

ElasticSearch Workshop

Everything you need to know to get started



Plan zajęć

1. Setup
2. Podstawowa praca z indeksami i dokumentami.
3. Bulk API oraz mapping.
4. Indeksowanie, wyliczanie score (TF/IDFS).
(trochę teorii)
5. Wyszukiwania.
6. Przypadki użycia.
7. Agregacje i statystyka.
8. Narzędzia (analizatory, explain)

Setup

<https://github.com/szyku/elasticsearchIntroWorkshop.git>

`git@github.com:szyku/elasticsearchIntroWorkshop.git`

`docker-compose up -d`

Setup

*.postman_collection.json – Gotowe instrukcje wykorzystane w workshope.

./exampleData/*.bulk – JSON z danymi używane w workshope.

./exampleQueries/*.query – Nomen omen.

./elasticsearch/data – Dane od ES.

./elasticsearch/config/elasticsearch.yml – Główny config.

./elasticsearch/config/scripts/* – Nomen omen.

Podstawowa praca z indeksami i dokumentami

1. Lazy Indexing
2. Przeglądanie mappingu
3. Lista indeksów
4. Aktualizowanie dokumentu
5. Atomowe aktualizowanie dokumentu
6. Wersjonowanie w Elasticsearch
7. Zadanko

Stwórz parę własnych ‘leniwych’ dokumentów.

Jak elastic mapuje poszczególne pola?

Czy wspiera zagnieżdżone obiekty? Obiekty w tablicach?

Czy po wprowadzeniu pierwszego dokumentu, mapping może się zmienić?

Bulk API

Bulk API

1. Wprowadzenie do Bulk API
2. Przeglądanie przykładowego dokumentu
3. Zadanko

Stwórz własny plik bulk. (3 rekordy stykną)

Jak zachowa się ES, kiedy mu NIE podamy endpoint (`{indeks}/{typ}`) przy `geoCache.bulk` i `geoCacheDivided.bulk`?

Jak się zachowa, kiedy podamy endpoint przy `geoCacheDivide.bulk`? Do jakiego typu zostaną przypisane dokumenty?

Czy można łączyć operacje?

Dlaczego zostało wprowadzonych 9 a nie 10 dokumentów?

Mapping

Na podstawie mappingu z geoCache.bulk.

Dodaj nowe pole (wymyśl se panie) i zaktualizuj wybrany dokument.
Czy natrafiłeś na jakiś problem?

Popraw mapping geoPoint oraz created.
Czy natrafiłeś na jakiś problem?

Jak zachowuje się ES, kiedy wprowadzamy dane o niepoprawnym formacie co do typu? Siedzi cicho czy protestuje, a może magia?

Popraw “Created” aby było typu date oraz format “YYYY-MM”. Załaduj ponownie dane geoCache.bulk.

Indeksowanie, TF/IDFS,
Zapytania, Paginacja

Indeksowanie

(Kolejność nie jest zachowana)

- Budowanie odwróconego indeksu ([Inverted Index](#))

Zawiera informacje:

- Gdzie występują klucze (w jakich dokumentach?)
- W jakiej ilości
- Dokument poddany jest tokenizacji i uproszczeniu słów kluczowych oraz filtrowaniu kluczy
- Generowane są meta informacje (np GeoHash, timestamp, auto-id itd.)
- Pola są przetwarzane (skracać precyzję, dokładanie pól wykonanych z skryptu)
- Dokument jest przydzielany do konkretnego shard, indexu, node...

Korzystając z Analize API, zbadaj simple, whitespace, stop, snowball wpisując teksty po angielsku.

Dla snowball korzystaj z czasowników w różnej formie (gerund, p. participle)

Co zauważyłeś? Co i jak zostało zamienione w token?

Inverted Index

[Image source](#)

Inverted Index Example

ID	Text	Term	Freq	Document ids
1	Baseball is played during summer months.	baseball	1	[1]
		during	1	[1]
2	Summer is the time for picnics here.	found	1	[3]
3	Months later we found out why.	here	2	[2], [4]
4	Why is summer so hot here	hot	1	[4]
↑	Sample document data	is	3	[1], [2], [4]
		months	2	[1], [3]
		summer	3	[1], [2], [4]
		the	1	[2]
		why	2	[3], [4]

Dictionary and posting lists →

Lucene TF/IDF(Similarity)

- Obliczenia na podstawie Inverted Index
- Rzadkie występowanie słowa – większy score dla dokumentu posiadającego dane słowo.
- Im mniejszy dokument, tym większy score za posiadanie słów.
- Nasze “boosty”.

[Szczegóły implementacji na stronie Apache Lucene](#) ← Dla zainteresowanych

[Algorytm TF/IDF](#) ← Dla turbo ciekawskich (generalnie statystyka)

Zapytania

Rodzaje:

1. Warunek (Term)
2. Pełnotekstowe (Full text)

Zapytania

Rodzaje:

1. Warunek (Term)



np. term, fuzzy

Nie poddawane analizie, bezpośrednie wyszukanie w Inverted Index.

Nie istotne jak się znalazło w II, ważne, że jest i tego będzie szukać.

Nisko poziomowe zapytanie.

2. Pełnotekstowe (Full text)



np. match, common words,
query_search

Respektuje mapowanie dokumentu.

Przetwarza skomplikowane zapytanie (np GeoBound) i przepisuje (query rewrite) do optymalnej postaci niskopoziomowej.

Pozwala na więcej :)

Search API *lite*

`/textsearch/shortdoc/_search?q="equal"&pretty`

(metody GET i POST)

Rodzaje Search API

Search API *lite*


/textsearch/shortdoc/_search?q="equal"&pretty

(metody GET i POST)

Full Search API

Występuje Payload (zapytanie POST)

```
{
  "common": {
    "body": {
      "query": "nelly the dog",
      "cutoff_frequency": 0.1,
      "low_freq_operator": "or"
    }
  }
}
```



Query Rewrite (common words query)

```
{  
  "common": {  
    "body": {  
      "query": "nelly the elephant as a cartoon",  
      "cutoff_frequency": 0.001,  
      "low_freq_operator": "and"  
    }  
  }  
}
```

Query Rewrite (common words query)

```
{
  "common": {
    "body": {
      "query": "nelly the elephant as a cartoon",
      "cutoff_frequency": 0.001,
      "low_freq_operator": "and"
    }
  }
}
```

Query Rewrite



```
{
  "bool": {
    "must": [
      { "term": { "body": "nelly" } },
      { "term": { "body": "elephant" } },
      { "term": { "body": "cartoon" } }
    ],
    "should": [
      { "term": { "body": "the" } }
      { "term": { "body": "as" } }
      { "term": { "body": "a" } }
    ]
  }
}
```

1. Poeksperymentuj z *lekkimi* zapytaniem. Wypróbuj wielkie i małe litery, tRaFcIe notation itp. Można ograniczyć pytanie do poszczególnych pól formatem `q=pole:wartosc AND|OR pole_kolejne:wartosc...`

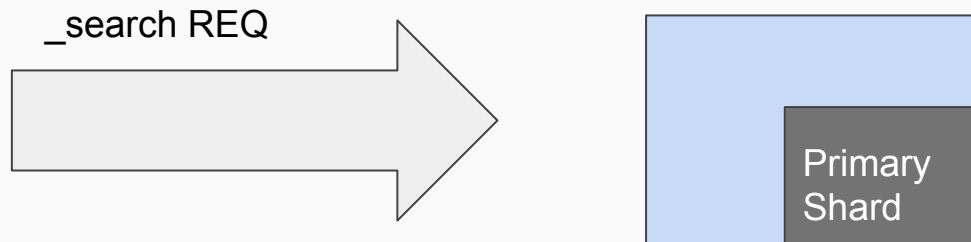
Co zauważyłeś? Czy wyniki są zgodne z oczekiwaniami?

2. Wykorzystując Full Search API (zapytania z payloadem), spróbuj wykonać podobne zapytania z punktu 1 z użyciem filtrowania. Postaraj się używać tych samych form zapisu, co używałeś w zadaniu 1 (przykładowe query w kodzie).

Czy zachowanie jest podobne do wypadkowej z punktu 1?

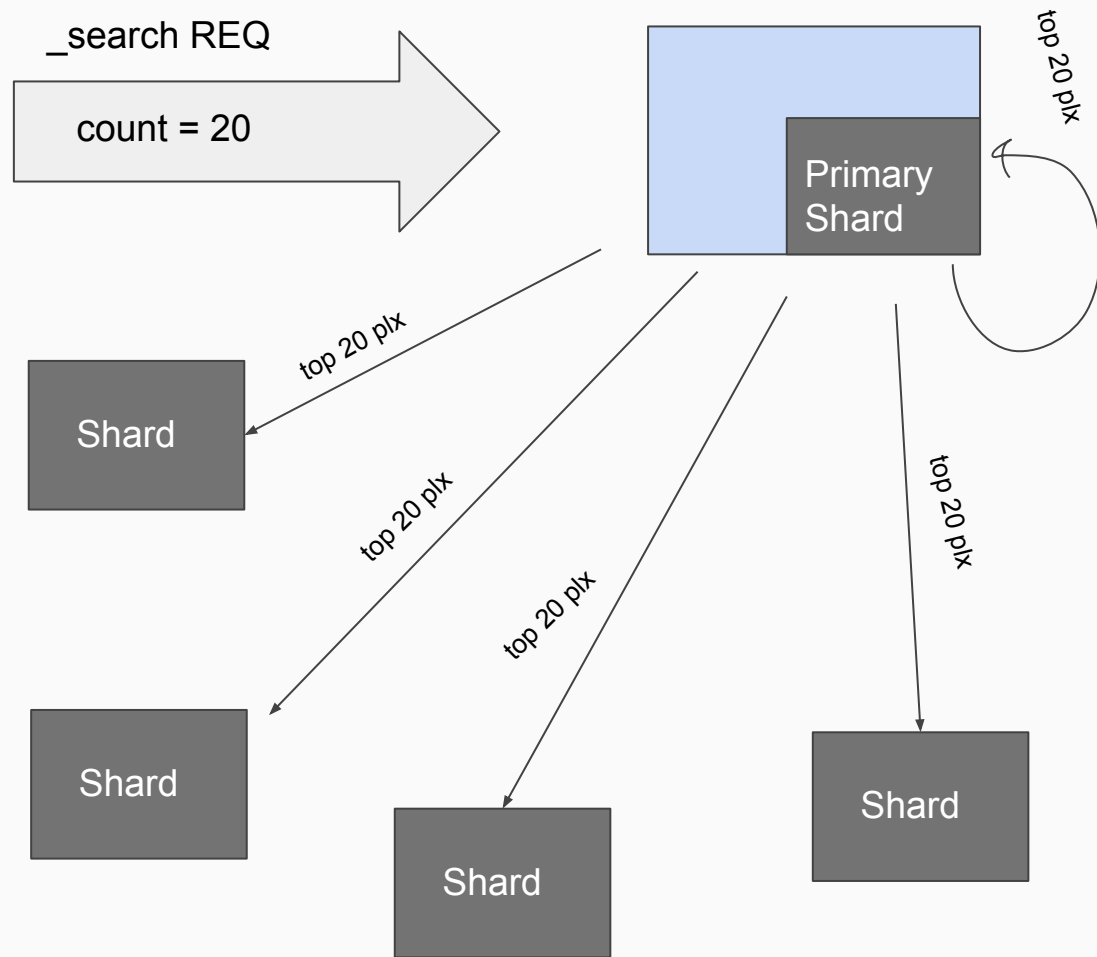
Before Pagination...

... Jak Elasticsearch
wykonuje zapytania?



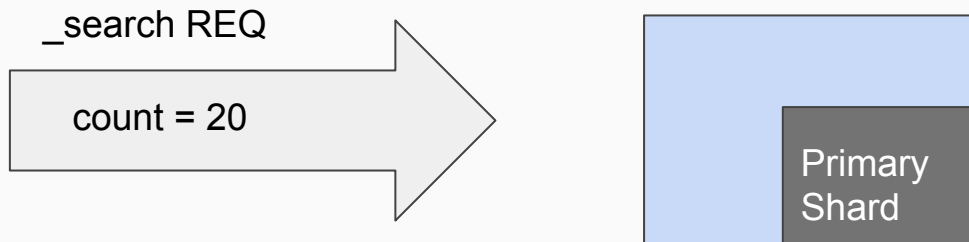
Before Pagination...

... Jak Elasticsearch
wykonuje zapytania?



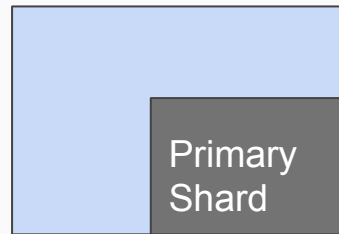
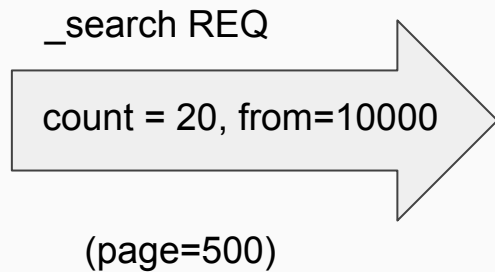
Before Pagination...

... Jak Elasticsearch
wykonuje zapytania?

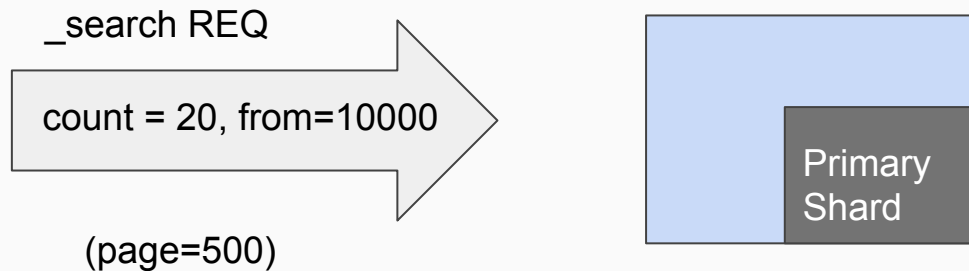


1. Złącz wszystkie dokumenty
2. Oblicz które są najlepsze
3. Zwróć top 20 z tego zestawu
4. Zwolnij zasoby (100 doków w pamięci)

Deep Pagination Problem

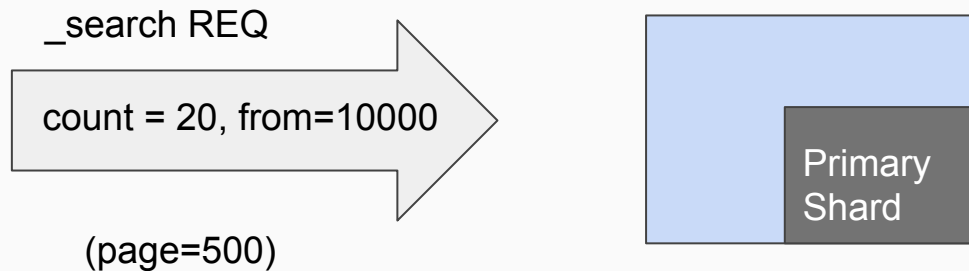


Deep Pagination Problem



To sprawi, że załaduje się $5 \times 10020 = 50100$ dokumentów do pamięci i obliczy co ma obliczyć.

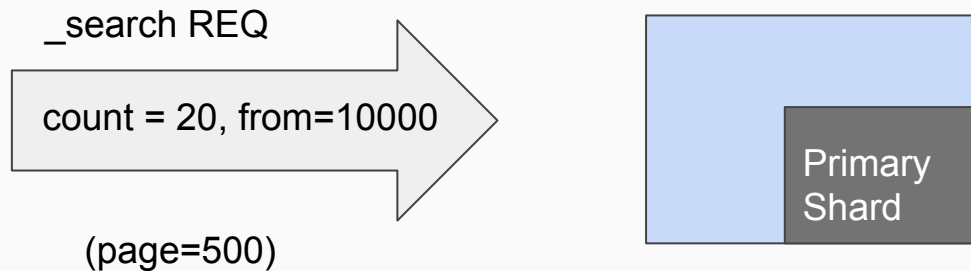
Deep Pagination Problem



To sprawi, że załaduje się $5 \times 10020 = 50100$ dokumentów do pamięci i obliczy co ma obliczyć.

Tylko po to, aby wyciągnąć te 20 dokumentów.

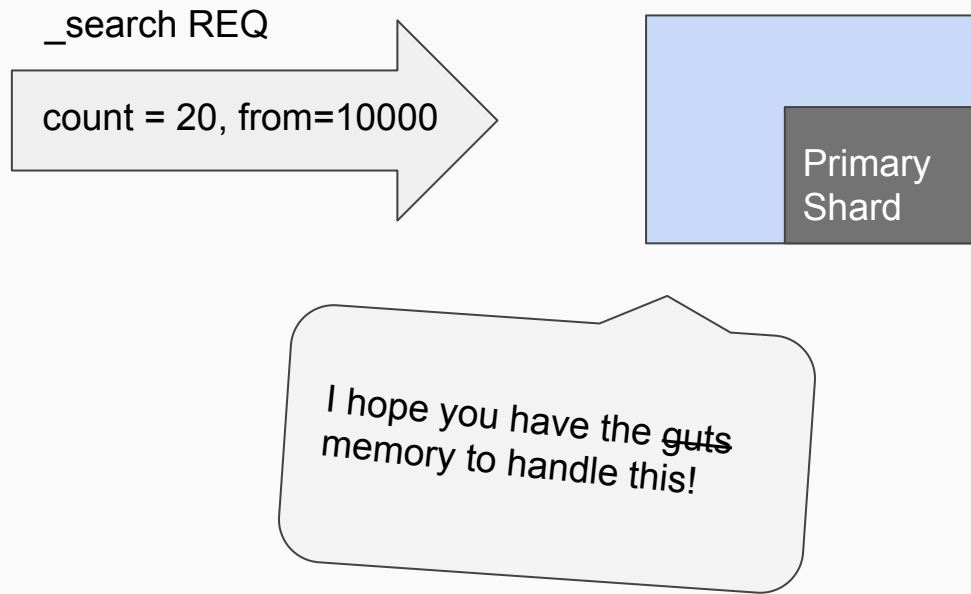
Deep Pagination Problem



To sprawi, że załaduje się $5 \times 10020 = 50100$ dokumentów do pamięci i obliczy co ma obliczyć.

Dlatego ogranicza się możliwość paginacji w wyszukiwarkach.

Deep Pagination Problem



Problem ten da się rozwiązać za pomocą Scan and Scroll, gdzie rezygnuje się z Real Time Search na rzecz obsługi wielkich ilości danych.

Jest to jednak temat na inne spotkanie.

Napisz własny skrypt w groovy i wykorzystaj go przy `_search`.

Use cases

Problem

Wyszukać dokumenty w zadanej odległości w różnych jednostkach metrycznych.

Geo document search – solution

```
"ur_mum": {  
  "type": "geo_point",  
  "fielddata": {  
    "precision": "3m"  
  },  
  "lat_lon": true,  
  "geohash": true  
}
```

Mapping pola: geo_point

```
{  
  "query": {  
    "filtered" : {  
      "query" : {  
        "match_all" : {}  
      },  
      "filter" : {  
        "geo_distance" : {  
          "distance" : "120km",  
          "ur_mum" : {  
            "lat" : 50,  
            "lon" : -50  
          }  
        }  
      }  
    }  
  }  
}
```

Query

Problem

Chcę aby ES podpowiedział co wpisać, ale i jednocześnie zwrócił odpowiednie dane (np ID) do dalszego przetwarzania.

Problem

Chcę aby ES podpowiedział co wpisać, ale i jednocześnie zwrócił odpowiednie dane (np ID) do dalszego przetwarzania.

Do tego służy Suggestion API!

API zapewnia szybki i natychmiastowy dostęp do danych. Sugestie zwracane są praktycznie natychmiastowo.

Mappowanie

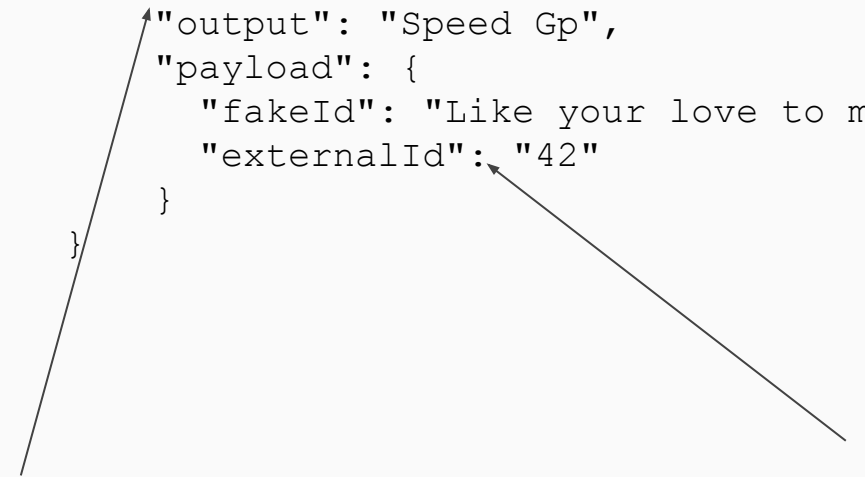
```
PUT /index
{
  "mappings": {
    "product" : {
      "properties" : {
        "name" : { "type" : "string" },
        "brand" : { "type" : "string" },
        "manufacturer" : { "type" : "string" },
        "suggest_part" : {
          "type" : "completion",
          "payloads" : true
        }
      }
    }
  }
}
```



Oczywiście, może być więcej niż jeden suggester.

Dodajmy dokumenty

```
POST /index/product
{
  "name": "Speed Gp",
  "brand": "Home&Garde",
  "manufacturer": "ACME",
  "suggest_part":
  {
    "input": ["Home&Grade", "Speed"],
    "output": "Speed Gp",
    "payload": {
      "fakeId": "Like your love to me",
      "externalId": "42"
    }
  }
}
```



Na co będzie reagować i czym
będzie odpowiadać

To co ta sugestia
będzie zawierała w
odpowiedzi

Suggestion API

```
POST /index/_suggest
{
  "products" : {
    "text" : "ho",
    "completion" : {
      "field" : "suggest_part"
    }
  }
}
```


Suggestion API

```
POST /index/_suggest
{
  "products" : {
    "text" : "ho",
    "completion" : {
      "field" : "suggest_part"
    }
  }
}
```

Dowolna etykieta

Treść z formatki

Zaimplementuj własny autosuggester.

Możesz użyć gotowych indeksów i mappingów albo stworzyć własne.

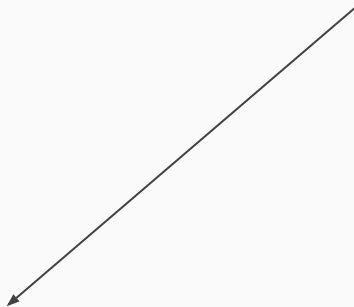
[Link do wyczerpującego artykułu od samej firmy elastic.](#)

Aggregations

Agregacje

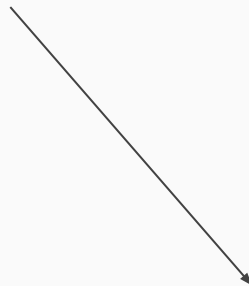
- Pozwalają klasyfikować dokumenty, według pewnych kryteriów (np. pacjenci z pewnym wskaźnikiem hemoglobiny, obiekty w pewnej odległościach od punktu geograficznego)
- Dostarczają metryk z pewnego podzbioru dokumentów o pewnym profilu
- Mogą być cachowanie
- **Mogą być zagnieżdżane** (możemy usystematyzować pacjentów z pewnymi objawami np. przedziały wiekowe)
- Dostarczają statystyki (min, max, odchylene std., percentyle itp.)
- Poddają się skryptowaniu (można tworzyć własne fn agregacji)
- Działają w czasie rzeczywistym

Aggregations



Bucketing

(term, geo distance, ip range, missing value)



Metric

(min, max, stats, value count)

Agregacje

```
"aggregations" : {  
  "<aggregation_name>" : {  
    "<aggregation_type>" : {  
      <aggregation_body>  
    }  
    [, "aggregations" : { [<sub_aggregation>]+ } ]?  
  }  
  [, "<aggregation_name_2>" : { ... } ]*  
}
```

← Agregacja agregacji :)

Na zbiorze danych fighters.

Wykonaj zagnieżdżoną agregację według twojego uznania.

Debugging
Analiza
Profilowanie

Tools

- Analyze API
- Validation API
- Explain API
- Indices API
- cat [sic!] API

- Analyze API



To już znamy

- Validation API
- Explain API
- Indices API
- cat [sic!] API

- Analyze API
- Validation API
- Explain API
- Indices API
- cat [sic!] API



Walidacja oraz tłumaczenie
ciężkich obliczeniowo
zapytań bez ich wykonywania

- Analyze API
- Validation API
- Explain API
- Indices API
- cat [sic!] API



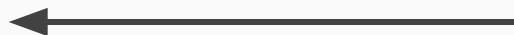
Tłumaczy obliczanie score
oraz czemu lub czemu nie
dany dokument został
wzięty pod uwagę.

- Analyze API
- Validation API
- Explain API
- Indices API
- cat [sic!] API



Zarządzanie indeksami. Info o operacjach jakie zaszły na indeksie, zużycie pamięci, optymalizacja pracy itp.

- Analyze API
- Validation API
- Explain API
- Indices API
- cat [sic!] API



“Czytelne” przedstawienie
informacji.
Sprawdź sam
GET /_cat

- Analyze API
- **Validation API**
- **Explain API**
- Indices API
- **cat [sic!] API**

Validation API

```
GET
/primary/example/_validate/query?q={&explain=true

{
  "valid": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "failed": 0
  },
  "explanations": [
    {
      "index": "primary",
      "valid": false,
      "error": "org.elasticsearch(...)"
    }
  ]
}
```


Explain API

GET

/fighters/dbpun/AVgbeEPvBwk-c0nJxC5r/_explain?q=address:Street

```
{
  (...)
  "matched": true,
  "explanation": {
    "value": 0.375,
    "description": "weight(address:street in 0)
[PerFieldSimilarity], result of:",
    "details": [
      {
        "value": 0.375,
        "description": "fieldWeight in 0, product
of:",
        "details": [ (...)
```

=^.^= API

```
GET /_cat
```

```
=^.^=
```

```
/_cat/allocation
```

```
/_cat/shards
```

```
/_cat/shards/{index}
```

```
/_cat/master
```

```
/_cat/nodes
```

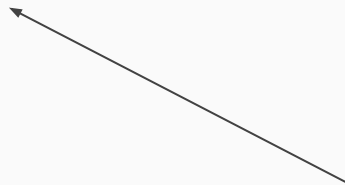
```
/_cat/indices
```

```
/_cat/indices/{index}
```

```
/_cat/segments
```

```
/_cat/segments/{index}
```

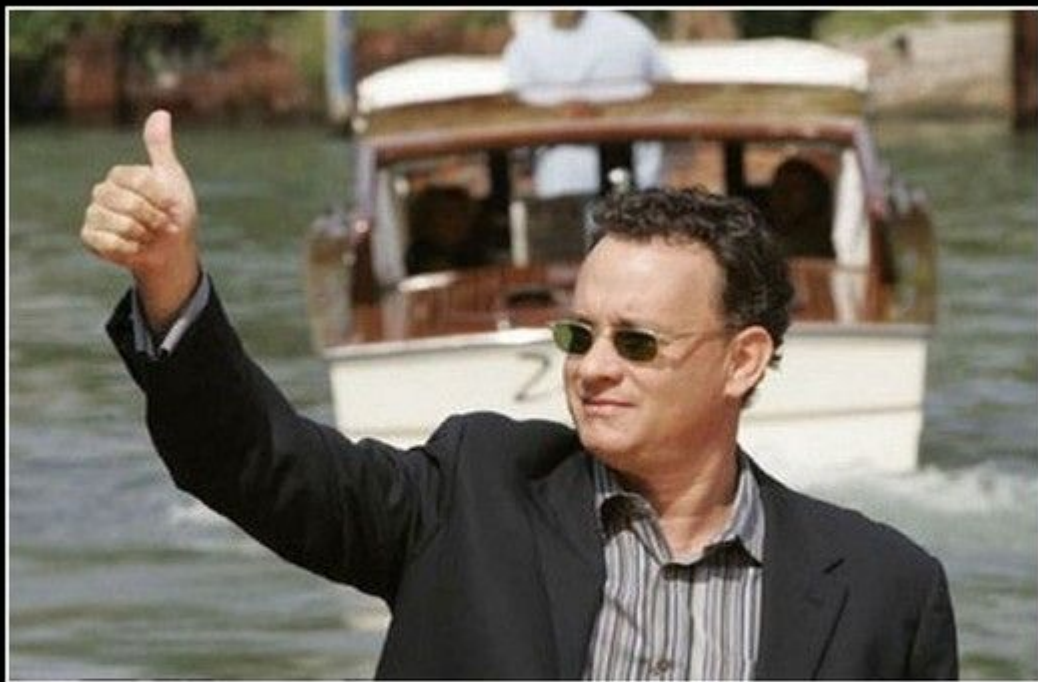
```
(...)
```



Give it a try!

add “?v” for nice format

?



T.Hanks!