Getting Down and Dirty with Elasticsearch

@clintongormley

Berlin Buzzwords 2013

elasticsearch.

Elasticsearch

Elasticsearch

real time,
search and
analytics engine

Elasticsearch

real time,

search and

analytics engine

distributed

scales

massively

Elasticsearch

real time,

search and

analytics engine

distributed

scales

massively

Elasticsearch

high availability

real time, search and analytics engine

distributed

RESTful
API

scales
massively

Elasticsearch

high
availability

real time,
search and
analytics engine

distributed

elasticsearch.

RESTful
API

JSON
over HTTP

scales
massively

Elasticsearch

high
availability

real time,
search and
analytics engine

distributed

RESTful API

JSON over HTTP

scales massively

Elasticsearch

high availability

real time, search and analytics engine

distributed

schema free

RESTful API

JSON over HTTP

scales massively

Elasticsearch

high availability

real time, search and analytics engine

distributed

schema free

multi tenancy

open-source

RESTful API

JSON over HTTP

scales massively

Elasticsearch

high availability

real time, search and analytics engine

distributed

schema free

multi tenancy

open-source

RESTful API

JSON over HTTP

scales massively

Elasticsearch

Lucene based

high availability

real time, search and analytics engine

distributed

schema free

multi tenancy

Cool.

Cool. Bonsai cool...

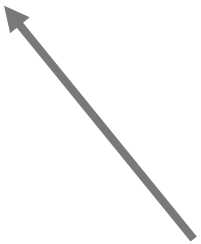
This is **WHY** we use it...

```
> ./bin/elasticsearch
> _
```

But **HOW** do we use it?

```
> curl -XGET localhost:9200/?pretty
```

```
> curl -XGET localhost:9200/?pretty
```

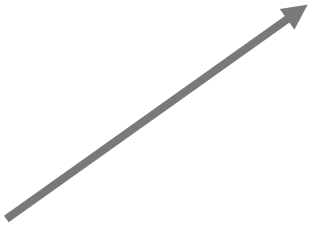


verb

```
> curl -XGET localhost:9200/?pretty
```

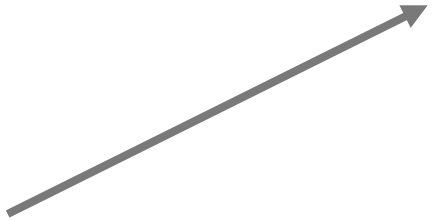
node

```
> curl -XGET localhost:9200/?pretty
```



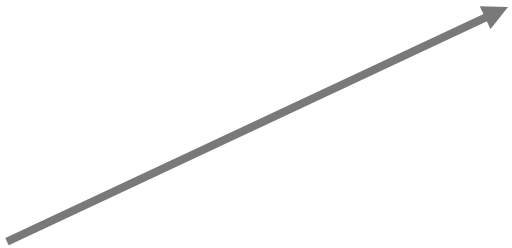
HTTP port

```
> curl -XGET localhost:9200/?pretty
```



path

```
> curl -XGET localhost:9200/?pretty
```



query string

```
> curl -XGET localhost:9200/?pretty
```

```
GET /
```

```
GET /
```

```
{
  "name"    : "Exploding Man",
  "tagline" : "You Know, for Search",
  "ok"      : true,
  "status"  : 200,
  "version" : {
    "number"          : "0.90.1",
   "snapshot_build" : false
 }
}
```

Where do we start?

With **data**

```
{
  "tweet": "I think #elasticsearch is AWESOME",
  "nick":  "@clintongormley",
  "name":  "Clinton Gormley",
  "date":  "2013-06-03",
  "rt"  :  5,
  "loc":   {
    "lat": 13.4,
    "lon": 52.5
  }
```

How to **put** it into ES?

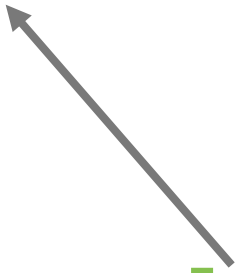
```
PUT /index/type/id
```

```
PUT /index/type/id
```

where?

```
PUT /myapp/type/id
```

```
PUT /myapp/type/id
```



what?

```
PUT /myapp/tweet/id
```

```
PUT /myapp/tweet/id
```

which?

```
PUT /myapp/tweet/1
```

```
PUT /myapp/tweet/1 -d '
{
  "tweet": "I think #elasticsearch is AWESOME",
  "nick":  "@clintongormley",
  "name":  "Clinton Gormley",
  "date":  "2013-06-03",
  "rt":    5,
  "loc":   {
    "lat": 13.4,
    "lon": 52.5
  }
}
'
```

```
# 201 CREATED

{
   "_index":    "myapp",
   "_type":     "tweet",
   "_id":       "1",
   "_version":   1,
   "ok":         true
}
```

Get

```
GET /myapp/tweet/1
```

```
# 200 OK

{
   "_index":    "myapp",
   "_type":     "tweet",
   "_id":       "1",
   "_version":   1,
   "exists":     true,
   "_source":    { ...OUR TWEET... }
}
```

Exists?

```
HEAD /myapp/tweet/1
```

```
HEAD /myapp/tweet/1   # 200 OK
```

```
HEAD /myapp/tweet/1   # 200 OK

HEAD /myapp/tweet/2   # 404 Not Found
```

Update

```
PUT /myapp/tweet/1 -d '
{
   "tweet": "I know #elasticsearch is AWESOME",
   "nick":  "@clintongormley",
   "name":  "Clinton Gormley",
   "date":  "2013-06-03",
   "rt":    5,
   "loc":   {
     "lat": 13.4,
     "lon": 52.5
   }
}
'
```

→ atomic DELETE & PUT

```
# 200 OK

{
   "_index":    "myapp",
   "_type":     "tweet",
   "_id":       "1",
   "_version":   2,
   "ok":         true
}
```

Delete

```
DELETE /myapp/tweet/1
```

```
# 200 OK

{
   "_index":    "myapp",
   "_type":     "tweet",
   "_id":       "1",
   "_version":   3,
   "ok":         true,
   "found":      true
}
```

Optimistic concurrency control

Optimistic concurrency control without locking

```
PUT /myapp/tweet/1?version=3 -d '
{
   ...
}
'

# 200 OK
```

```
PUT /myapp/tweet/1?version=2 -d '
{
   ...
}
'

# 409 Conflict
```

Update in place

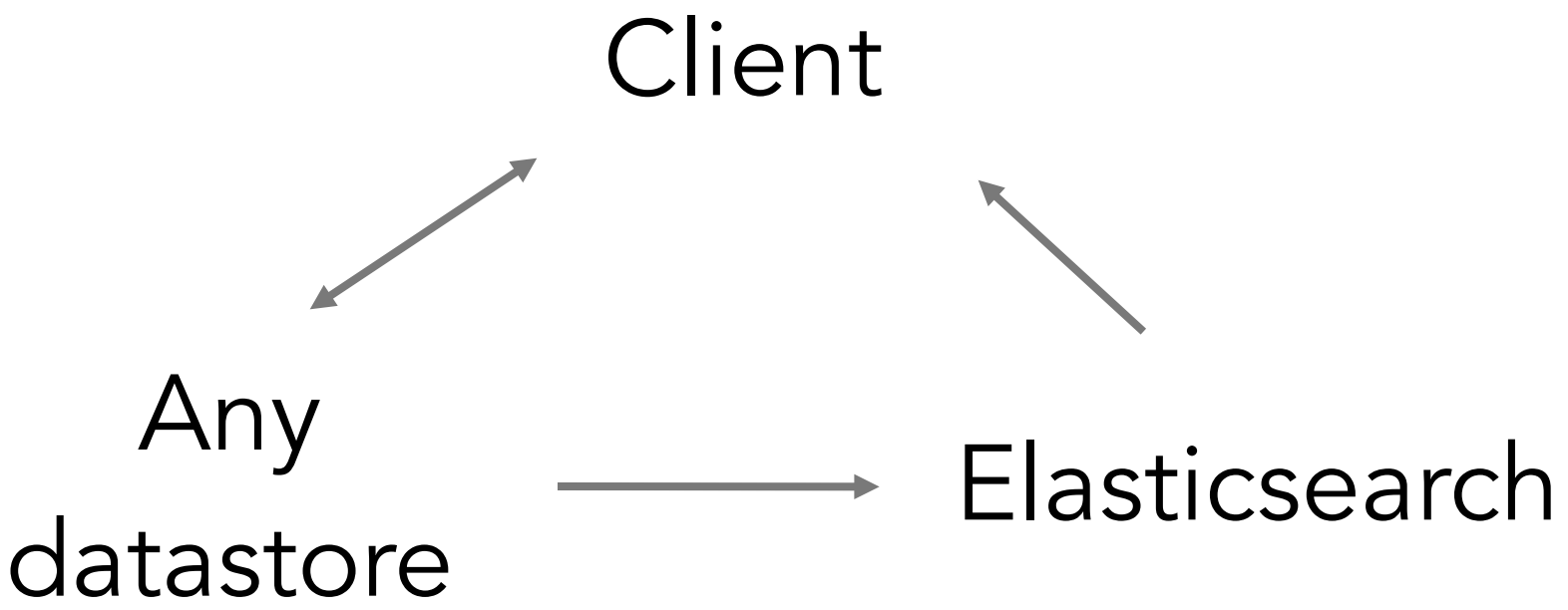
```
POST /myapp/tweet/1/_update -d '
{
 "script":                "ctx._source.count+=1",
 "retry_on_conflict":  3
}
'
```

```
POST /myapp/tweet/1/_update -d '
{
 "script":              "ctx._source.count+=1",
 "retry_on_conflict":  3
}
'
```

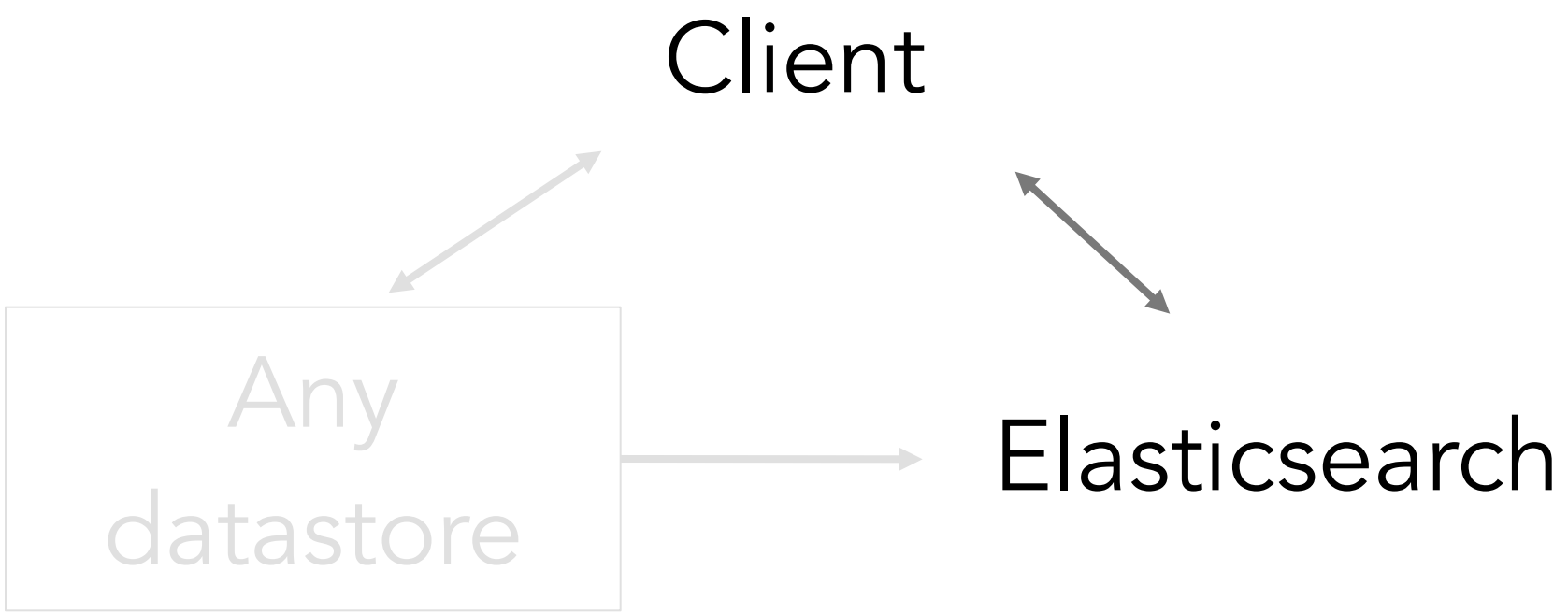
GET → change → PUT

Cheaper in bulk

Mirror external DB



Standalone



"Empty" Search

```
GET /_search
```

```
GET /_search
{
  "took" :          2,
}
```

```
GET /_search
{
  "took" :          2,
  "timed_out" :     false,




}
```

```
GET /_search
{
  "took" :          2,
  "timed_out" :     false,
  "_shards" : {
    "total" :       10,
    "successful" : 10,
    "failed" :      0
  },




}
```

GET /_search

```
{
  "took" :          2,
  "timed_out" :     false,
  "_shards" : {
    "total" :       10,
    "successful" : 10,
    "failed" :      0
  },
  "hits" : {
    "total" :       14,
    "max_score" :   1.0,
    "hits" :        [ { ... }]
  }
}
```

GET /_search

```
"hits" : [
  {
     "_index" :  "de",
     "_type" :   "tweet",
     "_id" :     "4",
     "_source" : { ... },
     "_score" :  1.0,
  },
  ...
]
```

Multi-index
Multi-type

```
GET /index/_search
```

```
GET /index/_search

GET /index1,index2/_search
```

```
GET /index/_search

GET /index1,index2/_search

GET /ind*/_search
```

```
GET /index/_search

GET /index1,index2/_search

GET /ind*/_search

GET /index/type/_search
```

```
GET /index/_search

GET /index1,index2/_search

GET /ind*/_search

GET /index/type/_search

GET /index/type1,type2/_search
```

```
GET /index/_search

GET /index1,index2/_search

GET /ind*/_search

GET /index/type/_search

GET /index/type1,type2/_search

GET /index/type*/_search
```

```
GET /index/_search

GET /index1,index2/_search

GET /ind*/_search

GET /index/type/_search

GET /index/type1,type2/_search

GET /index/type*/_search

GET /_all/type*/_search
```

Pagination

Pagination

`size` = num of results

Pagination

`size` = num of results

`from` = results to skip

```
GET /_search?size=5&from=0

GET /_search?size=5&from=5

GET /_search?size=5&from=10
```

Search *Lite*

Search *Lite*

```
GET /_search?q=name:john
```

```
+tweet:foo +name:john +date:>2013-05-01
```

```
+tweet:foo +name:john +date:>2013-05-01
```

→ percent encoding →

```
+tweet:foo +name:john +date:>2013-05-01
```

→ percent encoding →

```
?q=%2Btweet%3Afoo+%2Bname%3Ajohn+
    %2Bdate%3A%3E2013-05-01
```

```
GET /_search?q=mary
```

```
GET /_search?q=mary
```

→ user named "Mary"

→ tweets by "Mary"

→ tweet mentioning "@mary"

```
GET /_search?q=_all:mary
```

→ user named "Mary"

→ tweets by "Mary"

→ tweet mentioning "@mary"

_all field

string values from
all other fields

```
GET /_search?q=2013
→ 12 results
```

```
GET /_search?q=2013
→ 12 results

GET /_search?q=2013-06-03
→ 12 results!!
```

```
GET /_search?q=2013
→ 12 results

GET /_search?q=2013-06-03
→ 12 results!!

GET /_search?q=date:2013-06-03
→ 1 result
```

```
GET /_search?q=2013
→ 12 results

GET /_search?q=2013-06-03
→ 12 results!!

GET /_search?q=date:2013-06-03
→ 1 result

GET /_search?q=date:2013
→ 0 results!!
```

datatype differences?

check "mapping"

(field definitions)

```
GET /myapp/tweet/_mapping
```

GET /myapp/tweet/_mapping

```
{
  "tweet" : {
    "properties" : {
      "tweet" :   { "type" : "string" },
      "name" :    { "type" : "string" },
      "nick" :    { "type" : "string" },
      "date" :    { "type" : "date"   },
      "rt" :      { "type" : "long"   },
      "loc" : {
        "type": "object",
        "properties" : {
          "lat" : { "type" : "double" },
          "lon" : { "type" : "double" }
        }
      }
}}}
```

date = type:**date**

_all = type:**string**

Exact value vs Full text

Exact value vs Full text

```
10
4.5
2013-01-01
true
Foo
foo
```

Exact value vs Full text

```
10
4.5
2013-01-01
true
Foo
foo
```

```
The quick
brown fox
jumped
over the
lazy dog
```

Inverted index

"The quick brown fox jumped over the lazy dog"

"Quick brown foxes leap over lazy dogs in summer"

Inverted index

→ separate words / terms

```
"The quick brown fox jumped over the lazy dog"

"Quick brown foxes leap over lazy dogs in summer"
```

Inverted index

→ separate words / terms

`The,quick,brown,fox,jumped,over,the,lazy,dog`

`Quick,brown,foxes,leap,over,lazy,dogs,in,summer`

Inverted index

→ separate words / terms

→ sort unique terms

`The,quick,brown,fox,jumped,over,the,lazy,dog`

`Quick,brown,foxes,leap,over,lazy,dogs,in,summer`

Inverted index

→ separate words / terms

→ sort unique terms

```
The,brown,dog,fox,jumped,lazy,over,quick,the

Quick,brown,dogs,foxes,in,lazy,leap,over,summer
```

Inverted index

→ separate words / terms
→ sort unique terms
→ list docs containing terms

```
The,brown,dog,fox,jumped,lazy,over,quick,the

Quick,brown,dogs,foxes,in,lazy,leap,over,summer
```

Term	Doc 1	Doc 2
Quick	■	
The		■
brown	■	■
dog	■	
dogs		■
fox	■	
foxes		■
in		■
jumped	■	
lazy	■	■
leap		■
over	■	■
quick		■
summer		■
the	■	

Term	Doc 1	Doc 2
Quick		
The		
brown		
dog		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

q=quick brown

Term	Doc 1	Doc 2
Quick		
The		
brown		
dog		
dogs		
fox		
foxes		
in		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
Quick		
The		
brown		
dog		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

```
q=+Quick +foxes
```

Term	Doc 1	Doc 2
Quick		
The		
brown		
dog		
dogs		
fox		
foxes		
in		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
Quick		
The		
brown		
dog		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

No matches!

Improving recall

Term	Doc 1	Doc 2
Quick		
The		
brown		
dog		
dogs		
fox		
foxes		
in		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
brown		
dog		
dogs		
fox		
foxes		
in		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
brown		
dog		
dogs		
fox		
foxes		
in		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
brown		
dog		
fox		
in		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
brown		
dog		
fox		
in		
jumped		
lazy		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
brown		
dog		
fox		
in		
jump		
leap		
over		
quick		
summer		
the		

normalize terms

Term	Doc 1	Doc 2
brown		
dog		
fox		
in		
jump		
leap		
over		
quick		
summer		
the		

Term	Doc 1	Doc 2
brown		
dog		
fox		
jump		
lazy		
leap		
over		
quick		
summer		
the		

q=+Quick +foxes

Term	Doc 1	Doc 2
brown		
dog		
fox		
in		
jump		
leap		
over		
quick		
summer		
the		

normalize terms

in query too!

Term	Doc 1	Doc 2
brown		
dog		
fox		
jump		
lazy		
leap		
over		
quick		
summer		
the		

q=+Quick +foxes

Term	Doc 1	Doc 2
brown	■	■
dog	■	■
fox	■	■
jump	■	■
lazy	■	■
leap	■	■
over	■	■
quick	■	■
summer		■
the	■	■

```
q=+quick +foxes
```

Term	Doc 1	Doc 2
brown	■	■
dog	■	■
fox	■	■
jump	■	■
lazy	■	■
leap	■	■
over	■	■
quick	■	■
summer		■
the	■	■

```
q=+quick +fox
```

Term	Doc 1	Doc 2
brown		
dog		
fox		
in		
jump		
leap		
over		
quick		
summer		
the		

"Analysis"

"Analysis"

tokenization + normalization

"Analysers"

tokenizer + token filters

standard analyzer

```
"The Quick Brown Fox jumped
over the Lazy Dog!"
```

standard analyzer

→ standard tokenizer

```
"The Quick Brown Fox jumped
over the Lazy Dog!"
```

standard analyzer

→ standard tokenizer

`The,Quick,Brown,Fox,jumped,`

`over,the,Lazy,Dog`

standard analyzer

→ standard tokenizer

→ lowercase filter

```
The,Quick,Brown,Fox,jumped,
over,the,Lazy,Dog
```

standard analyzer

→ standard tokenizer

→ lowercase filter

```
the,quick,brown,fox,jumped,
over,the,lazy,dog
```

standard analyzer

→ standard tokenizer

→ lowercase filter

→ stopwords filter

```
the,quick,brown,fox,jumped,
over,the,lazy,dog
```

standard analyzer

→ standard tokenizer

→ lowercase filter

→ stopwords filter

```
,quick,brown,fox,jumped,
  over,    ,lazy,dog
```

english analyzer

→ standard tokenizer

→ lowercase filter

```
the,quick,brown,fox,jumped,
over,the,lazy,dog
```

english analyzer

→ standard tokenizer

→ lowercase filter

→ english stemmer

```
the,quick,brown,fox,jumped,
over,the,lazy,dog
```

english analyzer

→ standard tokenizer

→ lowercase filter

→ english stemmer

`the,quick,brown,fox,jumped,`

`over,the,lazy,dog`

english analyzer

→ standard tokenizer

→ lowercase filter

→ english stemmer

```
the,quick,brown,fox,jump,
over,the,lazy,dog
```

english analyzer

→ standard tokenizer

→ lowercase filter

→ english stemmer

→ english stopwords

```
the,quick,brown,fox,jump,
over,the,lazy,dog
```

english analyzer

→ standard tokenizer
→ lowercase filter
→ english stemmer
→ english stopwords

```
,quick,brown,fox,jump,
 over,    ,lazy,dog
```

date = type:**date**

_all = type:**string**

date = exact value

_all = full text

date = 2013-06-03

_all = 2013,06,03

```
GET /_search?q=2013
→ 12 results
```

```
GET /_search?q=2013
→ 12 results

GET /_search?q=2013-06-03
→ 12 results
```

```
GET /_search?q=2013
→ 12 results

GET /_search?q=2013 OR 06 OR 03
→ 12 results
```

```
GET /_search?q=2013
→ 12 results

GET /_search?q=2013-06-03
→ 12 results

GET /_search?q=date:2013-06-03
→ 1 result
```

```
GET /_search?q=2013
→ 12 results

GET /_search?q=2013-06-03
→ 12 results

GET /_search?q=date:2013-06-03
→ 1 result

GET /_search?q=date:2013
→ 0 results
```

Field mapping

Core field types

Strings:	string
Datetimes:	date
Whole numbers:	byte, short, integer, long
Floats:	float, double
Booleans:	boolean
Objects:	object

Core field types

Strings:	string
Datetimes:	date
Whole numbers:	byte, short, integer, long
Floats:	float, double
Booleans:	boolean
Objects:	object

Also: multi_field, ip, geo_point, geo_shape,

Dynamic detection

`"foo bar"`	**string**
`"2013-01-01"`	**date**
`10`	byte, short, integer, **long**
`10.0`	float, **double**
`true`	**boolean**
`{ foo: "bar" }`	**object**

Dynamic detection

`"foo bar"`	**string**
`"2013-01-01"`	**date**
`10`	byte, short, integer, **long**
`10.0`	float, **double**
`true`	**boolean**
`{ foo: "bar" }`	**object**
`["foo","bar"]`	No special mapping. Any field can have multi-vals

Most important: `type`

```
{
  "tweet" : {
    "properties" : {
      "tweet" :    { "type" : "string"     },
      "name" :     { "type" : "string"     },
      "nick" :     { "type" : "string"     },
      "date" :     { "type" : "date"       },
      "rt" :       { "type" : "long"       },
      "loc" : {
        "type": "object",
        "properties" : {
          "lat" : { "type" : "double"     },
          "lon" : { "type" : "double"     }
        }
      }

}}}
```

```
{
  "tweet" : {
    "properties" : {
      "tweet" :    { "type" : "string"    },
      "name" :     { "type" : "string"    },
      "nick" :     { "type" : "string"    },
      "date" :     { "type" : "date"      },
      "rt" :       { "type" : "long"      },
      "loc" :      { "type" : "geo_point" }
}}}
```

Full text vs Exact string

Full text: (default)

```
{ "type": "string", "index": "analyzed" }
```

Full text: (default)

```
{ "type": "string", "index": "analyzed" }
```

Exact string:

```
{ "type": "string", "index": "not_analyzed" }
```

Full text: (default)

```
{ "type": "string", "index": "analyzed" }
```

Exact string:

```
{ "type": "string", "index": "not_analyzed" }
```

Not searchable:

```
{ "type": "string", "index": "no" }
```

```
{
  "tweet" : {
    "properties" : {
      "tweet" :    { "type" : "string"    },
      "name" :     { "type" : "string"    },
      "nick" :     { "type" : "string"    },
      "date" :     { "type" : "date"      },
      "rt" :       { "type" : "long"      },
      "loc" :      { "type" : "geo_point" }
}}}
```

```
{
  "tweet" : {
    "properties" : {
      "tweet" :    { "type" : "string"     },
      "name" :     { "type" : "string"     },
      "nick" :     {
          "type"   : "string",
          "index" : "not_analyzed"
      },
      "date" :     { "type" : "date"       },
      "rt" :       { "type" : "long"       },
      "loc" :      { "type" : "geo_point" }
}}}
```

Analyzer

```
{
  "tweet" : {
    "properties" : {
      "tweet" :   { "type" : "string"    },
      "name" :    { "type" : "string"    },
      "nick" :    {
          "type"  : "string",
          "index" : "not_analyzed"
      },
      "date" :    { "type" : "date"      },
      "rt" :      { "type" : "long"      },
      "loc" :     { "type" : "geo_point" }
}}}
```

```
{
  "tweet" : {
    "properties" : {
      "tweet" :   {
          "type" :     "string",
          "analyzer" : "english"
      },
      "name" :    { "type" : "string"    },
      "nick" :    {
          "type"  : "string",
          "index" : "not_analyzed"
      },
      "date" :    { "type" : "date"      },
      "rt" :      { "type" : "long"      },
      "loc" :     { "type" : "geo_point" }
}}}
```

Updating mappings

Can: add new fields

Can: add new fields

```
PUT /myapp/tweet/_mapping -d '
{
    "tweet": {
        "properties": {
            ...
        }
    }
}
'
```

Cannot: change fields

Cannot: change fields

```
DELETE /myapp
```

Cannot: change fields

```
PUT /myapp -d '
{
    "mappings": {
        "tweet": {
            "properties": {
                ...
            }
        }
    }
}
'
```

Full body search

```
GET /_search -d '
{
    "query": {
        "match_all": {}
    },
    "from":  0,
    "size":  10
}
'
```

```
GET /_search -d '
{
    "query": {
        "match_all": {}
    },
    "from":  0,
    "size":  10
}
'
```

Query DSL

rich flexible query language

```
{
    "match": { "tweet": "search" }
}
```

```
GET /_search -d '{
{
    "query": {
        "match": { "tweet": "search" }
    }
}
```

Filters vs Queries

Filters vs Queries

`exact matching` `full text search`

Filters vs Queries

exact matching	full text search
binary yes/no	relevance scoring

Filters vs Queries

Filters	Queries
exact matching	full text search
binary yes/no	relevance scoring
fast	heavier

Filters vs Queries

Filters	Queries
exact matching	full text search
binary yes/no	relevance scoring
fast	heavier
cacheable	not cacheable

Combine filter & query

```
Query:   { "match": { "tweet": "search" }}

Filter:  { "term":  { "nick":  "@mary" }}
```

Combine filter & query

```
{
    "filtered": {
        "query": {
            "match": { "tweet": "search" }
        },
        "filter": {
            "term":  { "nick":  "@mary" }
        }
    }
}
```

Combine filter & query

```
GET /_search -d '
{
   "query": {
      "filtered": {
         "query": {
            "match": { "tweet": "search" }
         },
         "filter": {
            "term":  { "nick":  "@mary" }
         }
      }
   }
}
'
```

Just a filter

```
GET /_search -d '
{
   "query": {
      "filtered": {
         "query": {
            "match_all": {}
         },
         "filter": {
            "term":  { "nick":  "@mary" }
         }
      }
   }
}
'
```

Just a filter

```
GET /_search -d '
{
   "query": {
      "filtered": {
         "filter": {
            "term":  { "nick":  "@mary" }
         }
      }
   }
}
'
```

User's tweets by date

```
GET /_search -d '
{
   "query": {
      "filtered": {
         "filter": {
            "term":  { "nick":  "@mary" }
         }
      }
   },
   "sort": { "date": "desc" }
}
'
```

Tweets for last month

```
GET /_search -d '
{
  "query": {
     "filtered": {
        "filter": {
           "range":  {
             "date": {
               "gte": "2013-05-01",
               "lt":  "2013-06-01"
             }
           }
        }
}}}
'
```

Top tweeters

```
GET /_all/tweet/_search -d '
{
   "facets": {
       "top_tweeters": {
           "terms": {
               "field": "nick"
           }
       }
   }
}
'
```

Top tweeters for query

```
GET /_all/tweet/_search -d '
{
  "facets": {
      "top_tweeters": {
          "terms": {
              "field": "nick"
          }
      }
  },
  "query": {
      "match": { "tweet": "elasticsearch" }
  }
}
'
```

Tweets by month

```
GET /_all/tweet/_search -d '
{
  "facets": {
      "tweets_by_month": {
          "date_histogram": {
              "field":    "date",
              "interval": "month"
          }
      }
   }
}
'
```

Autocomplete

```
{ "match": { "name": "joh" }}
```

↓

John Smith

Johnny Depp

Lyndon **Joh**nson

Autocomplete

```
{ "match": { "name": "joh" }}
```

↓

John Smith

Johnny Depp

Lyndon **Joh**nson

But "joh" doesn't exist in the index

Autocomplete

N-grams == window-on-a-word:

Autocomplete

N-grams == window-on-a-word:

```
Length 1: j,o,h,n,s,m,i,t,h
```

Autocomplete

N-grams == window-on-a-word:

```
Length 1: j,o,h,n,s,m,i,t,h
Length 2: jo,oh,hn,sm,mi,it,th
```

Autocomplete

N-grams == window-on-a-word:

```
Length 1: j,o,h,n,s,m,i,t,h
Length 2: jo,oh,hn,sm,mi,it,th
Length 3: joh,ohn,smi,mit,ith
Length 4: john,smit,mith
```

Autocomplete

N-grams == window-on-a-word:

```
Length 1: j,o,h,n,s,m,i,t,h
Length 2: jo,oh,hn,sm,mi,it,th
Length 3: joh,ohn,smi,mit,ith
Length 4: john,smit,mith
```

Good for partial word matching

Autocomplete

Edge N-grams == anchored N-grams:

Autocomplete

Edge N-grams == anchored N-grams:

```
j
jo
joh
john
s
sm
smi
smit
smith
```

Autocomplete

Edge N-grams == anchored N-grams:

```
j
jo
joh
john
s
sm
smi
smit
smith
```

⇒ **Perfect for autocomplete**

Edge N-Gram token filter

```
{
  "filter": {
    "autocomplete": {
      "type":     "edge_ngram",
      "min_gram": 1,
      "max_gram": 20
    }
  }
}
```

Name field analyzers

```
{
  "analyzer": {
    "name": {
      "type":      "standard",
      "stopwords": []
    },
  }
}
```

Name field analyzers

```
{
  "analyzer": {
    "name": {
      "type":      "standard",
      "stopwords": []
    },
    "name_autocomplete": {
      "type":      "custom",
      "tokenizer": "standard",
      "filter":   ["lowercase","autocomplete"]
    }
  }
}
```

Name field mapping

```
{
  "name": {
    "type": "string"
  }
}
```

Name field mapping

```
{
  "name": {
    "type": "string"
  }
}
```

multi_field == one field, multi-purposes

Name field mapping

```
{
  "name": {
    "type":   "multi_field",
    "fields": {
      "name": {


      },
      "autocomplete": {



      }
}}
```

Name field mapping

```
{
  "name": {
    "type":   "multi_field",
    "fields": {
      "name": {

      },
      "autocomplete": {



      }
}}
```

Main field:

"**name**" or "**name.name**"

Name field mapping

```
{
  "name": {
    "type":   "multi_field",
    "fields": {
      "name": {
        "type":              "string",
        "analyzer":          "name"
      },
      "autocomplete": {



      }
}}
```

Name field mapping

```
{
  "name": {
    "type":   "multi_field",
    "fields": {
      "name": {
        "type":              "string",
        "analyzer":          "name"
      },
      "autocomplete": {

      }
}}
```

Sub field:

"name.autocomplete"

Name field mapping

```
{
  "name": {
    "type":   "multi_field",
    "fields": {
      "name": {
        "type":            "string",
        "analyzer":        "name"
      },
      "autocomplete": {
        "type":            "string",
        "index_analyzer":  "name_autocomplete",
        "search_analyzer": "name"
      }
}}
```

Recreate the index

```
DELETE /myapp
```

Recreate the index

```
PUT /myapp -d '
{
  "settings": {
    "analysis": {
      "analyzer":  {...},
      "filter":    {...}
    }
  },




}
```

Recreate the index

```
PUT /myapp -d '
{
  "settings": {
    "analysis": {
      "analyzer":  {...},
      "filter":    {...}
    }
  },
  "mappings": {
    "tweet": {
      "properties": {...}
    }
  }
}
```

Autocomplete query

```
{
  "match": {
    "name.autocomplete": "john smi"
  }
}
```

Autocomplete query

```
{
  "match": {
    "name.autocomplete": "john smi"
  }
}
```

Better: favor whole word matches

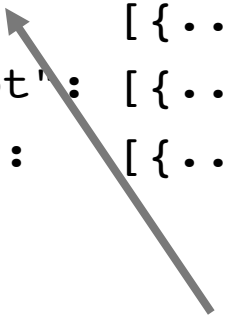
Autocomplete query

```
{
  "bool": {
    "must":     [{...},{...}],
    "must_not": [{...},{...}],
    "should":   [{...},{...}]
  }
}
```

Combines multiple query clauses

Autocomplete query

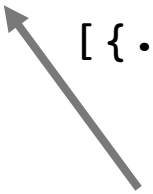
```
{
  "bool": {
    "must":     [{...},{...}],
    "must_not": [{...},{...}],
    "should":   [{...},{...}]
  }
}
```



MUST match

Autocomplete query

```
{
  "bool": {
    "must":     [{...},{...}],
    "must_not": [{...},{...}],
    "should":   [{...},{...}]
  }
}
```



MUST NOT match

Autocomplete query

```
{
  "bool": {
    "must":     [{...},{...}],
    "must_not": [{...},{...}],
    "should":   [{...},{...}]
  }
}
```

"More relevant" if these match

Autocomplete query

```
{
  "bool": {
    "must": {
      "match": {
        "name.autocomplete": "john smi"
      }
    },
    "should": {
      "match": {
        "name": "john smi"
      }
    }
```

Boost popular tweets

```
{
  "custom_score_query": {
    "query":  { "match": { "tweet": "search" }},
    "script": "_score * (1+log(doc['rt'].value))"
  }
}
```

Filter local tweets

```
{
  "filtered": {
    "query":  { "match": { "tweet": "search" }},
    "filter": {
      "geo_distance": {
        "distance": "100km",
        "loc": {
          "lat": 13.4,
          "lon": 52.5
        }
      }
    }
```

Boost local tweets

```
{
  "custom_filters_score_query": {
    "query":   { "match": { "tweet": "search" }},
    "filters": [

    ]
}}
```

Boost local tweets

```
{
  "custom_filters_score_query": {
    "query":   { "match": { "tweet": "search" }},
    "filters": [
      {
        "boost": 2,
        "filter": {
          "geo_distance": {
            "distance": "100km",
            "loc": { "lat": 13.4, "lon": 52.5 }
      }}}
    ]
}}
```

www.elasticsearch.org