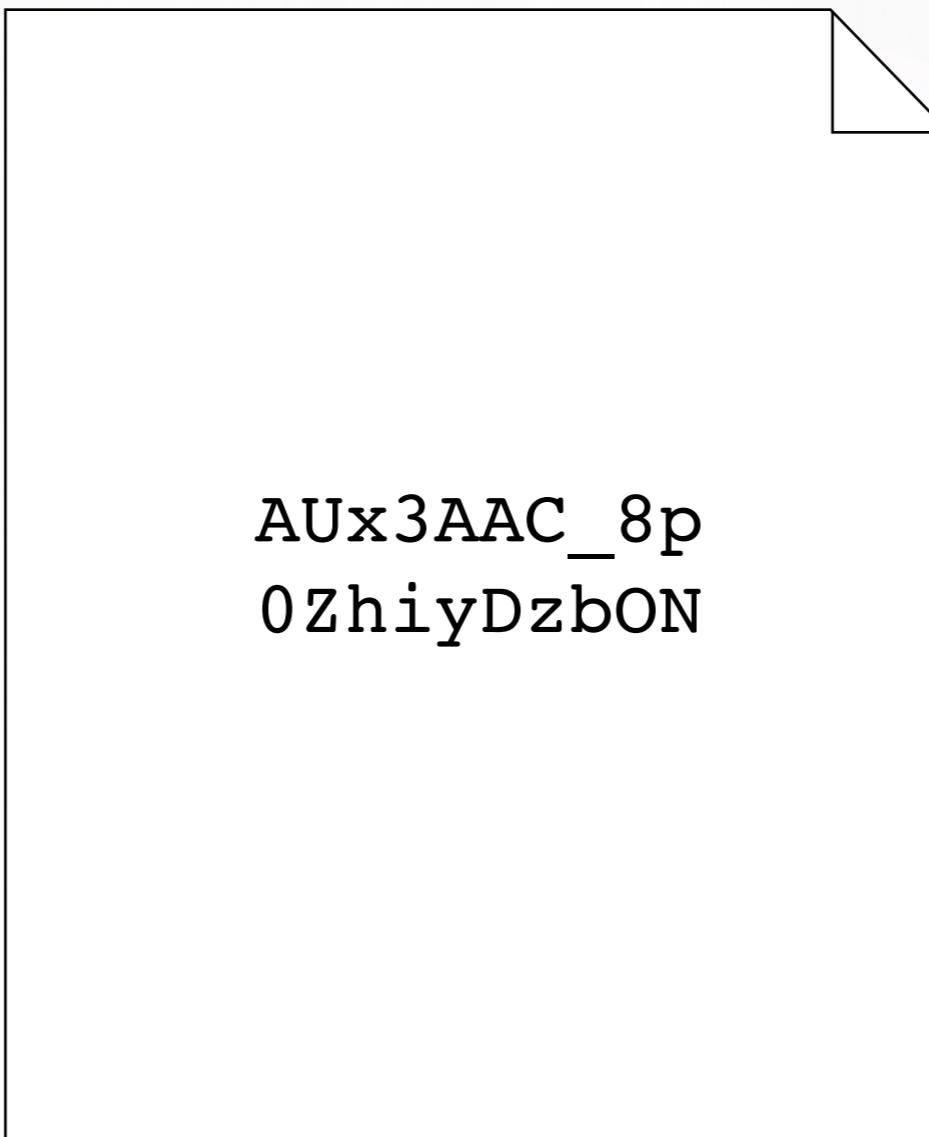


{"acknowledged":true}

IDs



IDs



AUX3AAC_8p
0ZhiyDzbON

exercise 2: make a doc

exercise 2: make a doc

```
# Make a doc:  
PUT /test/book/1  
{  
  "title": "All About Fish",  
  "author": "Fishy McFishstein",  
  "pages": 3015  
}
```

exercise 2: make a doc

```
# Make a doc:  
PUT /test/book/1  
{  
    "title": "All About Fish",  
    "author": "Fishy McFishstein",  
    "pages": 3015  
}  
  
# Make sure it's there:  
GET /test/book/1
```

exercise 2: make a doc

```
# Make a doc:  
PUT /test/book/1  
{  
    "title": "All About Fish",  
    "author": "Fishy McFishstein",  
    "pages": 3015  
}  
  
# Make sure it's there:  
GET /test/book/1  
↓  
{  
    "_index" : "test",  
    "_type" : "book",  
    "_id" : "1",  
    "_version" : 2,  
    "found" : true,  
    "_source" : {  
        "title": "All About Fish",  
        "author": "Fishy McFishstein",  
        "pages": 3015  
    }  
}
```

exercise 2: make a doc

```
# Delete the doc:  
DELETE /test/book/1
```

diplodocus	333
duodenum	201
dwaal	500, 119

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

row → 0, 1, 3

boat → 0, 1

chicken → 2

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

row

→ 0, 1, 3

boat

→ 0, 1

chicken

→ 2

row row
row your
boat

row the
row **boat**

chicken
chicken
chicken

the
front
row

0

1

2

3

row → 0, 1, 3

boat → 0, 1

chicken → 2

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

row → 0, 1, 3

boat → 0, 1

chicken → 2

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

row → 0, 1, 3

boat → 0, 1

chicken → 2

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

	→	doc	positions
row	→	0	[0, 1, 2]
		1	[0, 2]
boat	→	3	[2]
		0	[4]
chicken	→	1	[3]
		2	[0, 1, 2]

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

	→	doc	positions
row	→	0 1 3	[0, 1, 2] [0, 2] [2]
boat	→	0 1	[4] [3]
chicken	→	2	[0, 1, 2]

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

	→	doc	positions
row	→	0	[0, 1, 2]
	→	1	[0, 2]
boat	→	3	[2]
	→	0	[4]
chicken	→	1	[3]
	→	2	[0, 1, 2]

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

	→	doc	positions
row	→	0 1 3	[0, 1, 2] [0, 2] [2]
boat	→	0 1	[4] ← ? [3]
chicken	→	2	[0, 1, 2]

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

	→	doc	positions
row	→	0 1 3	[0, 1, 2] [0, 2] [2]
boat	→	0 1	[4] [3]
chicken	→	2	[0, 1, 2]

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

	→	doc	positions
row	→	0 1 3	[0, 1, 2] [0, 2] [2]
boat	→	0 1	[4] [3] ← ?
chicken	→	2	[0, 1, 2]

row row
row your
boat

row the
row boat

chicken
chicken
chicken

the
front
row

0

1

2

3

		doc	positions
row	232	→ 0	[0, 1, 2]
boat	78	→ 1	[0, 2]
chicken	91	→ 2	[2]

		doc	positions
row	232 →	0	[0, 1, 2]
		1	[0, 2]
		3	[2]
boat	78 →	0	[4]
		1	[3]
chicken	91 →	2	[0, 1, 2]

boat	78
• • •	
chicken	91
• • •	
row	232

indices on properties

```
"title": "All About Fish",  
"author": "Fishy McFishstein",  
"pages": 3015
```

```
"title": "Nothing About Pigs",  
"author": "Nopiggy Nopigman",  
"pages": 0
```

```
"title": "All About Everything",  
"author": "Everybody",  
"pages": 4294967295
```

inner objects

```
PUT /test/book/1
{
  "title": "All About Fish",
  "author": {
    "name": "Fisher McFishstein",
    "birthday": "1980-02-22",
    "favorite_color": "green"
  }
}
```

inner objects

```
PUT /test/book/1
{
  "title": "All About Fish",
  "author": {
    "name": "Fisher McFishstein",
    "birthday": "1980-02-22",
    "favorite_color": "green"
  }
}
```

title:	All About Fish
author.name:	Fisher McFishstein
author.birthday:	1980-02-22
author.favorite_color:	green

inner objects

```
PUT /test/book/1
{
    "title": "All About Fish",
    "author": {
        "name": "Fisher McFishstein",
        "birthday": "1980-02-22",
        "favorite_color": "green"
    }
}
```

```
GET/test/book/1
```

↓

```
{
    "_index": "test",
    "_type": "book",
    "_id": "1",
    "_version": 1,
    "found": true,
    "_source": {
        "title": "All About Fish",
        "author": {
            "name": "Fisher McFishstein",
            "birthday": "1980-02-22",
            "favorite_color": "green"
        }
    }
}
```

inner objects

```
PUT /test/book/1
```

```
{  
  "title": "All About Fish",  
  "author": {  
    "name": "Fisher McFishstein",  
    "birthday": "1980-02-22",  
    "favorite_color": "green"  
  }  
}
```

```
GET/test/book/1
```

```
↓  
{ "_index" : "test",  
  "_type" : "book",  
  "_id" : "1",  
  "_version" : 1,  
  "found" : true,  
  "_source" : {  
    "title": "All About Fish",  
    "author": {  
      "name": "Fisher McFishstein",  
      "birthday": "1980-02-22",  
      "favorite_color": "green"  
    }  
  }  
}
```

arrays

```
# Insert a doc containing an array:  
PUT /test/book/1  
{  
  "title": "All About Fish",  
  "tag": ["one", "two", "red", "blue"]  
}
```

arrays

```
# Insert a doc containing an array:
```

```
PUT /test/book/1
```

```
{
```

```
    "title": "All About Fish",
```

```
    "tag": ["one", "two", "red", "blue"]
```

```
}
```

```
["one",  
 "two",  
 "red",  
 "blue"]
```

doc 1

one	→	
two	→	
red	→	
blue	→	

doc	
1	
1	
1	
1	

exercise 3: array play

```
# Insert a bunch of different docs by changing the things  
in bold:
```

```
PUT /test/book/1  
{  
  "title": "All About Fish",  
  "tag": ["one", "two", "red", "blue"]  
}
```

exercise 3: array play

```
# Insert a bunch of different docs by changing the things  
in bold:
```

```
PUT /test/book/1  
{  
  "title": "All About Fish",  
  "tag": ["one", "two", "red", "blue"]  
}  
  "two"  
  ["one", "red"]
```

exercise 3: array play

```
# Insert a bunch of different docs by changing the things  
in bold:
```

```
PUT /test/book/1  
{  
  "title": "All About Fish",  
  "tag": ["one", "two", "red", "blue"]  
}  
  ["two"  
    ["one", "red"]
```

```
# A sample query--try changing the bold things:
```

```
GET /test/book/_search  
{  
  "query": {  
    "match_all": {}  
  },  
  "filter": {  
    "term": {"tag": ["two", "three"]}  
  }  
}  
  ["red"  
    ["blue"]
```



mappings

implicit mappings

implicit mappings

```
# Make a new album doc:
```

```
PUT /test/album/1
```

```
{
```

implicit mappings

```
# Make a new album doc:
```

```
PUT /test/album/1
```

```
{  
  "title": "Fish Sounds",
```

implicit mappings

```
# Make a new album doc:  
PUT /test/album/1  
{  
  "title": "Fish Sounds",  
  "gapless_playback": true,
```

implicit mappings

```
# Make a new album doc:
```

```
PUT /test/album/1
```

```
{  
  "title": "Fish Sounds",  
  "gapless_playback": true,  
  "length_seconds": 210000,
```

implicit mappings

```
# Make a new album doc:
```

```
PUT /test/album/1
```

```
{  
  "title": "Fish Sounds",  
  "gapless_playback": true,  
  "length_seconds": 210000,  
  "weight": 1.22,
```

implicit mappings

```
# Make a new album doc:  
PUT /test/album/1  
{  
  "title": "Fish Sounds",  
  "gapless_playback": true,  
  "length_seconds": 210000,  
  "weight": 1.22,  
  "released": "2013-01-23"  
}
```

implicit mappings

```
# Make a new album doc:
```

```
PUT /test/album/1
```

```
{  
  "title": "Fish Sounds",  
  "gapless_playback": true,  
  "length_seconds": 210000,  
  "weight": 1.22,  
  "released": "2013-01-23"  
}
```

```
# See what kind of mapping ES
```

```
guessed:
```

```
GET /test/album/_mapping
```

implicit mappings

```
# Make a new album doc:
```

```
PUT /test/album/1
```

```
{  
  "title": "Fish Sounds",  
  "gapless_playback": true,  
  "length_seconds": 210000,  
  "weight": 1.22,  
  "released": "2013-01-23"  
}
```

```
# See what kind of mapping ES  
guessed:
```

```
GET /test/album/_mapping
```

```
{  
  "test": {  
    "mappings": {  
      "album": {  
        "properties": {  
          "title": {  
            "type": "string"  
          },  
          "gapless_playback": {  
            "type": "boolean"  
          },  
          "length_seconds": {  
            "type": "long"  
          },  
          "weight": {  
            "type": "double"  
          },  
          "released": {  
            "type": "date",  
            "format": "dateOptionalTime"  
          }  
        }  
      }  
    }  
  }  
}
```

explicit mappings

```
{  
  "test" : {  
    "mappings" : {  
      "album" : {  
        "properties" : {  
          "title" : {  
            "type" : "string"  
          },  
          "gapless_playback" : {  
            "type" : "boolean"  
          },  
          "length_seconds" : {  
            "type" : "long"  
          },  
          "weight" : {  
            "type" : "double"  
          },  
          "released" : {  
            "type" : "date",  
            "format" :  
              "dateOptionalTime"  
          }  
        }  
      }  
    }  
  }  
}
```

explicit mappings

DELETE /test/album

```
{  
  "test" : {  
    "mappings" : {  
      "album" : {  
        "properties" : {  
          "title" : {  
            "type" : "string"  
          },  
          "gapless_playback" : {  
            "type" : "boolean"  
          },  
          "length_seconds" : {  
            "type" : "long"  
          },  
          "weight" : {  
            "type" : "double"  
          },  
          "released" : {  
            "type" : "date",  
            "format" :  
              "dateOptionalTime"  
          }  
        }  
      }  
    }  
  }  
}
```

explicit mappings

DELETE /test/album

```
{  
  "test" : {  
    "mappings" : {  
      "album" : {  
        "properties" : {  
          "title" : {  
            "type" : "string"  
          },  
          "gapless_playback" : {  
            "type" : "boolean"  
          },  
          "length_seconds" : {  
            "type" : "long"  
          },  
          "weight" : {  
            "type" : "double"  
          },  
          "released" : {  
            "type" : "date",  
            "format" :  
              "dateOptionalTime"  
          }  
        }  
      }  
    }  
  }  
}
```

explicit mappings

```
DELETE /test/album
```

```
PUT /test/_mapping/album
{
  "properties" : {
    "title" : {
      "type" : "string"
    },
    "gapless_playback" : {
      "type" : "boolean"
    },
    "length_seconds" : {
      "type" : "long"
    },
    "weight" : {
      "type" : "double"
    },
    "released" : {
      "type" : "date",
      "format" : "dateOptionalTime"
    }
  }
}
```



```
{
  "test" : {
    "mappings" : {
      "album" : {
        "properties" : {
          "title" : {
            "type" : "string"
          },
          "gapless_playback" : {
            "type" : "boolean"
          },
          "length_seconds" : {
            "type" : "long"
          },
          "weight" : {
            "type" : "double"
          },
          "released" : {
            "type" : "date",
            "format" :
              "dateOptionalTime"
          }
        }
      }
    }
  }
}
```

exercise 1: explicit mappings

1. Delete the “album” doctype, if you’ve made one by following along.
2. Think of an album which would prompt ES to guess a wrong type.
3. Insert it, and GET the `_mapping` to show the wrong guess.
4. Delete all “album” docs again so you can change the mapping.
5. Set a mapping explicitly so you can’t fool ES anymore.

Lurking Horrors



queries

queries

HTTP/REST (just like we've been doing)

queries

HTTP/REST (just like we've been doing)

- **String queries:** short “query-string queries”

queries

HTTP/REST (just like we've been doing)

- **String queries:** short “query-string queries”
- **JSON queries:** powerful “the query DSL” in the docs

exercise 1: bulk-loading

- Bulk-load a small test data set to use for querying.
- This is exercise_1 in the queries/ directory of the git repo, so you can cut and paste or execute it directly.

```
DELETE /test
```

```
POST /_bulk
```

```
{"index": ...}
```



```
{
```

```
  "took": 2,
```

```
  "errors": false,
```

exercise 2: getting

- Grab a single document to make sure the data is there.
- This is exercise_2 in the queries/ directory of the repo, so you can cut and paste.

GET /test/book/1

exercise 3: string queries

GET /test/book/_search?q=title:Python

Here are some other things to try. Try them in a web browser so you don't have to turn spaces into pluses:

exercise 3: string queries

GET /test/book/_search?q=title:Python

Here are some other things to try. Try them in a web browser so you don't have to turn spaces into pluses:

title:python OR sharks

title:python AND dogs

author:"Julia Smith"

author:Juli? Sm*

author:/Juli[ae] Smith/

pages:(>=100 AND <200)

string queries: meh

string queries: meh

- Good for mucking around on the command line

string queries: meh

- Good for mucking around on the command line
- Bad for power

string queries: meh

- Good for mucking around on the command line
- Bad for power
- Bad to construct

string queries: meh

- Good for mucking around on the command line
- Bad for power
- Bad to construct
- Bad to expose to user input

JSON queries

JSON queries

“The query DSL” (domain-specific-language)

JSON queries

“The query DSL” (domain-specific-language)

An AST (abstract syntax tree) of queries

JSON queries

“The query DSL” (domain-specific-language)

An AST (abstract syntax tree) of queries

- Easy to construct programmatically

JSON queries

“The query DSL” (domain-specific-language)

An AST (abstract syntax tree) of queries

- Easy to construct programmatically
- Easy to adapt from user input

JSON queries

“The query DSL” (domain-specific-language)

An AST (abstract syntax tree) of queries

- Easy to construct programmatically
- Easy to adapt from user input
- Able to draw on all of ES's abilities

JSON queries

“The query DSL” (domain-specific-language)

An AST (abstract syntax tree) of queries

- Easy to construct programmatically
- Easy to adapt from user input
- Able to draw on all of ES's abilities
- A complete pain to read or type

exercise 4: a JSON query

Run this query (copy & paste from exercise_4):

```
GET /test/book/_search
{
  "query": {
    "match": {"title": "python"}
  }
}
```

What do you notice about the results?

filters vs. queries

The Query DSL

filters vs. queries

The Query DSL

Filters

filters vs. queries

The Query DSL

Filters

Queries

filters vs. queries

Filters

Queries

filters vs. queries

Filters

Queries

- Boolean: document matches or it does not

filters vs. queries

Filters

Queries

- Boolean: document matches or it does not
- Faster

filters vs. queries

Filters

Queries

- Boolean: document matches or it does not
- Faster
- Cacheable

filters vs. queries

Filters

Queries

- Boolean: document matches or it does not
 - Faster
 - Cacheable
-
- Scoring: to what degree does it match?

filters vs. queries

Filters

- Boolean: document matches or it does not
- Faster
- Cacheable

Queries

- Scoring: to what degree does it match?
- Not cached

filters vs. queries

Filters

- Boolean: document matches or it does not
- Faster
- Cacheable

Queries

- Scoring: to what degree does it match?
- Not cached

Filter when you can; query when you must.

filters and queries together

```
GET /test/book/_search
{
  "query": {
    "filtered": {
      "filter": {
        "term": {
          "category": "Web Development"
        }
      },
      "query": {
        "bool": {
          "should": [
            {
              "match": {
                "title": "Python"
              }
            },
            {
              "match": {
                "summary": "Python"
              }
            }
          ]
        }
      }
    }
  }
}
```

filters and queries together

```
GET /test/book/_search
{
  "query": {
    "filtered": { ←
      "filter": {
        "term": {
          "category": "Web Development"
        }
      },
      "query": {
        "bool": {
          "should": [
            {
              "match": {
                "title": "Python"
              }
            },
            {
              "match": {
                "summary": "Python"
              }
            }
          ]
        }
      }
    }
  }
}
```

filters and queries together

```
GET /test/book/_search
{
  "query": {
    "filtered": { ←
      "filter": {
        "term": {
          "category": "Web Development"
        }
      },
      "query": {
        "bool": {
          "should": [
            {
              "match": {
                "title": "Python"
              }
            },
            {
              "match": {
                "summary": "Python"
              }
            }
          ]
        }
      }
    }
  }
}
```

filters and queries together

```
GET /test/book/_search
{
  "query": {
    "filtered": { ←
      "filter": {
        "term": {
          "category": "Web Development"
        }
      },
      "query": {
        "bool": {
          "should": [
            {
              "match": {
                "title": "Python"
              }
            },
            {
              "match": {
                "summary": "Python"
              }
            }
          ]
        }
      }
    }
  }
}
```

exercise 5: a filtered query

Run that query.

(Copy & paste from queries/exercise_5.)

```
GET /test/book/_search
{
  "query": {
    "filtered": {
      "filter": {
        "term": {
          "category": "Web Development"
        }
      },
      "query": {
        "bool": {
          "should": [
            {
              "match": {
                "title": "Python"
              }
            },
            {
              "match": {
                "summary": "Python"
              }
            }
          ]
        }
      }
    }
  }
}
```

exercise 5: oh noes!

Where are my results?!

exercise 6: explainin' to do

Like many relational DBs, ES provides an explainer.
Let's run it on this query (see `exercise_6`).

```
GET /test/book/4/_explain
{
  "query": {
    "filtered": {
      "filter": {
        "term": {
          "category": "Web Development"
        }
      },
      "query": {
        "bool": {
          "should": [
            {
              "match": {
                "title": "Python"
              }
            }
          ]
        }
      }
    }
  }
}
```

exercise 6 results

```
{  
  "_index": "test",  
  "_type": "book",  
  "_id": "4",  
  "matched": false,  
  "explanation": {  
    "value": 0,  
    "description": "failure to match filter:  
      cache(category:Web Development)",  
    "details": [  
      ...  
    ]  
  }  
}
```

tricky term queries

“Web Development” will be broken into the terms
“web” and **“development”**.

tricky term queries

“Web Development” will be broken into the terms “web” and “development”.

term	doc
web →	3 4 5 3 4 5
development →	

tricky term queries

“Web Development” will be broken into the terms “web” and “development”.

term	doc
web →	3 4 5
development →	3 4 5

The term “Web Development” is not indexed anywhere.

tricky term queries

```
"term": {  
    "category": "Web Development"  
}
```

term	doc
web →	3 4 5
development →	3 4 5

The term “Web Development” is not indexed anywhere.

tricky term queries

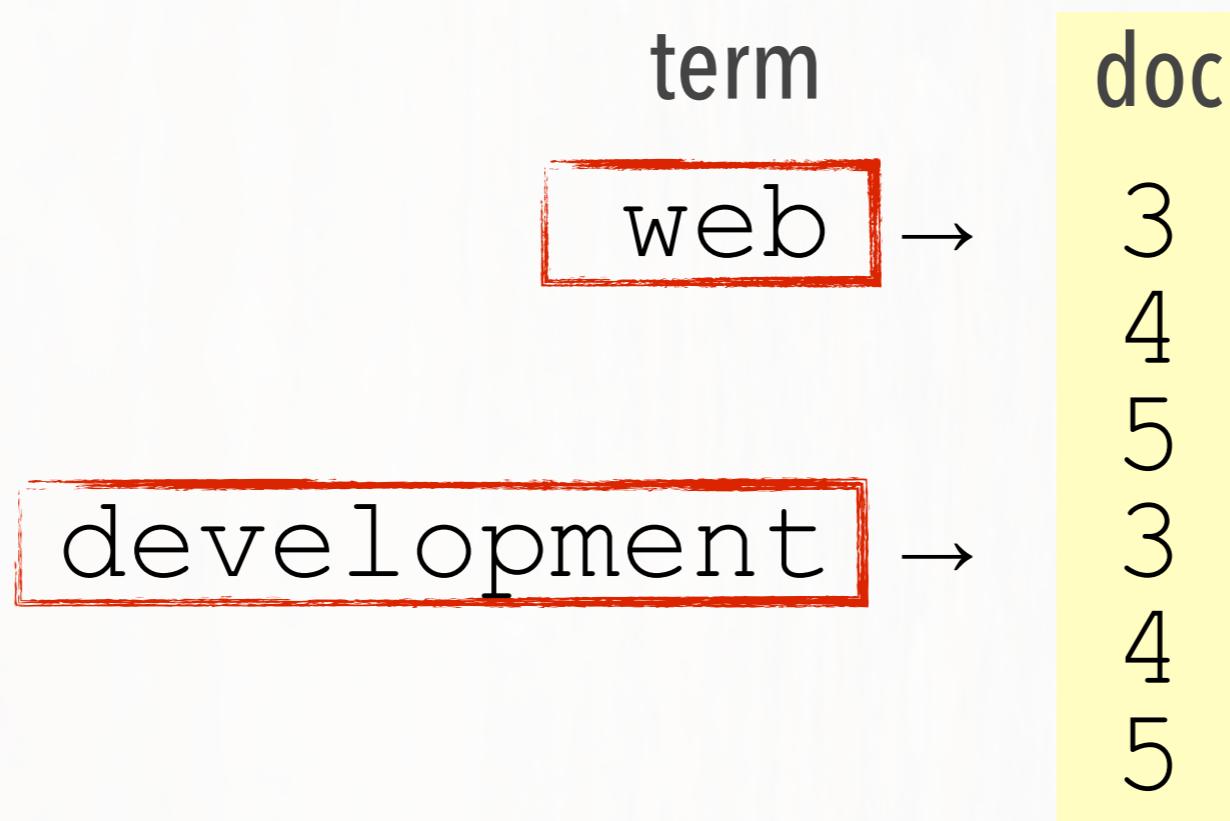
```
"term": {  
    "category": "Web Development"  
}
```

term	doc
web	3
development	4
	5
	3
	4
	5

The term “Web Development” is not indexed anywhere.

tricky term queries

```
"term": {  
    "category": "Web Development"  
}
```



The term “Web Development” is not indexed anywhere.

exercise 7

Write a query that finds the 3 books about Python in the Web Development category.

exercise 7

Write a query that finds the 3 books about Python in the Web Development category.

- Start with the query from exercise_6.

exercise 7

Write a query that finds the 3 books about Python in the Web Development category.

- Start with the query from exercise_6.
- Make up a new parameter for the **term** query so it returns something, given what we know about the index structure.

exercise 7

Write a query that finds the 3 books about Python in the Web Development category.

- Start with the query from exercise_6.
- Make up a new parameter for the **term** query so it returns something, given what we know about the index structure.
- Capital and lowercase params behave differently. Why?

exercise 7

Write a query that finds the 3 books about Python in the Web Development category.

- Start with the query from exercise_6.
- Make up a new parameter for the **term** query so it returns something, given what we know about the index structure.
- Capital and lowercase params behave differently. Why?
- One solution is in exercise_7. Take a couple minutes before peeking. There are many possible solutions.

exercise 7

Write a query that finds the 3 books about Python in the Web Development category.

- Start with the query from exercise_6.
- Make up a new parameter for the **term** query so it returns something, given what we know about the index structure.
- Capital and lowercase params behave differently. Why?
- One solution is in exercise_7. Take a couple minutes before peeking. There are many possible solutions.
- What's a downside of the exercise_7 solution?

term vs. match

A term filter (or term query) looks for its param as a whole term—they are *not analyzed*.

term vs. match

A term filter (or term query) looks for its param as a whole term—they are not *analyzed*.

```
"query": {  
    "match": {  
        "category": "web development"  
    }  
}
```

3 hits

term vs. match

A term filter (or term query) looks for its param as a whole term—they are not *analyzed*.

```
"query": {  
    "match": {  
        "category": "web development"  
    }  
}
```

3 hits

```
"query": {  
    "match": {  
        "category": "DeVeLoPmEnT web"  
    }  
}
```

3 hits

unanalyzed strings

```
...  
  'category': {  
    'type': 'string',  
    'index': 'not_analyzed'  
},  
...
```

match_phrase

		doc	positions
row	→	0	[0, 1, 2]
		1	[0, 2]
		3	[2]
boat	→	0	[4]
		1	[3]
chicken	→	2	[0, 1, 2]

match_phrase

```
"query": {  
    "match_phrase": {  
        "category": "web development"  
    }  
}
```

```
GET /test/book/_search  
{  
    "query": {  
        "match_phrase": {  
            "summary": {  
                "query": "old versions of browsers",  
                "slop": 2  
            }  
        }  
    }  
}
```

boolean queries

Where are my favorites, AND, OR, and NOT?

```
curl -s -XGET 'http://localhost:9200/test/book/_search?  
pretty' -d '{  
  "query": {  
    "bool": {  
      "must": {  
        "match": {  
          "summary": "drummers"  
        }  
      },  
      "should": [  
        {  
          "match": {  
            "title": "Python"  
          }  
        ...  
      ],  
      "must_not": {  
        ...  
      }  
    }  
  }  
}'
```

Nest them as much as you like.

boolean bonuses

- `minimum_should_match` is the number of **should** clauses that have to match.

exercise 8: bool query

Find books about Python targeted at drummers and scientists:

```
GET /test/book/_search
{
  "query": {
    "bool": {
      "must": {
        "bool": {
          "should": [
            {
              "match": {
                "summary": "drummers"
              }
            },
            {
              "match": {
                "summary": "scientists"
              }
            }
          ]
        }
      },
      "should": [
        {
          "match": {
            "title": "Python"
          }
        }
      ]
    }
  }
}
```

exercise 8: bool query

Edit the query to no longer return books that mention dogs.

exercise 8: bool query

Edit the query to no longer return books that mention dogs.

```
...
    "match": {
        "summary": "Python"
    }
},
"must_not": {
    "multi_match": {
        "query": "dogs canines",
        "fields": ["title", "summary"]
    }
}
```

filter and query smorgasbord

Filters

- And Filter
- Bool Filter
- Exists Filter
- Geo Bounding Box Filter
- Geo Distance Filter
- Geo Distance Range Filter
- Geo Polygon Filter
- GeoShape Filter
- Geohash Cell Filter
- Has Child Filter
- Has Parent Filter
- Ids Filter
- Indices Filter
- Limit Filter
- Match All Filter
- Missing Filter
- Nested Filter
- Not Filter
- Or Filter
- Prefix Filter
- Query Filter

Queries

- Query Filter
- Range Filter
- Regexp Filter
- Script Filter
- Term Filter
- Terms Filter
- Type Filter
- 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
- 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
- Span Or Query
- Span Term Query
- Term Query
- Terms Query
- Top Children Query
- Wildcard Query
- Minimum Should Match
- Multi Term Query Rewrite
- Template Query

aggregations

- Aggregations let you put returned documents into buckets and run metrics over those buckets.
- Useful for drill-down navigation of data

exercise 9: aggregations

Run a sample aggregation: exercise_9

```
GET /test/book/_search
{
  "size": 0,
  "aggs": {
    "category": {
      "terms": {
        "field": "category"
      }
    }
  }
}
```

exercise 9: aggregations

Run a sample aggregation: exercise_9

```
GET /test/book/_search
{
  "size": 0,
  "aggs": {
    "category": {
      "terms": {
        "field": "category"
      }
    }
  }
}

...
"aggregations": {
  "category": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 0,
    "buckets": [
      {
        "key": "development",
        "doc_count": 4
      },
      {
        "key": "web",
        "doc_count": 4
      },
      {
        "key": "games",
        "doc_count": 1
      },
      {
        "key": "pirate",
        "doc_count": 1
      }
    ]
  }
}
```

more aggregations

```
GET /test/book/_search
{
  "query": {
    "match": {"title": "Python"}
  },
  "aggs": {
    "most_pages": {
      "max": {"field": "pages"}
    }
  }
}
```

more aggregations

```
GET /test/book/_search
```

```
{  
  "query": {  
    "match": {"title": "Python"}  
  },  
  "aggs": {  
    "most_pages": {  
      "max": {"field": "pages"}  
    }  
  }  
}
```

- Min Aggregation
- Max Aggregation
- Sum Aggregation
- Avg Aggregation
- Stats Aggregation
- Extended Stats Aggregation
- Value Count Aggregation
- Percentiles Aggregation
- Percentile Ranks Aggregation
- Cardinality Aggregation
- Geo Bounds Aggregation
- Top hits Aggregation
- Scripted Metric Aggregation
- Global Aggregation
- Filter Aggregation
- Filters Aggregation
- Missing Aggregation
- Nested Aggregation
- Reverse nested Aggregation
- Children Aggregation
- Terms Aggregation
- Significant Terms Aggregation
- Range Aggregation
- Date Range Aggregation
- IPv4 Range Aggregation
- Histogram Aggregation
- Date Histogram Aggregation
- Geo Distance Aggregation
- GeoHash grid Aggregation

scoring

Adjusting relevance scores of results:

scoring

Adjusting relevance scores of results:

- **Boosting:** weigh one part of a query more heavily than others

scoring

Adjusting relevance scores of results:

- **Boosting:** weigh one part of a query more heavily than others
- **Function-score queries**

scoring

Adjusting relevance scores of results:

- **Boosting:** weigh one part of a query more heavily than others
- **Function-score queries**
- **Constant score queries:** pre-set scores for parts of a query. Excellent for filters.

scoring: boosting

```
"query": {  
    "bool": {  
        "should": [  
            {  
                "term": {  
                    "title": {  
                        "value": "python",  
                        "boost": 2.0  
                    }  
                }  
            },  
            {  
                "term": {  
                    "summary": "python"  
                }  
            }  
        ]  
    }  
}
```

scoring: functions

```
GET /test/book/_search
{
  "query": {
    "function_score": {
      "query": {
        "match": {"title": "Python"}
      },
      "script_score": {
        "script": "_score * doc['rating'].value"
      }
    }
  }
}
```

scripting languages

- Groovy (default as of 1.4)
- Plugins for:
 - JS
 - Python
 - Clojure
 - mvel



analysis

stock analyzers

original: Red-orange gerbils live at #43A Franklin St.

whitespace: Red-orange gerbils live at #43A Franklin St.

standard: red orange gerbils live at 43a franklin st

simple: red orange gerbils live at a franklin st

stop: red orange gerbils live franklin st

snowball: red orang gerbil live 43a franklin st

stock analyzers

original: Red-orange gerbils live at #43A Franklin St.

whitespace: Red-orange gerbils live at #43A Franklin St.

standard: red orange gerbils live at 43a franklin st

simple: red orange gerbils live at a franklin st

stop: red orange gerbils live franklin st

snowball: red orang gerbil live 43a franklin st

- stopwords

stock analyzers

original: Red-orange gerbils live at #43A Franklin St.

whitespace: Red-orange gerbils live at #43A Franklin St.

standard: red orange gerbils live at 43a franklin st

simple: red orange gerbils live at a franklin st

stop: red orange gerbils live franklin st

snowball: red orang gerbil live 43a franklin st

- stopwords
- stemming

stock analyzers

original: Red-orange gerbils live at #43A Franklin St.

whitespace: Red-orange gerbils live at #43A Franklin St.

standard: red orange gerbils live at 43a franklin st

simple: red orange gerbils live at a franklin st

stop: red orange gerbils live franklin st

snowball: red orang gerbil live 43a franklin st

- stopwords
- stemming
- punctuation

stock analyzers

original: Red-orange gerbils live at #43A Franklin St.

whitespace: Red-orange gerbils live at #43A Franklin St.

standard: red orange gerbils live at 43a franklin st

simple: red orange gerbils live at a franklin st

stop: red orange gerbils live franklin st

snowball: red orang gerbil live 43a franklin st

- stopwords
- stemming
- punctuation
- case-folding

GET /_analyze?analyzer=whitespace&text=Red-orange
gerbils live at #43A Franklin St.



```
{  
  "tokens" : [ {  
    "token" : "Red-orange",  
    "start_offset" : 0,  
    "end_offset" : 10,  
    "type" : "word",  
    "position" : 1  
  }, {  
    "token" : "gerbils",  
    "start_offset" : 11,  
    "end_offset" : 18,  
    "type" : "word",  
    "position" : 2  
  }, ...
```

exercise 1: find 10 stopwords

```
GET /_analyze?analyzer=snowball&pretty&text=The  
word "an" is a stopword.
```

Hint: Run the above and see what happens.

exercise 1: solution

GET /_analyze?analyzer=stop&text=The an is a with
that be for to and snookums

exercise 1: solution

GET /_analyze?analyzer=stop&text=The an is a with
that be for to and snookums



{

```
"tokens" : [ {  
    "token" : "snookums",  
    "start_offset" : 36,  
    "end_offset" : 44,  
    "type" : "word",  
    "position" : 11  
} ]
```

}

exercise 1: solution

GET /_analyze?analyzer=stop&text=The an is a with
that be for to and snookums



```
{  
  "tokens" : [ {  
    "token" : "snookums",  
    "start_offset" : 36,  
    "end_offset" : 44,  
    "type" : "word",  
    "position" : 11 —————→  
  } ]  
}
```

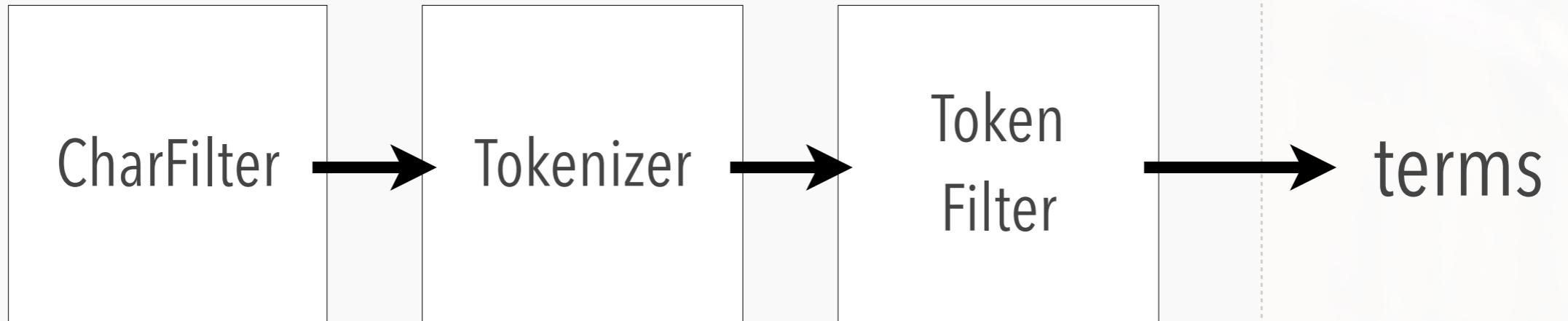
positions
[0 , 1 , 2]
[0 , 2]
[2]
[4]
[11]
[0 , 1 , 2]

analyzers in mappings

```
PUT /test/_mapping/album
{
  "properties": {
    "title": {
      "type": "string"
    },
    "description": {
      "type": "string",
      "analyzer": "snowball"
    },
    ...
  }
}
```

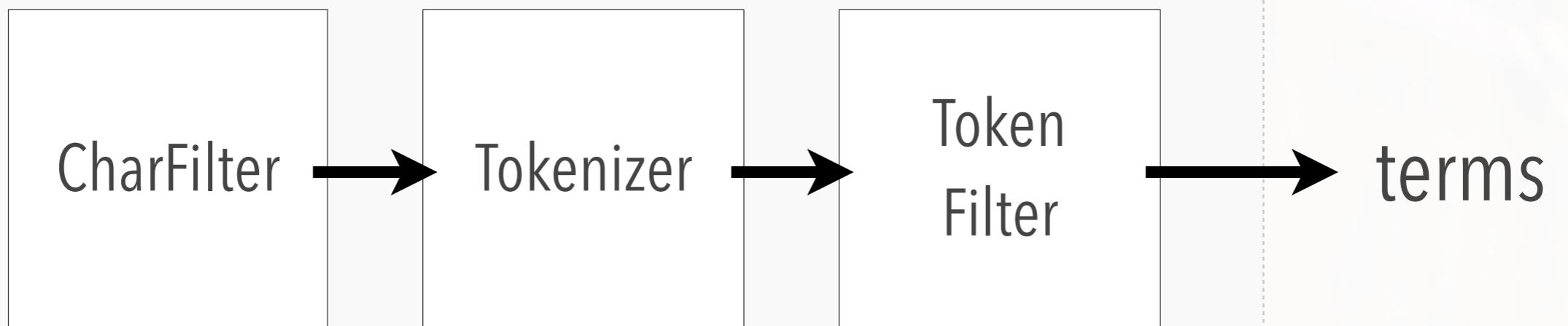
analyzer internals

name_analyzer



analyzer internals

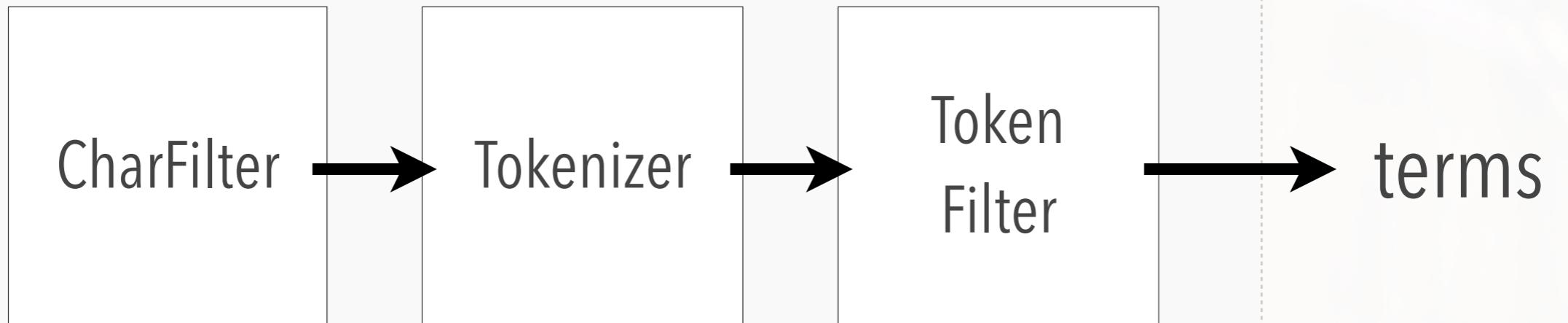
name_analyzer



O'Brien

analyzer internals

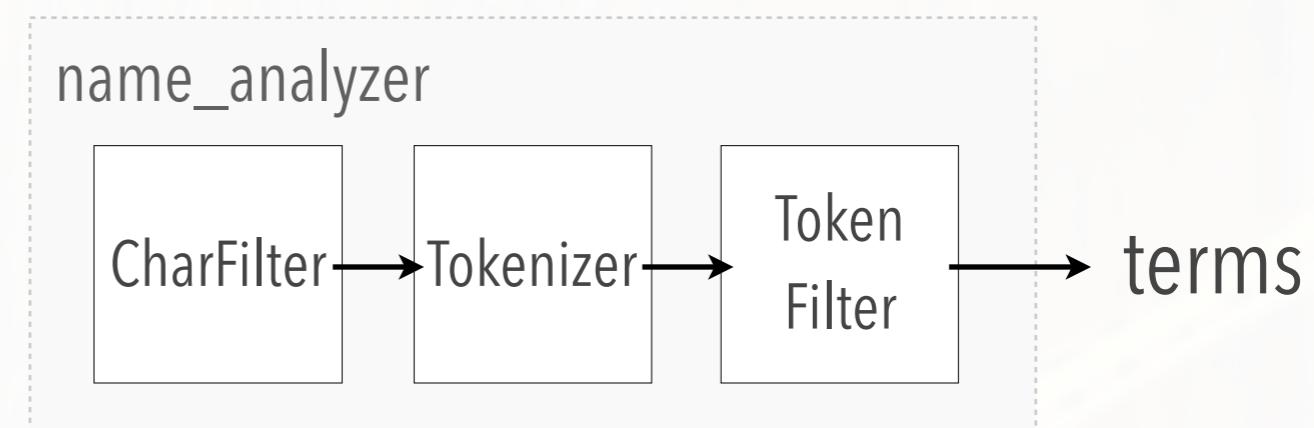
name_analyzer



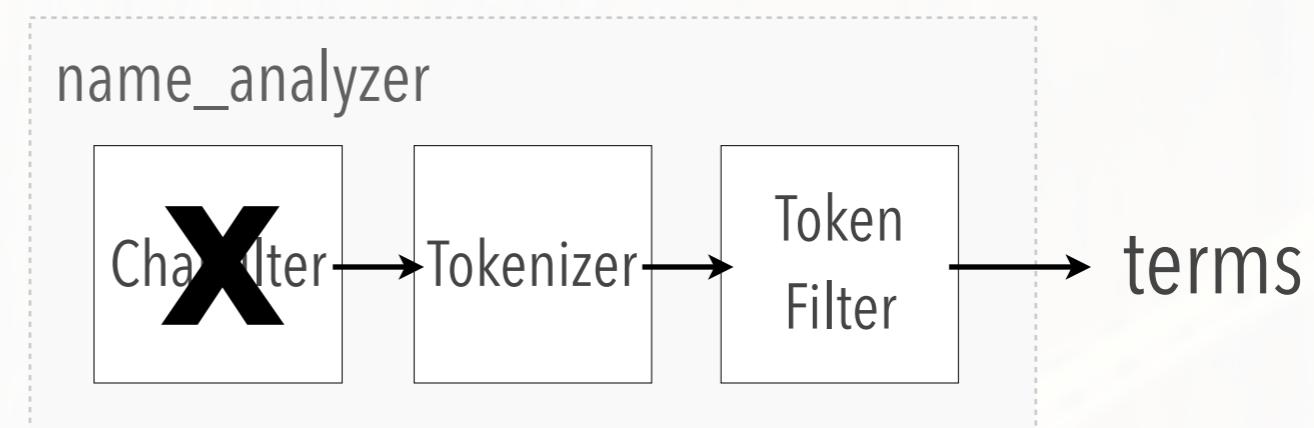
O

Brien

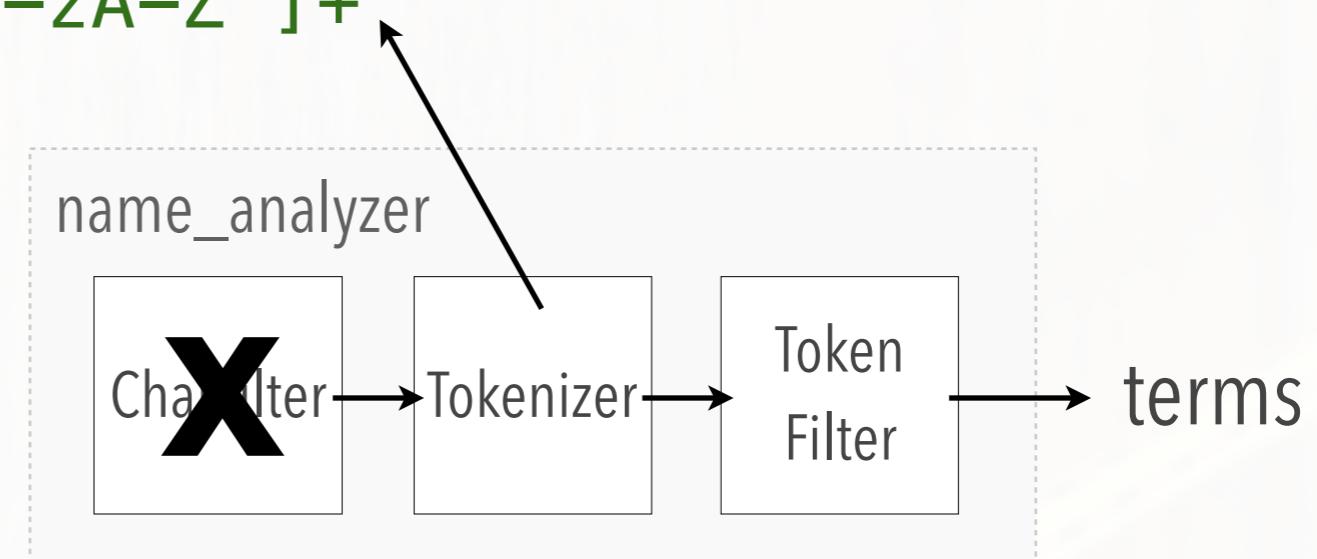
```
"analysis": {  
    "analyzer": {  
        "name_analyzer": {  
            "type": "custom",  
            "tokenizer": "name_tokenizer",  
            "filter": ["lowercase"]  
        }  
    },  
    "tokenizer": {  
        "name_tokenizer": {  
            "type": "pattern",  
            "pattern": "[^a-zA-Z']+"  
        }  
    }  
}
```



```
"analysis": {  
    "analyzer": {  
        "name_analyzer": {  
            "type": "custom",  
            "tokenizer": "name_tokenizer",  
            "filter": ["lowercase"]  
        }  
    },  
    "tokenizer": {  
        "name_tokenizer": {  
            "type": "pattern",  
            "pattern": "[^a-zA-Z']+"  
        }  
    }  
}
```

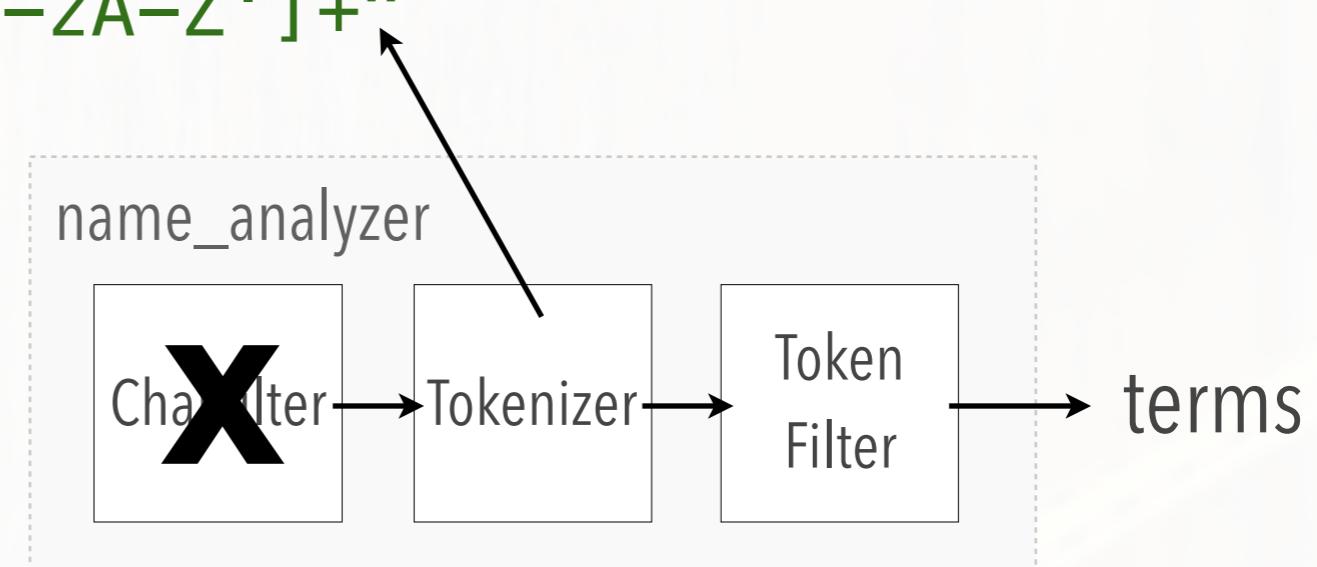


```
"analysis": {  
    "analyzer": {  
        "name_analyzer": {  
            "type": "custom",  
            "tokenizer": "name_tokenizer",  
            "filter": ["lowercase"]  
        }  
    },  
    "tokenizer": {  
        "name_tokenizer": {  
            "type": "pattern",  
            "pattern": "[^a-zA-Z']+"  
        }  
    }  
}
```



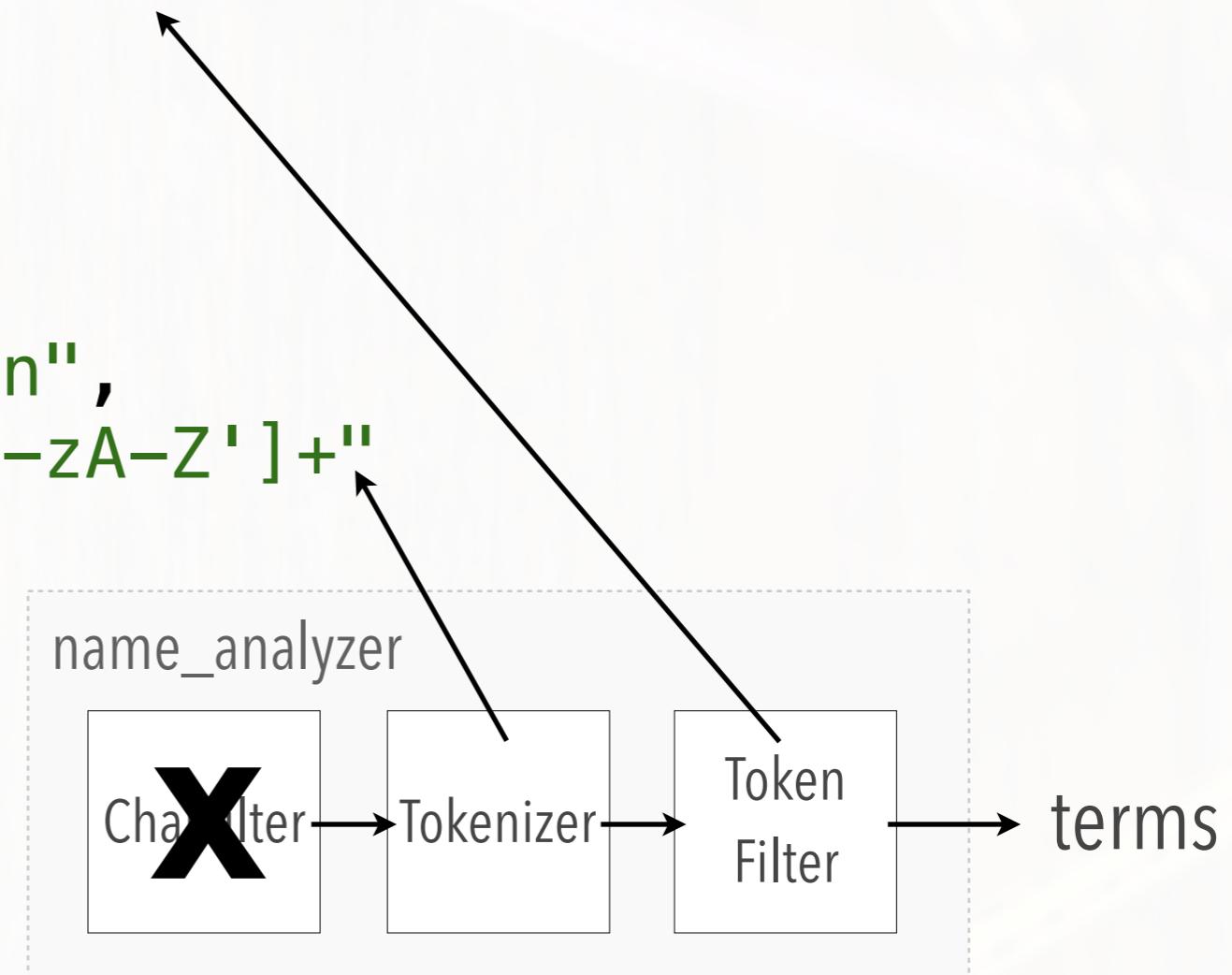
```
"analysis": {  
    "analyzer": {  
        "name_analyzer": {  
            "type": "custom",  
            "tokenizer": "name_tokenizer",  
            "filter": ["lowercase"]  
        }  
    },  
    "tokenizer": {  
        "name_tokenizer": {  
            "type": "pattern",  
            "pattern": "[^a-zA-Z']+"  
        }  
    }  
}
```

O'Brien



```
"analysis": {  
    "analyzer": {  
        "name_analyzer": {  
            "type": "custom",  
            "tokenizer": "name_tokenizer",  
            "filter": ["lowercase"]  
        }  
    },  
    "tokenizer": {  
        "name_tokenizer": {  
            "type": "pattern",  
            "pattern": "[^a-zA-Z']+"  
        }  
    }  
}
```

O'Brien



exercise 2: custom analyzer

exercise 2: custom analyzer

tags: "red, two-headed, striped, really dangerous"

exercise 2: custom analyzer

tags: "red, two-headed, striped, really dangerous"

GET /_analyze?analyzer=whitespace&text=**red, two-headed, striped, really dangerous**

exercise 2: custom analyzer

tags: "red, two-headed, striped, really dangerous"

GET /_analyze?analyzer=whitespace&text=**red, two-headed, striped, really dangerous**

red two-headed striped **really** **dangerous**

exercise 2: custom analyzer

tags: "red, two-headed, striped, really dangerous"

GET /_analyze?analyzer=whitespace&text=**red, two-headed, striped, really dangerous**

red two-headed striped **really** **dangerous**

GET /test/monster/_search
{
 "query": {
 "match_all": {}
 },
 "filter": {
 "term": {"tags": "dangerous"}
 }
}

exercise 2: custom analyzer

tags: "red, two-headed, striped, really dangerous"

GET /_analyze?analyzer=whitespace&text=**red, two-headed, striped, really dangerous**

red	two-headed	striped	really	dangerous
-----	------------	---------	--------	-----------

GET /test/monster/_search

```
{  
  "query": {  
    "match_all": {}  
  },  
  "filter": {  
    "term": {"tags": "dangerous"}  
  }  
}
```

```
...  
  "hits" : {  
    "total" : 1,  
    "max_score" : 1.0,  
    "hits" : [ {  
      "_index" : "test",  
      "_type" : "monster",  
      "_id" : "1",  
      "_score" : 1.0, "_source" : {  
        "title": "Scarlet Klackinblax",  
        "tags": "red, two-headed,  
        striped, really dangerous"  
      }  
    } ]  
  }  
}
```

exercise 2: custom analyzer

```
# How to update the "test" index's analyzers:  
PUT /test/_settings  
{  
  "analysis": {  
    "analyzer": {  
      "whitespace_analyzer": {  
        "filter": ["lowercase"],  
        "tokenizer": "whitespace_tokenizer"  
      }  
    },  
    "tokenizer": {  
      "whitespace_tokenizer": {  
        "type": "pattern",  
        "pattern": " +"  
      }  
    }  
  }  
}
```

exercise 2: custom analyzer

```
# How to update the "test" index's analyzers:  
PUT /test/_settings  
{  
  "analysis": {  
    "analyzer": {  
      "whitespace_analyzer": {  
        "filter": ["lowercase"],  
        "tokenizer": "whitespace_tokenizer"  
      }  
    },  
    "tokenizer": {  
      "whitespace_tokenizer": {  
        "type": "pattern",  
        "pattern": " +"  
      }  
    }  
  }  
}
```

```
GET /test/_analyze?analyzer=whitespace_analyzer&text=all your base are  
belong to us, dude
```

exercise 2: custom analyzer

```
# How to update the "test" index's analyzers:
```

```
PUT /test/_settings
{
  "analysis": {
    "analyzer": {
      "whitespace_analyzer": {
```

```
  {
    "error" : "ElasticsearchIllegalArgumentException[Can't
update non dynamic
settings[[index.analysis.analyzer.comma_delim.filter.0,
index.analysis.tokenizer.comma_delim_tokenizer.type,
index.analysis.tokenizer.comma_delim_tokenizer.pattern,
index.analysis.analyzer.comma_delim.tokenizer]] for open
indices[[test]]]",
    "status" : 400
  }
```

```
GET /test/_analyze?analyzer=whitespace_analyzer&text=all your base are
belong to us, dude
```

exercise 2: custom analyzer

```
# How to update the "test" index's analyzers:  
PUT /test/_settings  
{  
  "analysis": {  
    "analyzer": {  
      "whitespace_analyzer": {  
        "filter": ["lowercase"],  
        "tokenizer": "whitespace_tokenizer"  
      }  
    },  
    "tokenizer": {  
      "whitespace_tokenizer": {  
        "type": "pattern",  
        "pattern": " +"  
      }  
    }  
  }  
}
```

```
GET /test/_analyze?analyzer=whitespace_analyzer&text=all your base are  
belong to us, dude
```

exercise 2: custom analyzer

```
# How to update the "test" index's analyzers:  
PUT /test/_settings  
{  
  "analysis": {  
    "analyzer": {  
      "whitespace_analyzer": {  
        "filter": ["lowercase"],  
        "tokenizer": "whitespace_tokenizer"  
      }  
    },  
    "tokenizer": {  
      "whitespace_tokenizer": {  
        "type": "pattern",  
        "pattern": " +"  
      }  
    }  
  }  
}
```

```
GET /test/_analyze?analyzer=whitespace_analyzer&text=all your base are  
belong to us, dude
```

```
POST /test/_close
```

exercise 2: custom analyzer

```
# How to update the "test" index's analyzers:  
PUT /test/_settings  
{  
  "analysis": {  
    "analyzer": {  
      "whitespace_analyzer": {  
        "filter": ["lowercase"],  
        "tokenizer": "whitespace_tokenizer"  
      }  
    },  
    "tokenizer": {  
      "whitespace_tokenizer": {  
        "type": "pattern",  
        "pattern": " +"  
      }  
    }  
  }  
}
```

```
GET /test/_analyze?analyzer=whitespace_analyzer&text=all your base are  
belong to us, dude
```

```
POST /test/_close  
POST /test/_open
```

exercise 2: solution

exercise 2: solution

```
PUT /test/_settings
{
  "analysis": {
    "analyzer": {
      "comma_delim": {
        "filter": ["lowercase"],
        "tokenizer": "comma_delim_tokenizer"
      }
    },
    "tokenizer": {
      "comma_delim_tokenizer": {
        "type": "pattern",
        "pattern": ", +"
      }
    }
  }
}
```

exercise 2: solution

```
PUT /test/_settings
{
  "analysis": {
    "analyzer": {
      "comma_delim": {
        "filter": ["lowercase"],
        "tokenizer": "comma_delim_tokenizer"
      }
    },
    "tokenizer": {
      "comma_delim_tokenizer": {
        "type": "pattern",
        "pattern": ", +"
      }
    }
  }
}
```

exercise 2: solution

```
PUT /test/_settings
{
  "analysis": {
    "analyzer": {
      "comma_delim": {
        "filter": ["lowercase"],
        "tokenizer": "comma_delim_tokenizer"
      }
    },
    "tokenizer": {
      "comma_delim_tokenizer": {
        "type": "pattern",
        "pattern": ", +"
      }
    }
  }
}
```

```
GET /test/_analyze?analyzer=comma_delim&text=red, two-headed, striped, really dangerous
"token": "red" ... "token": "two-headed" ... "token": "striped" ...
"token": "really dangerous"
```

exercise 2: solution

```
PUT /test/_settings
{
  "analysis": {
    "analyzer": {
      "comma_delim": {
        "filter": ["lowercase"],
        "tokenizer": "comma_delim_tokenizer"
      }
    },
    "tokenizer": {
      "comma_delim_tokenizer": {
        "type": "pattern",
        "pattern": ", +"
      }
    }
  }
}
```

GET /test/_analyze?analyzer=comma_delim&text=**red, two-headed, striped, really dangerous**

"token": "red" ... "token": "two-headed" ... "token": "striped" ...
"token": **really dangerous**

ngrams

```
'analyzer': {  
    # A lowercase trigram analyzer  
    'trigramalyzer': {  
        'filter': ['lowercase'],  
        'tokenizer': 'trigram_tokenizer'  
    }  
},  
'tokenizer': {  
    'trigram_tokenizer': {  
        'type': 'nGram',  
        'min_gram': 3,  
        'max_gram': 3  
        # Keeps all kinds of chars by default.  
    }  
}
```

ngrams

“Chemieingenieurwesen”

ngrams

“Chemieingenieurwesen”

...ing
nge
gen.
enj
nie
ieu
eur...

ngrams

“

Chemieingenieurwesen

...ing nge gen eni nie ieu eur.

”

ngrams

“

Chemieingenieurwesen

”

...ing nge gen eni nie ieu eur.

ingenieur → ing nge gen eni nie ieu eur



clustering

shards



shards

```
curl -XPUT 'http://localhost:9200/twitter/' -d '  
index:  
  number_of_shards: 3  
'
```

replicas

```
curl -XPUT 'http://localhost:9200/twitter/' -d '  
index:  
  number_of_shards: 3  
  number_of_replicas: 2  
'
```

exercise: provisioning

How would you provision a cluster if we were doing lots of CPU-expensive queries on a large corpus, but only a small subset of the corpus was "hot"?

going to extremes

recommendations

recommendations

- At least 1 replica

recommendations

- At least 1 replica
- Plenty of shards—but not a million

recommendations

- At least 1 replica
- Plenty of shards—but not a million
- At least 3 nodes.

recommendations

- At least 1 replica
- Plenty of shards—but not a million
- At least 3 nodes.

Avoid split-brain:

```
discovery.zen.minimum_master_nodes: 2
```

real-life examples

real-life examples



real-life examples



too friendly

too friendly

- `discovery.zen.ping.multicast.enabled: false`

too friendly

- `discovery.zen.ping.multicast.enabled: false`
- `discovery.zen.ping.unicast.hosts: ["master1", "master2"]`

too friendly

- `discovery.zen.ping.multicast.enabled: false`
- `discovery.zen.ping.unicast.hosts:`
[“master1”, “master2”]
- `cluster.name: something_weird`

too friendly

- **discovery.zen.ping.multicast.enabled: false**
- **discovery.zen.ping.unicast.hosts:**
["master1", "master2"]
- **cluster.name: something_weird**
- Protect with a firewall, buy Shield, try the elasticsearch-http-basic plugin, or use a proxy.

adding nodes w/o downtime

adding nodes w/o downtime

- Puppet out new config file:
`discovery.zen.ping.unicast.hosts:`
`["old.example.com", ..., "new.example.com"]`

adding nodes w/o downtime

- Puppet out new config file:
`discovery.zen.ping.unicast.hosts:`
`["old.example.com", ..., "new.example.com"]`
- Bring up the new node.

beware inconsistent config

be wary of upgrades

monitoring

```
curl -XGET -s 'http://localhost:9200/_cluster/health?pretty'  
{  
  "cluster_name" : "grinchertoo",  
  "status" : "yellow",  
  "timed_out" : false,  
  "number_of_nodes" : 1,  
  "number_of_data_nodes" : 1,  
  "active_primary_shards" : 29,  
  "active_shards" : 29,  
  "relocating_shards" : 0,  
  "initializing_shards" : 0,  
  "unassigned_shards" : 26  
}
```

monitoring

```
curl -XGET -s 'http://localhost:9200/_cluster/health?pretty'  
{  
  "cluster_name" : "grinchertoo",  
  "status" : "yellow",  
  "timed_out" : false,  
  "number_of_nodes" : 1,  
  "number_of_data_nodes" : 1,  
  "active_primary_shards" : 29,  
  "active_shards" : 29,  
  "relocating_shards" : 0,  
  "initializing_shards" : 0,  
  "unassigned_shards" : 26  
}
```

```
curl -XGET -s 'http://localhost:9200/_cluster/state?pretty'  
{  
  "cluster_name" : "elasticsearch",  
  "version" : 3,  
  "master_node" : "ACuIytIIQ7G7b_Rg_G7wnA",  
  ...
```

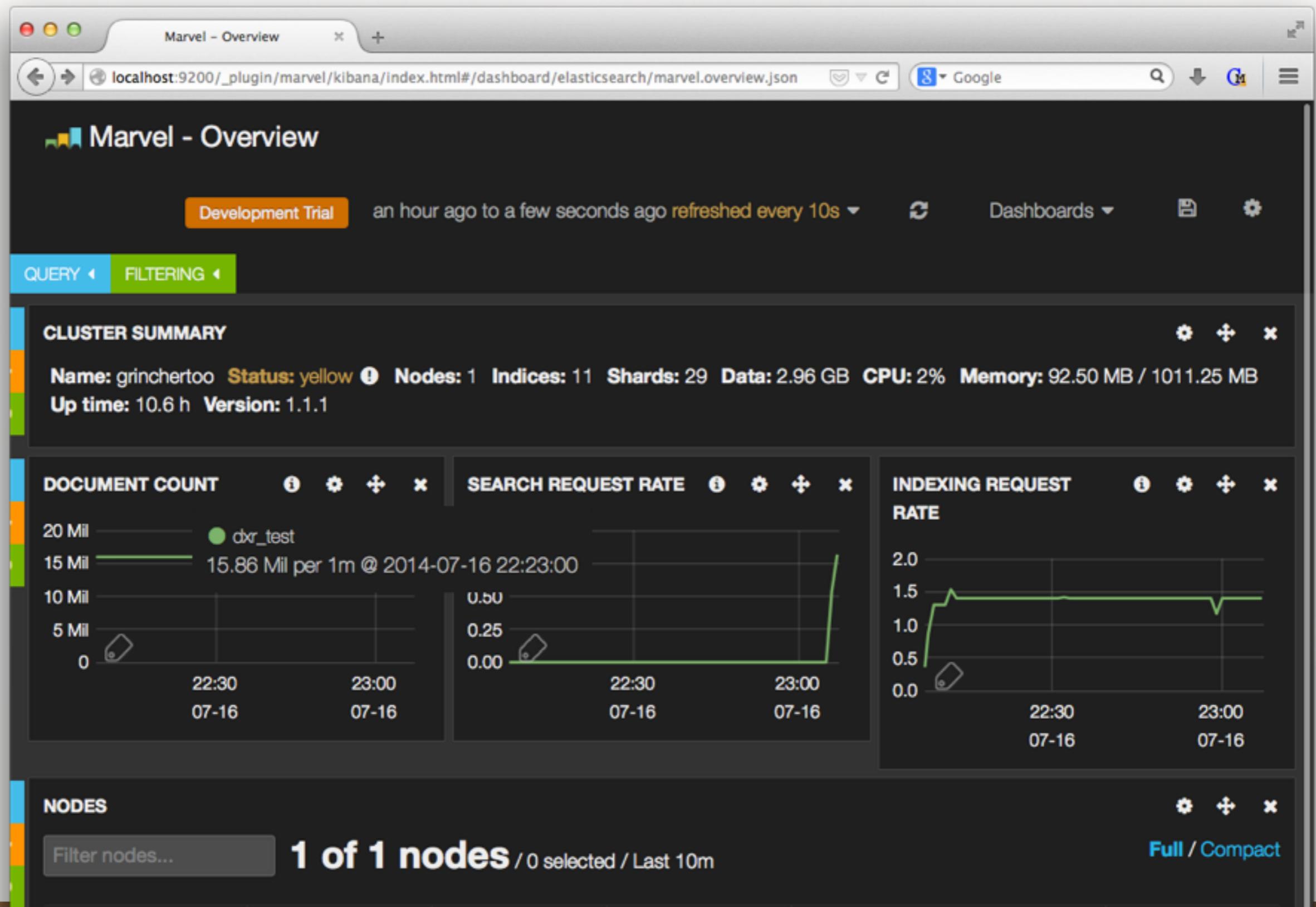
exercise: monitoring

Why is just checking for cluster color insufficient?

What could we check in addition?

```
"cluster_name" : "grinchertoo",
"status" : "yellow",
"timed_out" : false,
"number_of_nodes" : 1,
"number_of_data_nodes" : 1,
"active_primary_shards" : 29,
"active_shards" : 29,
"relocating_shards" : 0,
"initializing_shards" : 0,
"unassigned_shards" : 26
```

monitoring with marvel





m

INDICES

⚙️ + ✕

Filter indices...

11 of 11 indices / 0 selected / Last 10m

Compact view. Filter indices to 5 or less, or set the page refresh rate to 2m or greater for more options.

indices	Dashboard	Documents	Index Rate	Search Rate	Merge Rate	Field Data
■ album ⓘ		0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ http: ⓘ		1.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ test ⓘ		4.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ twitter ⓘ		0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ .marvel-2014.07.14 ⓘ		6.7 K ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ .marvel-2014.07.15 ⓘ		31.7 K ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ .marvel-2014.07.16 ⓘ		56.5 K ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ .marvel-2014.07.17 ⓘ		14.9 K ↘━	1.4 ↘━	0.8 ↗━	70.2 KB ↘━	513.1 KB ↗━
■ .marvel-kibana ⓘ		2.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ dxr_test		15.9 M ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━
■ test_vacancies_de_slave		0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━	0.0 ━━━━

<https://www.elastic.co/products/marvel>



optimization

bootstrap.mlockall: true

ES_HEAP_SIZE:
half of RAM
< 32GB

open files

open files

```
/etc/security/limits.conf:  
es_user soft  nofile 65535  
es_user hard  nofile 65535
```

open files

/etc/security/limits.conf:
es_user soft nproc 65535
es_user hard nproc 65535

/etc/init.d/elasticsearch:
ulimit -n 65535
ulimit -l unlimited

Use default stores.

RAM & JVM tuning

MySQL

shrinking indices

shrinking indices

```
% vmstat -S m -a 2
```

procs		memory				swap		io	
r	b	swpd	free	inact	active	si	so	bi	bo
1	0	4	37	54	55	0	0	0	1
0	0	4	37	54	55	0	0	0	0
0	0	4	37	54	55	0	0	0	0

shrinking indices

```
% vmstat -S m -a 2
procs -----memory----- ----swap-- -----io-----
 r b    swpd   free  inact active    si    so    bi    bo
 1 0      4     37     54     55    0    0    0    1
 0 0      4     37     54     55    0    0    0    0
 0 0      4     37     54     55    0    0    0    0
```

```
"some_doctype" : {
    "_source" : {"enabled" : false}
}
```

shrinking indices

```
% vmstat -S m -a 2
procs -----memory----- ----swap-- -----io-----
 r b    swpd   free  inact active    si    so    bi    bo
 1 0      4     37     54     55      0      0      0      1
 0 0      4     37     54     55      0      0      0      0
 0 0      4     37     54     55      0      0      0      0
```

```
"some_doctype" : {
    "_all" : {"enabled" : false}
}
```

shrinking indices

```
% vmstat -S m -a 2
```

procs		memory				swap		io	
r	b	swpd	free	inact	active	si	so	bi	bo
1	0	4	37	54	55	0	0	0	1
0	0	4	37	54	55	0	0	0	0
0	0	4	37	54	55	0	0	0	0

```
"some_doctype" : {  
    "some_field" : {"include_in_all" : false}  
}
```

shrinking indices

```
% vmstat -S m -a 2
procs -----memory----- ----swap-- -----io-----
 r b    swpd    free   inact active    si    so    bi    bo
 1 0      4     37     54     55      0      0      0      1
 0 0      4     37     54     55      0      0      0      0
 0 0      4     37     54     55      0      0      0      0
```

```
"some_doctype" : {
    "some_field" : {"index" : no}
}
```

filter caching

filter caching

```
"filter": {  
    "terms": {  
        "tags": ["red", "green"],  
        "execution": "plain"  
    }  
}
```

filter caching

```
"filter": {  
    "terms": {  
        "tags": ["red", "green"],  
        "execution": "plain"  
    }  
}  
  
"filter": {  
    "terms": {  
        "tags": ["red", "green"],  
        "execution": "bool"  
    }  
}
```



dealing with the future

mappings

expensive updates

how to reindex

how to reindex

- Use Bulk API.

how to reindex

- Use Bulk API.
- Turn off auto-refresh:

```
curl -XPUT localhost:9200/test/_settings -d '{
  "index" : {
    "refresh_interval" : "-1"
  }
}'
```

how to reindex

- Use Bulk API.
- Turn off auto-refresh:

```
curl -XPUT localhost:9200/test/_settings -d '{
  "index" : {
    "refresh_interval" : "-1"
  }
}'
```

- Find your optimal bulk size.

how to reindex

- Use Bulk API.
 - Turn off auto-refresh:

```
curl -XPUT localhost:9200/test/_settings -d '{
  "index" : {
    "refresh_interval" : "-1"
  }
}'
```

- Find your optimal bulk size.
- Set replicas to 0.

how to reindex

- Use Bulk API.
 - Turn off auto-refresh:

```
curl -XPUT localhost:9200/test/_settings -d '{
  "index" : {
    "refresh_interval" : "-1"
  }
}'
```

- Find your optimal bulk size.
- Set replicas to 0.
- Use multiple feeder processes.

how to reindex

- Use Bulk API.
 - Turn off auto-refresh:

```
curl -XPUT localhost:9200/test/_settings -d '{
  "index" : {
    "refresh_interval" : "-1"
  }
}'
```
 - Find your optimal bulk size.
 - Set replicas to 0.
 - Use multiple feeder processes.
 - Put everything back.

backups

- Backups used to be fairly cumbersome, but now there's an API for that!
- Set it up:

```
curl -XPUT 'http://localhost:9200/_snapshot/backups' -d '{  
  "type": "fs",  
  "settings": {  
    "location": "/somewhere/backups",  
    "compress": true  
  }  
}'
```

- Run a backup:

```
curl -XPUT "localhost:9200/_snapshot/backups/july20"
```

A close-up photograph of a coiled black rope with yellow and red stripes. The rope is resting on a dark, textured wooden surface. In the background, there are blurred horizontal bands of color in shades of blue, yellow, green, and purple, creating a bokeh effect.

python libs

history

pyes → pyelasticsearch → elasticsearch-py

elasticsearch-py

elasticsearch-py

- Cross-language homogeneity

```
es.indices.create('some_index') # in Python, JS, ...
es.search(...)
```

elasticsearch-py

- Cross-language homogeneity
`es.indices.create('some_index') # in Python, JS, ...
es.search(...)`
- Comprehensive feature coverage

pyelasticsearch

pyelasticsearch

```
es = ElasticSearch('http://localhost:9200')
```

pyelasticsearch

```
es = ElasticSearch('http://localhost:9200')  
  
es.search('name:joe OR name:freddy',  
          index='contacts')
```

pyelasticsearch

```
es = ElasticSearch('http://localhost:9200')

es.search('name:joe OR name:freddy',
          index='contacts')

es.search({'query': ...},
          index='contacts')
```

pyelasticsearch

```
es = ElasticSearch('http://localhost:9200')

es.search('name:joe OR name:freddy',
          index='contacts')

es.search({'query': ...},
          index='contacts')

es.index('contacts',
          'person',
          {'name': 'Joe Tester',
           'birthday': datetime(1980, 2, 23),
           'silly': True},
          id=1)
```

pyelasticsearch

```
es = ElasticSearch('http://localhost:9200')

es.search('name:joe OR name:freddy',
          index='contacts')

es.search({'query': ...},
          index='contacts')

es.index('contacts',
          'person',
          {'name': 'Joe Tester',
           'birthday': datetime(1980, 2, 23),
           'silly': True},
          id=1)

es.cluster_state()
es.optimize()
es.refresh()
```

pyelasticsearch

pyelasticsearch

- Pythonic

pyelasticsearch

- Pythonic
- Safe: `delete()` vs. `delete_all()`

pyelasticsearch

- Pythonic
- Safe: `delete()` vs. `delete_all()`
- Simple

pyelasticsearch

- Pythonic
- Safe: `delete()` vs. `delete_all()`
- Simple
- Docs

pyelasticsearch

- Pythonic
- Safe: `delete()` vs. `delete_all()`
- Simple
- Docs
- Best bulk in the business

pyelasticsearch

```
from pyelasticsearch import bulk_chunks

def documents():
    for book in books:
        yield es.index_op({'title': book.title, 'pages': book.pages})
    # index_op() also takes kwargs like index= and id= in case
    # you want more control.
    #
    # You could also yield some delete_ops or update_ops here.

# bulk_chunks() breaks your documents into smaller requests for speed:
for chunk in bulk_chunks(documents(),
                          docs_per_chunk=500,
                          bytes_per_chunk=10000):
    # We specify a default index and doc type here so we don't
    # have to repeat them in every operation:
    es.bulk(chunk, doc_type='book', index='library')
```

pyelasticsearch

```
from pyelasticsearch import bulk_chunks

def documents():
    for book in books:
        yield es.index_op({'title': book.title, 'pages': book.pages})
    # index_op() also takes kwargs like index= and id= in case
    # you want more control.
    #
    # You could also yield some delete_ops or update_ops here.

# bulk_chunks() breaks your documents into smaller requests for speed:
for chunk in bulk_chunks(documents(),
                         docs_per_chunk=500,
                         bytes_per_chunk=10000):
    # We specify a default index and doc type here so we don't
    # have to repeat them in every operation:
    es.bulk(chunk, doc_type='book', index='library')
```

pyelasticsearch

```
from pyelasticsearch import bulk_chunks

def documents():
    for book in books:
        yield es.index_op({'title': book.title, 'pages': book.pages})
    # index_op() also takes kwargs like index= and id= in case
    # you want more control.
    #
    # You could also yield some delete_ops or update_ops here.

# bulk_chunks() breaks your documents into smaller requests for speed:
for chunk in bulk_chunks(documents(),
                          docs_per_chunk=500,
                          bytes_per_chunk=10000):
    # We specify a default index and doc type here so we don't
    # have to repeat them in every operation:
    es.bulk(chunk, doc_type='book', index='library')
```

pyelasticsearch

```
from pyelasticsearch import bulk_chunks

def documents():
    for book in books:
        yield es.index_op({'title': book.title, 'pages': book.pages})
    # index_op() also takes kwargs like index= and id= in case
    # you want more control.
    #
    # You could also yield some delete_ops or update_ops here.

# bulk_chunks() breaks your documents into smaller requests for speed:
for chunk in bulk_chunks(documents(),
                          docs_per_chunk=500,
                          bytes_per_chunk=10000):
    # We specify a default index and doc type here so we don't
    # have to repeat them in every operation:
    es.bulk(chunk, doc_type='book', index='library')
```

pyelasticsearch

```
from pyelasticsearch import bulk_chunks

def documents():
    for book in books:
        yield es.index_op({'title': book.title, 'pages': book.pages})
    # index_op() also takes kwargs like index= and id= in case
    # you want more control.
    #
    # You could also yield some delete_ops or update_ops here.

# bulk_chunks() breaks your documents into smaller requests for speed:
for chunk in bulk_chunks(documents(),
                          docs_per_chunk=500,
                          bytes_per_chunk=10000):
    # We specify a default index and doc type here so we don't
    # have to repeat them in every operation:
    es.bulk(chunk, doc_type='book', index='library')
```

pyelasticsearch

```
from pyelasticsearch import bulk_chunks

def documents():
    for book in books:
        yield es.index_op({'title': book.title, 'pages': book.pages})
    # index_op() also takes kwargs like index= and id= in case
    # you want more control.
    #
    # You could also yield some delete_ops or update_ops here.

# bulk_chunks() breaks your documents into smaller requests for speed:
for chunk in bulk_chunks(documents(),
                          docs_per_chunk=500,
                          bytes_per_chunk=10000):
    # We specify a default index and doc type here so we don't
    # have to repeat them in every operation:
    es.bulk(chunk, doc_type='book', index='library')
```

pyelasticsearch

```
from pyelasticsearch import bulk_chunks

def documents():
    for book in books:
        yield es.index_op({'title': book.title, 'pages': book.pages})
    # index_op() also takes kwargs like index= and id= in case
    # you want more control.
    #
    # You could also yield some delete_ops or update_ops here.

# bulk_chunks() breaks your documents into smaller requests for speed:
for chunk in bulk_chunks(documents(),
                          docs_per_chunk=500,
                          bytes_per_chunk=10000):
    # We specify a default index and doc type here so we don't
    # have to repeat them in every operation:
    es.bulk(chunk, doc_type='book', index='library')
```

A close-up photograph of a coiled, multi-colored rope made of black, yellow, and red fibers. The rope is resting on a dark, textured wooden surface. In the background, there is a blurred, colorful object with horizontal stripes in shades of purple, blue, yellow, and green.

bonus!
synonyms

synonyms

```
"filter": {  
    "synonym": {  
        "type": "synonym",  
        "synonyms": [  
            "albert => albert, al",  
            "allan => allan, al"  
        ]  
    }  
}
```

original query: Allan Smith
after synonyms: [allan, al] smith

original query: Albert Smith
after synonyms: [albert, al] smith

synonym gotchas

synonym gotchas

- You can set up synonyms at indexing or at query time.

synonym gotchas

- You can set up synonyms at indexing or at query time.
 - Choose query time for size.

synonym gotchas

- You can set up synonyms at indexing or at query time.
 - Choose query time for size.
- You can store synonyms in a file and reference that file in your mapping.

synonym gotchas

- You can set up synonyms at indexing or at query time.
 - Choose query time for size.
- You can store synonyms in a file and reference that file in your mapping.
 - Don't.



thank you



thank you

@ErikRose
erik@mozilla.com