**Chapter 1: Introduction**

ElasticSearch: Phat trien dua tren Apache Lucene

inverted index table: map giua term <-> document chua term

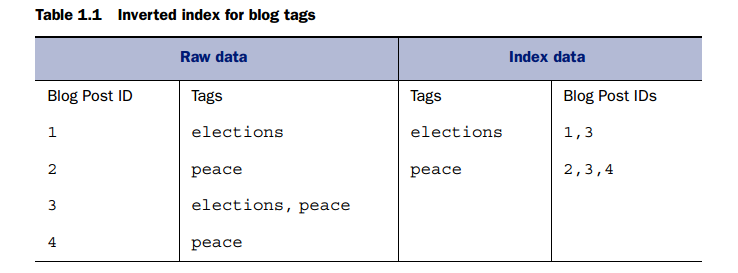
talk to it: rest api (json)

Co the deploy on top of rdms/nosql to index and speed up sql query. Cung co the dung de store data as a nosql database (with enhanced search capability) (as standalone database).

Concepts:

+ index: data structure create base on "real data" to faster searching (like database)

inverted index: list of where each word belong to -> search ‘elections’ -> 1,3: really faster’



Elasticsearch ability

* Performance
* It provide **relevancy** too….ex.

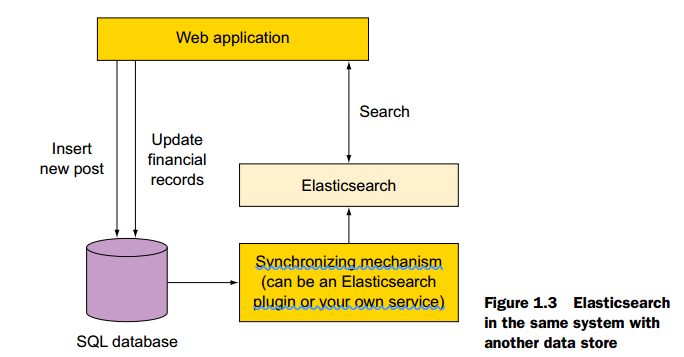
Relevancy score -> sort the result: elasticsearch have some algorithms (default is TF-IDF) to do this with many cretirias (trọng số khác nhau: match exactly, match title (over body), newer over older...)

* It also provide **searching beyond exact match** (synonym, typo (fuzzy query)…), derivative, statistic, suggestion, flexible sorting
* Scale
* …..
* ES do not support transaction.

Data ----feed to----> Elastic search ----> call Elastic search

3 type of using ES

* Use as durable storage (with search enhanced)
* Use with other datastorage (RDMS, NoSQL…)



* Integrate elasticsearch with other component: find tools first

Tools: Sense (google chrome), curl, elasticsearch-head plugin

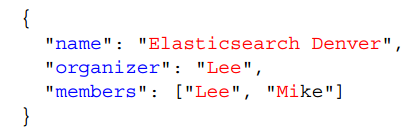
Need to learn (native and java api) for now

* Query (search)
* Indexing, update, delete
* Analyzing
* Search with relevancy
* Aggregation
* Relation among documents
* How to scale?
* Apache Lucene

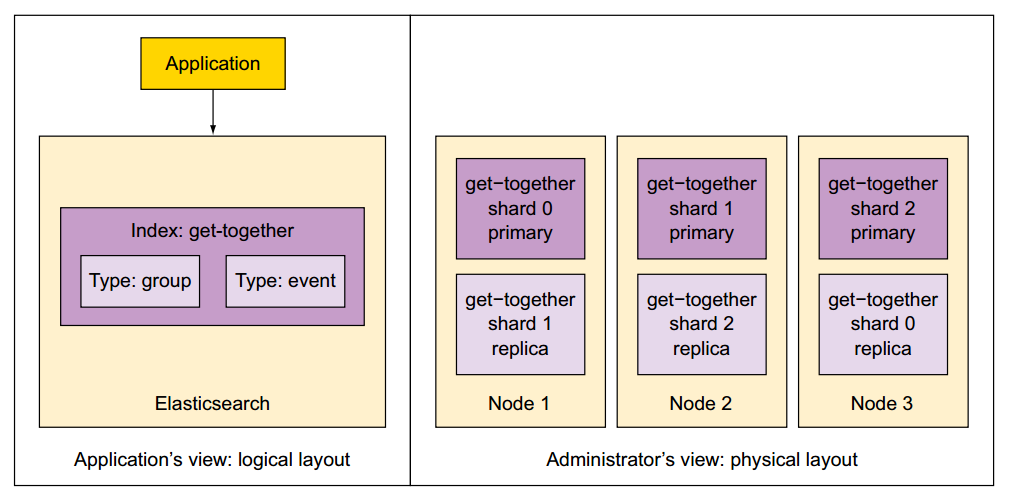
**Chapter 2: Getting started with elasticsearch functionality**

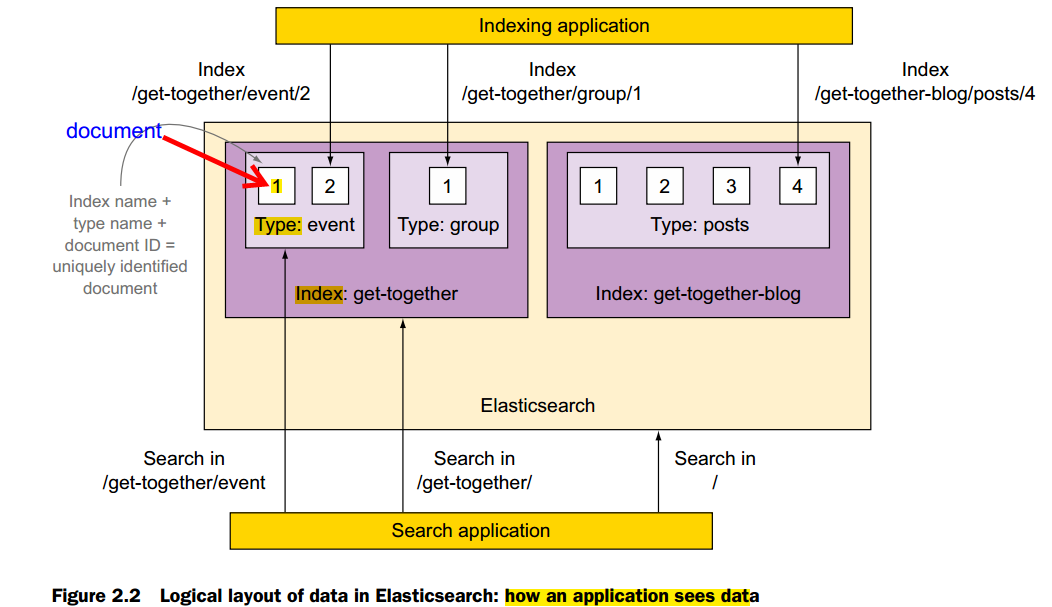
How Elasticsearch data is organized?

**Logical layout**: document, type, index.

* Document: smallest unit of data to index and search. Is the json represent of data, schema free.  
  Ex:  
  
* Type (mapping type): contain documents (like table).
* Index: Contain types. Can search on 1 or more indices (like database).

**Physical**: shard





* Node: When run elasticsearch -> 1 node
* Cluster: 1 or more nodes
* Shard: directory of files containing converted index.

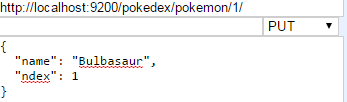
Index have 1 or more primary shard, 1 shard have 0 or more replica (copy)

Cluster status

**Some operation with elasticsearch (intro)**

Indexing document by PUT/POST

Creating a document



\_source filed in response

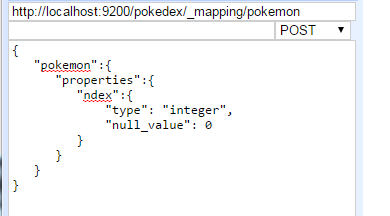
\_version filed in response

Updating document all and partially

Deleting with DELETE request

Check existence with HEAD request/GET

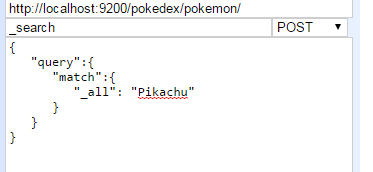
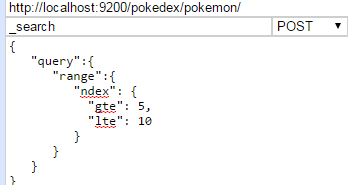
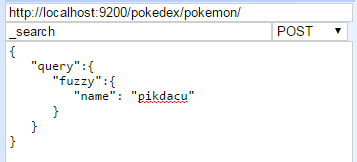
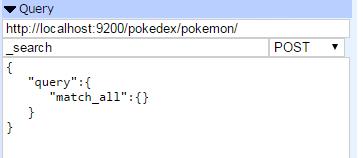
Create type mapping (type): field and type of field: \_mapping (/pokedex/\_mapping/pokemon) - pokemon is type



Run mapping from file with curl

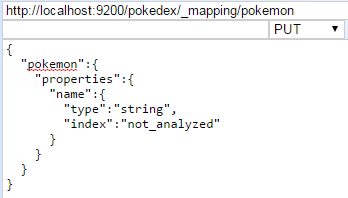
Using bulk api to indexing data from json file

**Query**

* match  
  
* Range  
  
* Fuzzy  
  
* match\_all  
  

Some standard types:

+ string type  
Ex:

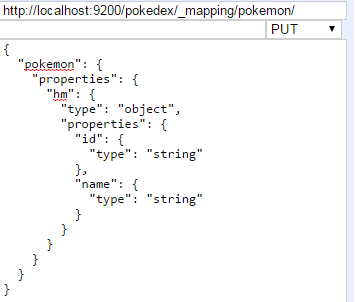
* index attribute
  + analyzed: default - for searching
  + not-analyzed: index but just search exactly -for aggregation and sorting
  + no: no index -> no searchable  
    
* include\_in\_all:
  + True
  + False: no find in generic search
* boost: Number value: e.g 2.0, when search with “\_all” -> filed with this property > 1 become more important (trọng số)
* copy\_to: copy attributes to particular field (for composition searching)  
  

Operator: “and” or “or” - when search 2 word (trở lên) -> apply rule cho từng term trong search text (nếu là analyzed) then return 2 boolean result -> and/or these boolean result-> final

Important note about query: for analyzed field - execute the query trên từng term.

Other type: integer, boolean, date…

Object type

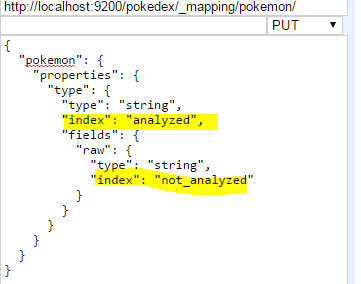


Nested type:

Use when index array of object and remain the relationship between object properties



Multi field: index same filed in different way (for multiple puposes: ex: analyzed and not-analyzed for “index” attribute.

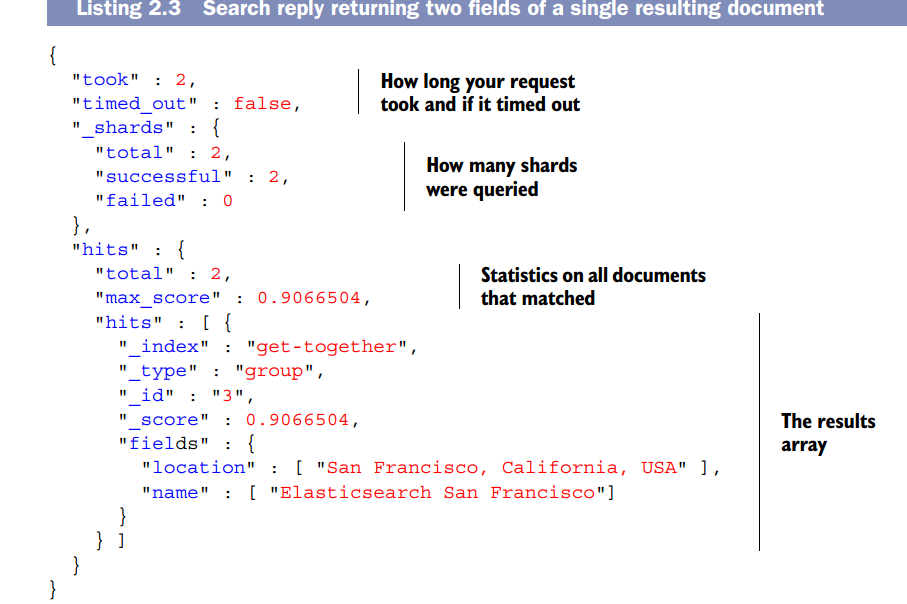


Till now: \_mapping, \_bulk, \_search api

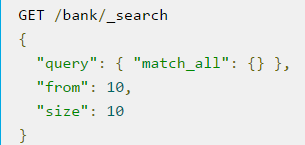
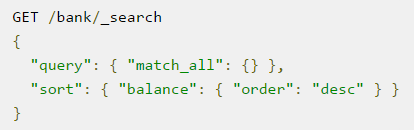
**Understanding elastic searching**

* Exact term search
* Full text search

Reply content: matched document and other information for checking performance



How to search:

* Uri request (simple, for test)
* Request with json - using query DSL  
    
  Or  
  

=> mapping to java api

**Chapter 3: Indexing, update, delete document**

1. **Mapping**

*Get current mapping type:* Using elasticsearch-head (index metadata)  
Or using curl

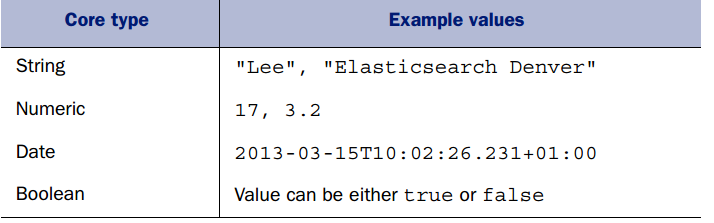
  
Or using http GET  
<http://localhost:9200/pokedex/_mapping/pokemon/>

*Define new mapping:*   


Extend mapping: when add mapping type with existing name -> extend.  
Note: when define mapping, elasticsearch auto create predefined filed: eg. \_all for different purpose.

**2. Mapping types**

**2.1. Core types**



**2.2. Array and multi-field**

**3. Predefined field**

Start with \_, auto po;ulated by elasticsearch

* Control how to store and sort document: \_source, \_all
* Identify document: \_uid, \_id, \_type, \_index
* New property for document: \_timestamp, \_ttl, \_size
* Control the shard where document is routed to: \_parent, \_routing

**4. Update existing document**

Using \_update api

**5. Delete existing document**

Using delete request

**Chapter 4: Searching**

* Url based request
* Json body based request

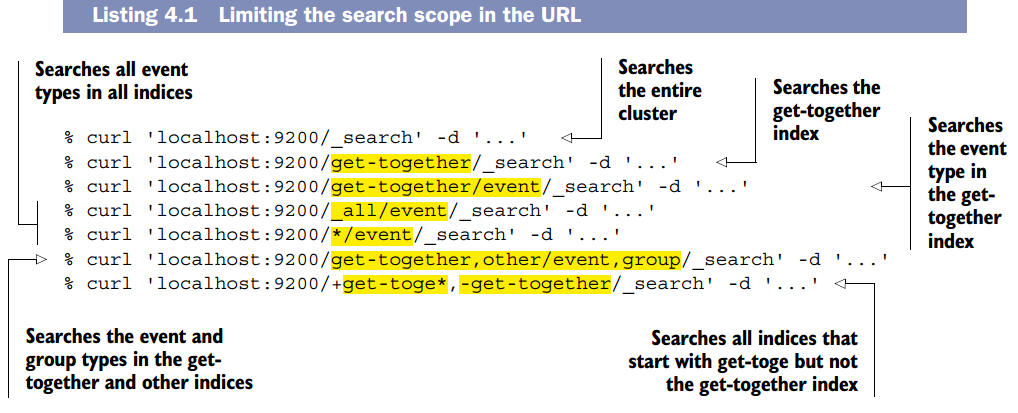
Search request ---send to api---> node----send to----> all shards ---> get documents result.

This search routing is configurable. Default is “query\_then\_fetch”.

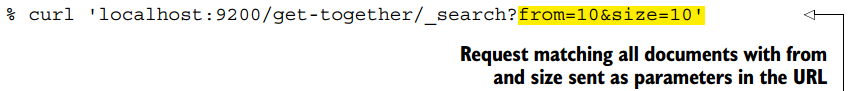
1. **Structure of search request**

URL based request or JSON based request  
All search request using \_search end point, can be GET or POST

**1.1. Search scope** (search from where?)



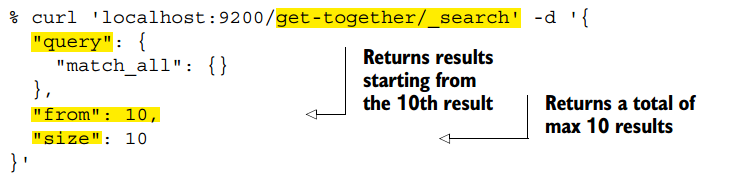
**1.2. Basic component of search request** (specific how many document returned, select best document, config document don’t want to return) - Search API

* query: most important
* size:
* from: together with size for pagination  
  ex with URL based request:  
  
* \_source: filter fields returned (vs using fields attribute, prefer \_source)
* sorting: sorting based on document score

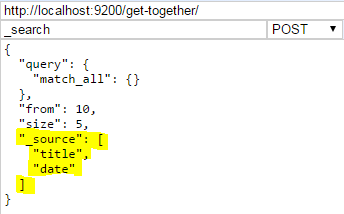
**1.3. URL based request**

**1.4. JSON based request:** more flexible and more option (có thể dùng URL based mix JSON based). Can be used with Sense (or other plugin), elasticsearch-head, curl.

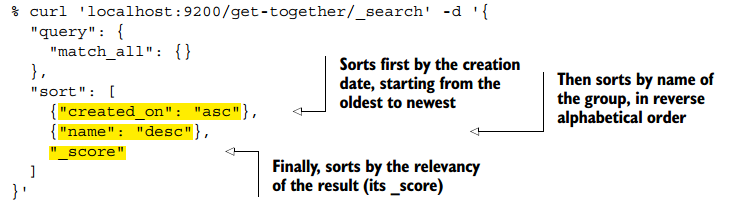
ex:



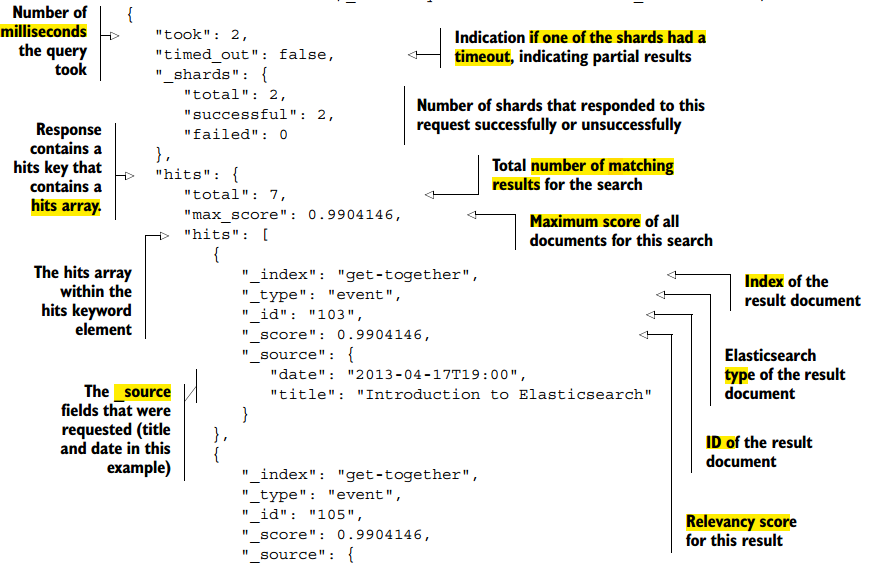
(if no \_source component -> entire fields in document, in case \_source isn’t store (define mapping) -> just return metadata like \_index, \_type...of returned document).  
Ex:



Sorting and filtering result with sort component



**2. Structure of response**



**3. Query and filter DSL** (query component of request)

Phân loại

* Base on ….: 2 types of query
  + Leaf query: look for particular value in particular field
  + Compound query:
* Base on how query is treated
  + Term query: search exact text of string query (as a term), rarely use directly
  + Full text query:
    - On not-analyzed field: like single term
    - On analyzed field: analyze string query to separate terms then search on each term

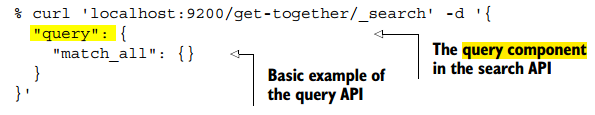
The behavior of ***query clause*** depends on

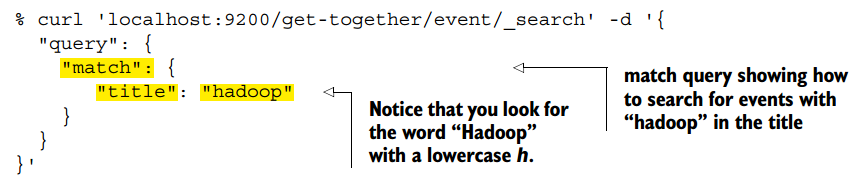
* the query context: how well document match? (\_score), need score -> use this
* filter context: does this document match this query clause. No score, use yes/no, use for filtering (“filter” parameter)

  
Here, “term”:{}...is query clause

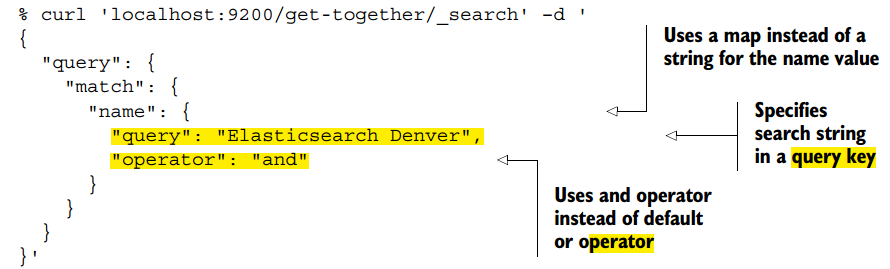
**3.1. Full text query**

*Match\_all query*

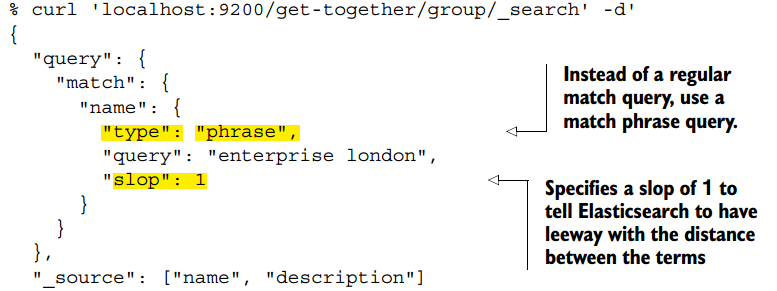
 *match query (most use)*



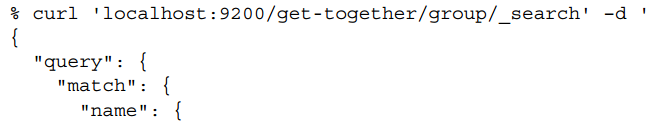
boolean behavior: If match for “Elasticsearch Denver” => Elasticsearch OR Denver

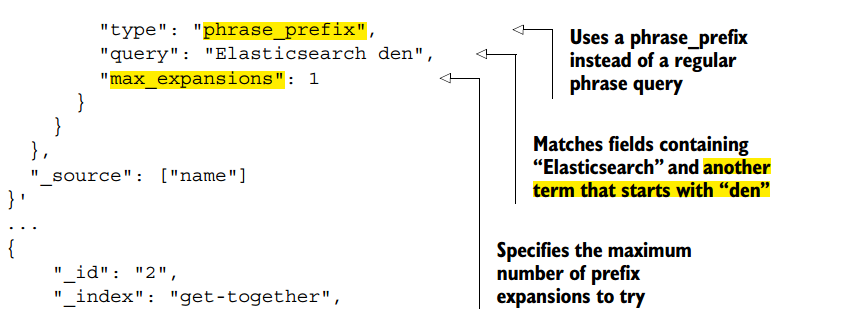


phase behavior: use when search for “phrase” (not must continuous)

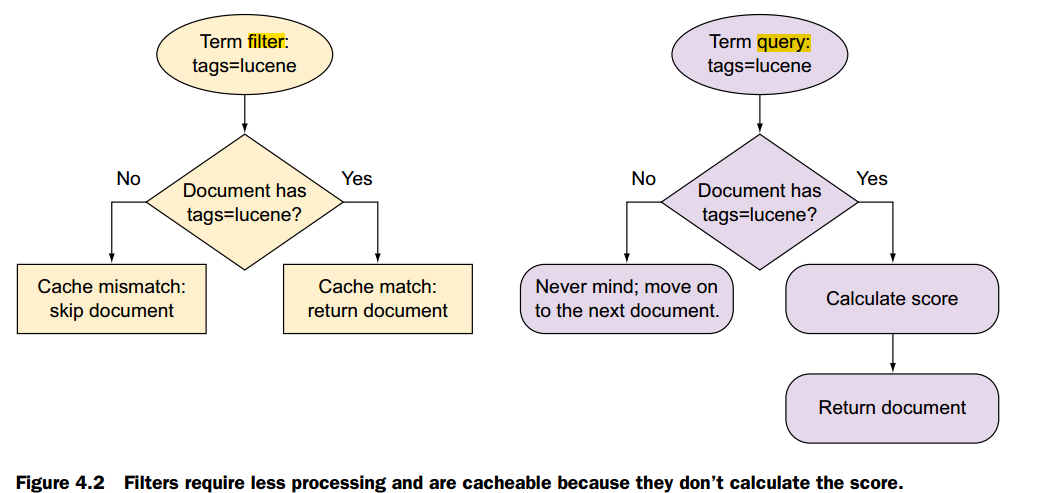


phrase\_prefix behavior: allow prefix matching on the last term in the phrase, useful for autocomplete function.



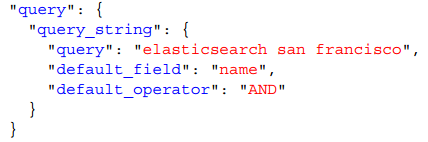


Filter vs query



Filtered query contains 2 components: query and filter - deprecated, change to **bool query**.

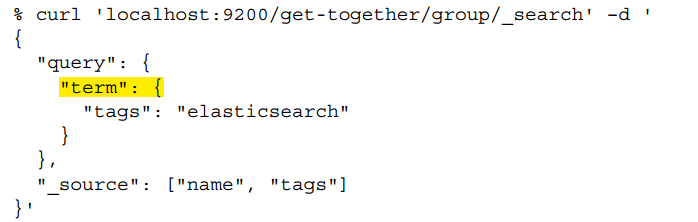
*query\_string query:* Can use Lucene query syntax for this query.



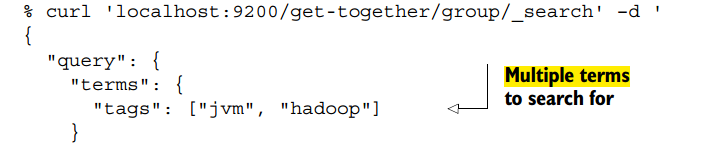
Powerful but recommend other query before this query because it hard to maintain.

***3.2. Term level query: match exact term***

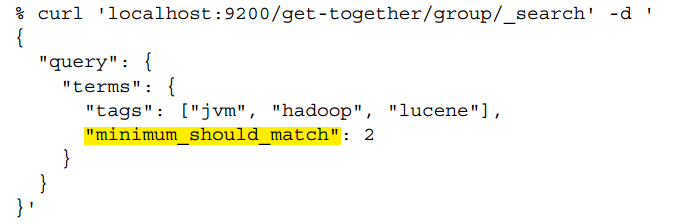
*term query: match exactly (not-analyzed)*



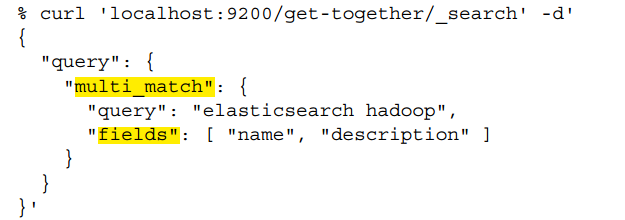
*terms query*



minimun\_should\_match param



*multi\_match query: search across multiple field (“query” using OR)*

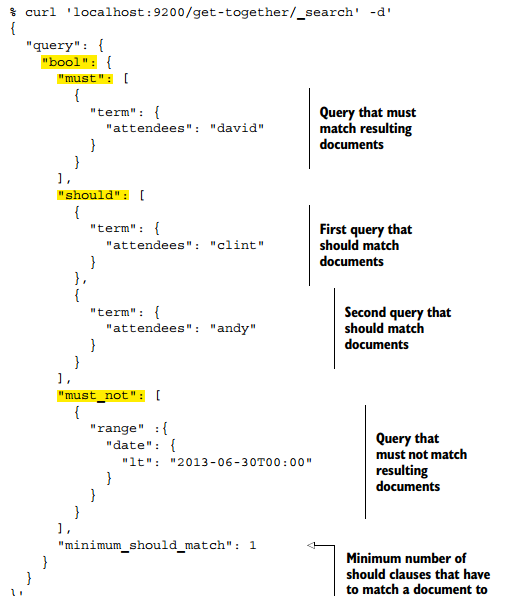


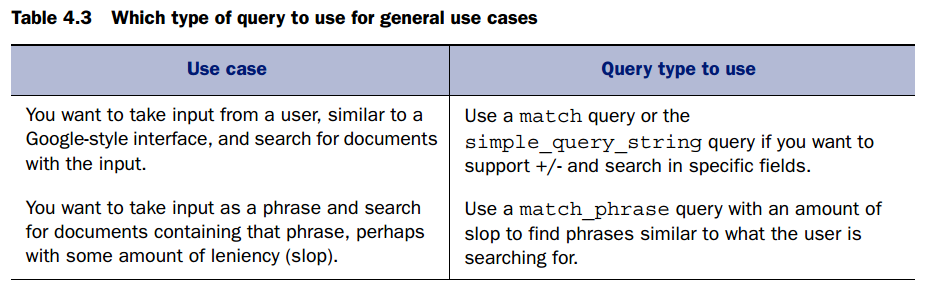
Can use boolean, phase, phase\_prefix like match query.

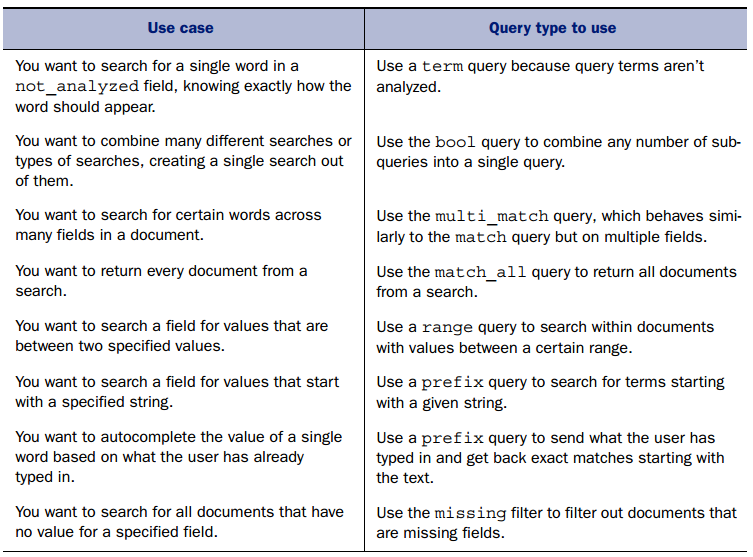
Search API vs Query DSL: Search API is about “query”, “sort”, “size”....(components), Query DSL is about body of “query” component.

4. Compound query

Bool query: must, should, must\_not: trong bool query contains other query (compound voi nhau)







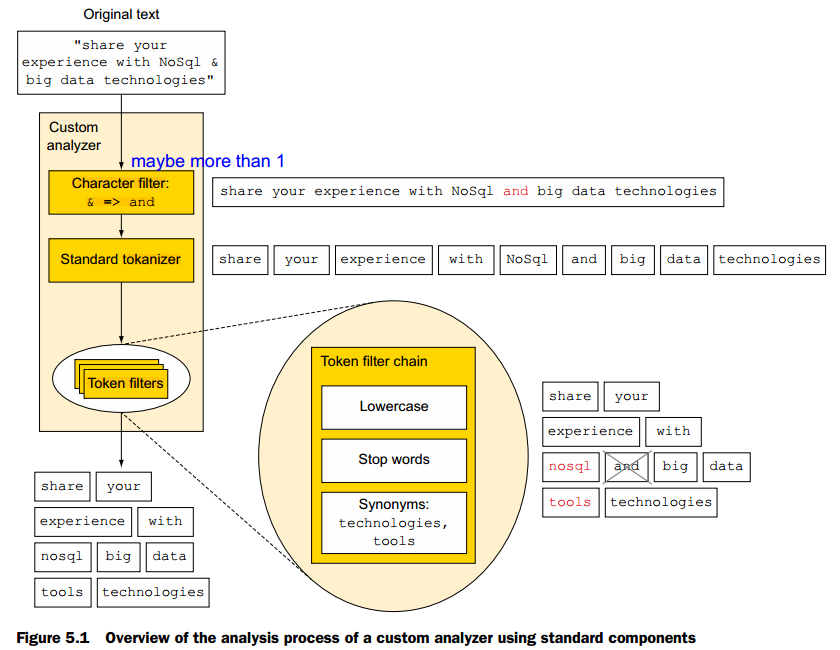
**Chapter 5: Analyzing data**

Analysis: Process body of query before send to inverted index: indexing

In searching: full text query will be analyzed, term query will not.

For every **analyzed** filed, do steps:

* Character filtering
* Breaking text to tokens: tokenize
* Token filtering
* Token indexing



can specify in the mapping which individual tokenizer and token filters to use for an analyzer and which analyzer to use for which field.

2 ways to specify analyzer

* Setting when create index (mapping - for particular index). Khi change analyzer ko cần restart ES. Prefer
* Add global analyzer in configuration file, sử dụng khi apply 1 analyzer for nhiều indices và ít thay đổi - using put mapping api.  
  Khi change analyzer cần restart ES to get effect.

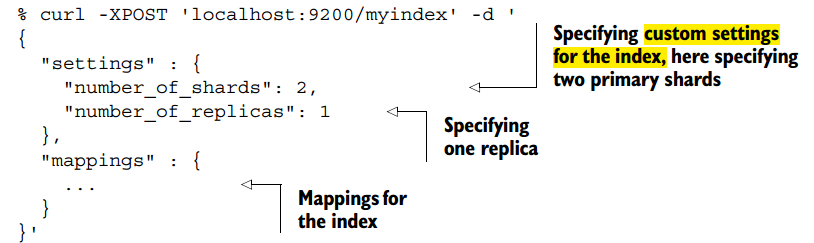
Can kết hợp cả 2.

Use analyzer

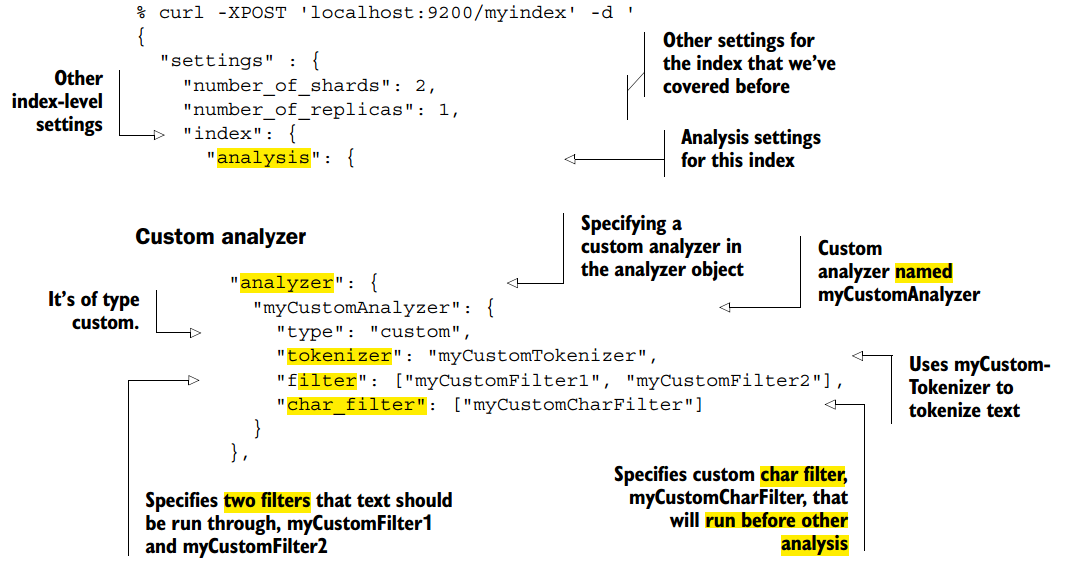
* Indexing: cần chỉ rõ field nào dùng analyzer nào, field ko được declared -> using standard
* Searching:

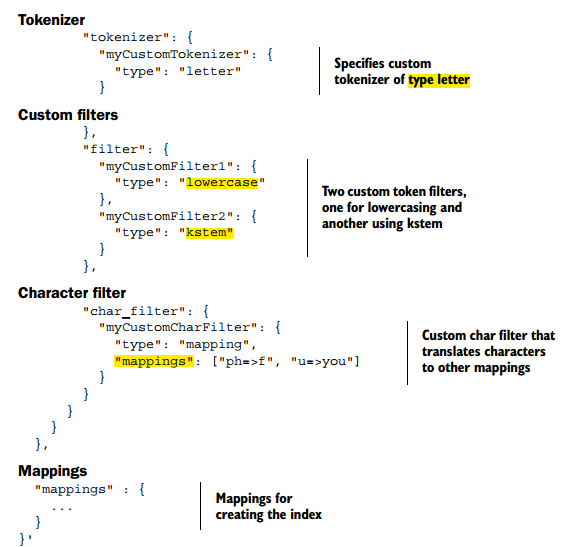
**5.1. Add analyzer when create index**

Review index setting

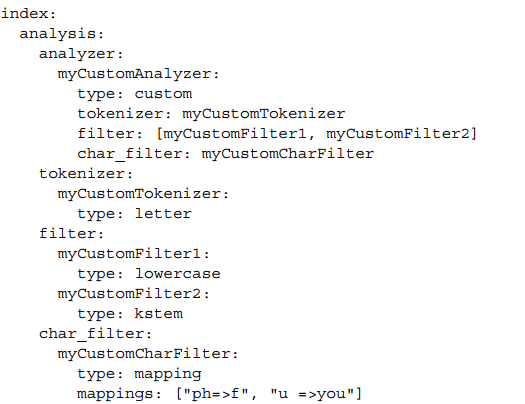


Add custom analyzer

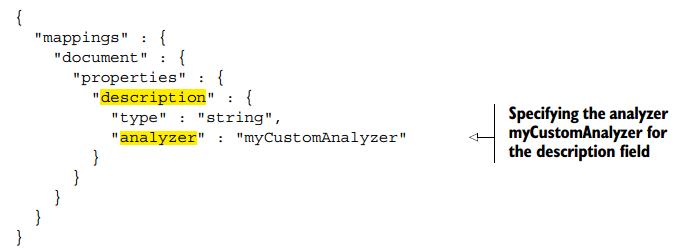




**5.2. Add analyzer when to the Elasticsearch configuration**

Specify in elasticsearch.yml  


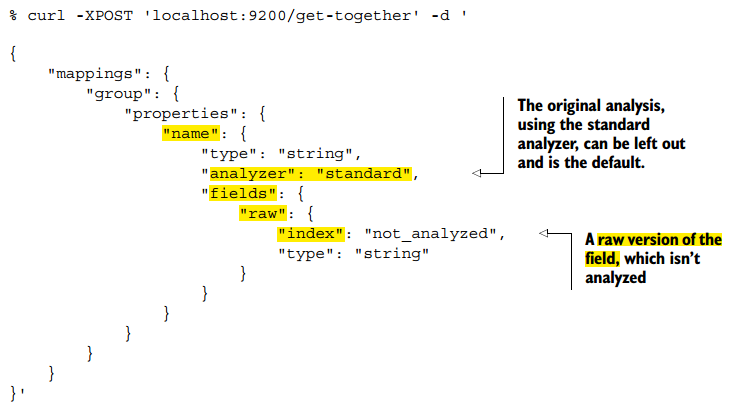
**5.3. Specify analyzer for fields (which field will be affected)**



Default sẽ có analyzer (with lowercase character filter…) run, want to disable all -> “index”:”not-analyzed”.

Nếu muốn vừa có analyzer and not-analyzed for a field -> using multiple filed.

Thường thì muốn search on both analyzed and not-analyzed for a field (duplicate field).

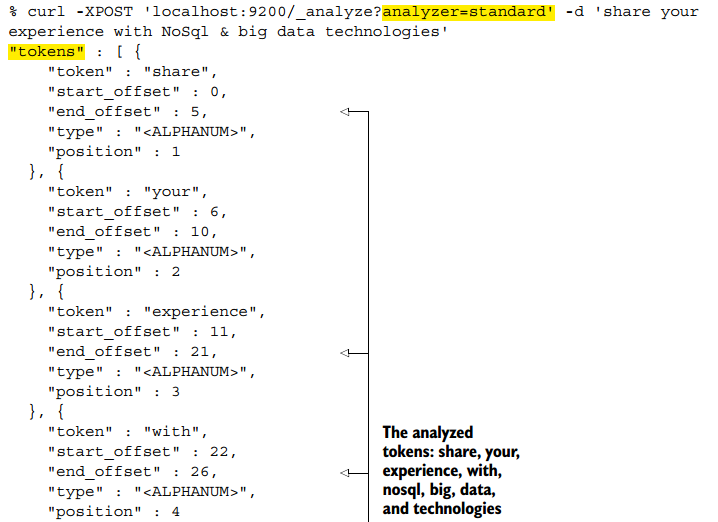
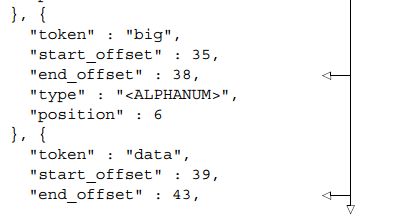


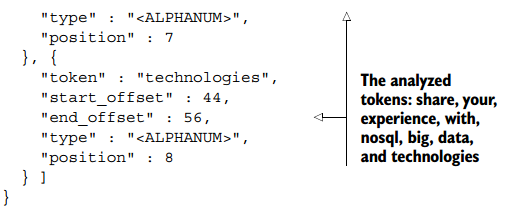
**5.4. Analyzing text with the analyze API**

Use to test analyze process: tracking how information is stored (how data is analyzed). This API

allows you to send any text to Elasticsearch, specifying what analyzer, tokenizer, or

token filters to use, and get back the analyzed tokens



5.4.1. Selecting an analyzer

Built in analyzers (‘standard’ is 1 of them) and custom analyzer

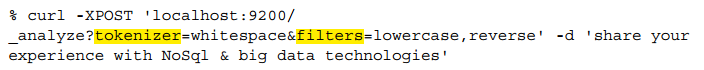
Global



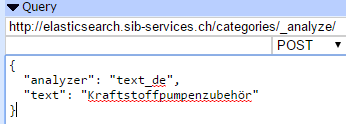
and index-specific analyzer



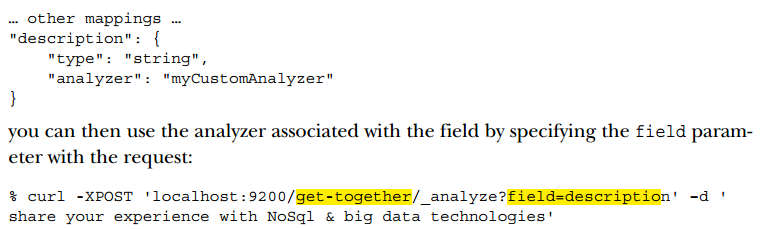
5.4.2. Impromptu analyzer



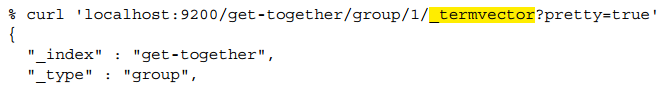
With head-plugin

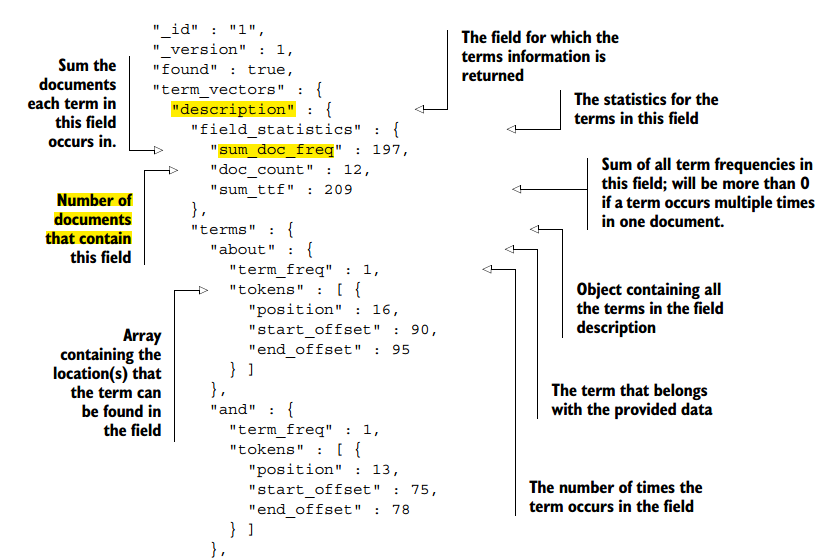


5.4.3. Analyze base on field mapping

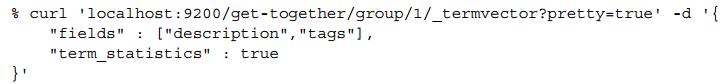


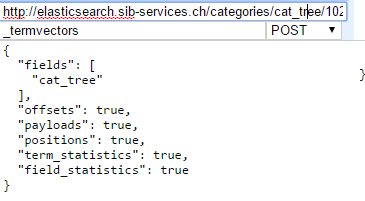
5.4.4. \_termvector endpoint: More info about terms in document



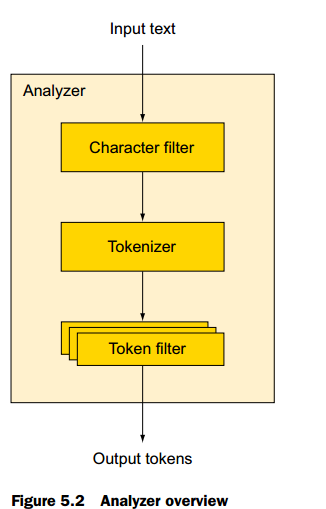


Also with fields





**5.5. Analyzer, tokenizer and token filter**



Note: define analyzer in index, then field nào muốn apply thì phải

an analyzer consists of an optional character filter, a single tokenizer, and zero or more token

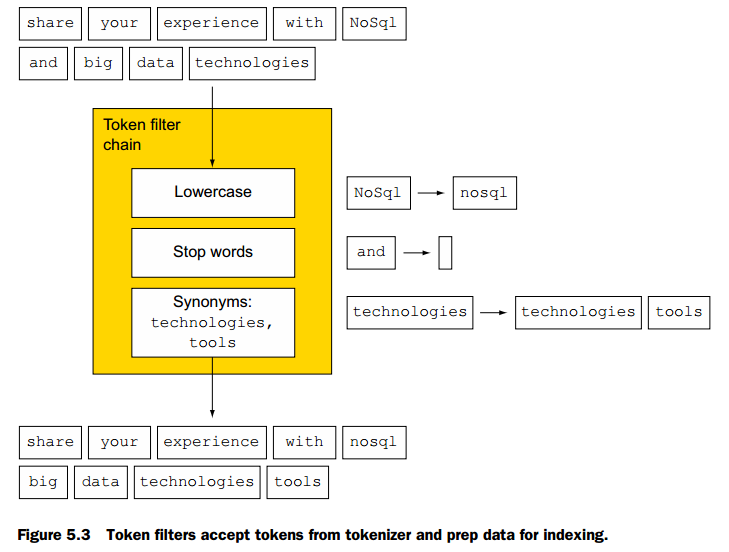
filters .

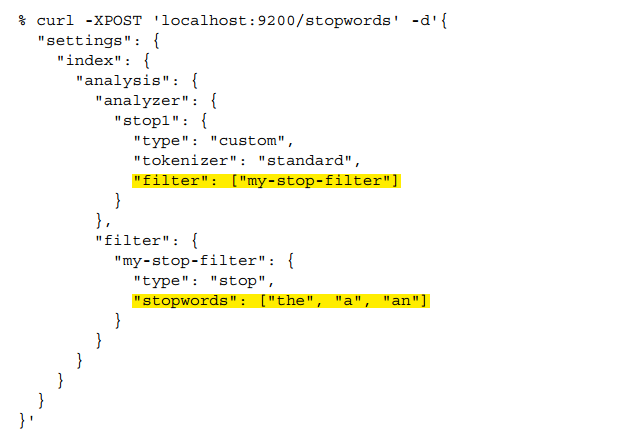
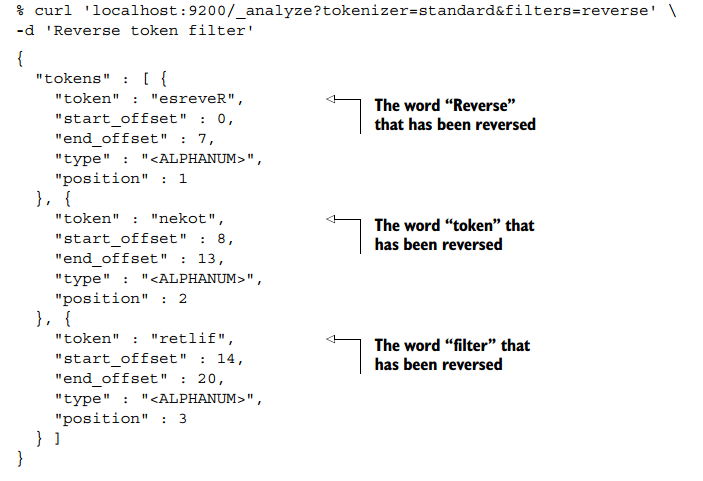
Some built in analyzer

* Standard analyzer: default
* Simple analyzer: just lowercase
* Whitespace analyzer:
* Srop analyzer
* Keyword analyzer
* Pattern analyzer
* Language specific analyzers: many, if don’t exist, find plug in
* Snowball

Some built in tokenizer

* Standard
* Keywork: like not-analyzer
* Letter: break to letter (not punctuation…)
* lowercase
* Whitespace
* Pattern: breaking by pattern
* Uax\_url\_email
* Path\_hierachy:

Some built in token filters:  


* Standard: does nothing
* Lowercase:
* Length: min and max
* Stop: remove stop words  
  Ex for English  
    
  Can specify by
  + direct  
    
  + File:
  + For language: \_dutch\_, ….
* Truncate, trim, max\_token\_count
* Reverse:   
    
  Useful when searching with wildcard (performance)
* Unique
* Asciifolding
* Synonyms  
  

**5.5. Ngrams, edge ngrams, shingles (tokenizer)**

Ngram: split single word to parts

Unigram, bigram, trigram, min/max gram

Advantages: longword (like German), multiple languages…(later)

Edgengram: can specify direction (from start or from end)  


Shingles: ngram at word level: ex for ‘foo bar baz’ text with min=2, max=3 (1 is original tokens, can turn off)  


**5.6. Steaming (token filter)**

Reduce words to root word. Make searching is more flexible

If the word is “administrations,” the root of the word is “administr.” This allows you to match all of the other roots for this word, like “administrator,” “administration,” and “administrate.”

Can apply by using algorithmic, one-to-one mapping or dictionary.

5.6.1. Algorithmic steaming:

* Snow ball
* Porter stem
* kstem

5.6.2. Using dictionary: with hunspell token filter

Prevent words from stemmed rule: using keyworkmarker filter before stemmer.

Override using stemmreoverride filter before stemmer

Kiem elasticsearch practical query book/tutorial to practice.

Nested type and nested query note, highlight search note.

**Chapter 6:**

Dealing with human language:

1. Fuzziness: Handling typo/misspelling

0,1,2 distance (can auto base on string length, recommend: 1)  
You will seldom use it directly yourself, but understanding how it works will help you to use fuzziness in the higher-level match query.  
Performance consideration with prefix\_length and max\_expansions

1. Synonym:

* Using file
* Add directly in database through analysis

Real index: chưa làm, sẽ làm sau, mapping và index sẽ chạy theo configure trên zenkins.

Chapter 8: Elasticsearch Java API

Running script vs directly with java