Elasticsearch

Jakub Podeszwik / infoShare ACADEMY / Yameo

Agenda

- Full-text search
- Apache Lucene
- Elasticsearch / Elastic stack

Full-text search

- Full-text database contains complete text of books, articles, e.t.c. They are stored as documents.
- examine all words in every stored document and return only documents that match
- serial scanning vs indexing

Inverted index

1. Tom has a cat

2. Kate has a dog

3. Mike has an owl

Term	Documents
a	1, 2
an	3
cat	1
dog	2
has	1, 2, 3
kate	2
mike	3
owl	3
tom	1

Inverted index with fields

1.
title: The cat
text: The cat sits

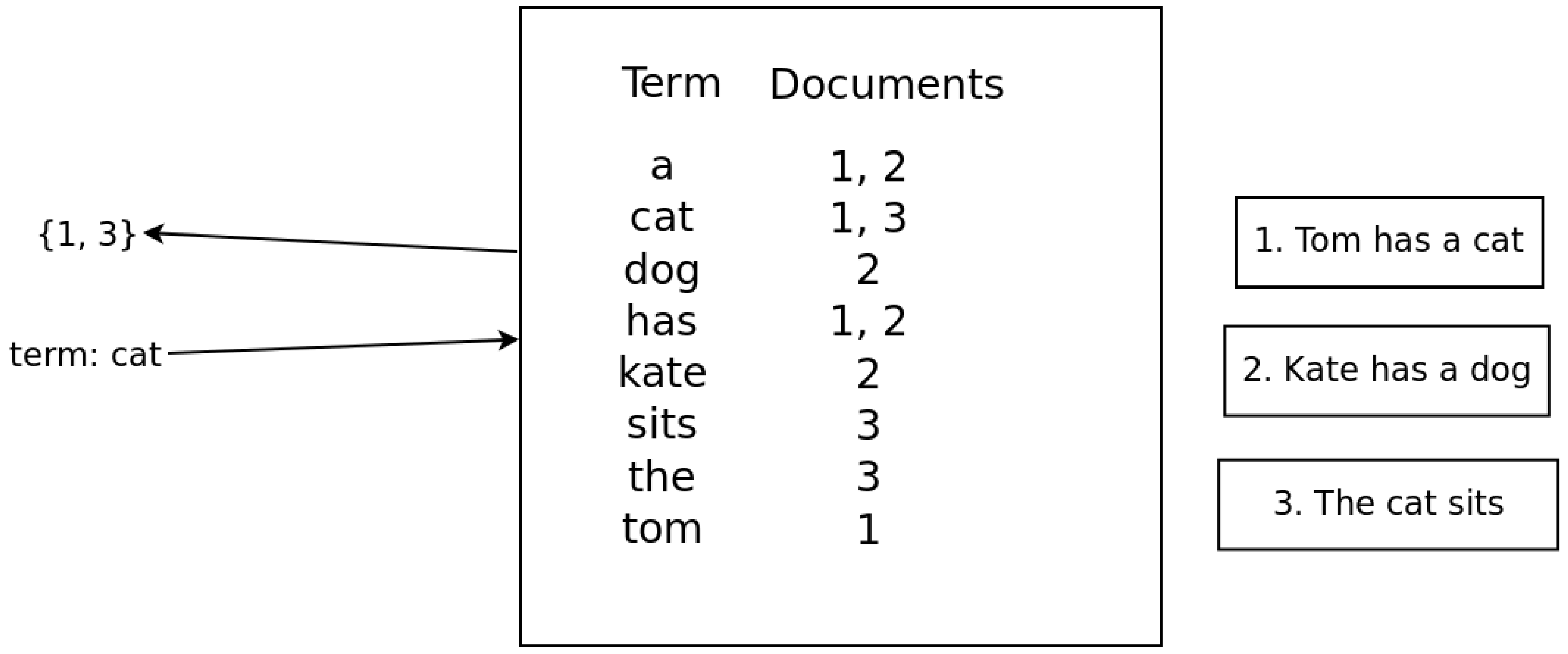
2.
title: The dog
text: The dog stands

Term	Documents
text:cat	1
text:dog	2
text:sits	1
text:stands	2
text:the	1, 2
title:cat	1
title:dog	2
title:the	1,2

Apache lucene

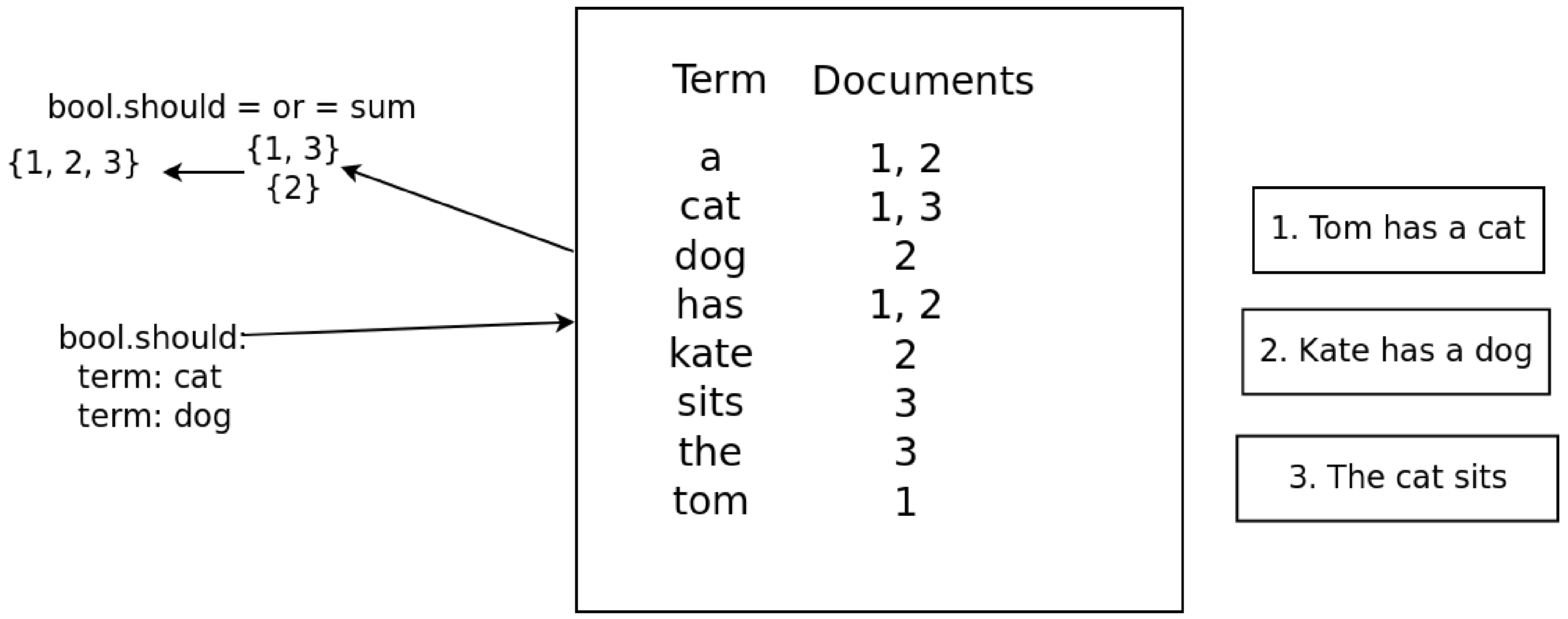
- high performance full-text search library written in Java
- open source (Apache license)
- initially released in 1999

Term query



Term	Documents
a	1, 2
cat	1, 3
dog	2
has	1, 2
kate	2
sits	3
the	3
tom	1

Boolean query



Term	Documents
a	1, 2
cat	1, 3
dog	2
has	1, 2
kate	2
sits	3
the	3
tom	1

https://github.com/jpodeszwik/elasticsearch-workshop-01-2019

Inverted index

1. Tom has a cat

2. Kate has a dog

3. Mike has an owl

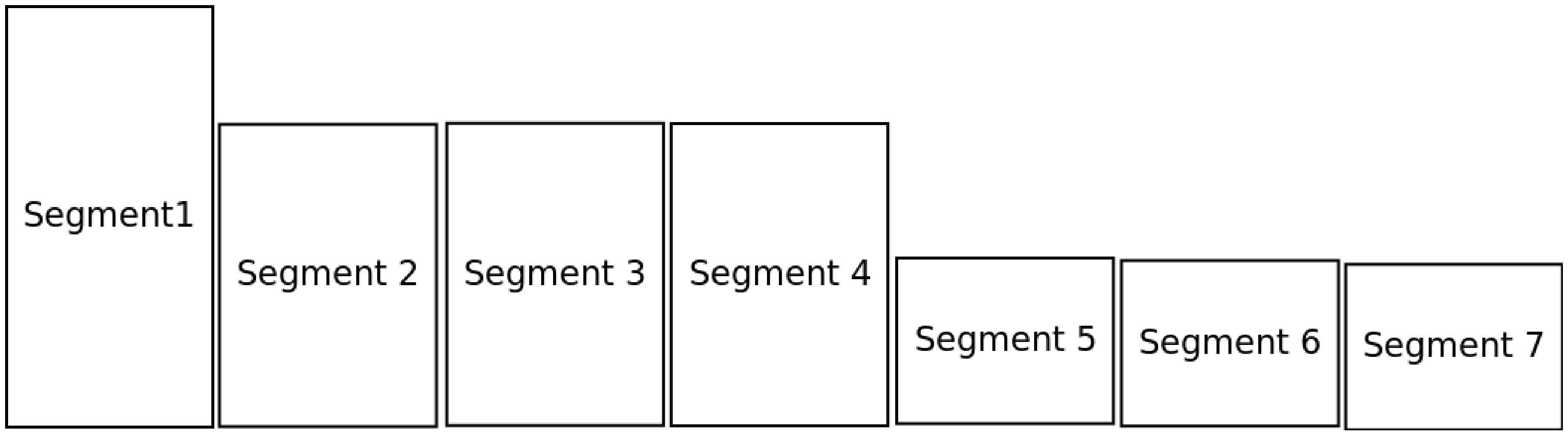
Term	Documents
a	1, 2
an	3
cat	1
dog	2
has	1, 2, 3
kate	2
mike	3
owl	3
tom	1

Lucene segment

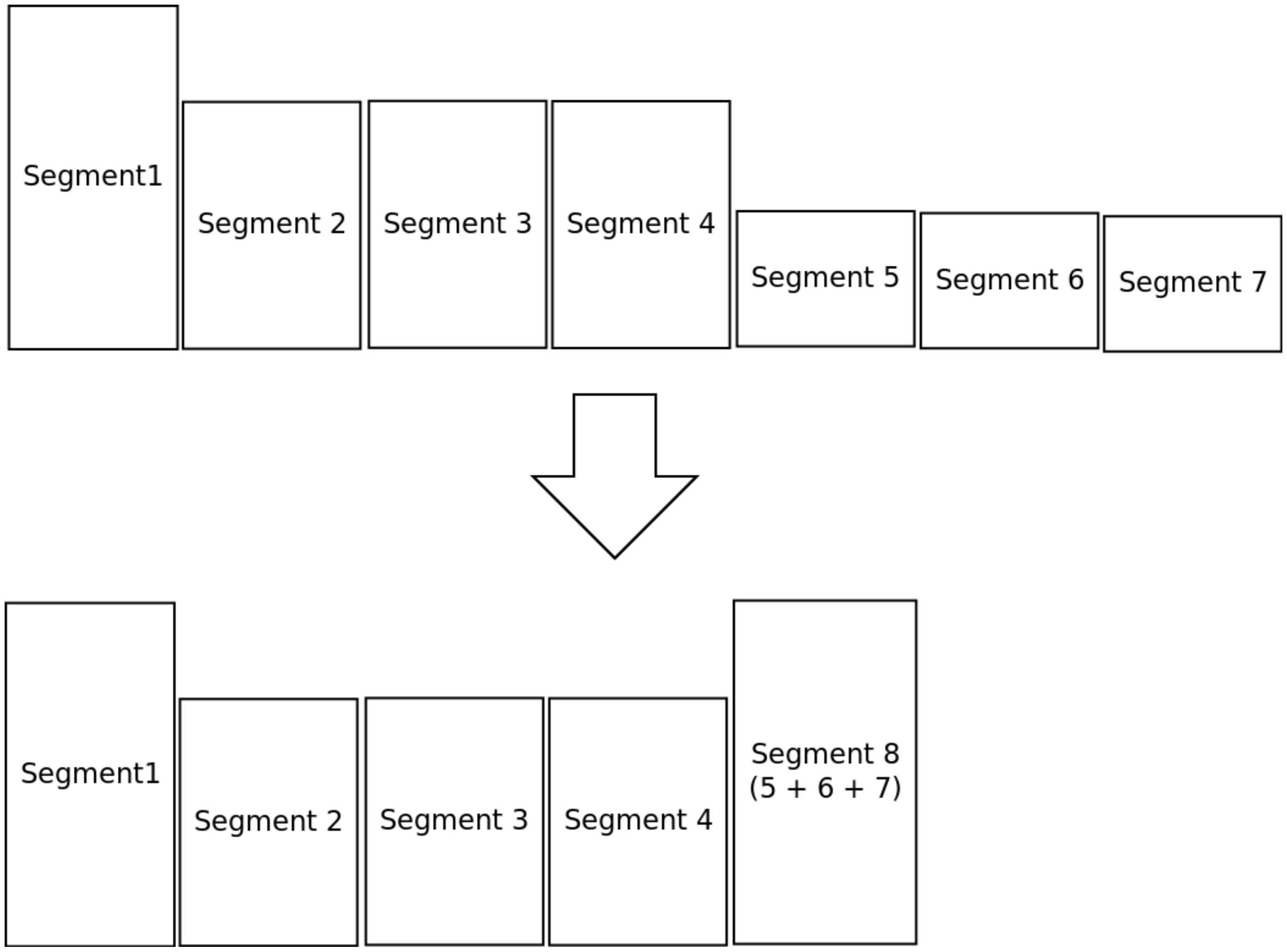
immutable, stored on disk data structure constisting of:

- inverted index
- fielddata cache / doc_values
- _source
- live documents bitset (mutable)

Lucene segments



Segment merging

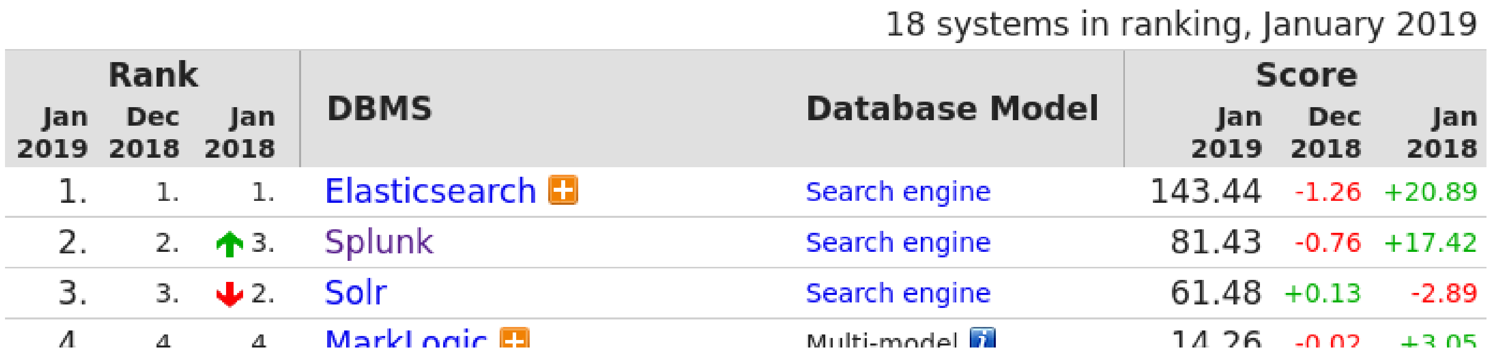


Elasticsearch

- full text search engine built on top of lucene
- log analytics
- horizontally scalable writes
- open source (Apache license)
- v 0.4 released in 2010
- v 1.0 released in 2014
- RESTful api
- resilence
- some features are paid

Apache Solr

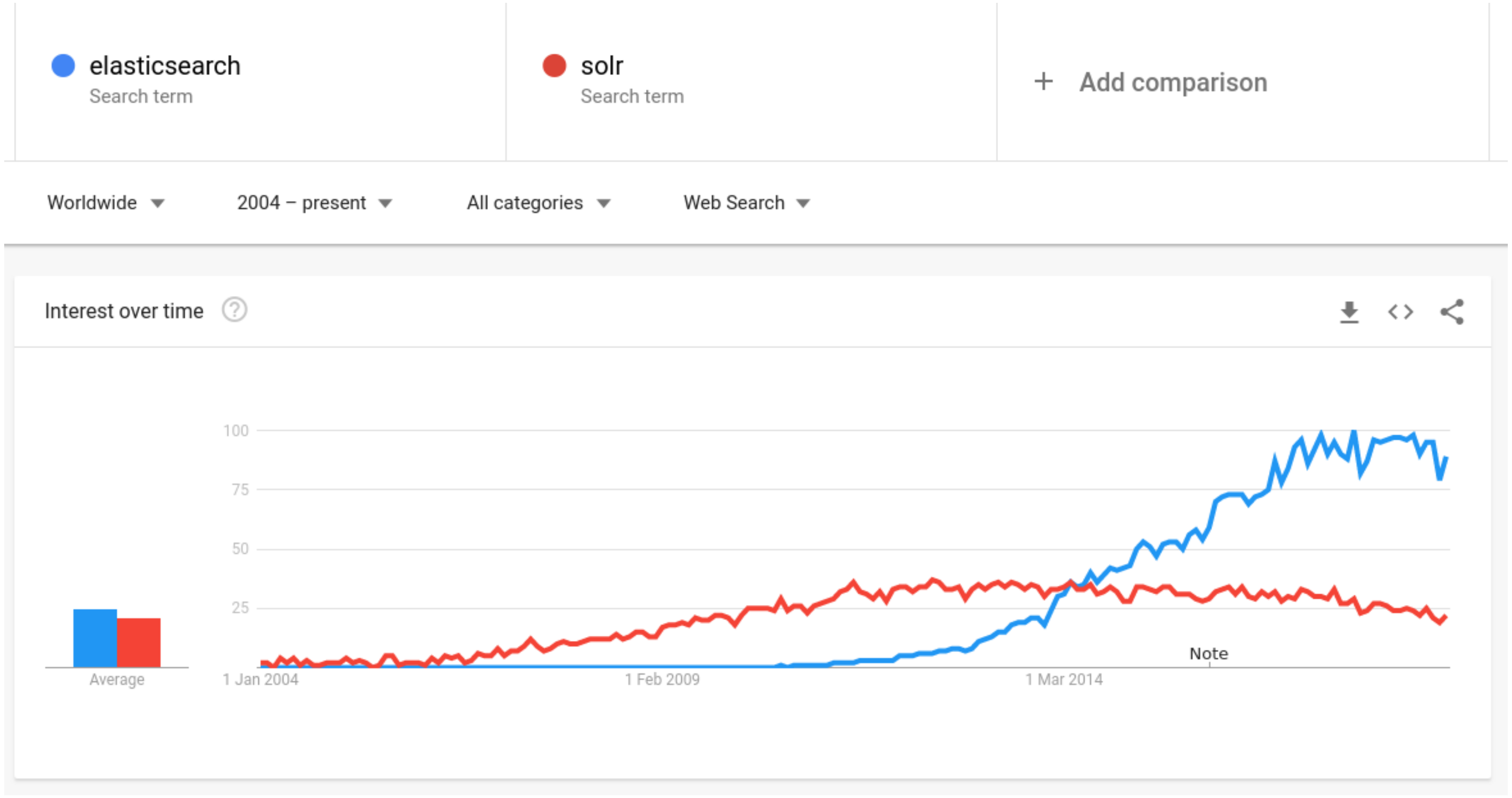
- in 2010 Elasticsearch's competetive project apache Solr joined lucene as a subproject
- features are similar
- elasticsearch is more popular



18 systems in ranking, January 2019

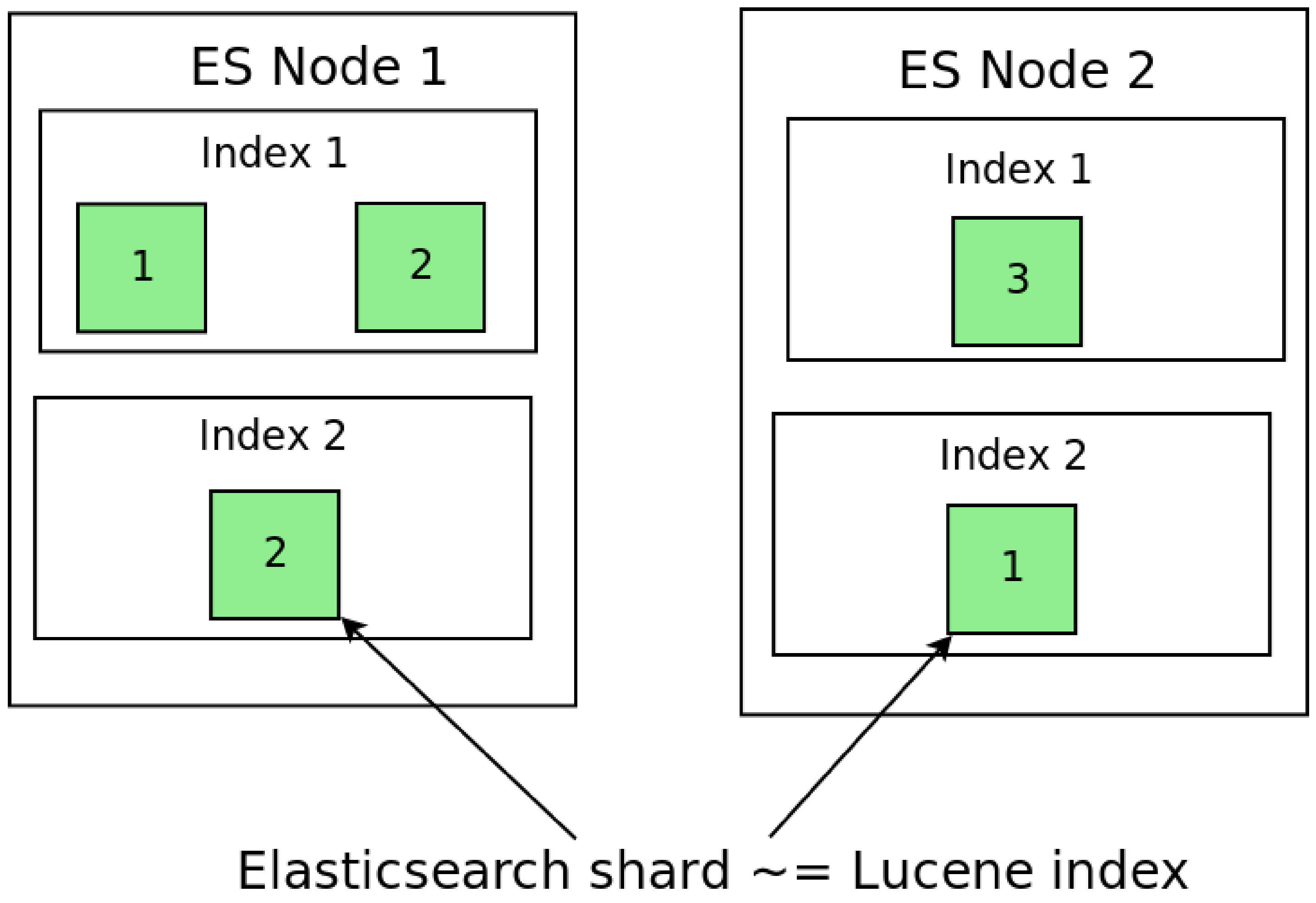
Rank			DBMS	Database Model	Score		
Jan 2019	Dec 2018	Jan 2018			Jan 2019	Dec 2018	Jan 2018
1.	1.	1.	Elasticsearch	Search engine	143.44	-1.26	+20.89
2.	2.	↑ 3.	Splunk	Search engine	81.43	-0.76	+17.42
3.	3.	↓ 2.	Solr	Search engine	61.48	+0.13	-2.89
4.	4.	4.	MarkLogic	Multi-model	14.26	-0.02	+3.05

Apache Solr

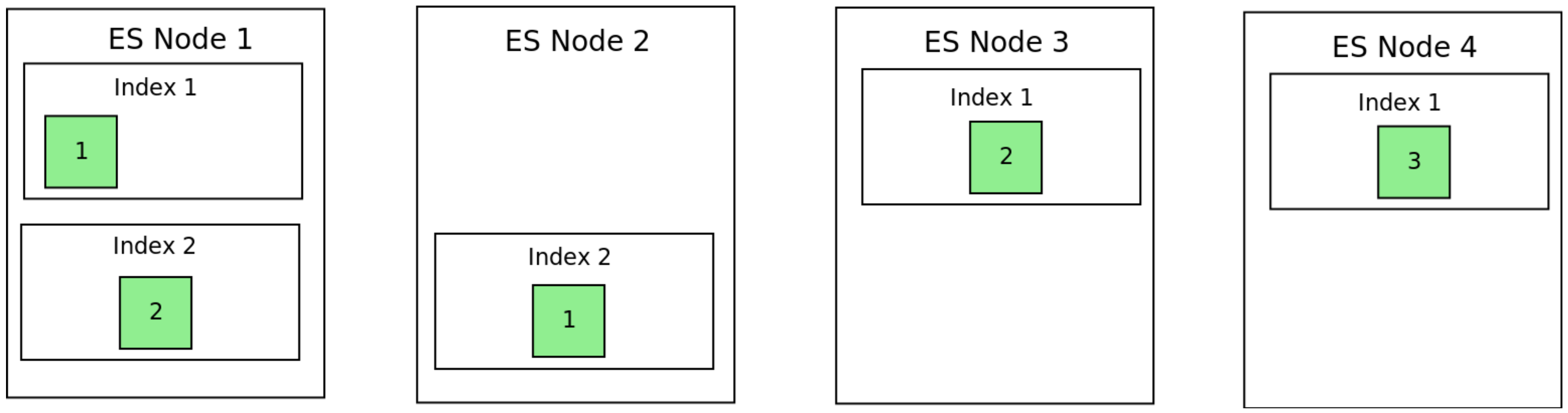


● elasticsearch
Search term
● solr
Search term
+ Add comparison
Worldwide
2004 – present
All categories
Web Search
Interest over time
100
75
50
25
Average
1 Jan 2004
1 Feb 2009
1 Mar 2014
Note

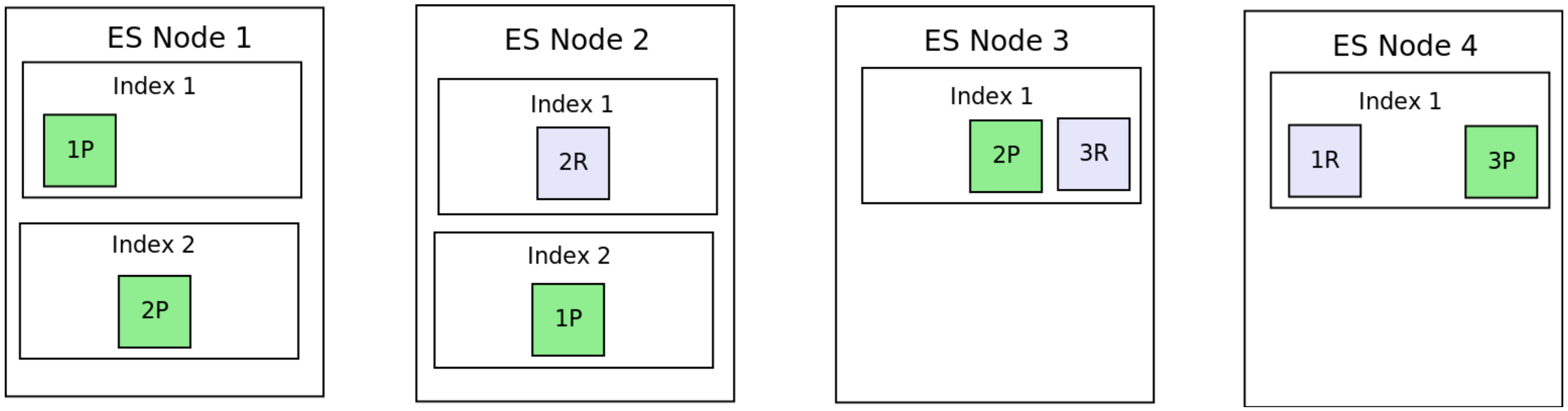
Elasticsearch index



Elasticsearch index



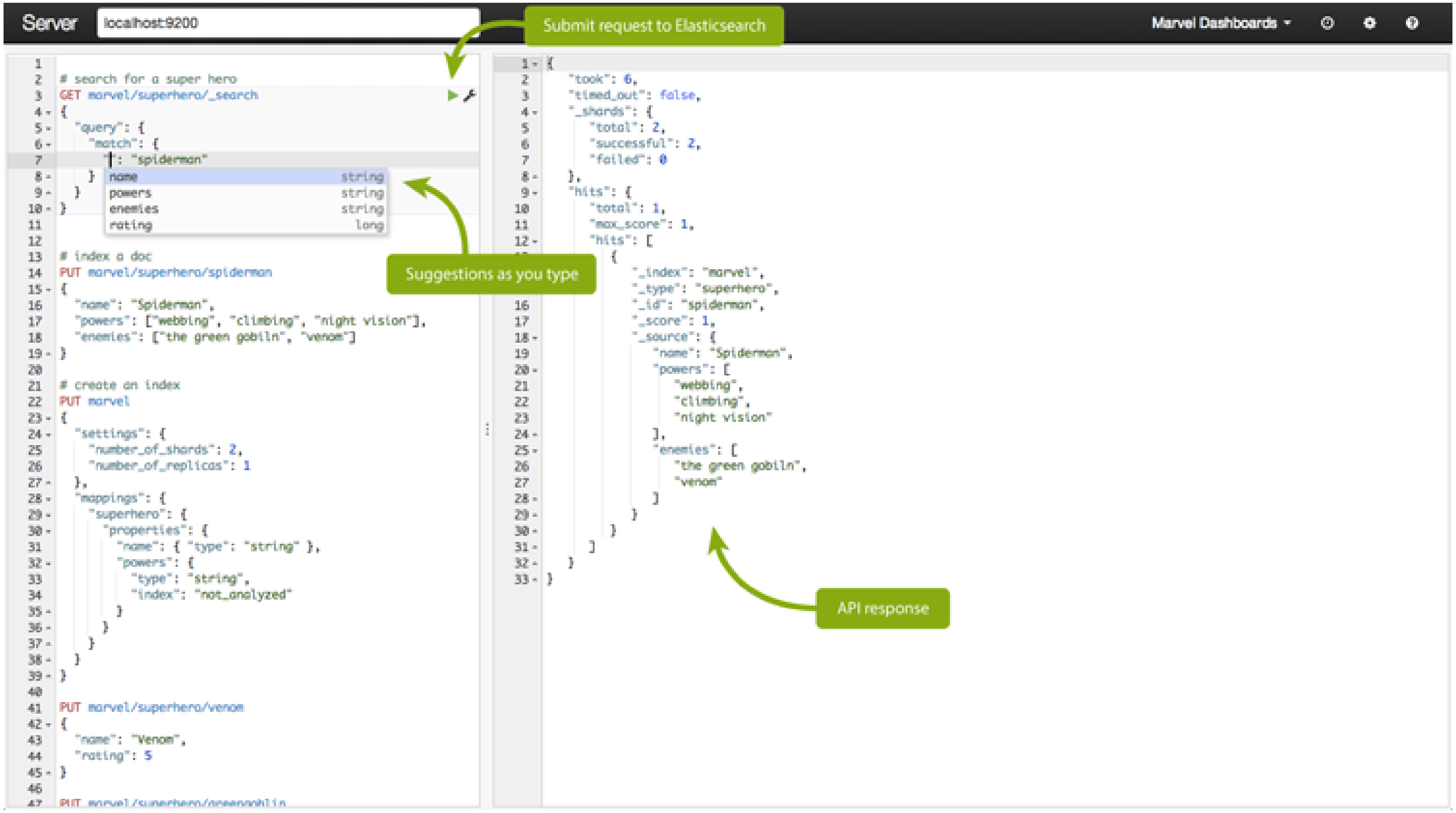
Elasticsearch index



Kibana

- dev tools
- admin interface
- visualizations
- clickable dashboards
- managing elasticsearch

Kibana



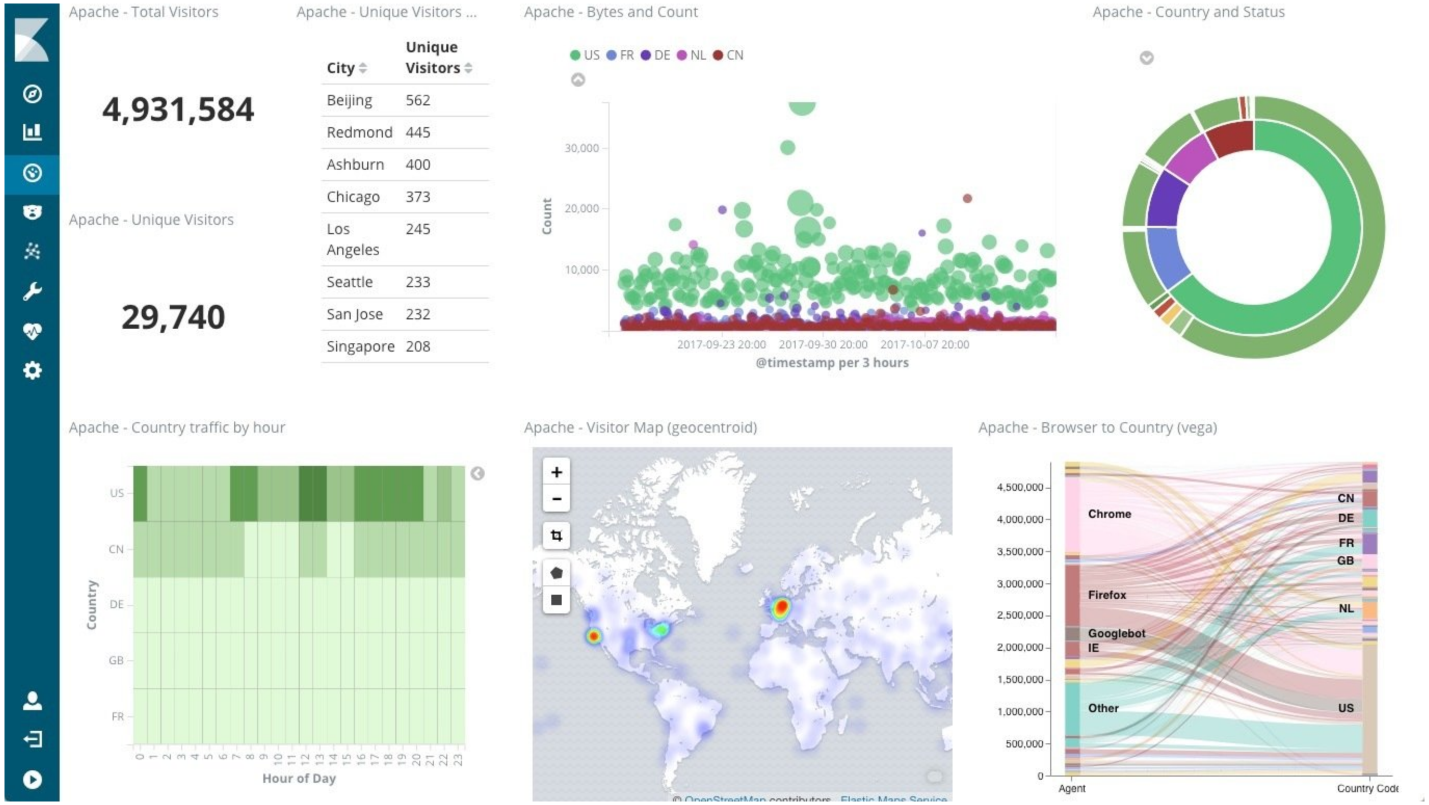
```
# search for a super hero
GET marvel/superhero/_search
{
  "query": {
    "match": {
      "": "spiderman"
    }
  }
}
```

```
name        string
powers      string
enemies     string
rating      long
```

```
# index a doc
PUT marvel/superhero/spiderman
{
  "name": "Spiderman",
  "powers": ["webbing", "climbing", "night vision"],
  "enemies": ["the green gobiln", "venom"]
}

# create an index
PUT marvel
{
  "settings": {
    "number_of_shards": 2,
    "number_of_replicas": 1
  },
  "mappings": {
    "superhero": {
      "properties": {
        "name": { "type": "string" },
        "powers": {
          "type": "string",
          "index": "not_analyzed"
        }
      }
    }
  }
}

PUT marvel/superhero/venom
{
  "name": "Venom",
  "rating": 5
}

PUT marvel/superhero/greengoblin
```

```
{
  "took": 6,
  "timed_out": false,
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 1,
    "hits": [
      {
        "_index": "marvel",
        "_type": "superhero",
        "_id": "spiderman",
        "_score": 1,
        "_source": {
          "name": "Spiderman",
          "powers": [
            "webbing",
            "climbing",
            "night vision"
          ],
          "enemies": [
            "the green gobiln",
            "venom"
          ]
        }
      }
    ]
  }
}
```

Kibana


Apache - Total Visitors

4,931,584

Apache - Unique Visitors

29,740

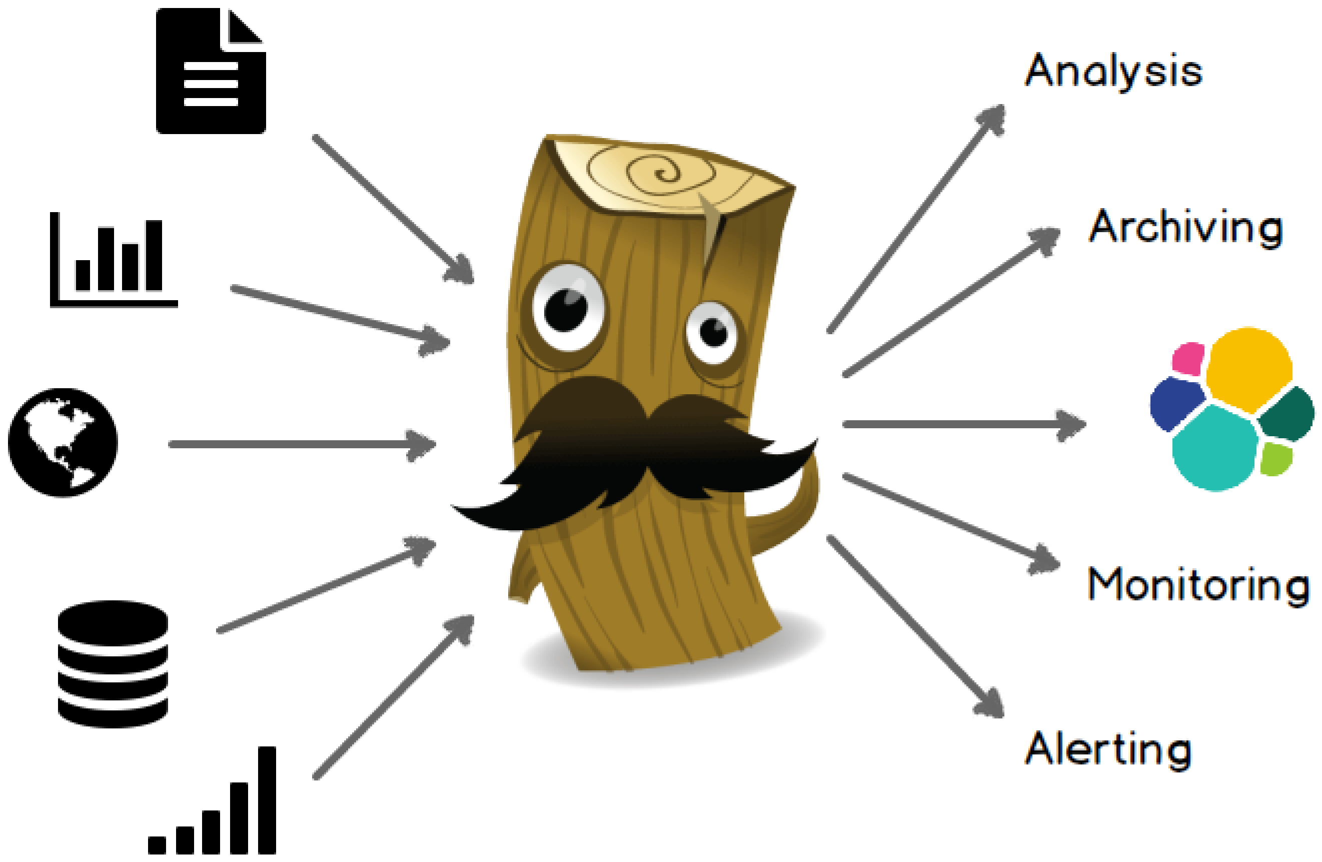
Apache - Unique Visitors ...

City	Unique Visitors
Beijing	562
Redmond	445
Ashburn	400
Chicago	373
Los Angeles	245
Seattle	233
San Jose	232
Singapore	208

Logstash

- Log forwarder
- many inputs (tcp, file, kafka)
- filters for modifying / enriching the data
- outputs to elasticsearch and some other

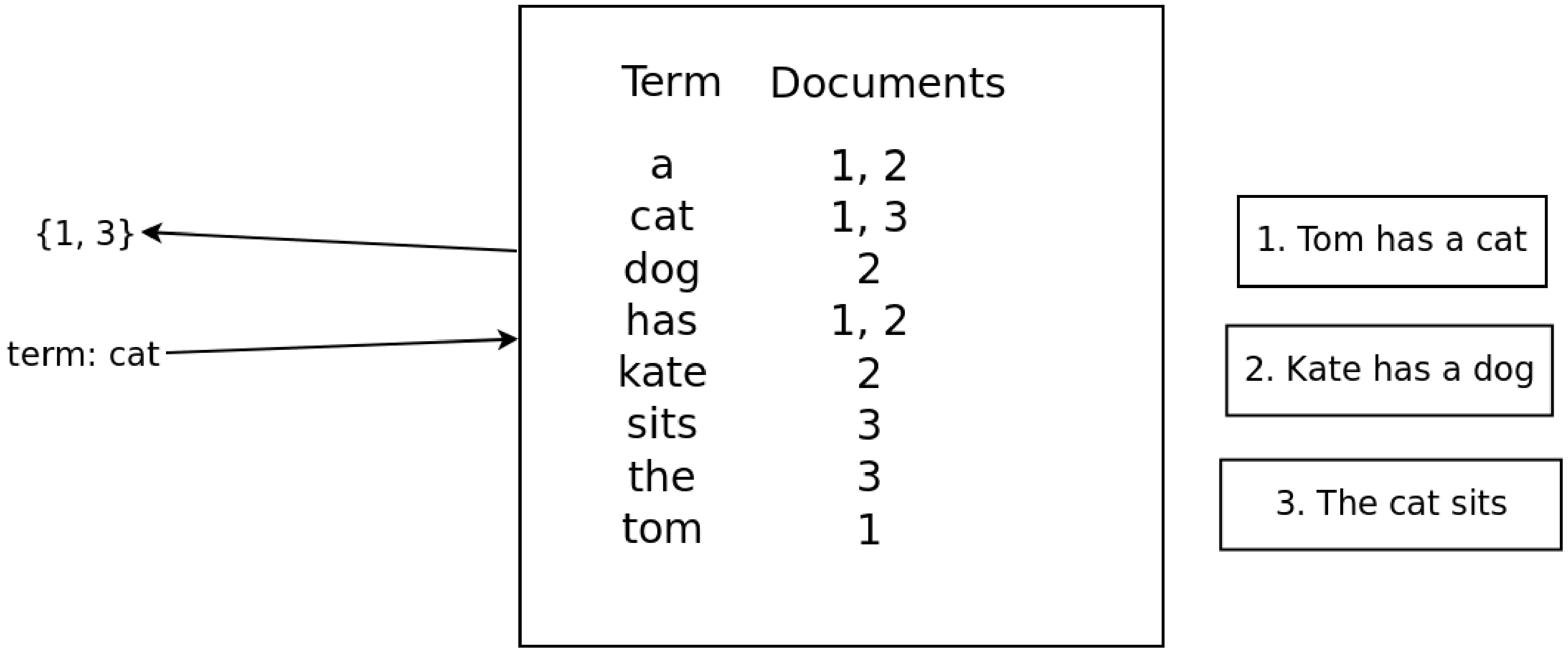
Logstash



Query types

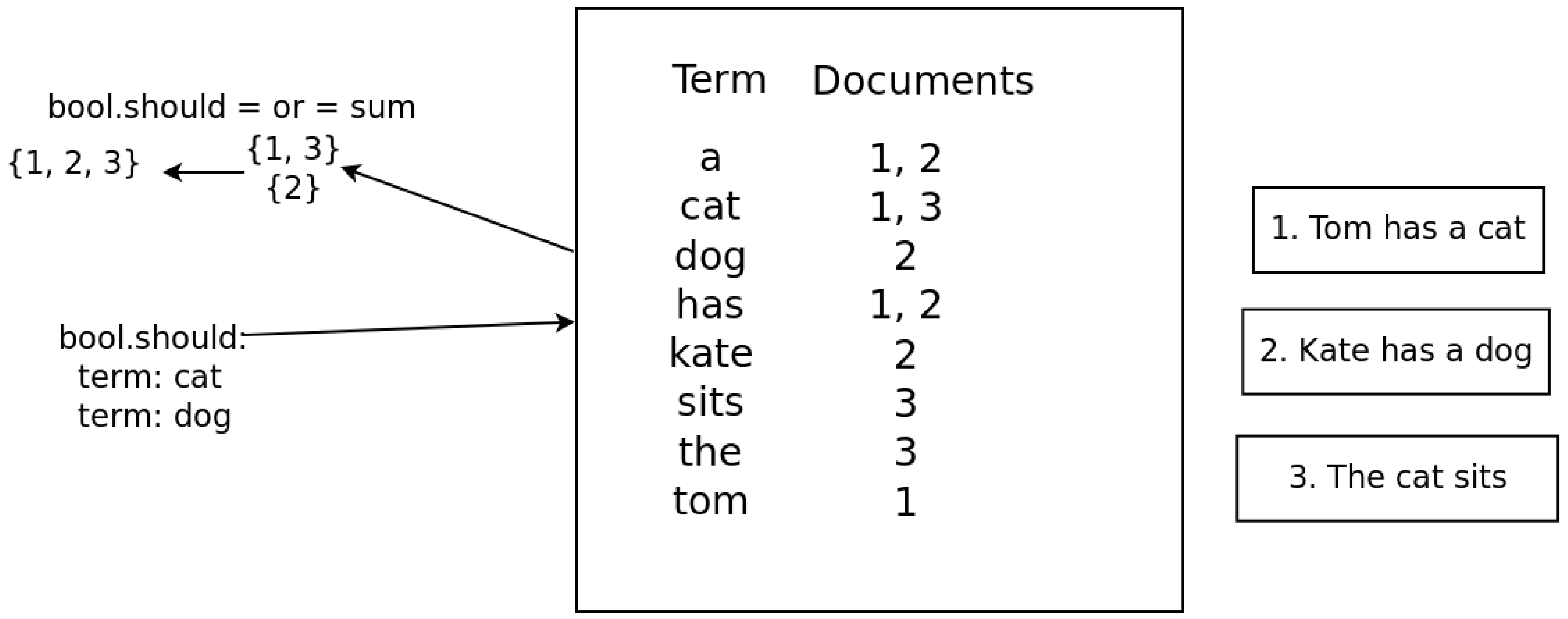
- term
- bool
- prefix
- wildcard
- fuzzy
- match, match_all
- query_string

Term query



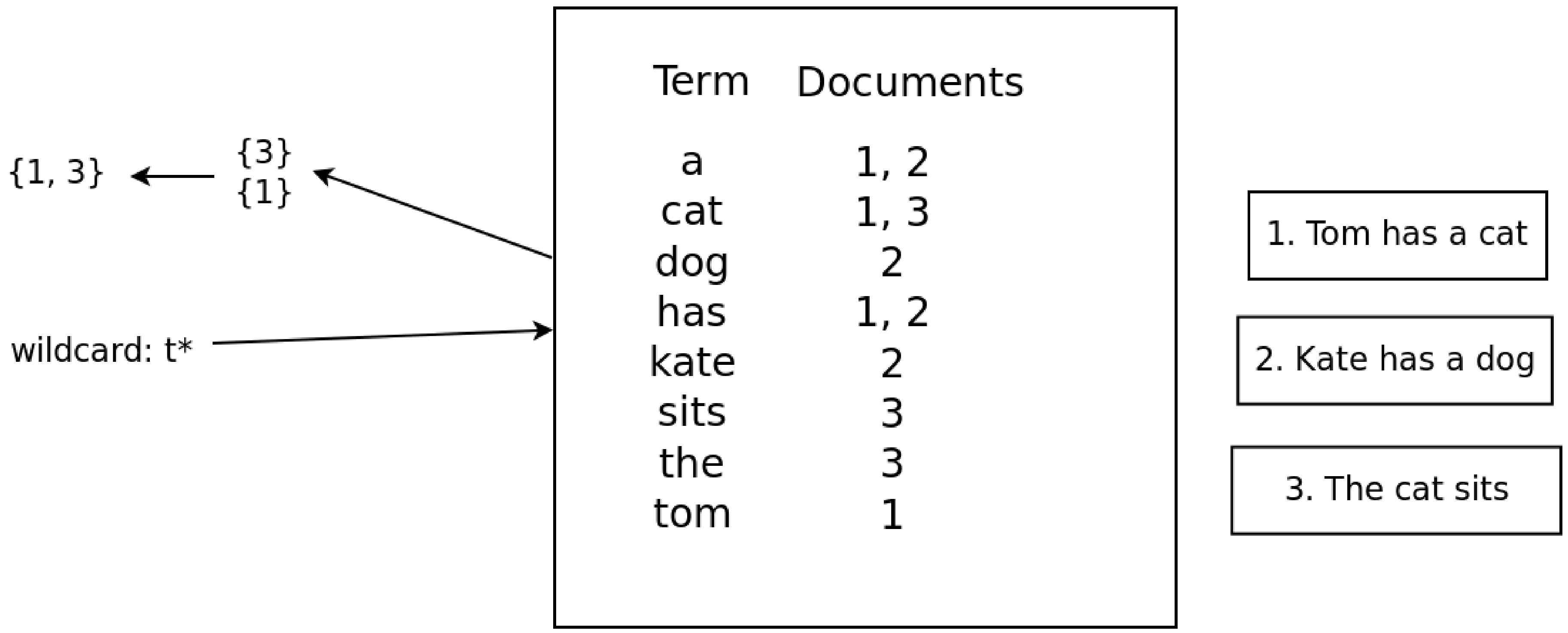
Term	Documents
a	1, 2
cat	1, 3
dog	2
has	1, 2
kate	2
sits	3
the	3
tom	1

Bool query



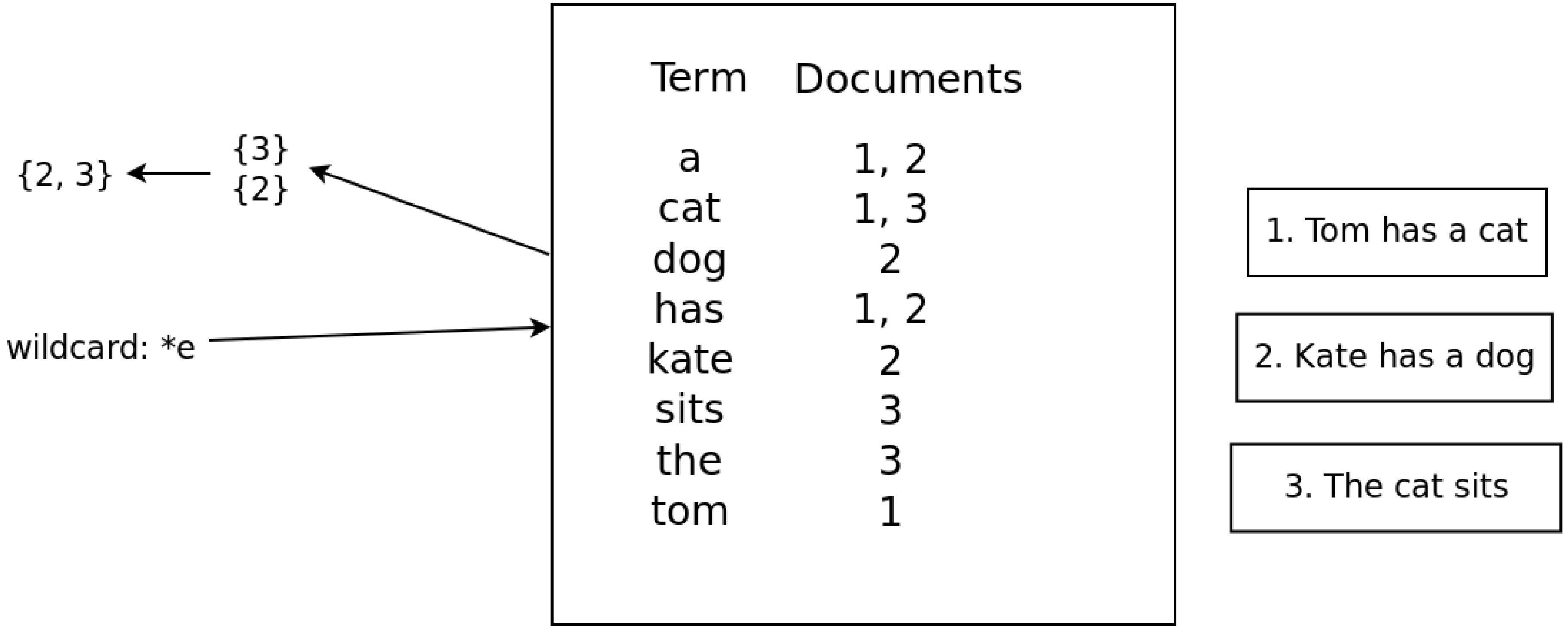
Term	Documents
a	1, 2
cat	1, 3
dog	2
has	1, 2
kate	2
sits	3
the	3
tom	1

Wildcard query



Term	Documents
a	1, 2
cat	1, 3
dog	2
has	1, 2
kate	2
sits	3
the	3
tom	1

Wildcard query

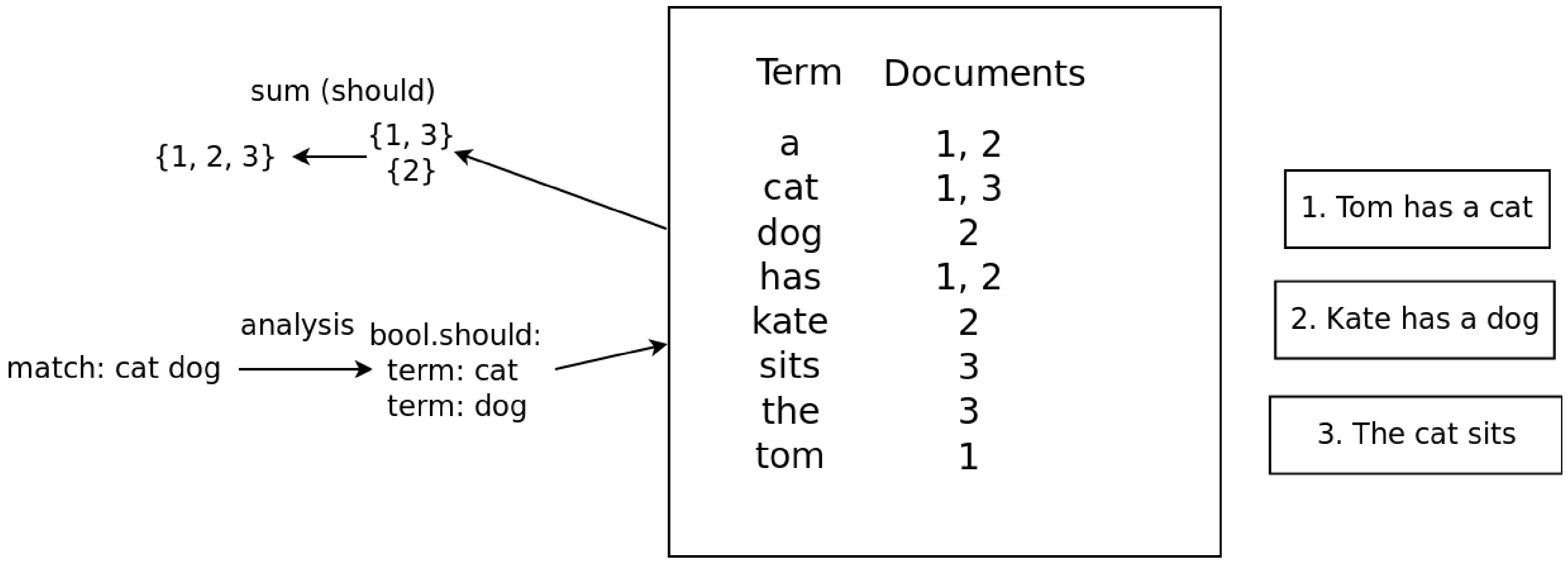


Term	Documents
a	1, 2
cat	1, 3
dog	2
has	1, 2
kate	2
sits	3
the	3
tom	1

Fuzzy query

- Levenshtein distance
- fuzziness value: 1, 2, AUTO

Match query



Term	Documents
a	1, 2
cat	1, 3
dog	2
has	1, 2
kate	2
sits	3
the	3
tom	1

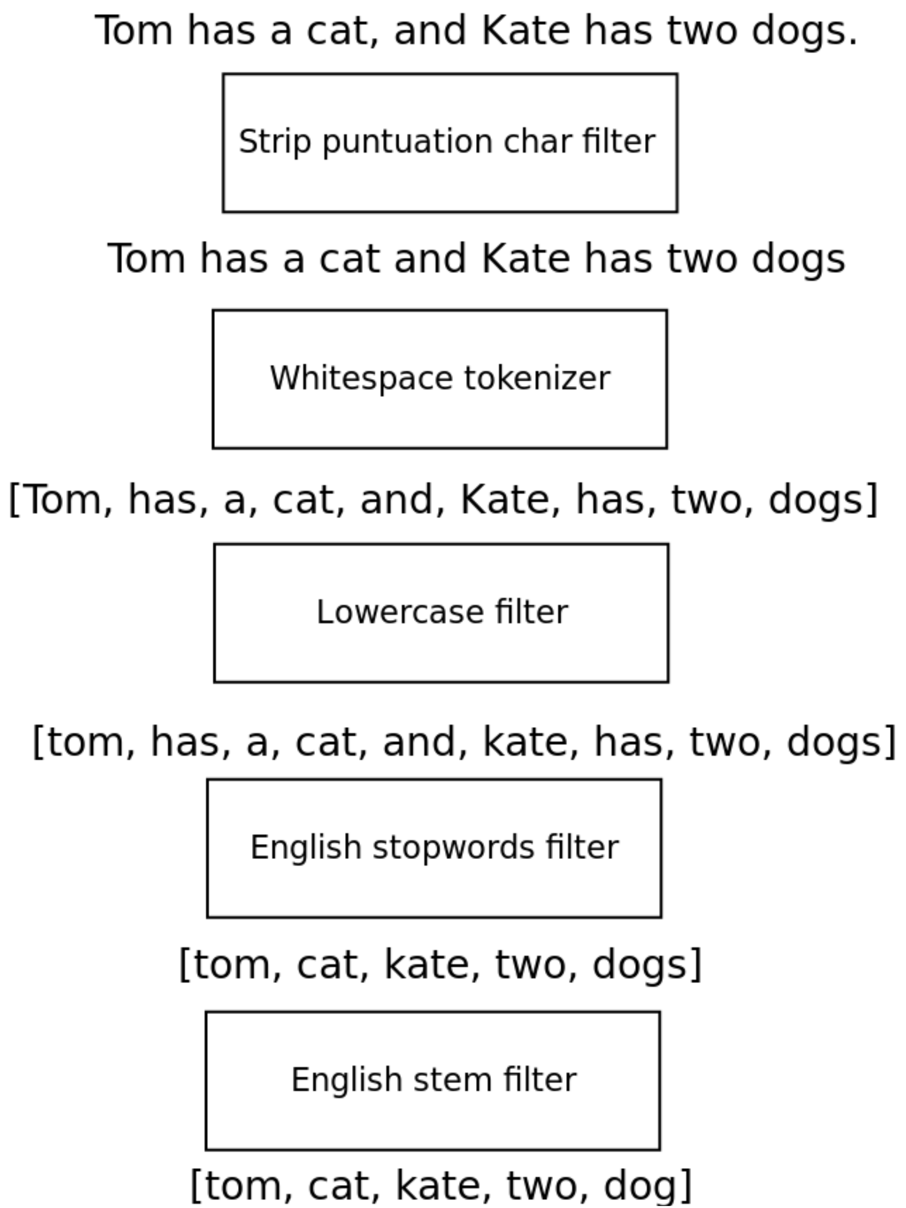
Analyzers

- whitespace
- standard - default analyzer for text fields
- keyword - noop

Custom analyzers

- char_filters - removes / changes / adds characters before text goes to tokenizer
- tokenizer - splits text into tokens
- filters - removes / changes / adds tokens

Custom analyzer


Tom has a cat, and Kate has two dogs.
Strip puntuation char filter
Tom has a cat and Kate has two dogs
Whitespace tokenizer
[Tom, has, a, cat, and, Kate, has, two, dogs]
Lowercase filter
[tom, has, a, cat, and, kate, has, two, dogs]
English stopwords filter
[tom, cat, kate, two, dogs]
English stem filter
[tom, cat, kate, two, dog]

Scoring

$$score(q,d) =$$

$$queryNorm(q) * coord(q,d) *$$

$$\sum (tf(t\ in\ d) * idf(t)^2 * t.getBoost() * norm(t,d))(t\ in\ q)$$

Scoring

- score(q,d) - the relevance score of document d for query q
- queryNorm(q) - query normalization factor
- coord(q,d) is the coordination factor (rewards for higher percentage of query terms contained in document)
- tf(t in d) - term frequency for term t in document d,
- idf(t) - inverse document frequency for term
- t.getBoost() - boost that has been applied to the query,
- norm(t,d) - field-length norm, combined with the index-time field-level boost

Aggregations

- aggregate results into buckets
- count metrics for each of buckets
- can be nested
- fielddata vs doc_values

Kibana dashboards

Nested documents

- can be used to model one to many relations
- each nested document is treated as separate document

Routing

- possibility to split documents into shards basing on a value
- documents with the same routing value will be stored in the same shard
- documents with different routing value might or might not be stored in different shards
- in low cardinality fields can lead to putting all documents into single shard.

Parent child

- another way to store one to many relations
- search performance is significantly lower

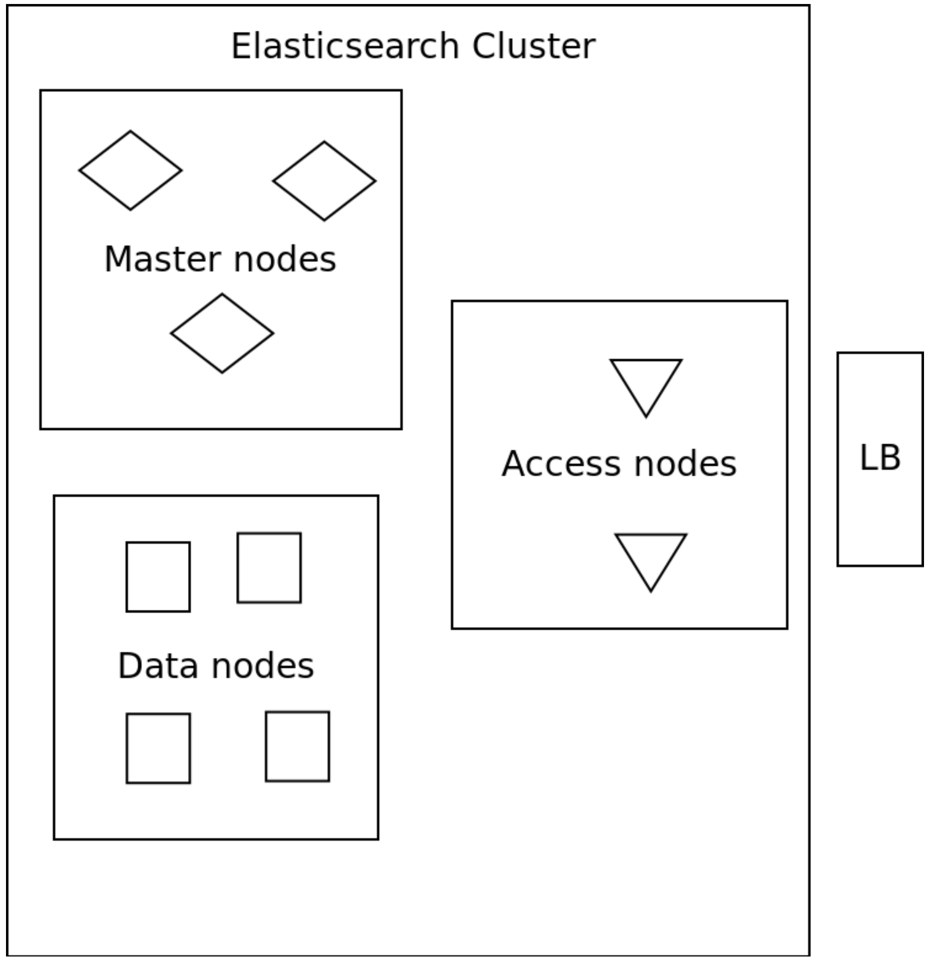
Cluster

- Master node - keeping, updating, broadcasting state of the cluster (list of nodes, mappings)
- Data node - storing data and executing searches on shard
- Access node - forwarding requests to data nodes and merging results
- Ingest node - preprocess documents before indexing them

Leader election

- zen discovery
- quorum

Cluster


Elasticsearch Cluster
Master nodes
Access nodes
LB
Data nodes

_cat api

- indices
- shards
- nodes
- master
- ?v - legend

Premium features

- security (alternative search guard)
- machine learning
- alerting
- graph exploration api