Elasticsearch

Untuk Mata Kuliah

Big Data

Ahmad Rio Adriansyah
STT Terpadu Nurul Fikri

Elasticsearch

- Server mesin pencari berbasiskan Apache Lucent
- Dikembangkan oleh Shay Banon dan dipublikasikan tahun 2010
- Real-time-distributed, **open source** (Apache licence ver. 2.0), berfungsi sebagai mesin pencari dan analitik
- Scalable hingga beberapa petabyte data baik terstruktur ataupun tidak
- Digunakan oleh organisasi besar seperti Wikipedia, The Guardian, Stackoverflow, GitHub, dll.

Konsep

- Node : sebuah instance dari Elasticsearch
- Cluster
- Index
- Type/Mapping
- Document
- Shard
- Replica

Keunggulan

- Dikembangkan dalam bahasa java, kompatibel dengan hampir semua platform
- Real time, data yang baru dimasukkan langsung masuk ke dalam pencariannya
- Distributed, mudah diskalakan, dan diintegrasikan
- Berkomunikasi menggunakan RESTful-API
- Menggunakan objek JSON sebagai responnya, sehingga memungkinkan servernya bekerja sama dengan berbagai bahasa pemrograman
- Mampu memproses hampir semua tipe dokumen (selain yang tidak mensupport text rendering)

Kelemahan

- Tidak mensupport banyak cara untuk menangani request dan response (hanya JSON). Apache Solr memungkinkan format CSV, XML, dan JSON, tetapi Elasticsearch lebih mudah menangani multi-tenant dibanding Apache Solr
- Split brain situation (jarang terjadi)

Elasticsearch vs RDBMS

Elasticsearch	RDBMS
Index	Database
Shard	Shard
Type/Mapping	Table
Field	Field
JSON Object	Tuple

Instalasi

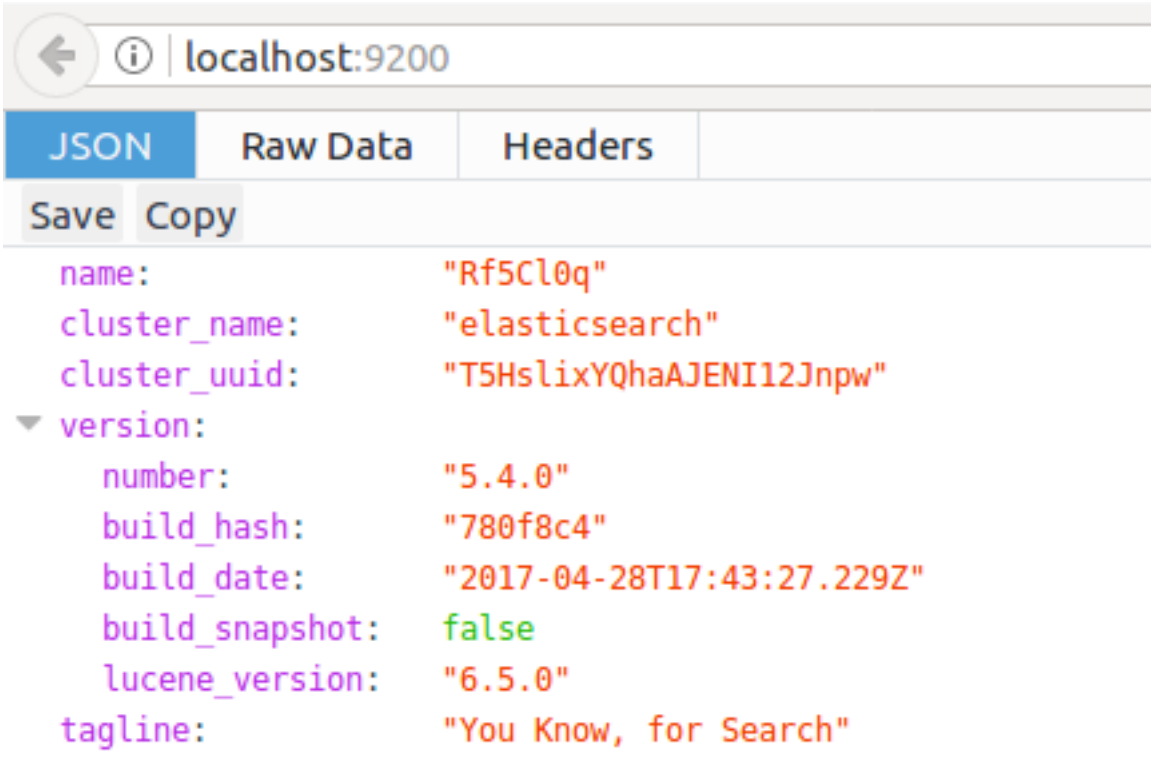
- Download Java (Elasticsearch versi 6.2. membutuhkan minimal Java 8)
  https://www.java.com/en/
- Download Elasticsearch dari link berikut
  # versi terbaru 7.0.1 (1 Mei 2019)
  https://www.elastic.co/downloads/elasticsearch
- Download Postman, Fiddler, Sense (atau client webservice lain, boleh cli seperti curl)
  https://www.getpostman.com/

Instalasi

- Setelah diekstrak, elasticsearch sudah siap dijalankan
- Jalankan menggunakan
  ```
  $ cd elasticsearch-<version>
  $ ./bin/elasticsearch
  ```
- Lakukan pengujian dengan membuka localhost di port 9200 (bisa lewat browser atau curl)
  ```
  $ curl 'http://localhost:9200/?pretty'
  ```

Instalasi

- Jika muncul respon berikut, berarti sebuah node elasticsearch sudah berjalan dan siap digunakan



Data

- Data dituliskan dalam elasticsearch dengan hirarki berikut
  /<index>/<type>/<id>
- Misalkan kita mau memasukkan dokumen dalam **index** twitter dengan **type** tweet dan **id** 1

```
PUT twitter/tweet/1
{
    "user" : "kimchy",
    "post_date" : "2009-11-15T14:12:12",
    "message" : "trying out Elasticsearch"
}
```

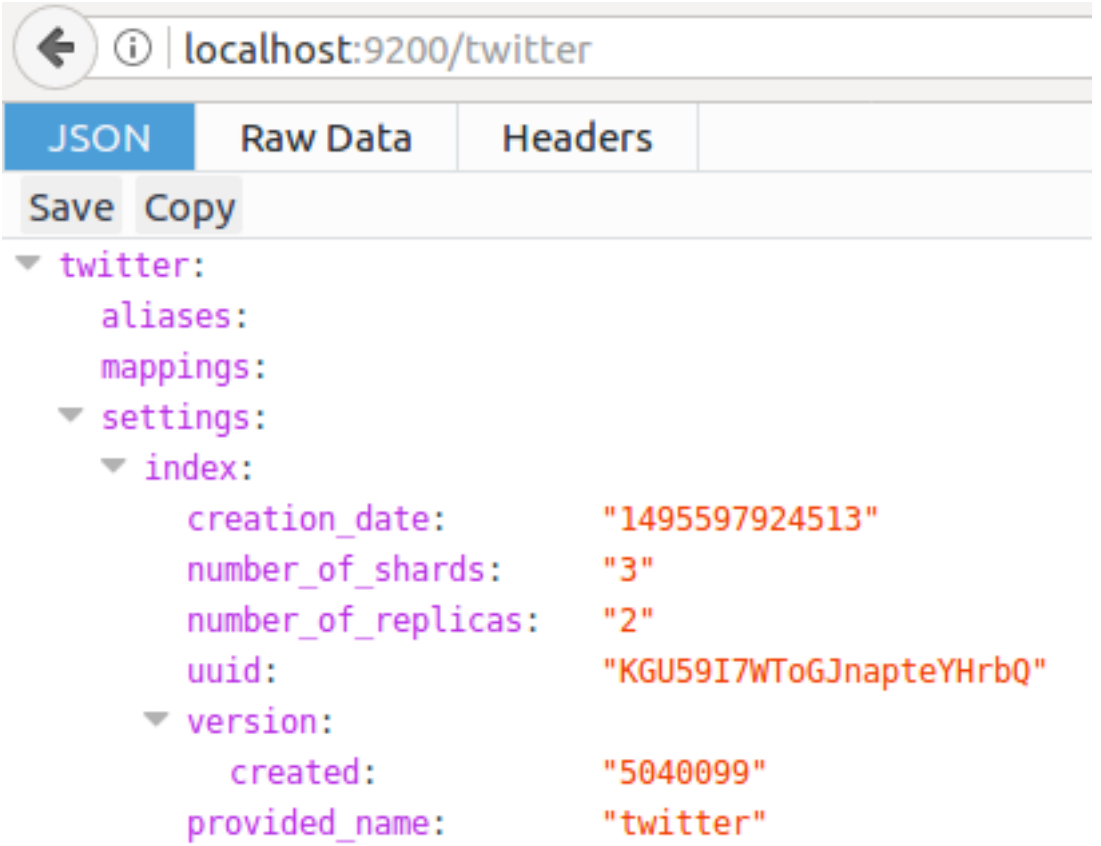
Data

Ada 2 cara untuk memasukkan data ke dalam elasticsearch :

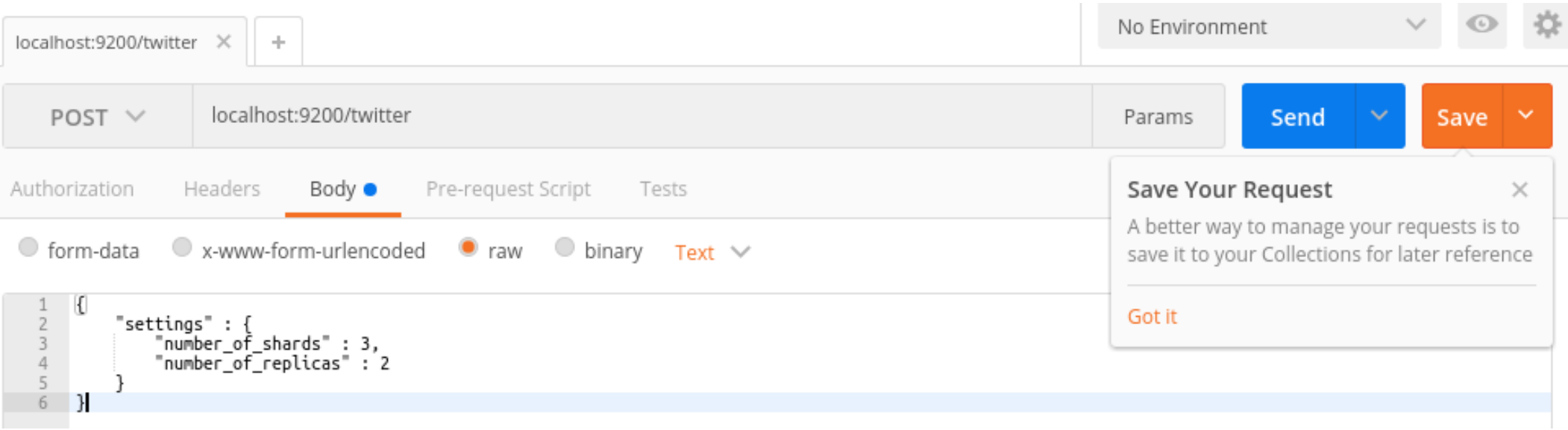
- **JSON over HTTP**
  pesan dikemas dalam format JSON dan dikomunikasikan menggunakan REST API
- **Native Client**
  overheadnya lebih rendah dibandingkan menggunakan REST, tapi harus menggunakan versi yang sama dengan elasticsearchnya

Create Index

- **Index** pada elasticsearch setara dengan **database** yang kita gunakan pada SQL
- Nama **index** harus huruf kecil semua (lowercase) atau angka
- Contoh : **index** bernama twitter



Create Index



- Menggunakan perintah POST (atau PUT)
- Setting tambahan dapat diberikan pada body

- Menggunakan curl :
  $ `curl -XPUT 'http://localhost:9200/twitter'`

  * pastikan bahwa content-type berupa json

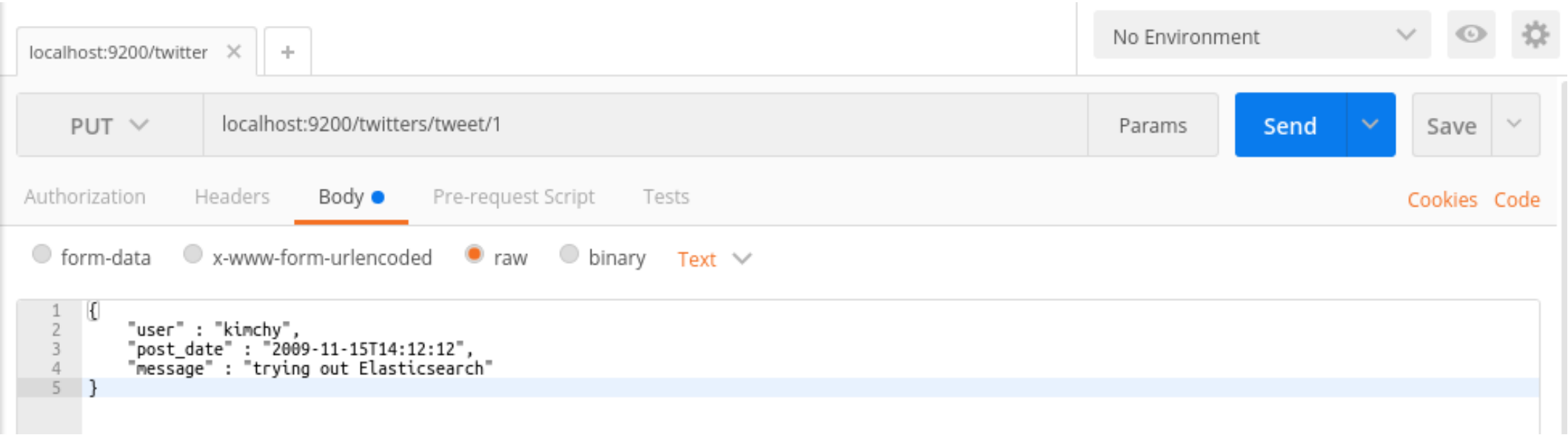
Index

- Membuat Index
  POST /<nama index> , atau
  PUT /<nama index>
- Melihat Index
  GET /<nama index> , atau difilter menggunakan
  GET /<nama index>/_settings,_mappings,_aliases
- Melihat Semua Index yang Terdaftar
  GET /_cat/indices?v
- Menghapus Index
  DELETE /<nama index>
  bisa digunakan untuk menghapus semua menggunakan wildcard (*) atau
  DELETE /_all

Index

- Untuk memeriksa apakah sebuah index ada atau tidak, bisa mengunakan
  HEAD /<nama index>
  - Respon 200 artinya ada
  - Respon 404 artinya tidak ada
- Menutup atau membuka index (index tertutup meringankan beban cluster dan tertutup atas operasi read/write)
  POST /<nama index>/_close
  POST /<nama index>/_open

Menuliskan Data ke Dalam Index



- Menggunakan curl :

```
$ curl -XPUT 'http://localhost:9200/twitter/tweet/1'
-d '{
    "user" : "kimchy",
    "post_date" : "2009-11-15T14:12:12",
    "message" : "trying out Elasticsearch"
}'
```

Latihan 1 : Memasukkan Data

- Misalnya kita mau mendaftarkan orang dan pekerjaannya dalam database kita
  - Nama = “John”
  - Nama Keluarga = “Smith”
  - Pekerjaan = “IT Analyst”
- Akan dimasukkan ke dalam **/akun/karyawan/1**
- Masukkan beberapa nama karyawan lain dan pekerjaannya ke id yang berbeda (5 orang cukup)

Membaca Data dari Index

- Jika datanya ada, kita bisa panggil langsung dengan urlnya melalui browser atau menggunakan perintah GET jika melalui REST client.
- Hasilnya akan mengandung metadata dan juga dokumennya.

```
$ curl -XGET 'http://localhost:9200/akun/karyawan/1'
```

Mengupdate Data

- Misalnya ada data yang salah, mau diperbaiki, atau mau menambahkan data baru (misalnya tanggal lahir), kita bisa menggunakan fungsi **_update**

```
$ curl -XPOST 'http://localhost:9200/akun/karyawan/1/_update' -d '{
    "doc" : {
            "pekerjaan":"CTO",
            "tahunlahir":"1980"
    }
}'
```

Mengupdate Data

- Perhatikan bahwa update yang dilakukan diletakkan pada **“doc”** dalam jsonnya, tidak langsung.
- Jika kita panggil menggunakan perintah GET, versi dokumen tersebut terlihat sudah diupdate, tetapi bukan berarti dokumen versi sebelumnya tersimpan otomatis.
- Kita **tidak bisa** secara langsung mengambil versi sebelumnya dari dokumen yang sudah diupdate. (bisa diakali dari cara penyimpanan dokumennya)

Mencari Data

- Selama ini kita memanggil data menggunakan id dari dokumennya (link lengkap). Kita bisa mencari data yang mengandung kata tertentu dari seluruh dokumen yang ada di **index** atau **type** tertentu dengan menggunakan fungsi **_search**
- Formatnya **_search?q=katayangdicari**

```
$ curl -XGET
'http://localhost:9200/akun/karyawan/_search?q=john'
```

Mencari Data

- Pencarian dengan _**search?q=katayangdicari** mengembalikan semua nilai pada semua key.
- Jika ingin mencari kata pada key tertentu, maka kita gunakan filter tambahan
- Formatnya _**search?q=keyfilter:value**

```
$ curl -XGET
'http://localhost:9200/akun/karyawan/_search?
q=nama:ahmad'
```

Latihan 2

- Dari data karyawan yang tadi dimasukkan carilah :
  - Karyawan yang pekerjaannya mengandung kata IT
  - Karyawan yang namanya Ahmad
  - Karyawan yang id nya 5
- Buat index lain dan isi dengan beberapa data
- Carilah data lintas index yang memunculkan informasi minimal 1 dari masing masing indexnya

Menghapus Data

- Data yang ada bisa dihapus dengan menggunakan perintah DELETE ke index, type, atau id

```
$ curl -XDELETE ‘
http://localhost:9200/akun/karyawan/1’

$ curl -XDELETE ‘
http://localhost:9200/akun/karyawan/’

$ curl -XDELETE ‘http://localhost:9200/akun/’
```

Menggunakan Data yang Lebih Besar

- Download dataset di

  https://www.elastic.co/guide/en/kibana/6.2/tutorial-load-dataset.html
- Ada 3 dataset yang tersedia, download **accounts.zip** untuk digunakan sebagai contoh. **Shakespeare** dan **logs** dapat digunakan untuk latihan dan eksplorasi

Dataset Accounts

Dataset accounts memiliki skema seperti berikut

{

"**account_number**": INT, "**balance**": INT,
"**firstname**": "String", "**lastname**": "String",
"**age**": INT, "**gender**": "M or F",
"**address**": "String", "**employer**": "String",
"**email**": "String", "**city**": "String",
"**state**": "String"

}

Bulk Insert

- Menggunakan fungsi _**bulk**

```
$ curl -H 'Content-Type:application/x-ndjson' -XPOST 'http://localhost:9200/bank/account/_bulk?pretty' --data-binary @accounts.json
```

- Untuk yang tidak bisa menggunakan curl, bisa merefer ke https://www.elastic.co/guide/en/elasticsearch/reference/6.2/docs-bulk.html

Query DSL

- Dapat melakukan pencarian dengan query yang lebih kompleks daripada query melalui url seperti pada contoh sebelumnya

```
$ curl -XGET
'http://localhost:9200/bank/_search' -d
'{"query":{
      "match":{
          "age":25
          }
      }
}'
```

Query DSL

- Klausa pencarian pada elasticsearch bisa dibagi menjadi
  - Klausa tunggal (leaf query clause)
    mencari nilai tertentu pada field tertentu. Contohnya **match**, **term**, atau **range**
  - Klausa gabungan (compound query clause)
    mengkombinasikan klausa-klausa tunggal atau gabungan dalam logika tertentu

Match

- Menganalisis dan menyusun query dari inputan yang diberikan

```
$ curl -XGET
'localhost:9200/bank/account/_search'
-d '{"query":
      {"match":
        {"address": "300 narrow avenue"}
      }
    }'
```

Match

- Query pada slide sebelumnya akan mengembalikan dokumen-dokumen yang mengandung kata yang mirip dengan “300 narrow avenue” pada field addressnya.
- Nilai kemiripannya ditampilkan pada **_score** . Semakin tinggi nilainya artinya semakin mirip.
- Untuk mempercantik tampilannya, tambahkan parameter pretty di akhir url

```
$ curl -XGET 'localhost:9200/bank/account/_search?pretty'
```

Term

- Akan mengembalikan dokumen yang memiliki nilai tertentu dalam inverted indexnya

```
$ curl -XGET
'localhost:9200/bank/account/_search'
-d '{"query":
      {"match":
        {"address": "avenue"}
      }
    }'
```

Range

- Mengembalikan dokumen yang memiliki nilai pada rentang tertentu
- Cara elasticsearch memproses query tergantung dari tipe field pada querynya
  - Field bertipe string akan diproses dengan **TermRangeQuery**
  - Field bertipe number/date akan diproses dengan **NumericRangeQuery**

Range

- Misalnya mau mencari yang umurnya (field age) pada rentang tertentu
  (antara 20 dan 25 tahun, inklusif pada contoh)

```
$ curl -XGET 'http://localhost:9200/bank/_search' -d
'{"query":{
            "range":{
                "age":{
                    "gte":20,
                    "lte":25
                }
            }
        }
}'
```

Range

- Parameter yang diterima oleh range query adalah :
  - **gte** (greater than or equal) ; (>=)
  - **gt** (greater than) ; (>)
  - **lte** (less than or equal) ; (<=)
  - **lt** (less than) ; (<)
  - **boost** (untuk menekankan tingkat kepentingan querynya, defaultnya 1.0)

Date Range

- Untuk tanggal, nilai yang diberikan pada query mengikuti date math
- Pada contoh data bank account tidak ada field yang berisi tanggal, jadi digunakan contoh lain

```
$ curl -XGET 'localhost:9200/twitter/_search?pretty' -d
'{"query":{
           "range":{
               "post_date":{
                  "gte": "now-1d/d",
                  "lte": "now/d"
               }
           }
      }
}'
```

Date Range

```
now - 1d/d
```

- now = waktu saat query dilakukan
- 1d = 1 hari
- /d = dibulatkan sampai hari

Date Range

- Unit waktu yang diizinkan (case sensitive)
  - y = years (tahun)
  - M = months (bulan)
  - w = weeks (minggu)
  - d = days (hari)
  - H/h = hours (jam)
  - m = minutes (menit)
  - s = seconds (detik)

Query DSL

- Informasi lebih lanjut bisa dibaca di

  https://www.elastic.co/guide/en/elasticsearch/reference/6.2/query-dsl.html

Latihan 3

- Dari data akun yang diberikan, carilah orang yang nama emailnya (bisa nama alamatnya atau domainnya) dimulai dari huruf **a**. Ada berapa?
- Ada berapa dari data tersebut yang laki-laki?
- Ada berapa yang balancenya lebih besar dari 40000?
- Cari id dari orang yang bernama “Marion Schneider”

ELK Stack

- Elasticsearch bekerja dengan baik saat digabungkan dengan dua buah aplikasi lainnya yang saling mendukung yaitu Logstash dan Kibana
- Digunakan terutama untuk analisis log di lingkungan IT
- Ketiganya ini disebut sebagai ELK Stack atau Elastic Stack

Instalasi Kibana, Logstash, dan Sense

- Kibana adalah dashboard untuk elasticsearch. Menyediakan eksplorasi visual dan analisis real-time terhadap data.
- Logstash adalah apps untuk data collection pipeline. Ini adalah komponen dari Elastic Stack yang mengumpulkan data dan memasukkannya ke dalam elasticsearch.
- Sense adalah Kibana apps yang menyediakan konsol interaktif untuk mengirimkan request dari browser ke node elasticsearch.

Data Replication

- Setiap index pada elasticsearch dibagi menjadi *shards* dan tiap *shard* dapat memiliki salinan (*replicas*) yang harus tersingkronisasi saat dokumen ditambahkan atau dihapus.
- Ketidaksingkronan akan mengakibatkan pembacaan yang berbeda saat dibaca dari satu salinan dan yang lain.
- Proses memastikan bahwa salinan dari shard tetap singkron disebut sebagai *data replication model*.

- Ref :

  https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html