

# Requests for Elastic Search / Kibana - Advanced Topics in NoSQL

Amine MILIANI      Louis GAILLET      Dimitrije DJOKIC  
Nathan IMMACOLATO

## Contents

<b>Datamodel and Import</b>	<b>1</b>
<b>Requests</b>	<b>2</b>
Easy requests . . . . .	2
Medium requests . . . . .	3
Hard requests . . . . .	4

---

## Datamodel and Import

- As for the work done on Cassandra, we decided to edit every line of the JSON data to make each its own individual insert, with a script written in Common Lisp, found in the resources folder.

```
{"index":{"_index": "restaurants","_type":"restaurant","_id":1}}
{"fields" :
  {"address": {"building": "1007",
    "coord":{"type":"Point", "coordinates" : [-73.856077, 40.848447]},
    "street": "Morris Park Ave", "zipcode": "10462"},
    "borough": "Bronx", "cuisine": "Bakery",
    "grades": [{"date": {"$date": 1393804800000},
      "grade": "A", "score": 2}, {"date": {"$date": 1378857600000},
      "grade": "A", "score": 6}, {"date": {"$date": 1358985600000},
      "grade": "A", "score": 10}, {"date": {"$date": 1322006400000},
      "grade": "A", "score": 9}, {"date": {"$date": 1299715200000},
      "grade": "B", "score": 14}], "name": "Morris Park Bake Shop",
    "restaurant_id": "30075445"}}
```

*for example here a first line*

- Once have our fixed JSON file containing all the data in a way that can be given to Elastic Search, we do so with:

```
curl -XPUT localhost:9200/_bulk -H"Content-Type: application/json" \
--data-binary @fixed-restaurants.json
```

---

## Requests

### Easy requests

#### All restaurants in Brooklyn

```
GET restaurants/_search {
  "query": {
    "match_phrase": {
      "fields.borough": "Brooklyn"
    }
  }
}
```

All Italian restaurants in Brooklyn without Pizza cuisine (because there is a cuisine type named “Pizza/Italian”, this query could be useful for the people who don’t want to eat in Italian restaurant specialized in pizzas)

```
GET restaurants/_search {
  "query": {
    "bool": {
      "must": [{
        "match_phrase": {
          "fields.borough": "Brooklyn"
        }, {
          "match": {
            "fields.cuisine": "Italian"
          }
        }
      ],
      "must_not": {
        "match": {
          "fields.cuisine": "Pizza"
        }
      }
    }
  }
}
```

#### All restaurants where cuisine type is Bakery

```
GET restaurants/_search {
  "query": {
    "match_phrase": {
      "fields.cuisine": "Bakery"
    }
  }
}
```

#### Restaurant data knowing its name

```
GET restaurants/_search {
  "query": {
    "match_phrase": {
      "fields.name": "Phillip Morris International"
    }
  }
}
```

### All Bronks restaurants

POST restaurants/\_search

```
{
  "query": {
    "query_string": {
      "query": "Bronx",
      "fields": ["fields.borough"]}}}

```

### Restaurant data knowing its id

POST restaurants/\_search {

```
  "query": {
    "query_string": {
      "query": "30075445",
      "fields": ["fields.restaurant_id"]}}}

```

### delete restaurant by ID

POST restaurants/\_delete\_by\_query {

```
  "query": {
    "query_string": {
      "query": "30075445",
      "fields": ["fields.restaurant_id"]}}}

```

### Insert of new data

PUT restaurants/restaurant/25358

```
{
  "address": {
    "building": "666",
    "coord": {
      "type": "Point",
      "coordinates": [-73.97992870000002, 40.7833573]
    },
    "street": "Broadway",
    "zipcode": "10024"
  },
  "borough": "Manhattan",
  "cuisine": "Caf\u00e9/Coffee/Tea",
  "grades": [
    {
      "date": {
        "$date": "1399939200000"
      },
      "grade": "A",
      "score": 7
    }
  ],
  "name": "Chez Kibana",
  "restaurant_id": "30079564"
}
```

---

## Medium requests

### Count of all restaurants of each cuisines

GET restaurants/\_search {

```
  "aggs" : {
    "nb_per_category" : {
      "terms" : {
        "field" : "fields.cuisine.keyword"
      }
    },
    "size": 0
  }
}
```

---

## Hard requests

### Get all restaurants from a group of cuisines that we define

- We first close the database, to be able to change settings.
- We then add in the settings some synonyms. We use the synonyms to group different cuisines together. So by giving “Asian” as a synonym of “Japanese” or “Korean”, we can just search for “Asian” without having to specify “Japanese and Korean”.
- We reopen the database.
- Then we have 2 examples of the actual request, which just searches for the group of cuisines we defined.

POST restaurants/\_close

PUT /restaurants/\_settings

```
{
  "settings": {
    "analysis": {
      "filter": {
        "my_synonym_filter": {
          "type": "synonym",
          "synonyms": [
            "ave,avenue,av","street,st","road,rd","boulevard,blvd,bvd",
            "Asian,Korean","Asian,Japanese","Asian,Chinese","Asian,Thai",
            "Asian,Indian","European,Mediterranean","European,Italian",
            "European,Irish","European,French","European,English",
            "European,Pizza","European,Spanish","European,Russian",
            "European,Greek"]}},
        "analyzer": {
          "my_synonyms": {
            "tokenizer": "standard",
            "filter": [
              "my_synonym_filter"]}}}}}
```

POST restaurants/\_open

POST restaurants/\_search

```
{
  "query": {
    "match" : {
      "fields.cuisine.keyword": {
        "query" : "Asian",
        "analyzer": "my_synonyms"}}}}
```

```
POST restaurants/_search
{
  "query": {
    "match": {
      "fields.cuisine.keyword": {
        "query": "European",
        "analyzer": "my_synonyms"}}}}}
```