



Documents are Immutable

- Elasticsearch documents are immutable (!)
- We actually *replaced* documents in this lecture
- The Update API did some things for us, making it *look* like we updated documents
- The Update API is simpler and saves some network traffic



How update API Works

- The current document is retrieved
- The field values are changed
- The existing document is *replaced* with the modified document
- We *could* do the exact same thing at the application level



Managing Documents – Adding Documents

Kibana Dev Tools

Console Search Profiler

```
1 POST /product/default
2 {
3   "name": "Processing Events with Logstash",
4   "instructor": {
5     "firstName": "Bo",
6     "lastName": "Andersen"
7   }
8 }
```

▶ 🔍

```
1 {
2   "_index": "product",
3   "_type": "default",
4   "_id": "AV0I2WNjswE-NdQiooh2",
5   "_version": 1,
6   "result": "created",
7   "_shards": {
8     "total": 2,
9     "successful": 1,
10    "failed": 0
11  },
12  "created": true
13 }
```

...

Udemy



Managing Documents – Adding Documents

Kibana
Adding documents
Section 4, Lecture 27

Discover Visualize Dashboard Timelion Machine Learning Graph Dev Tools Management

Go to Dashboard

Dev Tools

Console Search Profiler

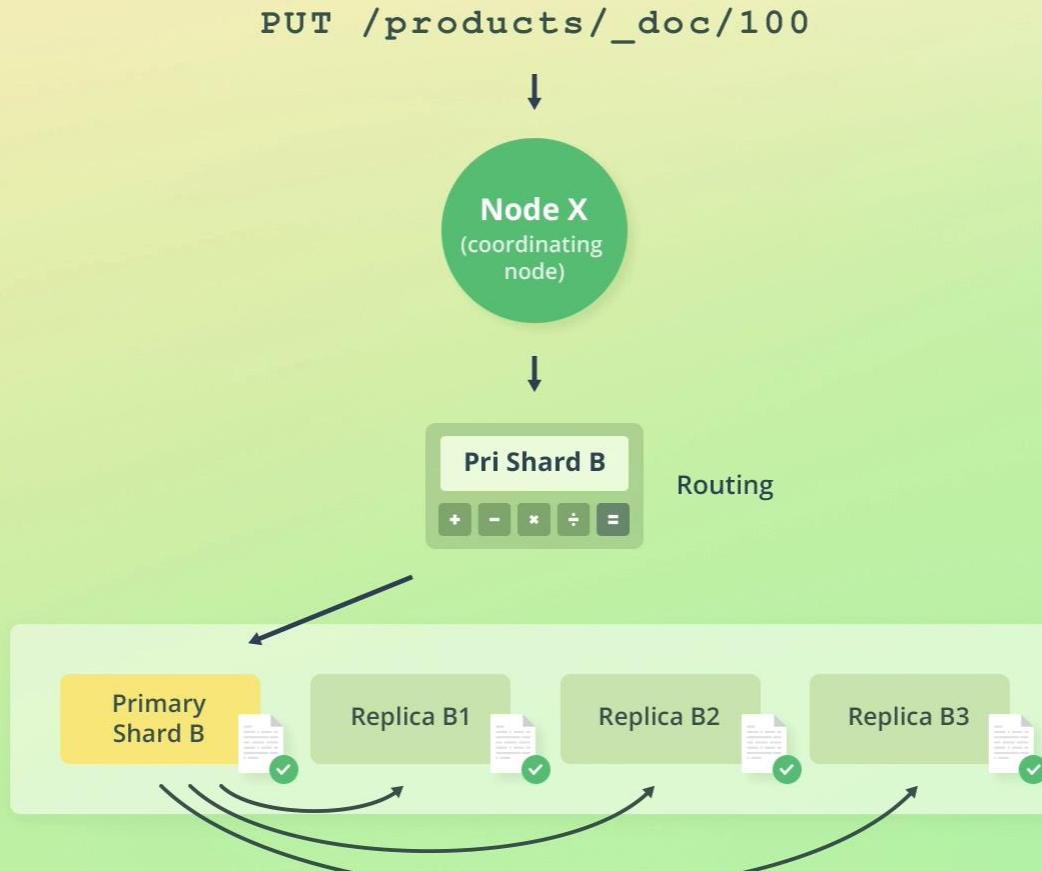
```
1 | POST /product/default
2 | {
3 |   "name": "Processing Events with Logstash",
4 |   "instructor": {
5 |     "firstName": "Bo",
6 |     "lastName": "Andersen"
7 |   }
8 |
9 |
10 | PUT /product/default/_1
11 | {
12 |   "name": "Complete Guide to Elasticsearch",
13 |   "instructor": {
14 |     "firstName": "Bo",
15 |     "lastName": "Andersen"
16 |   }
17 | }
```

A teal arrow points from the highlighted `PUT` command in the console to the corresponding JSON response in the results pane.

```
1 | [
2 |   {
3 |     "_index": "product",
4 |     "_type": "default",
5 |     "_id": "1",
6 |     "_version": 1,
7 |     "result": "created",
8 |     "_shards": {
9 |       "total": 2,
10 |       "successful": 1,
11 |       "failed": 0
12 |     },
13 |     "created": true
14 |   }
15 | ]
```



Writing Documents





GET /products/_doc/100



Node X
(coordinating node)

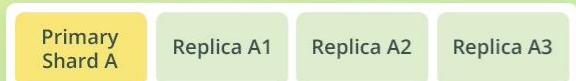


Rep Group B
+ - × ÷ =

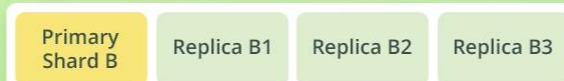
Routing



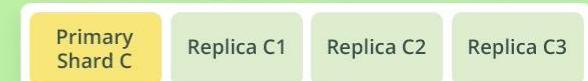
Replication group A



Replication group B



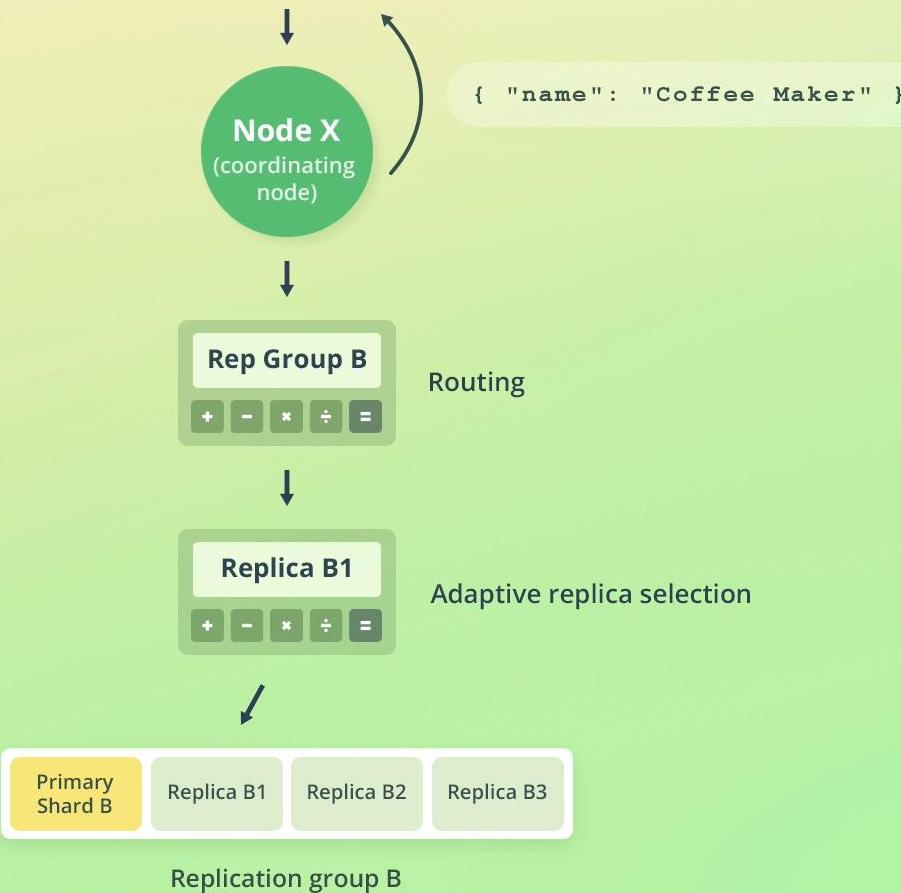
Replication group C





Reading Documents

GET /products/_doc/100





Managing Documents – Retrieving Documents

Kibana Dev Tools

Console Search Profiler

1 GET /product/default/1

```
1 {  
2   "_index": "product",  
3   "_type": "default",  
4   "_id": "1",  
5   "_version": 1,  
6   "found": true,  
7   "_source": {  
8     "name": "Complete Guide to Elasticsearch",  
9     "instructor": {  
10       "firstName": "Bo",  
11       "lastName": "Andersen"  
12     }  
13   }  
14 }
```

...

Udemy



Managing Documents – Replace Documents

Kibana Dev Tools

Console Search Profiler

```
1 PUT /product/default/1
2 {
3   "name": "Complete Guide to Elasticsearch",
4   "instructor": {
5     "firstName": "Bo",
6     "lastName": "Andersen"
7   },
8   "price": 195
9 }
10
11
12 GET /product/default/1| ➤ ↻
```

```
1 {
2   "_index": "product",
3   "_type": "default",
4   "_id": "1",
5   "_version": 2,
6   "found": true,
7   "_source": {
8     "name": "Complete Guide to Elasticsearch",
9     "instructor": {
10       "firstName": "Bo",
11       "lastName": "Andersen"
12     },
13     "price": 195
14   }
15 }
```

...

Udemy



Managing Documents – Update Documents

kibana

Dev Tools

Console Search Profiler

```
1 POST /product/default/1/_update
2 {
3   "doc": { "price": 95, "tags": [ "Elasticsearch" ] }
4 }
5
6 GET /product/default/1|
```

▶ 🔍

1 | {
2 | "_index": "product",
3 | "_type": "default",
4 | "_id": "1",
5 | "_version": 3,
6 | "found": true,
7 | "_source": {
8 | "name": "Complete Guide to Elasticsearch",
9 | "instructor": {
10 | "firstName": "Bo",
11 | "lastName": "Andersen"
12 | },
13 | "price": 95,
14 | "tags": [
15 | "Elasticsearch"
16 |]
17 | }
18 | }

⋮

Udemy



Managing Documents – Scripting Documents

Kibana Dev Tools

Console Search Profiler

```
1 POST /product/default/1/_update
2 {
3     "script": "ctx._source.price += 10"
4 }
5
6 GET /product/default/1
```

▶ 🔍

```
1 {
2     "_index": "product",
3     "_type": "default",
4     "_id": "1",
5     "_version": 4,
6     "found": true,
7     "_source": {
8         "name": "Complete Guide to Elasticsearch",
9         "instructor": {
10             "firstName": "Bo",
11             "lastName": "Andersen"
12         },
13         "price": 105,
14         "tags": [
15             "Elasticsearch"
16         ]
17     }
18 }
```

...

Udemy



Managing Documents – Scripting Documents

elasticsearch Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

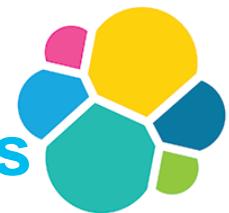
History Settings Help

200 - OK 19 ms

```
1 POST /customers/_update/100
2 {
3   "script": 
4     | "ctx._source.customerId+=Math.random()*100000"
5
6 }
7
8 GET /customers/_doc/100
```

GET method
PUT method
POST method
DELETE method

```
1 #! Elasticsearch built-in security features are not enabled.
2 Without authentication, your cluster could be accessible to
3 anyone. See https://www.elastic.co/guide/en/elasticsearch
4 /reference/7.15/security-minimal-setup.html to enable security.
5
6 {
7   "_index" : "customers",
8   "_type" : "_doc",
9   "_id" : "100",
10  "_version" : 13,
11  "_seq_no" : 12,
12  "_primary_term" : 1,
13  "found" : true,
14  "_source" : {
15    "customerId" : 438816918,
16    "firstName" : "Vignesh Manickam",
17    "lastName" : "Bala",
18    "dob" : "1995-12-07",
19    "status" : "active"
20  }
21 }
```



Managing Documents – Upserting Documents

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1  DELETE /member/default/1
2
3  POST /member/default/1/_update
4  {
5    "script": "ctx._source.fees+=500",
6    "upsert": {
7      "fees": 2000
8    }
9  }
10
11 GET /member/default/1| ➡️ 🔒 ⚙️ ⋮
12
13 POST /member/default/1/_update
14 {
15   "script": "ctx._source.fees+=500",
16   "upsert": {
17     "fees": 2000
18   }
19 }
```

History Settings Help

The screenshot shows the Kibana Dev Tools interface. On the left, there's a sidebar with icons for Discover, Visualize, Dashboard, Timelion, APM, Dev Tools (which is selected), Monitoring, and Management. The main area is titled 'Dev Tools' and contains three tabs: 'Console', 'Search Profiler', and 'Grok Debugger'. The 'Console' tab is active, displaying a sequence of Elasticsearch operations. It starts with a DELETE request to remove document ID 1 from the 'member/default' index. This is followed by a POST request to update the document. The update script increments the 'fees' field by 500. An 'upsert' block is present, which defines a new document to be inserted if the update fails. This upsert document has a 'fees' value of 2000. After the update, a GET request is shown to retrieve the document. The response on the right side shows the updated document with an '_index' of 'member', '_type' of 'default', '_id' of '1', '_version' of 7, 'found' as true, and '_source' containing a 'fees' value of 2500.



Managing Documents – Upserting Documents

Screenshot of the Elasticsearch Dev Tools Console interface showing an upsert operation on a document.

The Dev Tools sidebar shows various icons for different features like History, Settings, and Help.

The main area displays the following code:

```
1 POST /bookdb_index/book/3/_update
2 {
3     "script": "ctx._source.title=ctx._source.title
4         .toUpperCase()",
4     "upsert": {
5         "title": "actions"
6     }
7 }
8
9
10
11
12
13 GET /bookdb_index/book/3
```

The response from the POST request (line 13) is shown on the right:

```
1 #! Deprecation: [types removal] Specifying types in document get requests is
2 #! deprecated, use the("/{index}"/_doc/{id}) endpoint instead.
3
4 {
5     "_index": "bookdb_index",
6     "_type": "book",
7     "_id": "3",
8     "_version": 11,
9     "_seq_no": 37,
10    "_primary_term": 1,
11    "found": true,
12    "_source": {
13        "summary": "build scalable search applications using Elasticsearch without
14            having to do complex low-level programming or understand advanced data
15            science algorithms",
16        "publisher": "manning",
17        "num_reviews": 259,
18        "title": "ELASTICSEARCH IN ACTION",
19        "publish_date": "2015-12-03",
20        "authors": [
21            "radu gheorge",
22            "matthew lee hinman",
23            "roy russo"
24        ]
25    }
26}
```

A preview of the document is shown on the right side of the interface.



Managing Documents – Deleting Documents

Complete Guide to Elasticsearch Elastic Kibana

localhost:5601/app/kibana#/dev_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 251 ms

```
1 DELETE /products/_doc/123 [ ] {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "123",  
5   "_version" : 6,  
6   "result" : "deleted",  
7   "_shards" : {  
8     "total" : 3,  
9     "successful" : 1,  
10    "failed" : 0  
11  },  
12  "_seq_no" : 5,  
13  "_primary_term" : 1  
14 }  
15 }
```

Type here to search

Windows taskbar icons: File Explorer, Task View, Microsoft Edge, Google Chrome, Microsoft Word, Microsoft Powerpoint, Microsoft Word, Microsoft Excel, Microsoft Word, Microsoft Word.

System tray: ENG IN 02/06/2020 01:30 21



Managing Documents – Deleting Documents

Kibana Dev Tools

Console Search Profiler

```
1 | DELETE /product/default/1
2 |
3 | POST /product/default
4 | {
5 |   "name": "Processing Events with Logstash",
6 |   "category": "course"
7 | }
8 |
9 | POST /product/default
10| {
11|   "name": "The Art of Scalability",
12|   "category": "book"
13| }
14|
15| POST /product/_delete_by_query
16| {
17|   "query": {
18|     "match": {
19|       "category": "book"
20|     }
21|   }
22| }
```

▶ 🔍

```
1 | {
2 |   "took": 65,
3 |   "timed_out": false,
4 |   "total": 1,
5 |   "deleted": 1,
6 |   "batches": 1,
7 |   "version_conflicts": 0,
8 |   "noops": 0,
9 |   "retries": {
10|     "bulk": 0,
11|     "search": 0
12|   },
13|   "throttled_millis": 0,
14|   "requests_per_second": -1,
15|   "throttled_until millis": 0,
16|   "failures": []
17| }
```

...

Udemy



Managing Documents – Delete By Index

Kibana Dev Tools

Console Search Profiler

1 DELETE /product

2 { "acknowledged": true }

3 }

Discover Visualize Dashboard Timelion Machine Learning Graph Dev Tools Management

Udemy



Introduction to Routing

- How does Elasticsearch know where to store documents?
- How are documents found once they have been indexed?
- The answer is *routing*
- Routing is the process of resolving a shard for a document



Introduction to Routing

```
shard_num = hash(_routing) % num_primary_shards
```



Introduction to Routing

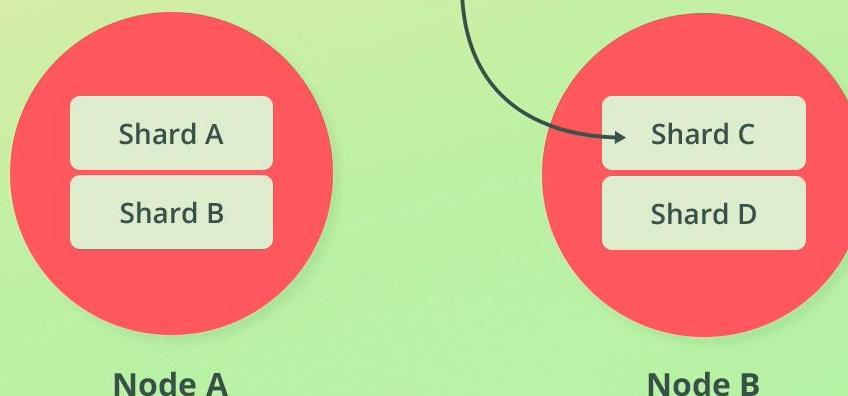
GET /products/_doc/100



```
shard_num = hash(_routing) % num_primary_shards
```

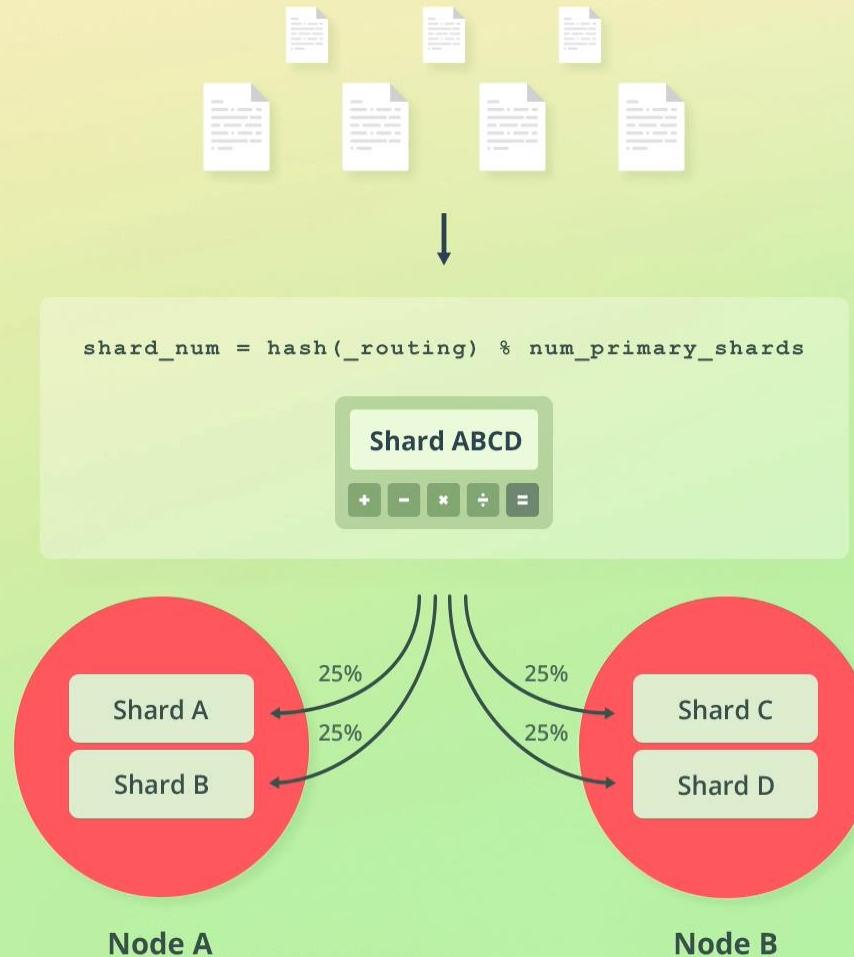
Shard C

+ - * ÷ =





Introduction to Routing





Introduction to Routing





Introduction to Routing

GET /products/_doc/100



```
shard_num = hash(_routing) % num_primary_shards
```

5





Introduction to Read Document

```
GET /products/_doc/100
```

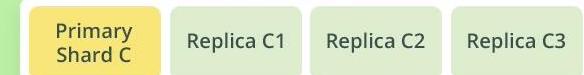
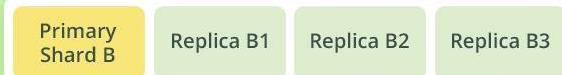
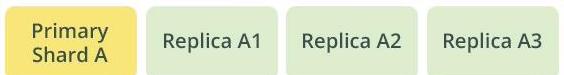
↓
Node X
(coordinating node)

↓
Rep Group B
Routing
+ - × ÷ =

Replication group A

Replication group B

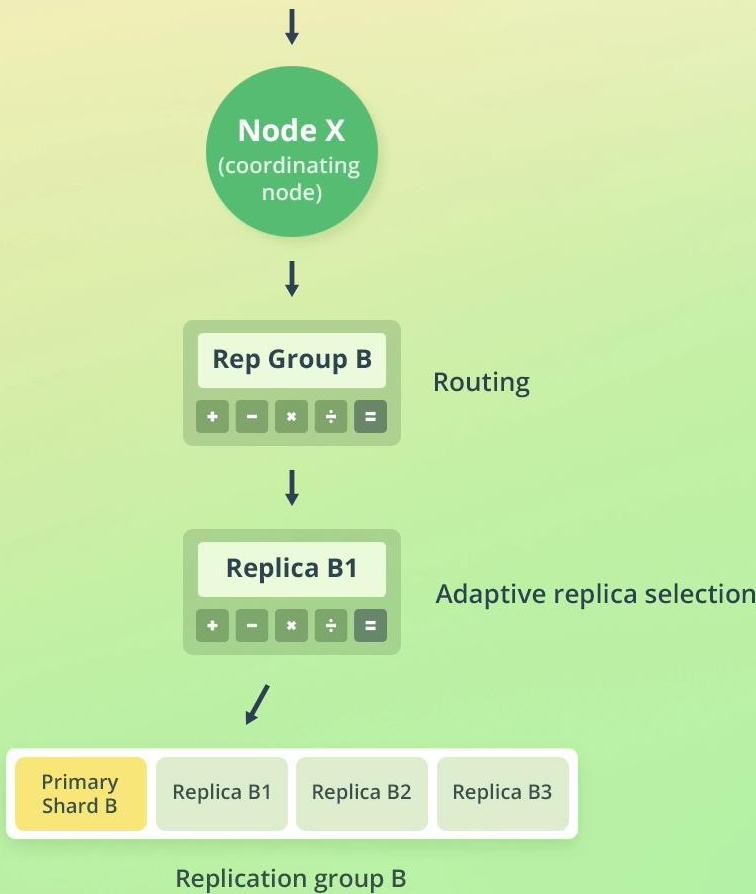
Replication group C





Introduction to Read Document

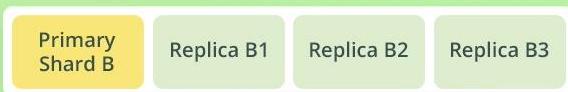
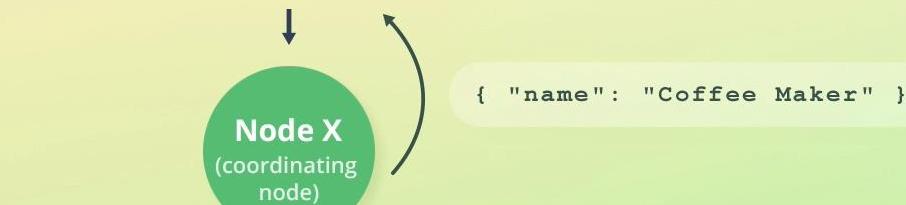
GET /products/_doc/100





Introduction to Read Document

GET /products/_doc/100





Introduction to Routing

Lecture summary

- A read request is received and handled by a *coordinating node*
- Routing is used to resolve the document's replication group
- ARS is used to send the query to the best available shard
 - ARS is short for *Adaptive Replica Selection*
 - ARS helps reduce query response times
 - ARS is essentially an intelligent load balancer
- The coordinating node collects the response and sends it to the client



Writing Documents

```
PUT /products/_doc/100
```



Node X
(coordinating node)



Pri Shard B
+ - × ÷ =

Routing

Valid!



Primary
Shard B

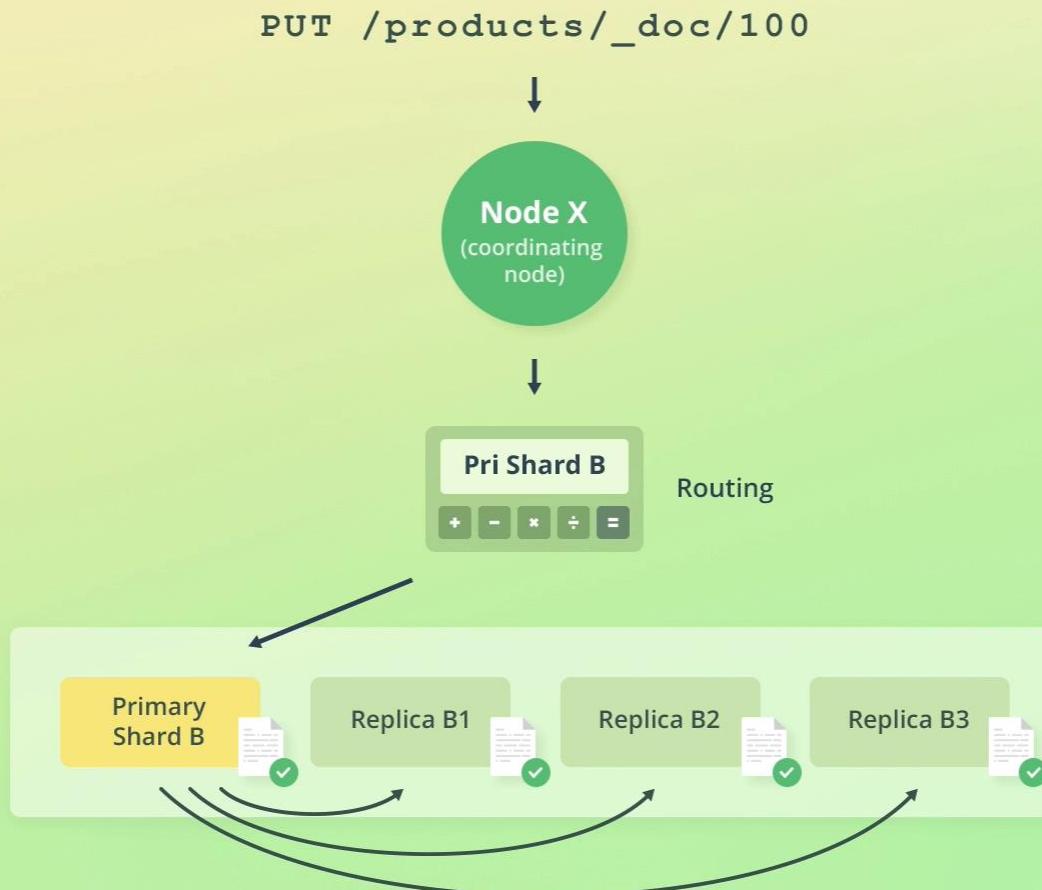
Replica B1

Replica B2

Replica B3



Writing Documents





Multiple Update

- We already covered how to update *one* document at a time
- Let's now update *multiple* documents within a single query
 - Similar to an UPDATE WHERE query in a RDBMS
- The query uses three concepts that we have just covered
 - Primary terms
 - Sequence numbers
 - Optimistic concurrency control



Multiple Update

Apps Projects Gmail YouTube Maps Pluralsight

D Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 2292 ms

POST /products/_update_by_query

```
1 POST /products/_update_by_query
2 {
3   "script": {
4     "source": "ctx._source.qty++"
5   },
6   "query": {
7     "match_all": {}
8   }
9 }
10
11 }
```

1 {
2 "took" : 1400,
3 "timed_out" : false,
4 "total" : 2,
5 "updated" : 2,
6 "deleted" : 0,
7 "batches" : 1,
8 "version_conflicts" : 0,
9 "noops" : 0,
10 "retries" : {
11 "bulk" : 0,
12 "search" : 0
13 },
14 "throttled_millis" : 0,
15 "requests_per_second" : -1.0,
16 "throttled_until_millis" : 0,
17 "failures" : []
18 }
19



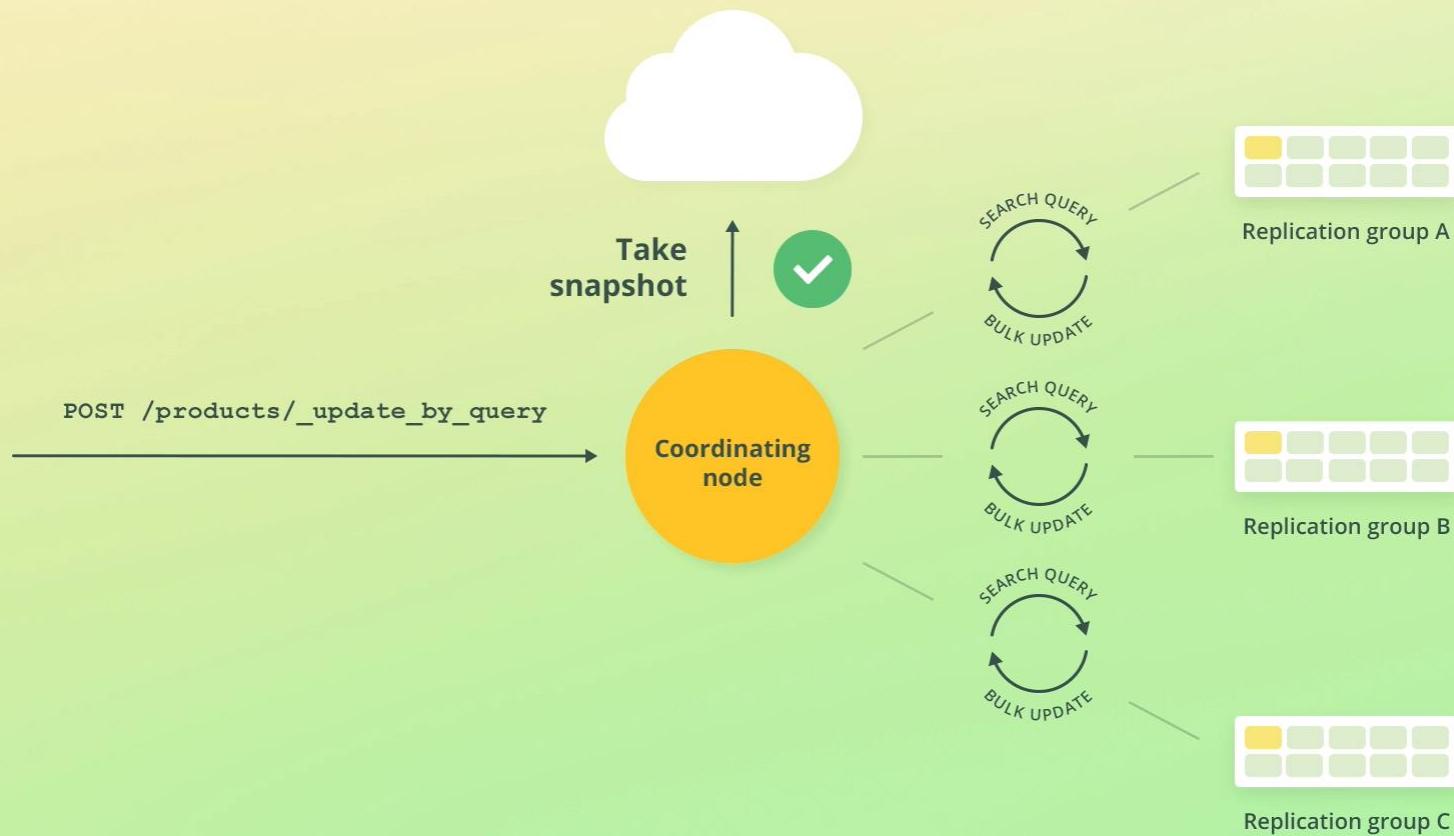
Multiple Update

```
localhost:9200/products/_search?pretty
Apps Projects Gmail YouTube Maps Pluralsight

{
  "total" : {
    "value" : 2,
    "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
    {
      "_index" : "products",
      "_type" : "_doc",
      "_id" : "dGJNcXIBxt9H8GTT2c0_",
      "_score" : 1.0,
      "_source" : {
        "productId" : 4974,
        "qty" : 90,
        "name" : "flask"
      }
    },
    {
      "_index" : "products",
      "_type" : "_doc",
      "_id" : "156",
      "_score" : 1.0,
      "_source" : {
        "productId" : 4974,
        "qty" : 90,
        "name" : "flask"
      }
    }
  ]
}
```

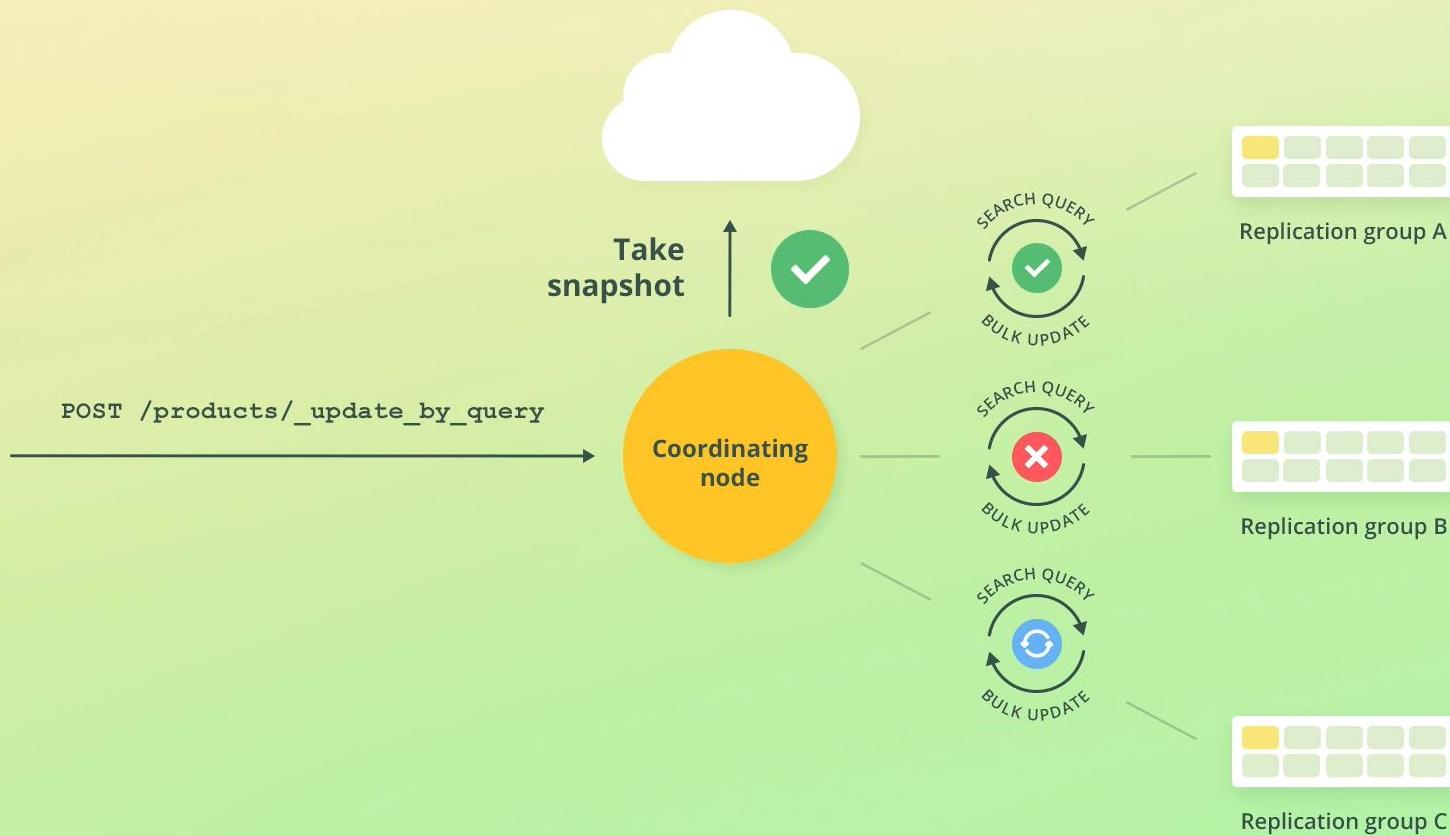


Multiple update(Scroll API)





Multiple update(Scroll API)





Delete Multiple Documents

History Settings Help

Console Search Profiler Grok Debugger

```
1 POST /products/_delete_by_query
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
```

1 [2 "took" : 141,
3 "timed_out" : false,
4 "total" : 2,
5 "deleted" : 2,
6 "batches" : 1,
7 "version_conflicts" : 0,
8 "noops" : 0,
9 "retries" : {
10 "bulk" : 0,
11 "search" : 0
12 },
13 "throttled_millis" : 0,
14 "requests_per_second" : -1.0,
15 "throttled_until millis" : 0,
16 "failures" : []
17]
18 }



Bulk API

- You learned how to index, update, and delete documents
- Let's see how we can perform these actions on *many* documents with a *single* query
 - That's done with the Bulk API
- The Bulk API expects data formatted using the NDJSON specification

```
action_and_metadata\n
optional_source\n
action_and_metadata\n
optional_source\n
```



Bulk API

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

```
1 POST /_bulk
2 { "index": { "_index": "products", "_id": 200 } }
3 { "name": "Espresso Machine", "price": 199, "in_stock": 5 }
4 { "create": { "_index": "products", "_id": 201 } }
5 { "name": "Milk Frother", "price": 149, "in_stock": 14 }
6
7 POST /_bulk
8 { "update": { "_index": "products", "_id": 201 } }
9 { "doc": { "price": 129 } }
10 { "delete": { "_index": "products", "_id": 200 } }

11
12
13
14
15
16
17
18 GET /products/_search
19 {
20   "query": {
21     "match_all": {}
22   }
23 }
```

▶ 🔍

```
1 [
2   "took" : 634,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 2,
6     "successful" : 2,
7     "skipped" : 0,
8     "failed" : 0
9   },
10   "hits" : {
11     "total" : {
12       "value" : 2,
13       "relation" : "eq"
14     },
15     "max_score" : 1.0,
16     "hits" : [
17       {
18         "_index" : "products",
19         "_type" : "_doc",
20         "_id" : "200",
21         "_score" : 1.0,
22         "_source" : {
23           "name" : "Espresso Machine",
24           "price" : 199,
25           "in_stock" : 5
26         }
27       }
28     ]
29   }
30 }
```

Udemy



Bulk Insert

Curl –H “Content-Type: application/x-ndjson” –XPOST
“localhost:9200/receipe/doc/_bulk?pretty” –data-binary @test-data.json

```
Administrator: RabbitMQ Command Prompt (sbin dir) - X
Complete Guide to Elasticsearch | Loading sample data | Kibana User + ←
G:\Local disk\ELK>curl -H "Content-Type: application/x-ndjson" -XPOST "localhost:9200/receipe/doc/_bulk?pretty" --data-binary @test-data.json
{
  "took" : 3039,
  "errors" : false,
  "items" : [
    {
      "index" : {
        "_index" : "receipe",
        "_type" : "doc",
        "_id" : "1",
        "_version" : 1,
        "result" : "created",
        "_shards" : {
          "total" : 2,
          "successful" : 1,
          "failed" : 0
        },
        "_seq_no" : 0,
        "_primary_term" : 1,
        "status" : 201
      }
    }
  ]
}
```



Bulk API

New line data json -- ndjson

Things to be aware of (1/3)

- The HTTP Content-Type header should be set as follows
 - Content-Type: application/x-ndjson
 - application/json is accepted, but that's not the correct way
- The Console tool handles this for us
 - The Elasticsearch SDKs also handle this for us
 - Using HTTP clients, we need to handle this ourselves
- You will see how to do this in the next lecture



Bulk API

Things to be aware of (2/3)

- Each line **must** end with a newline character (`\n` or `\r\n`)
 - This **includes** the last line
 - In a text editor, this means that the last line should be empty
 - Automatically handled with the Console tool
 - Typically a script will generate the bulk file, in which case you need to handle this
 - Don't type out `\n` or `\r\n` in a text editor 😊



Bulk API

Things to be aware of (3/3)

- A failed action will **not** affect other actions
 - Neither will the bulk request as a whole be aborted
- The Bulk API returns detailed information about each action
 - Inspect the `items` key to see if a given action succeeded
 - The order is the same as the actions within the request
 - The `errors` key conveniently tells us if any errors occurred



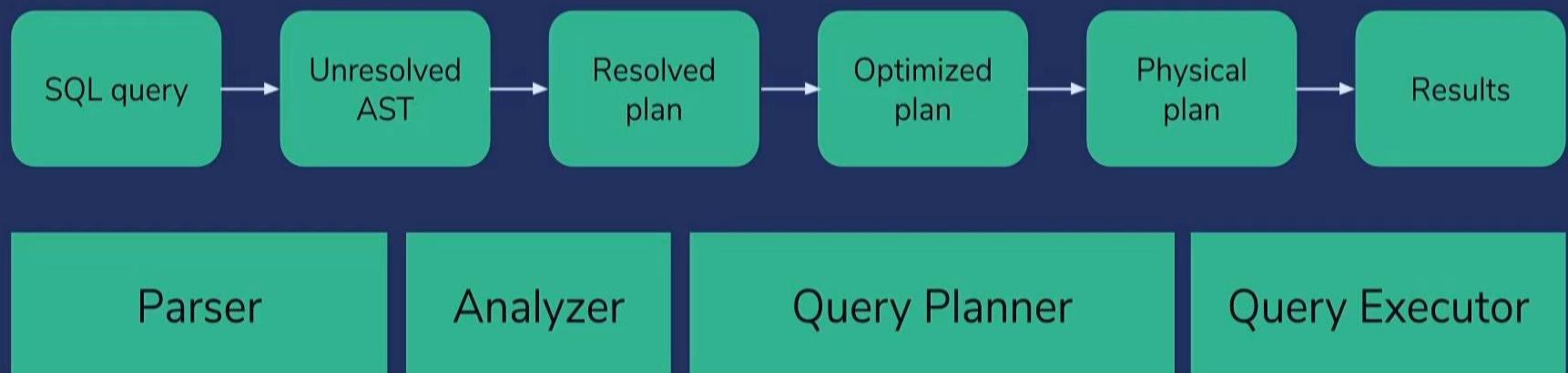
Why Bulk API

- When you need to perform lots of write operations at the same time
 - E.g. when importing data or modifying lots of data
- The Bulk API is more efficient than sending individual write requests
 - A lot of network round trips are avoided



Elastic Sql query

How It Works





Elastic SQL Query

Gmail Elastic Kibana Getting Started with SQL | Elastic

localhost:5601/app/kibana#/dev_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

K D Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 18620 ms

1 POST /_sql?format=txt
2 {
3 | "query": "SELECT * FROM products"
4 }

	name	productId	qty
1			
2			
3	flask	4974	90
4	flask	4974	90
5			

Type here to search



Elastic SQL Query

- POST /_sql?format=txt
- {
- "query": "SELECT * FROM library WHERE release_date < '2000-01-01'"
- }



Elastic SQL(elastic-sql-cli) from elastic bin

```
c:\ Administrator: Command Prompt - elasticsearch-sql-cli
          asticElasticE
          ElasticE  sticEla
      sticEl  ticEl           Elast
      lasti  Elasti           tic
      cEl     ast             icE
      icE     as              cEl
      icE     as              cEl
      icEla   las             El
  sticElasticElast           icEla
  las       last            ticElast
  El       asti            asti    stic
  El       asticEla         Elas    icE
  El       Elas  cElasticE  ticEl    cE
  Ela      ticEl           ticElasti  cE
  las       astic           last      icE
  sticEla  asti            asti    stic
  icEl     sticElasticEla
  icE      sticE  ticEla
  icE      sti   cEla
  icEl    sti   Ela
  cEl     sti   cEl
  Ela      astic  ticE
  asti     ElasticElasti
  ticElasti  lasticEla
  ElasticElast

          SQL
          7.7.0

sql> -
```



02:26 03/06/2020 ENG



Bulk Insert

Curl –H “Content-Type: application/x-ndjson” –XPOST
“localhost:9200/receipt/doc/_bulk?pretty” - -data-binary @test-data.json

```
Complete Guide to Elasticsearch | X Loading sample data | Kibana User | +  
Administrator: RabbitMQ Command Prompt (sbin dir)  
G:\Local disk\ELK>curl -H "Content-Type: application/x-ndjson" -XPOST "localhost:9200/receipt/doc/_bulk?pretty" --data-binary @test-data.json  
{  
  "took" : 3039,  
  "errors" : false,  
  "items" : [  
    {  
      "index" : {  
        "_index" : "receipt",  
        "_type" : "doc",  
        "_id" : "1",  
        "_version" : 1,  
        "result" : "created",  
        "_shards" : {  
          "total" : 2,  
          "successful" : 1,  
          "failed" : 0  
        },  
        "_seq_no" : 0,  
        "_primary_term" : 1,  
        "status" : 201  
      }  
    }  
  ]  
}
```



Mapping

```
CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    dob DATE,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

MySQL



```
PUT /employees
{
  "mappings": {
    "properties": {
      "id": { "type": "integer" },
      "first_name": { "type": "text" },
      "last_name": { "type": "text" },
      "dob": { "type": "date" },
      "description": { "type": "text" },
      "created_at": { "type": "date" }
    }
  }
}
```

Elasticsearch



What is Mapping

- Defines the structure of documents (e.g. fields and their data types)
 - Also used to configure how values are indexed
- Similar to a table's schema in a relational database
- Explicit mapping
 - We define field mappings ourselves
- Dynamic mapping
 - Elasticsearch generates field mappings for us



Keyword Data Type

- Used for exact matching of values
- Typically used for filtering, aggregations, and sorting
- E.g. searching for articles with a status of PUBLISHED
- For full-text searches, use the `text` data type instead
 - E.g. searching the body text of an article



Keyword Data Type

object

boolean

double

float

integer

long

short

text

date



Data Types

- Core datatypes
- string
- text and keyword
- Numeric
- long, integer, short, byte, double, float, half_float, scaled_float
- Date
- date
- Date nanoseconds



Data Types

- date_nanos
- Boolean
- boolean
- Binary
- binary
- Range
- integer_range, float_range, long_range, double_range, date_range, ip_range



Data Types

- Complex datatypes
- Object
- object for single JSON objects
- Nested
- nested for arrays of JSON objects
- Geo datatypes
- Geo-point
- geo_point for lat/lon points
- Geo-shape
- geo_shape for complex shapes like polygons



Data Types

- Specialised datatypes
- IP
- ip for IPv4 and IPv6 addresses
- Completion datatype
- completion to provide auto-complete suggestions
- Token count
- token_count to count the number of tokens in a string
- mapper-murmur3



Data Types

- Specialised datatypes
- IP
- ip for IPv4 and IPv6 addresses
- Completion datatype
- completion to provide auto-complete suggestions
- Token count
- token_count to count the number of tokens in a string
- mapper-murmur3



Data Types

- Join
- Defines parent/child relation for documents within the same index
- Rank feature
- Record numeric feature to boost hits at query time.
- Rank features
- Record numeric features to boost hits at query time.
- Dense vector
- Record dense vectors of float values.
- Sparse vector
- Record sparse vectors of float values.
- Search-as-you-type



Data Types

- A text-like field optimized for queries to implement as-you-type completion
- Alias
- Defines an alias to an existing field.
- Flattened
- Allows an entire JSON object to be indexed as a single field.
- Shape
- shape for arbitrary cartesian geometries.
- Histogram
- histogram for pre-aggregated numerical values for percentiles aggregations.
- Constant keyword
- Specialization of keyword for the case when all documents have the same value.
- Arrays



Meta Fields

- Meta fields customize how a document's associated metadata is treated.
- Each document has associated metadata such as the _index, mapping _type, and _id meta-fields.
- The behavior of some of these meta-fields could be custom when a mapping type was created.



Meta Fields

- Identity meta-fields
 - `_index`: The index to which the document belongs.
 - `_uid`: A composite field consisting of the `_type` and the `_id`.
 - `_type`: The document's mapping type.
 - `_id`: The document's ID.
- Document source meta-fields
 - `_source`: The original JSON representing the body of the document.
 - `_size`: The size of the `_source` field in bytes, provided by the mapper-size plugin.



Meta Fields

- Indexing meta-fields
 - `_all`: A catch-all field that indexes the values of all other fields.
 - `_field_names`: All fields in the document that contain non-null values.
 - `_timestamp`: A timestamp associated with the document, either specified manually or auto-generated.
 - `_ttl`: How long a document should live before it is automatically deleted.
- Routing meta-fields
 - `_parent`: Used to create a parent-child relationship between two mapping types.
 - `_routing`: A custom routing value that routes a document to a particular shard.



Mapping

```
{  
    "name": "Coffee Maker",  
    "price": 64.2,  
    "in_stock": 10,  
    "is_active": true,  
    "manufacturer": {  
        "name": "Nespresso",  
        "country": "Switzerland",  
        "owned_by": {  
            "name": "Nestlé Group",  
            "country": "Switzerland"  
        }  
    }  
}
```



```
PUT /products  
{  
    "mappings": {  
        "properties": {  
            ...  
            "manufacturer": {  
                "properties": {  
                    "name": { "type": "text" },  
                    "country": { "type": "text" },  
                    "owned_by": {  
                        "properties": {  
                            "name": { "type": "text" },  
                            "country": { "type": "text" }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```



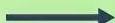
Mapping

```
{
  "name": "Coffee Maker",
  "price": 64.2,
  "in_stock": 10,
  "is_active": true,
  "manufacturer": {
    "name": "Nespresso",
    "country": "Switzerland"
  }
}
```



```
{
  "name": "Coffee Maker",
  "price": 64.2,
  "in_stock": 10,
  "is_active": true,
  "manufacturer.name": "Nespresso",
  "manufacturer.country": "Switzerland"
}
```

```
{
  "name": "Coffee Maker",
  "price": 64.2,
  "in_stock": 10,
  "is_active": true,
  "manufacturer": {
    "name": "Nespresso",
    "country": "Switzerland",
    "owned_by": {
      "name": "Nestlé Group",
      "country": "Switzerland"
    }
  }
}
```



```
{
  "name": "Coffee Maker",
  "price": 64.2,
  "in_stock": 10,
  "is_active": true,
  "manufacturer.name": "Nespresso",
  "manufacturer.country": "Switzerland",
  "manufacturer.owned_by.name": "Nestlé Group",
  "manufacturer.owned_by.country": "Switzerland"
}
```



Introduction to Arrays

- There is no such thing as an `array` data type
- Any field may contain zero or more values
 - No configuration or mapping needed
 - Simply supply an array when indexing a document
- We did this for the `tags` field for the `products` index



Arrays

Console Search Profiler Grok Debugger

```
1 POST /_analyze
2 {
3   "text": ["Strings are simply", "merged together."],
4   "analyzer": "standard"
5 }
```

```
1 [
2   "tokens" : [
3     {
4       "token" : "strings",
5       "start_offset" : 0,
6       "end_offset" : 7,
7       "type" : "<ALPHANUM>",
8       "position" : 0
9     },
10    {
11      "token" : "are",
12      "start_offset" : 8,
13      "end_offset" : 11,
14      "type" : "<ALPHANUM>",
15      "position" : 1
16    },
17    {
18      "token" : "simply",
19      "start_offset" : 12,
20      "end_offset" : 18,
21      "type" : "<ALPHANUM>",
22      "position" : 2
23    },
24    {
25      "token" : "merged",
26      "start_offset" : 19,
27      "end_offset" : 25,
28      "type" : "<ALPHANUM>",
29      "position" : 3
30    },
31    {
32      "token" : "together",
33      "start_offset" : 26,
34      "end_offset" : 34,
35      "type" : "<ALPHANUM>",
36      "position" : 4
37    }
38  ]
39 ]
```



Arrays

```
// Correct data types
```

- ✓ ["electronics", "expensive", "popular"]
- ✓ [37, 45, 9]
- ✓ [true, false, true]
- ✓ [{ "name": "Coffee Maker" }, { "name": "Toaster" }, { "name": "Blender" }]

```
// Coercion
```

- ! [true, false, "true"]
- ! ["electronics", "expensive", 47]
- ! [37, 45, "9"]
- ! [true, false, "true"]

```
// Cannot coerce
```

- ✗ [{ "name": "Coffee Maker" }, { "name": "Toaster" }, false]



Nested Arrays

- Arrays may contain nested arrays
- Arrays are flattened during indexing
- `[1, [2, 3]]` becomes `[1, 2, 3]`



Mapping by Data Type

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

```
PUT /reviews
{
  "mappings": {
    "properties": {
      "rating": { "type": "float" },
      "content": { "type": "text" },
      "product_id": { "type": "integer" },
      "author": {
        "properties": {
          "first_name": { "type": "text" },
          "last_name": { "type": "text" },
          "email": { "type": "keyword" }
        }
      }
    }
  }
}

1 ↴ [
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "reviews"
] ↴
```

Udemy



Mapping in Data Types

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

```
1 PUT /reviews
2 {
3   "mappings": {
4     "properties": {
5       "rating": { "type": "float" },
6       "content": { "type": "text" },
7       "product_id": { "type": "integer" },
8       "author": {
9         "properties": {
10           "first_name": { "type": "text" },
11           "last_name": { "type": "text" },
12           "email": { "type": "keyword" }
13         }
14       }
15     }
16   }
17 }

18
19 PUT /reviews/_doc/1
20 {
21   "rating": 5.0,
22   "content": "Outstanding course! Bo really taught me a lot about Elasticsearch!",
23   "product_id": 123,
24   "author": {
25     "first_name": "John",
26     "last_name": "Doe",
27     "email": "johndoe123@example.com"
28   }
29 }
```

```
1 [
2   "_index": "reviews",
3   "_type": ".doc",
4   "_id": "1",
5   "_version": 1,
6   "result": "created",
7   "_shards": {
8     "total": 2,
9     "successful": 1,
10    "failed": 0
11  },
12  "_seq_no": 0,
13  "_primary_term": 1
14 ]
15
```



Mapping in Data Types

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

1 GET /reviews/_mapping

```
1  [
2    "reviews" : {
3      "mappings" : {
4        "properties" : {
5          "author" : {
6            "properties" : {
7              "email" : {
8                "type" : "keyword"
9              },
10             "first_name" : {
11               "type" : "text"
12             },
13             "last_name" : {
14               "type" : "text"
15             }
16           }
17         },
18         "content" : {
19           "type" : "text"
20         },
21         "product_id" : {
22           "type" : "integer"
23         },
24         "rating" : {
25           "type" : "float"
26         }
27       }
28     }
29   }
30 ]
31
```



Mapping in Data Types

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

```
1 PUT /reviews/_doc/2
2 {
3   "rating": 4.5,
4   "content": "Not bad. Not bad at all!",
5   "product_id": 123,
6   "created_at": "2015-03-27",
7   "author": {
8     "first_name": "Average",
9     "last_name": "Joe",
10    "email": "avgjoe@example.com"
11  }
12}
13
14 PUT /reviews/_doc/3
15 {
16   "rating": 3.5,
17   "content": "Could be better",
18   "product_id": 123,
19   "created_at": "2015-04-15T13:07:41Z",
20   "author": {
21     "first_name": "Spencer",
22     "last_name": "Pearson",
23     "email": "sppearson@example.com"
24  }
25}
26
27 PUT /reviews/_doc/4
28 {
29   "rating": 5.0,
30   "content": "Incredible!",
31   "product_id": 123,
32   "created_at": "2015-01-28T09:21:51+01:00",
33   "author": {
34     "first_name": "Adam",
35     "last_name": "Jones",
36     "email": "adam.jones@example.com"
37  }
38}
39
40 # 2015-07-04T12:01:24Z
41 PUT /reviews/_doc/5
42 {
43   "rating": 4.5,
44   "content": "Very useful",
45   "product_id": 123,
46   "created_at": 1436011284000,
47   "author": {
48     "first_name": "Taylor",
49     "last_name": "West",
```

Add Mapping to existing index



Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

```
1 PUT /reviews/_mapping
2 {
3   "properties": {
4     "created_at": {
5       "type": "date"
6     }
7   }
8 }
9
10 GET /reviews/_mapping|
```

A screenshot of the Kibana Dev Tools interface. On the left, there's a sidebar with various icons. The main area shows a code editor with two tabs: one for PUT /reviews/_mapping and one for GET /reviews/_mapping. The GET tab is active, showing a JSON mapping configuration. A cursor is hovering over the GET button.

```
1 [
2   "reviews": {
3     "mappings": {
4       "properties": {
5         "author": {
6           "properties": {
7             "email": {
8               "type": "keyword"
9             },
10            "first_name": {
11              "type": "text"
12            },
13            "last_name": {
14              "type": "text"
15            }
16          }
17        },
18        "content": {
19          "type": "text"
20        },
21        "created_at": {
22          "type": "date"
23        },
24        "product_id": {
25          "type": "integer"
26        },
27        "rating": {
28          "type": "float"
29        }
30      }
31    }
32  }
33 }
```

The mapping defines an 'reviews' type with nested properties for 'author', 'content', 'created_at', 'product_id', and 'rating'. The 'author' property contains fields for 'email', 'first_name', and 'last_name', all of which are of type 'text'. The 'content' field is also of type 'text'. The 'created_at' field is of type 'date'. The 'product_id' field is of type 'integer'. The 'rating' field is of type 'float'.



Introduction to Date Fields

- Specified in one of three ways;
 - Specially formatted strings
 - Milliseconds since the epoch (long)
 - Seconds since the epoch (integer)
- Epoch refers to the 1st of January 1970
- Custom formats are supported



Default Behaviour of Date Fields

- Three supported formats;
 - A date *without* time
 - A date *with* time
 - Milliseconds since the epoch (long)
- UTC timezone assumed if none is specified
- Dates must be formatted according to the ISO 8601 specification



Date

```
22     "last_name": "Pearson",
23     "email": "spearson@example.com"
24   }
25 }
26
27 PUT /reviews/_doc/4
28 {
29   "rating": 5.0,
30   "content": "Incredible!",
31   "product_id": 123,
32   "created_at": "2015-01-28T09:21:51+01:00",
33   "author": {
34     "first_name": "Adam",
35     "last_name": "Jones",
36     "email": "adam.jones@example.com"
37   }
38 }
39
40 # 2015-07-04T12:01:24Z
41 PUT /reviews/_doc/5
42 {
43   "rating": 4.5,
44   "content": "Very useful",
45   "product_id": 123,
46   "created_at": 1436011284000,
47   "author": {
48     "first_name": "Taylor",
49     "last_name": "West",
50     "email": "twest@example.com"
51   }
52 }
53
54 GET /reviews/_search
55 {
56   "query": {
57     "match_all": {}
58   }
59 }
```



Missing Fields

- All fields in Elasticsearch are optional
- You can leave out a field when indexing documents
- E.g. unlike relational databases where you need to allow `NULL` values
- Some integrity checks need to be done at the application level
 - E.g. having required fields
- Adding a field mapping does *not* make a field required

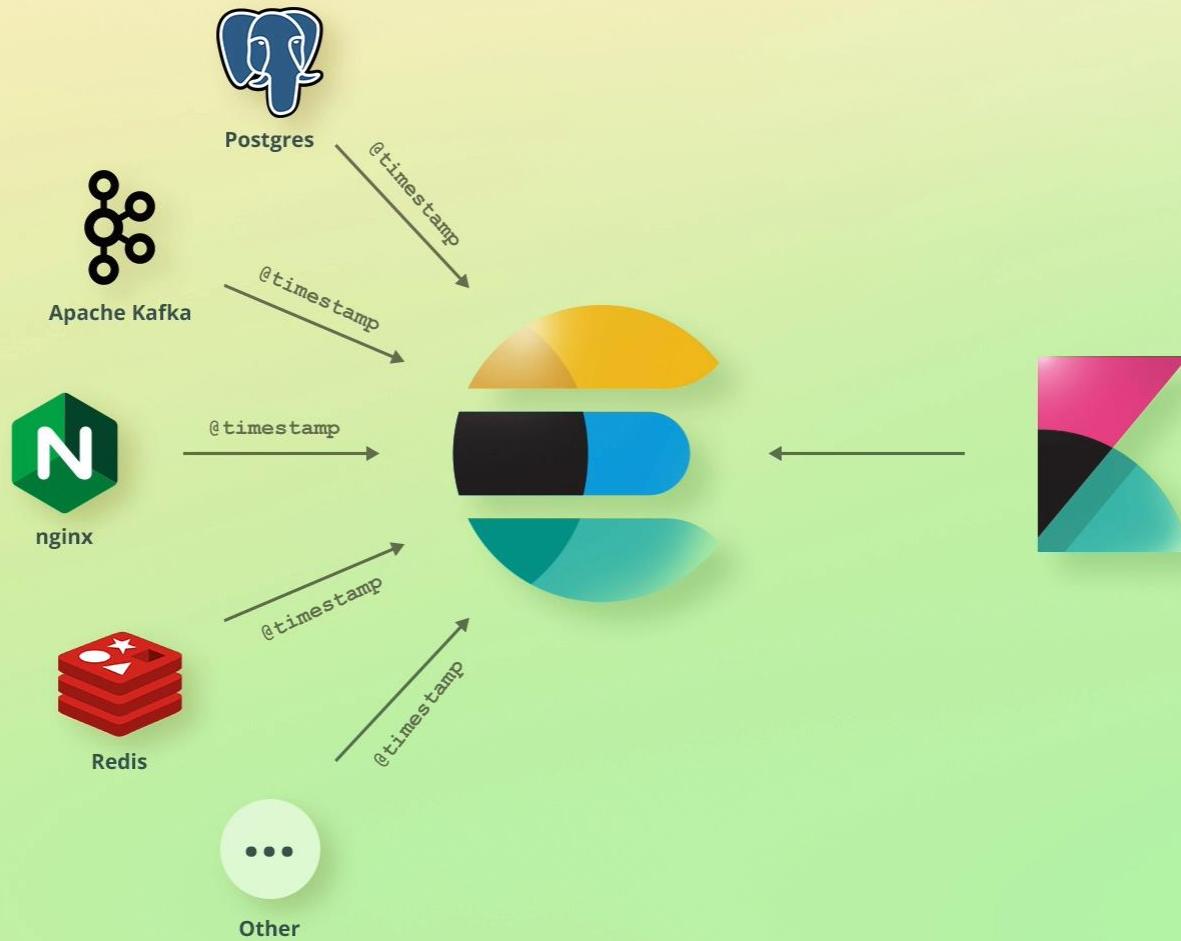


Limitation to Update Mappings

- Being able to update mappings would be problematic for existing documents
 - Text values have already been analyzed, for instance
 - Changing between some data types would require rebuilding the whole data structure



Elastic Common Schema





Uses of ECS

- In ECS, documents are referred to as *events*
 - ECS doesn't provide fields for non-events (e.g. products)
- Mostly useful for standard events
 - E.g. web server logs, operating system metrics, etc.
- ECS is automatically handled by Elastic Stack products
 - If you use them, you often won't have to actively deal with ECS



Elastic Common Schema

Heartbeat -c heartbeat.yml -e -d *

```
Administrator: Command Prompt - heartbeat -c heartbeat.yml -e -d *
"fe80::7570:a615:c73c:c95",
"10.102.37.150",
"169.254.12.149",
"fe80::bc62:acf0:4b32:24",
"192.168.0.7",
"fe80::a806:d6d1:284:2f3b",
"169.254.47.59"
],
"mac": [
    "58:8a:5a:02:6b:0a",
    "8e:15:9b:65:bc:9b",
    "00:15:5d:fc:f0:03",
    "02:00:4c:4f:4f:50",
    "0a:00:27:00:00:08",
    "f8:34:41:ac:d4:70",
    "fa:34:41:ac:d4:6f",
    "00:ff:28:1b:32:b7",
    "f8:34:41:ac:d4:6f",
    "f8:34:41:ac:d4:73"
]
}
}
2020-06-03T19:19:47.567+0530 DEBUG [scheduler] scheduler/scheduler.go:195 Job 'auto-http-0X3F1F767F45156CB3' returned at
2020-06-03 19:19:47.5674444 +0530 IST m+=601.120184301
2020-06-03T19:19:47.740+0530 DEBUG [elasticsearch] elasticsearch/client.go:217 PublishEvents: 2 events have been published to
elasticsearch in 72.9698ms.
2020-06-03T19:19:47.748+0530 DEBUG [publisher] memqueue/ackloop.go:160 ackloop: receive ack [59: 0, 2]
2020-06-03T19:19:47.749+0530 DEBUG [publisher] memqueue/eventloop.go:535 broker ACK events: count=2, start-seq=60, end-s
eq=61

2020-06-03T19:19:47.750+0530 DEBUG [publisher] memqueue/ackloop.go:128 ackloop: return ack to broker loop:2
2020-06-03T19:19:47.750+0530 DEBUG [publisher] memqueue/ackloop.go:131 ackloop: done send ack
```



Type here to search



19:19
ENG
03/06/2020



Elastic Common Schema

localhost:5601/app/kibana#/discover?_g=(filters:!(),refreshInterval:(pause:1t,value:0),time:(from:now-15m,to:now))&_a=(columns:!(_source),filters:!(),index:'4aad6750-a5a0-11ea-bfde-b...',)

Discover

New Save Open Share Inspect

Search KQL Last 15 minutes Show dates Refresh

+ Add filter

heart* ▾

Search field names

Filter by type 0

Selected fields _source

Available fields @timestamp add

Top 5 values in 78 / 78 records

Value	Count
Jun 3, 2020 @ 19:00:31.153	1.3%
Jun 3, 2020 @ 19:15:27.005	1.3%
Jun 3, 2020 @ 19:01:00	1.3%
Jun 3, 2020 @ 19:02:00	1.3%
Jun 3, 2020 @ 19:03:00	1.3%
Jun 3, 2020 @ 19:04:00	1.3%
Jun 3, 2020 @ 19:05:00	1.3%
Jun 3, 2020 @ 19:06:00	1.3%
Jun 3, 2020 @ 19:07:00	1.3%
Jun 3, 2020 @ 19:08:00	1.3%
Jun 3, 2020 @ 19:09:00	1.3%
Jun 3, 2020 @ 19:10:00	1.3%
Jun 3, 2020 @ 19:11:00	1.3%
Jun 3, 2020 @ 19:12:00	1.3%
Jun 3, 2020 @ 19:13:00	1.3%
Jun 3, 2020 @ 19:14:00	1.3%
Jun 3, 2020 @ 19:15:00	1.3%

78 hits Jun 3, 2020 @ 19:00:31.153 - Jun 3, 2020 @ 19:15:31.153 — Auto

Count @timestamp per 30 seconds

Time ▾ _source

> Jun 3, 2020 @ 19:15:27.005 @timestamp: Jun 3, 2020 @ 19:15:27.005 monitor.duration.us: 6,101 monitor.type: http monitor.timespan: { "gte": "2020-06-03T13:45:27.005Z", "lt": "2020-06-03T13:45:37.005Z" } monitor.id: auto-http-0X3F1F767F45156CB3 monitor.name: monitor.check_group: 7553b97a-a5a0-11ea-9009-588a5a026b0a monitor.ip: 127.0.0.1 monitor.status: up resolve.ip: 127.0.0.1



Elastic Common Schema

Screenshot of a web browser showing an Elastic Kibana dashboard titled "Uptime".

The dashboard displays the following information:

- 2 Monitors**: A donut chart showing 1 Down monitor (red) and 1 Up monitor (light blue).
- Pings over time**: A bar chart showing ping counts from 11:00 to 11:12. Most bars are red (approx. 3 pings), except for 11:12 which is light blue (approx. 1 ping).
- Monitor status**: A table listing two monitors:

Status	Name	Url	Downtime history
Up 3m ago	Unnamed - auto-http-0X2DC84F14213F9C15-e21e441f5a592bc8	http://localhost:5601	--
Down 3m ago	Unnamed - auto-http-0X2DC84F14213F9C15-f64771baa43c6c9f	https://customer-cf2020.cfapps.io	



Dynamic Mapping

```
POST /my-index/_doc
{
  "tags": ["computer", "electronics"],
  "in_stock": 4,
  "created_at": "2020/01/01 00:00:00"
}
```



```
{
  "created_at": {
    "type": "date",
    "format": "yyyy/MM/dd HH:mm:ss||yyyy/MM/dd||epoch_millis"
  },
  "in_stock": {
    "type": "long"
  },
  "tags": {
    "type": "text",
    "fields": {
      "keyword": {
        "type": "keyword",
        "ignore_above": 256
      }
    }
  }
}
```



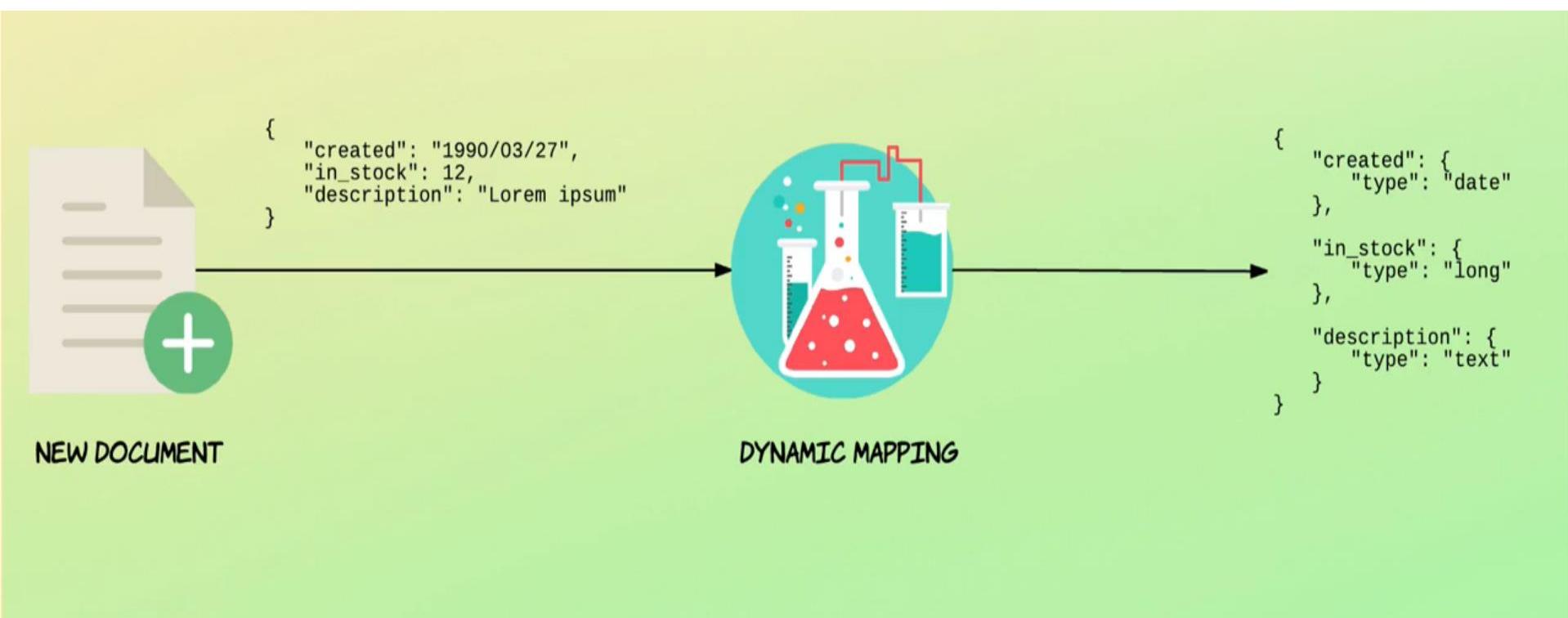
Dynamic Mapping

JSON	ELASTICSEARCH
string	One of the following: <ul style="list-style-type: none">• <code>text</code> field with <code>keyword</code> mapping• <code>date</code> field• (<code>float</code> or <code>long</code> field)
integer	<code>long</code>
floating point number	<code>float</code>
boolean (<code>true</code> or <code>false</code>)	<code>boolean</code>
object	<code>object</code>
array	Depends on the first non-null value

```
[ "strict_date_optional_time", "yyyy/MM/dd HH:mm:ss Z||yyyy/MM/dd Z" ]
```



Managing Documents Dynamic Mapping





Managing Documents Dynamic Mapping

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET /product/_mapping
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
```

```
{
  "name": {
    "type": "text",
    "fields": {
      "keyword": {
        "type": "keyword",
        "ignore_above": 256
      }
    }
  },
  "price": {
    "type": "long"
  },
  "salesprice": {
    "type": "long"
  },
  "tags": {
    "type": "text",
    "fields": {
      "keyword": {
        "type": "keyword",
        "ignore_above": 256
      }
    }
  }
}
```



Use Explicit Mappings

- Dynamic mapping is convenient, but often not a good idea in production
- Save disk space with optimized mappings when storing many documents
- Set `dynamic` to "strict", not `false`
 - Avoids surprises and unexpected results



Mapping of text fields

- Don't always map strings as both `text` and `keyword`
 - Typically only one is needed
 - Each mapping requires disk space
- Do you need to perform full-text searches?
 - Add a `text` mapping
- Do you need to do aggregations, sorting, or filtering on exact values?
 - Add a `keyword` mapping



Disable Coercion

- Coercion forgives you for not doing the right thing
- Try to do the right thing instead
- Always use the correct data types whenever possible



Use appropriate numeric data types

- For whole numbers, the `integer` data type might be enough
 - `long` can store larger numbers, but also uses more disk space
- For decimal numbers, the `float` data type might be precise enough
 - `double` stores numbers with a higher precision but uses 2x disk space
 - Usually, `float` provides enough precision



Mapping Parameters

- Set `doc_values` to `false` if you don't need sorting, aggregations, and scripting
- Set `norms` to `false` if you don't need relevance scoring
- Set `index` to `false` if you don't need to filter on values
 - You can still do aggregations, e.g. for time series data
- Probably only worth the effort when storing lots of documents
 - Otherwise it's probably an over complication
- Worst case scenario, you will need to reindex documents



Managing Documents Custom Data Type

Dev Tools

Console Search Profiler Grok Debugger

Discover Visualize Dashboard Timelion APM Dev Tools Monitoring Management

```
1 POST member/default/1/_update
2 {
3
4     "doc": {"doj": "2018/12/12"}
5
6 }
7
8 PUT member/default/_mapping
9 {
10     "properties":
11     {
12         "doj": {
13             "type": "date",
14             "format": ["basic_date_time", "basic_date", "basic_date_time_no_millis"]
15         }
16     }
17 }
18
19 DELETE member
20
21 PUT /member
```

1 {
2 "member": {
3 "mappings": {
4 "default": {
5 "dynamic": "false",
6 "properties": {
7 "doj": {
8 "type": "date"
9 },
10 "fees": {
11 "type": "integer"
12 }
13 }
14 }
15 }
16 }
17 }



Managing Documents Custom Data Type

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
18
19 DELETE member
20
21 PUT /member
22 {
23   "mappings": {
24     "default": {
25       "dynamic": false,
26       "properties": {
27         "fees": {
28           "type": "integer"
29         },
30         "doj": {
31           "type": "date"
32         }
33       }
34     }
35   }
36 }
37
38 POST member/default/1/
39 {
40 }
```

1 {
2 "member": {
3 "mappings": {
4 "default": {
5 "dynamic": "false",
6 "properties": {
7 "doj": {
8 "type": "date"
9 },
10 "fees": {
11 "type": "integer"
12 }
13 }
14 }
15 }
16 }
17 }



Managing Documents Custom Data Type

Kibana Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

```
23 "mappings":{  
24     "default":{  
25         "dynamic":false,  
26         "properties": {  
27             "fees":{  
28                 "type":"integer"  
29             },  
30             "doj":{  
31                 "type":"date"  
32             }  
33         }  
34     }  
35 }  
36 }  
37  
38 POST member/default/1/  
39 {  
40  
41     "doc":{ "doj": "2018/12/12" }  
42  
43 }  
44  
45 GET member/default/_mapping
```

Member mapping:

```
1 {  
2     "member": {  
3         "mappings": {  
4             "default": {  
5                 "dynamic": "false",  
6                 "properties": {  
7                     "doj": {  
8                         "type": "date"  
9                     },  
10                    "fees": {  
11                        "type": "integer"  
12                    }  
13                }  
14            }  
15        }  
16    }  
17 }
```



Managing Documents Custom Data Type

```
curl -H "Content-Type: application/json" -XPOST "http://localhost:9200/product/default/_bulk?pretty" --data-binary "@products-bulk.json"
```

mats



Managing Documents Changing Mapping

Kibana Dev Tools Console Search Profiler

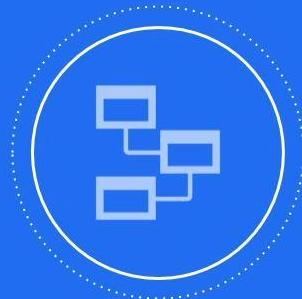
```
PUT /product/default/_mapping
{
  "properties": {
    "discount": {
      "type": "integer"
    }
  }
}

DELETE /product

PUT /product
{
  "mappings": {
    "default": {
      "dynamic": false,
      "properties": {
        "in_stock": {
          "type": "integer"
        },
        "is_active": {
          "type": "integer"
        },
        "price": {
          "type": "integer"
        },
        "sold": {
          "type": "long"
        }
      }
    }
  }
}

{"acknowledged": true,
 "shards_acknowledged": true}
```

Udemy

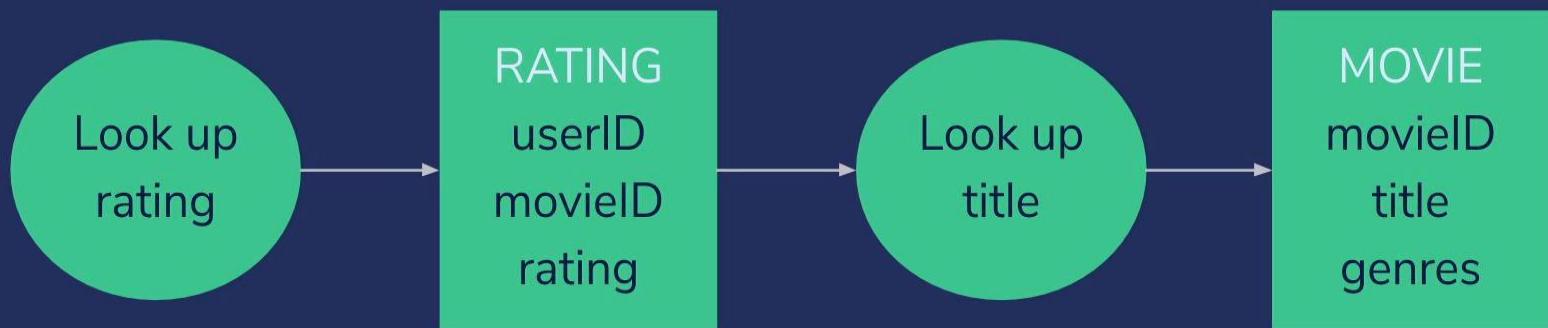


Data Modeling



Strategies For Relational Data

Normalized data

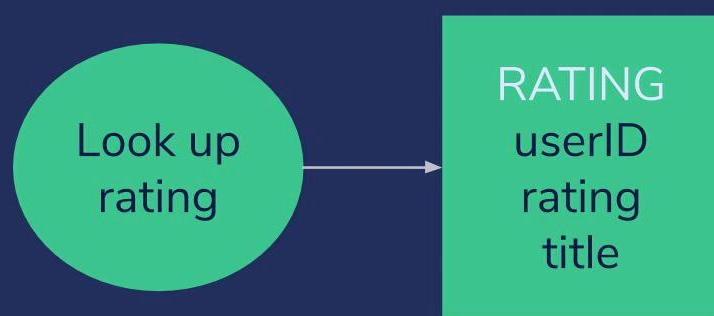


Minimizes storage space, makes it easy to change titles
But requires two queries, and storage is cheap!



Strategies For Relational Data

Denormalized Data



Titles are duplicated, but only one query



Strategies For Relational Data

Parent / Child Relationship

Star Wars

A New Hope

Empire
Strikes Back

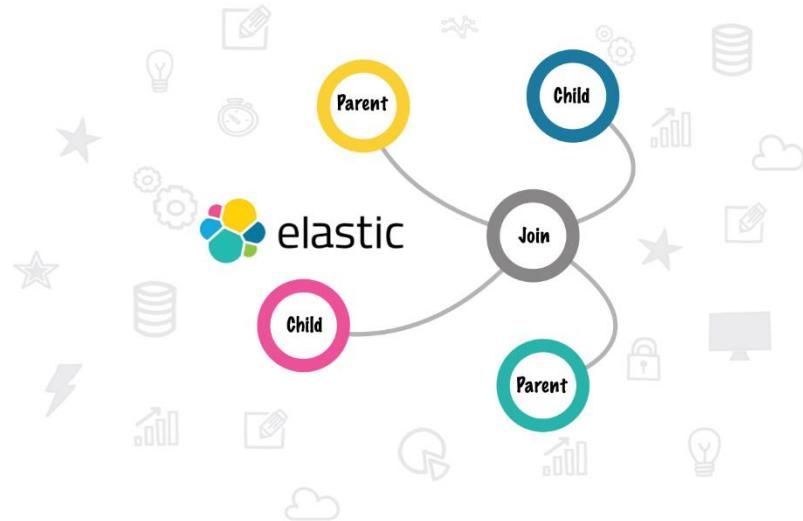
Return of the
Jedi

The Force
Awakens



Parent child Relationship

```
• PUT /music-5_6
• {
•   "settings": {
•     "number_of_shards": 1, "number_of_replicas": 0,
•     "mapping.single_type": true
•   },
•   "mappings": {
•     "doc": {
•       "properties": {
•         "artist": { "type": "text" },
•         "song": { "type": "text" },
•         "user": { "type": "keyword" },
•         "artist_relations": {
•           "type": "join",
•           "relations": {
•             "artist": "song",
•             "song": "user"
•           }
•         }
•       }
•     }
•   }
• }
```



Refer github.com/rpsvelkjan2023



Parent child Relationship

- POST /music-5_6/doc/_bulk
- {"index":{"_id":1}}
- {"name":"John Legend","artist_relations":{"name":"artist"}}
- {"index":{"_id":2}}
- {"name":"Ariana Grande","artist_relations":{"name":"artist"}}



Parent child Relationship

- PUT music-5_6/doc/3?routing=1
- {"song":"All of Me","artist_relations":{"name":"song","parent":1}}

- PUT music-5_6/doc/4?routing=1
- {"song":"Beauty and the Beast","artist_relations":{"name":"song","parent":1}}

- PUT music-5_6/doc/5?routing=2
- {"song":"Beauty and the Beast","artist_relations":{"name":"song","parent":2}}



Parent child Relationship

- POST music-5_6/doc/_bulk?routing=3
- {"index":{"_id":"I-1"}}
- {"user":"Gabriel","artist_relations":{"name":"user","parent":3}}
- {"index":{"_id":"I-2"}}
- {"user":"Berte","artist_relations":{"name":"user","parent":3}}
- {"index":{"_id":"I-3"}}
- {"user":"Emma","artist_relations":{"name":"user","parent":3}}



Parent child Relationship

- GET music-5_6/_search
- {
- "query": {
- "has_parent": {
- "parent_type": "artist",
- "query": { "match": { "name": "John Legend" } } }
- }
- }
- }



Parent child Relationship

- GET music-5_6/_search
- {
- "query": {
- "has_parent": {
- "parent_type": "song",
- "query": {
- "match": { "song": "all of Me" } }
- }
- }
- }
- }



Parent child Relationship

- GET music/_search
- {
- "query": {
- "has_child": {
- "type": "song",
- "min_children": 1, "max_children": 10,
- "query": { "match_all": {} } }
- }
- }
- }

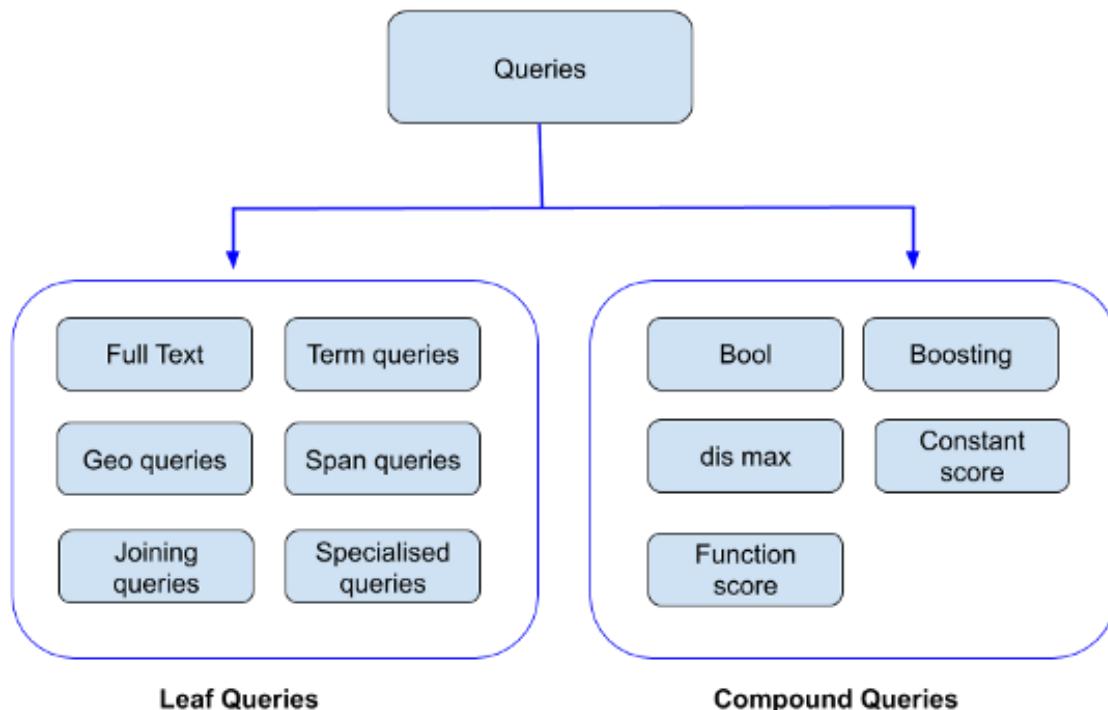


Parent child Relationship

- GET music-5_6/doc/3?routing=1
- {
- "_index": "music-5_6",
- "_type": "doc",
- "_id": "3",
- "_version": 3,
- "_routing": "1",
- "found": true,
- "_source": {
- "song": "All of Me",
- "artist_relations": {
- "name": "song",
- "parent": 1
- }
- }
- }



Queries





Search Methods Term Search

Apps Projects Gmail YouTube Maps Pluralsight

K D Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 1580 ms

```
1 GET /bookdb_index/book/_search
2
3 {
4   "query": {
5     "match": {
6       "title": "in action"
7     }
8   }
9 }
10 }
```

```
/ successful : 1,
8   "skipped" : 0,
9   "failed" : 0
10 },
11 "hits" : {
12   "total" : {
13     "value" : 2,
14     "relation" : "eq"
15   },
16   "max_score" : 1.2393079,
17   "hits" : [
18     {
19       "_index" : "bookdb_index",
20       "_type" : "book",
21       "_id" : "4",
22       "_score" : 1.2393079,
23       "_source" : {
24         "summary" : "Comprehensive guide to implementing a scalable search engine using Apache Solr",
25         "publisher" : "manning",
26         "num_reviews" : 63,
27         "title" : "Solr in Action",
28         "publish_date" : "2014-04-05",
29         "authors" : [
30           "trey grainger",
31           "timothy potter"
32         ]
33       }
34     }
35   ]
36 }
```



Request URI Search

- GET /bookdb_index/book/_search?q=title:in action

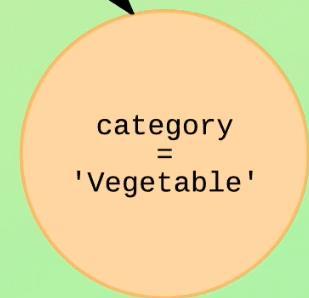
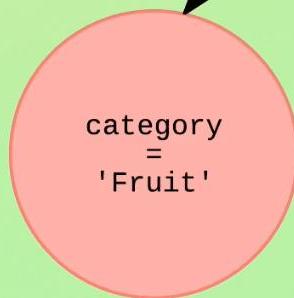
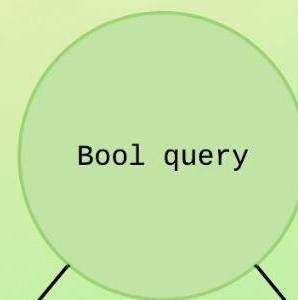
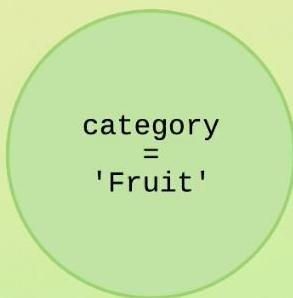


Introducing the Query DSL



COMPOUND QUERY

LEAF QUERY



category
= 'Fruit'

OR

category
= 'Vegetable'



Query DSL

localhost:5601/app/kibana#/dev_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 325 ms

```
1 GET /bookdb_index/book/_search
2 {
3   "query": {
4     "query_string": {
5       "query": "title:in action"
6     }
7   }
8 }
9 }
10 }
```

```
3   "took" : 219,
4   "timed_out" : false,
5   "_shards" : {
6     "total" : 1,
7     "successful" : 1,
8     "skipped" : 0,
9     "failed" : 0
10 },
11   "hits" : {
12     "total" : {
13       "value" : 2,
14       "relation" : "eq"
15     },
16     "max_score" : 1.2393079,
17     "hits" : [
18       {
19         "_index" : "bookdb_index",
20         "_type" : "book",
21         "_id" : "4",
22         "_score" : 1.2393079,
23         "_source" : {
24           "summary" : "Comprehensive guide to implementing a scalable search engine using Apache Solr",
25           "publisher" : "manning",
26           "num_reviews" : 63,
27           "title" : "Solr in Action",
28         }
29       }
30     ]
31   }
32 }
```

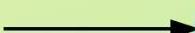


Understanding relevance scores





GET /product/default/_search?q=pizza



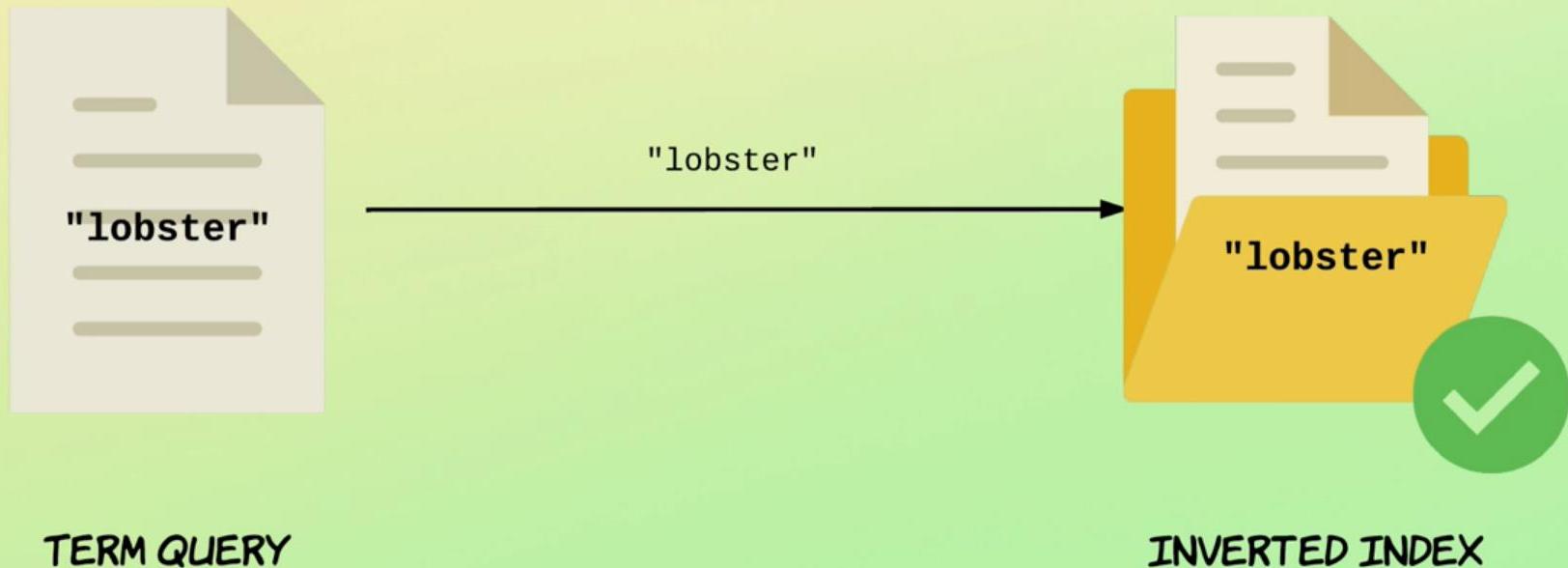
Find documents
matching the query

- Pizza Hawaii
■ Pizza Vegetariana
■ Pizza Margherita





Term Level Queries





Term Level Queries

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET syslog-logglyindex/doc/_search
2 {
3   "query": {
4     "term": {
5       "@timestamp": "2018-12-19T07:27:54.786Z"
6     }
7   }
8 }
9
10
11
12 GET syslog-logglyindex/doc/_search
```

History Settings Help

took: 450, timed_out: false, _shards: { total: 5, successful: 5, skipped: 0, failed: 0 }, hits: { total: 20, max_score: 1, hits: [{ _index: "syslog-logglyinde", _type: "doc", _id: "cr1dxWcBjynLt7N6KjB3", _score: 1, _source: { path: "D:/logstash configurations/loggly_events_2018-12-19T07:27:54.786Z" } }] }



Term Level Queries

kibana

Dev Tools

Console Search Profiler Grok Debugger

Discover Visualize Dashboard Timelion APM Dev Tools Monitoring Management

Collapse

```
1 GET syslog-logglyindex/doc/_search
2 {
3   "query":{
4     "term":{
5       "@timestamp":{
6         "value": "2018-12-19T07:27:54.786Z"
7     }
8   }
9 }
10 }
11 }
12 }
13 }
14 }
15 }
16 GET syslog-logglyindex/doc/_search
```

History Settings Help

```
1 {
2   "took": 2,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10   "hits": {
11     "total": 20,
12     "max_score": 1,
13     "hits": [
14       {
15         "_index": "syslog-logglyinde",
16         "_type": "doc",
17         "_id": "cr1dxWcBjynLt7N6KjB3",
18         "_score": 1,
19         "_source": {
20           "path": "D:/logstash
configurations/loggly_events_2018-12-19T07:27:54.786Z"
21         }
22       }
23     ]
24   }
25 }
```



Term Level Queries

Kibana Dev Tools

Console Search Profiler Grok Debugger History Settings Help

```
1 GET syslog-logglyindex/doc/_search
2 {
3     "query":{
4
5         "terms":{
6             "@timestamp": [
7                 "2018-12-19T07:27:54.786Z",
8                 "2018-12-19T07:27:54.787Z"
9             ]
10        }
11    }
12 }
13 }
```

```
1 {
2     "took": 11,
3     "timed_out": false,
4     "_shards": {
5         "total": 5,
6         "successful": 5,
7         "skipped": 0,
8         "failed": 0
9     },
10    "hits": {
11        "total": 42,
12        "max_score": 1,
13        "hits": [
14            {
15                "_index": "syslog-logglyinde",
16                "_type": "doc",
17                "_id": "cr1dxWcBjynLt7N6KjB3",
18                "_score": 1,
19                "_source": {
20                    "path": "D:/logstash
configurations/loggly_events_2018-12-19-07-27-54-786.log"
21                }
22            }
23        ]
24    }
25 }
```



Retrieving Documents By Id

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET syslog-logglyindex/doc/_search
2 {
3     "query":{
4
5         "ids":{
6             "values":
7                 ["cr1dxWcBjynLt7N6KjB3",
8                 "h71dxWcBjynLt7N6KjB3"]
9
10        }
11
12    }
13 }
14
15 GET syslog-logglyindex/doc/_search
```

took: 189,
timed_out: false,
_shards: {
 total: 5,
 successful: 5,
 skipped: 0,
 failed: 0
},
hits: {
 total: 2,
 max_score: 1,
 hits: [
 {
 _index: "syslog-logglyindex",
 _type: "doc",
 _id: "cr1dxWcBjynLt7N6KjB3"
,
 _score: 1,
 _source: {
 path: "D:/logstash/configurations/loggly_events_2018-12-07-01-45570.json"
 }
 }
]
}



Matching Documents with Range Query

Kibana Dev Tools

Console Search Profiler Grok Debugger History Settings Help

```
1 GET bitcoin-prices/_search
2 {
3   "query":{
4     "range":{
5       "Low": {
6         "gte":200,
7         "lt": 900
8       }
9     }
10 }
11 }
12 }
13 }
14 }
15 }
16 }
17 }
18 GET syslog-logglyindex/_search
```

1 {
2 "took": 390,
3 "timed_out": false,
4 "_shards": {
5 "total": 5,
6 "successful": 5,
7 "skipped": 0,
8 "failed": 0
9 },
10 "hits": {
11 "total": 1032,
12 "max_score": 1,
13 "hits": [
14 {
15 "_index": "bitcoin-prices",
16 "_type": "doc",
17 "_id": "Nlu5vGcB1da7Pa9b1bav",
18 "_score": 1,
19 "_source": {
20 "Low": "200.42",
21 "High": "209.79",
22 "Close": "206.9".



Matching Documents with Range Query

localhost:5601/app/kibana#/dev_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

200 - OK 461 ms

```
115  {
116    "query": {
117      "ids": {
118        "values": [2,4]
119      }
120    }
121  }
122  }
123  }
124
125 POST /heartbeat-7.7.0-2020.06.03-000001/_search
126  {
127    "query": {
128      "range": {
129        "@timestamp": {
130          "gte": "2020-06-03T13:18:09.719Z",
131          "lte": "2020-06-04T13:18:09.719Z"
132        }
133  },
134  },
135  "_source": ["@timestamp"]
136  }
137
138
139
140
141
142
```

33 ▲
34 ▲
35 ▾
36
37
38
39
40 ▾
41
42 ▾
43 ▾
44 ▾
45
46
47
48
49 ▾
50
51 ▾
52 ▾
53 ▾
54
55
56
57
58 ▾

```
{
  "_index" : "heartbeat-7.7.0-2020.06.03-000001",
  "_type" : "_doc",
  "_id" : "G3RWenIBXgkoffJ8YCXR",
  "_score" : 1.0,
  "_source" : {
    "@timestamp" : "2020-06-03T13:19:39.744Z"
  }
},
{
  "_index" : "heartbeat-7.7.0-2020.06.03-000001",
  "_type" : "_doc",
  "_id" : "AXRVenIBXgkoffJ8nSV_",
  "_score" : 1.0,
  "_source" : {
    "@timestamp" : "2020-06-03T13:18:49.741Z"
  }
},
{
  "_index" : "heartbeat-7.7.0-2020.06.03-000001",
  "_type" : "_doc",
  "_id" : "CnRVenIBXgkoffJ86yWa",
  "_score" : 1.0,
  "_source" : {
    "@timestamp" : "2020-06-03T13:18:49.742Z"
  }
}
```



Working with Date Relevant Dates

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET bitcoin-prices/_search
2 {
3   "query":{
4     "range":{
5       "Date": {
6         "gte": "2014-04-30 || -1y"
7       }
8     }
9   }
10
11
12 }
13
14 }
15 }
16
17 GET syslog-logglyindex/_search
```

91 "Weighted Price": "134
92 .819293969",
93 "Volume (BTC)": "17710
94 .2452671",
95 "message": "2013-10-04,130
96 .97902,139.8,128.5,136.82222,17710
97 .2452671,2387682.76294,134.819293969
98 ",
99 "@timestamp": "2018-12
100 -17T15:12:32.864Z",
101 "Date": "2013-10-04"
102 }
103 },
104 {
105 "_index": "bitcoin-prices",
106 "_type": "doc",
107 "_id": "PVu5vGcB1da7Pa9b1bav
108 ",
109 "_score": 1,
110 "_source": {
111 "Low": "142.11",
112 "High": "148.5",
113 "Close": "144.0",
114 "Open": "142.11",
115 "Volume": "17710"
116 }
117 }



Working with Date Relevant Dates

Operator	Rounding direction	Before	After
gt	up	2010-01-20	2010-01-31
gte	down	2010-01-20	2010-01-01
lt	down	2010-01-20	2010-01-01
lte	up	2010-01-20	2010-01-31



Working with non null values

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET bitcoin-prices/doc/_search
2 {
3   "query": {
4     "exists": {
5       "field": "Open"
6     }
7   }
8 }
9 }
10 }
11 }
12 }
13 }
14 GET syslog-logglyindex/doc/_search
```

1 {
2 "took": 82,
3 "timed_out": false,
4 "_shards": {
5 "total": 5,
6 "successful": 5,
7 "skipped": 0,
8 "failed": 0
9 },
10 "hits": {
11 "total": 2640,
12 "max_score": 1,
13 "hits": [
14 {
15 "_index": "bitcoin-prices",
16 "_type": "doc",
17 "_id": "K1u5vGcB1da7Pa9b1bav",
18 "
19 "_score": 1,
20 "_source": {
21 "Low": "903.4",
22 "High": "950.0",
23 "Close": "917.05879",
24 }
25 }
26 }
27 }
28 }



Matching on Prefixes

Kibana

Dev Tools

Console Search Profiler Grok Debugger

```
1 GET bitcoin-prices/doc/_search
2 {
3   "query":{
4     "prefix":{
5       "message":"2012"
6     }
7   }
8 }
9 }
10 }
11 }
12 }
13 }
14 GET syslog-logglyindex/doc/_search
```

30 .5102,11.765,11.45,11.62399,30340
.953643,353092.674073,11.6374942669"
,
31 " @timestamp": "2018-12
-17T15:12:32.976Z",
32 "Date": "2012-08-12"
33 }
34 }
35 }
36 {
37 "_index": "bitcoin-prices",
38 "_type": "doc",
39 "_id": "_lu5vGcB1da7Pa9b1b0m
",
40 "_score": 1,
41 "_source": {
42 "Low": "8.8183",
43 "High": "9.23355",
44 "Close": "8.86999",
45 "host": "DESKTOP-55AGI0I",
46 "path": "D:/logstash
configurations/BCHARTS-MTGOXUSD.csv"
,



Searching with wild cards

```
1  GET /products/_search
2  {
3  "query": {
4  "match_all": {}
5  }
6  }
7
8  GET /products/_search
9  {
10 "query": {
11 "wildcard": {
12   "name.keyword": {
13     "value": "N*"
14   }
15 }
16 }
17 }
```

Click to send request

```
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
  "is_active": true,
  "created": "2011/11/06"
},
{
  "_index": "products",
  "_type": "doc",
  "_id": "14",
  "_score": 1.0,
  "_ignored": [
    "description.keyword"
  ],
  "_source": {
    "name": "Nori Sea Weed - Gold Label",
    "price": 177,
    "in_stock": 21,
    "sold": 411,
    "tags": [
      "Seafood"
    ],
  }
},
```



Searching with Regular Expressions

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

Discover Visualize Dashboard Timelion APM Dev Tools Monitoring Management Collapse

```
1 GET bitcoin-prices/doc/_search
2 {
3   "query":{
4     "regexp":{
5       "message": "[0-9]+"
6     }
7   }
8 }
9 }
10 }
11 }
12 }
13
14 GET syslog-logglyindex/doc/_search
```

▶ 🔒

```
1 {
2   "took": 26,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10   "hits": {
11     "total": 2640,
12     "max_score": 1,
13     "hits": [
14       {
15         "_index": "bitcoin-prices",
16         "_type": "doc",
17         "_id": "K1u5vGcB1da7Pa9b1bav",
18         "_score": 1,
19         "_source": {
20           "Low": "903.4",
21           "High": "950.0",
22           "Close": "917.05879",
23           ...
24         }
25       }
26     ]
27   }
28 }
```



Full Text Queries

Dev Tools History Settings Help

Console Search Profiler Grok Debugger

1 GET receipt/doc/_mapping

```
1 {  
2   "receipt": {  
3     "mappings": {  
4       "doc": {  
5         "properties": {  
6           "created": {  
7             "type": "date",  
8             "format": "yyyy/MM/dd HH  
:mm:ss||yyyy/MM/dd||epoch_millis"  
9           },  
10          "description": {  
11            "type": "text",  
12            "fields": {  
13              "keyword": {  
14                "type": "keyword",  
15                "ignore_above": 256  
16              },  
17            },  
18          },  
19          "ingredients": {  
20            "properties": {  
21              "name": {  
22                "type": "text",  
23              },  
24            },  
25          },  
26        },  
27      },  
28    },  
29  }  
30 }
```



Flexi Queries

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET receipe/doc/_search
2 {
3   "query":{
4     "match": {
5       "title":"Receipe with pasta or Spaghetti"
6     }
7   }
8 }
9 }
```

```
1 {
2   "took": 286,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10   "hits": {
11     "total": 17,
12     "max_score": 2.6974046,
13     "hits": [
14       {
15         "_index": "receipe",
16         "_type": "doc",
17         "_id": "11",
18         "_score": 2.6974046,
19         "_source": {
20           "title": "Spaghetti Puttanesca (Pasta or Spaghetti With Capers, Olives, and Anchovies)",
21           "description": "Puttane"
22         }
23       }
24     ]
25   }
26 }
```

Discover Visualize Dashboard Timelion APM Dev Tools Monitoring Management Collapse

Type here to search



Phrase Query

Kibana

Dev Tools

Console Search Profiler Grok Debugger

Discover Visualize Dashboard Timelion APM Dev Tools Monitoring Management Collapse

```
1 GET receipe/doc/_search
2 {
3   "query":{
4     "match_phrase": {
5       "title": "Spaghetti With Black Pepper"
6     }
7   }
8 }
```

1 {
2 "took": 110,
3 "timed_out": false,
4 "_shards": {
5 "total": 5,
6 "successful": 5,
7 "skipped": 0,
8 "failed": 0
9 },
10 "hits": {
11 "total": 1,
12 "max_score": 3.8519454,
13 "hits": [
14 {
15 "_index": "receipe",
16 "_type": "doc",
17 "_id": "7",
18 "_score": 3.8519454,
19 "_source": {
20 "title": "Cacio e Pepe
(Spaghetti With Black Pepper and
Pecorino Romano)",
21 "description": "If you were



Multi Field Query

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET receipe/doc/_search
2 {
3   "query":{
4     "multi_match": {
5       "query": "pasta",
6       "fields": ["title","description"]
7     }
8   }
9 }
```

```
1 {
2   "took": 132,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10   "hits": {
11     "total": 16,
12     "max_score": 1.2756016,
13     "hits": [
14       {
15         "_index": "receipe",
16         "_type": "doc",
17         "_id": "2",
18         "_score": 1.2756016,
19         "_source": {
20           "title": "Pasta With
Butternut Squash and Sage Brown
Butter",
21           "description": "Brown
Butter"
22         }
23       }
24     ]
25   }
26 }
```



Managing Documents Full Text Query

Kibana Dev Tools

Console Search Profiler

Discover Visualize Dashboard Timelion Machine Learning Graph Dev Tools Management

History Settings Help

```
1 | GET /recipe/default/_search
2- {
3-   "query": {
4-     "match": {
5-       "title": "Recipes with pasta or spaghetti"
6-     }
7-   }
8- }
9
10 GET /recipe/default/_search
11- {
12-   "query": {
13-     "match": {
14-       "title": {
15-         "query": "pasta| spaghetti",
16-         "operator": "and"
17-       }
18-     }
19-   }
20 }
```

1+ [
2 "took": 5,
3 "timed_out": false,
4 "_shards": {
5 "total": 5,
6 "successful": 5,
7 "failed": 0
8 },
9 "hits": {
10 "total": 1,
11 "max_score": 1.6718876,
12 "hits": [
13 {
14 "_index": "recipe",
15 "_type": "default",
16 "_id": "11",
17 "_score": 1.6718876,
18 "_source": {
19 "title": "Spaghetti Puttanesca (Pasta or Spaghetti With Capers, Olives, and Anchovies)",
20 "description": """Puttanesca"" literally translates to ""in the style of prostitutes,"" supposedly because the pungent aromas of garlic, anchovies, capers, and olives tossed with pasta were how Neapolitan prostitutes would lead customers to their doors. This is one of those stories that seem, in the words of Douglas Adams, apocryphal or at least wildly inaccurate. That said, it's a fitting title-puttanesca packs an aromatic punch and then some."",
21 "preparation_time_minutes": 15,
22 "servings": {
23 "min": 2,
24 "max": 3
25 },
26 "steps": [
27 "Place spaghetti in a large skillet, saut\u00e9 pan, or saucepan and cover with water. Add a small pinch of salt. Bring to a boil over high heat, stirring occasionally to prevent pasta from sticking.",
28 "Meanwhile, in a medium skillet, combine 4 tablespoons (60ml) oil, garlic, anchovies, and red pepper flakes. Cook over medium heat until garlic is very lightly golden, about 5 minutes. (Adjust heat as necessary to keep it gently sizzling.) Add capers and olives and stir to combine.",
29 "Add tomatoes, stir to combine, and bring to a bare simmer. Continue to simmer until pasta is cooked to just under al dente (about 1 minute less than the package recommends).",
30 "Using tongs, transfer pasta to sauce. Alternatively, drain pasta through a colander, reserving 1 cup of the cooking water. Add drained pasta to sauce.",
31 "Add a few tablespoons of pasta water to sauce and increase heat to bring pasta and sauce to a vigorous simmer. Cook, stirring and shaking the pan and adding more pasta water as necessary to keep sauce loose, until pasta is perfectly al dente, 1 to 2 minutes longer. (The pasta will cook more slowly in the sauce than it did in the water.) Stir in remaining olive oil, parsley, and cheese.",
32 "Season with salt and pepper. (Be generous with the pepper and scant with the salt—the dish will be plenty salty from the other ingredients.) If using, stir in canned tuna and break it up with a fork. Serve immediately with more grated cheese at the table."
33],
34 "ingredients": [
35 {
36 "name": "Dried spaghetti",
37 "quantity": "225g"
38 },

Collapse

Ubiquity



Managing Documents multiple fields

Kibana Dev Tools

Console Search Profiler

Discover Visualize Dashboard Timelion Machine Learning Graph Dev Tools Management

```
1 | GET /recipe/default/_search
2 - {
3 -   "query": {
4 -     "multi_match": {
5 -       "query": "pasta",
6 -       "fields": [ "title", "description" ]
7 -     }
8 -   }
9 - }
```

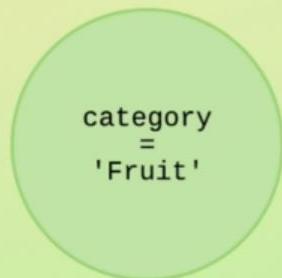
1,]
2, "created": "2002/01/04",
3, "ratings": [
4, 3,
5, 2.5,
6, 4,
7, 4.5,
8, 3.5,
9, 4
10,]
11, }
12, {
13, "_index": "recipe",
14, "_type": "default",
15, "_id": "12",
16, "_score": 1.3220012,
17, "_source": {
18, "title": "Penne With Melted-Vegetable Sauce",
19, "description": "Made with vegetables that have been cooked until meltingly soft, this penne pasta dish is one of those great examples of what makes classic rustic Italian cooking so special: It makes the most of humble and unassuming ingredients, turning them into something downright delicious.",
20, "preparation_time_minutes": 30,
21, "servings": {
22, "min": 4,
23, "max": 4
24, },
25, "steps": [
26, "In a medium pot of salted boiling water, cook potato until a piece is easily crushed between fingers, about 5 minutes. Using fine strainer, transfer to large mixing bowl. Working one vegetable at a time, continue by boiling carrots, string beans, fennel, and onion until each is well done, about 5 minutes each; add each vegetable to mixing bowl as it is ready.",
27, "Add penne to boiling water and cook until al dente, following timing on package. Drain, reserving 1 cup of cooking water.",
28, "Meanwhile, add garlic, olive oil, and parsley to vegetables, and mix thoroughly until potatoes have broken down for form a chunky puree.",
29, "Add penne and a healthy grating of Parmigiano-Reggiano to vegetable sauce and stir to combine, adding cooking water 1 tablespoon at a time if sauce is too thick. Spoon into bowls, top with additional grated cheese, and serve."
30,],
31, "ingredients": [
32, {
33, "name": "Kosher salt"
34, },
35, {
36, "name": "Large russet potato",
37, "quantity": "1"
38, },
39, {
40, "name": "Medium carrots",
41, "quantity": "2"
42, }
43,]
44, }
45, }

Collapse

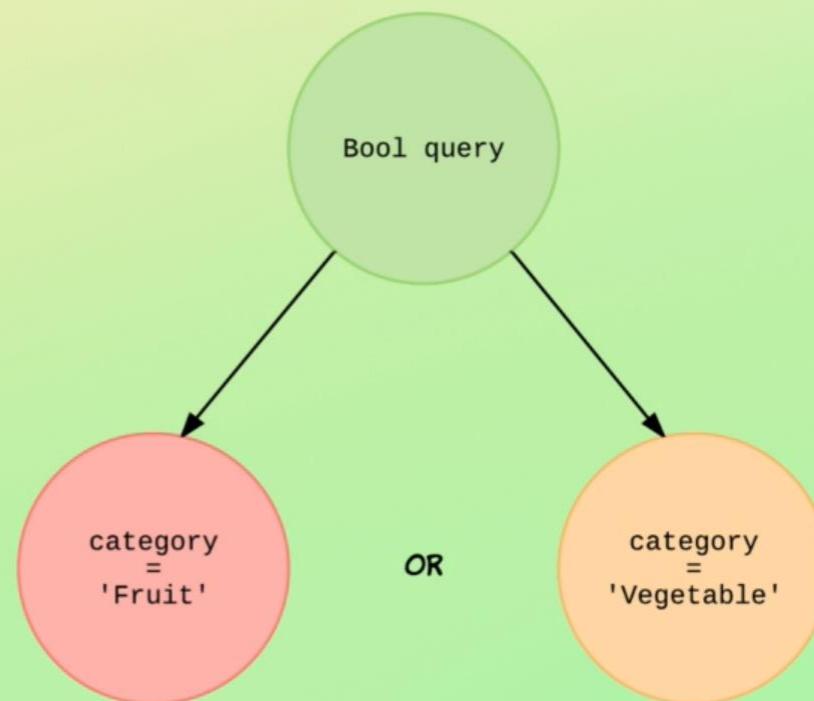


Compound Queries

LEAF QUERY



COMPOUND QUERY





Compound Queries

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET receipt/doc/_search
2 {
3   "query":{
4     "bool": {
5       "must": [
6         {
7           "match": {
8             "ingredients.name": "pasta"
9           }
10        }
11      ],
12      "should": [
13        {
14          "match": {
15            "ingredients.name": "parmeson"
16          }
17        }
18      ]
19    }
20  }
21 }
```

History Settings Help

```
1 {
2   "took": 76,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 6,
12    "max_score": 0.72217846,
13    "hits": [
14      {
15        "_index": "receipt",
16        "_type": "doc",
17        "_id": "10",
18        "_score": 0.72217846,
19        "_source": {
20          "title": "Penne With Hot
-As-You-Dare Arrabbiata Sauce",
21          "description": "Exceedin
gly simple in concept and execution,
... "
22        }
23      }
24    ]
25  }
26 }
```



Document

Doc ID	Content
1	This is a brown fox
2	This is a brown dog
3	This dog is really brown
4	The dog is brown but this document is very very long
5	There is also a white cat
6	The quick brown fox jumps over the lazy dog



General Query

- "query": {
- "match": {
- "content": {
- "query": "quick brown dog"
- }
- }
- }



Document

Pos	Doc ID	Content	Score
1	6	The quick brown fox jumps over the lazy dog	0.81502354
2	2	This is a brown dog	0.26816052
3	3	This dog is really brown	0.26816052
4	4	The dog is brown but this document is very very long	0.15323459
5	1	This is a brown fox	0.055916067



Query

- {
- "query": {
- "match": {
- "content": {
- "query": "quick brown dog",
- "minimum_should_match": 75%
- }
- }
- }
- }



Phrase Query

- "query": {
- "match_phrase": {
- "content": "brown dog"
- }
- }
- }



Proximity Query

- {
- "query": {
- "match_phrase": {
- "content": {
- "query": "brown dog",
- "slop": 3
- }
- }
- }
- }



Result

Pos	Doc ID	Content	Score
1	2	This is a brown dog	0.9547657
2	4	The dog is brown but this document is very very long	0.2727902



Query

- {
- "query": {
- "match_phrase": {
- "content": {
- "query": "brown dog",
- "slop": 4
- }
- }
- }
- }



Result

Pos	Doc ID	Content	Score
1	2	This is a brown dog	0.9547657
2	3	This dog is really brown	0.4269842
3	4	The dog is brown but this document is very very long	0.2727902



Proximity Search

	Position 1	Position 2	Position 3
Document	tomato	sauce	spicy
Query	spicy	sauce	
Slop 1	→	spicy sauce	
Slop 2		sauce	→ spicy



Proximity Search(Order Relevance – slop)

☰ D Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 19 ms

```
1 GET /recipes/_search
2 {
3   "query": {
4     "match_phrase": {
5       "title": {
6         "query": "puttanesca spaghetti",
7         "slop": 2
8       }
9     }
10   }
11 }
12 }
13 }
```

The screenshot shows a developer tools interface for Elasticsearch's Dev Tools. The main area displays a search request and its results. A blue arrow points from the 'slop' value in the search query (line 7) to the resulting document's '_score' field (line 20), which is highlighted in green. Another blue arrow points from the 'slop' value to the detailed description of the recipe (lines 28-70), which also contains several green highlights.

```
20 _type : doc ,
21 _id : "11",
22 _score : 2.7049189,
23 _ignored : [
24   "description.keyword",
25   "steps.keyword"
26 ],
27 _source : {
28   "title" : "Spaghetti Puttanesca (Pasta or Spaghetti With Capers, Olives, and Anchovies)",
29   "description" : "\"Puttanesca\" literally translates to \"in the style of prostitutes,\" supposedly because the pungent aromas of garlic, anchovies, capers, and olives tossed with pasta were how Neapolitan prostitutes would lead customers to their doors. This is one of those stories that seem, in the words of Douglas Adams, apocryphal or at least wildly inaccurate. That said, it's a fitting title – puttanesca packs an aromatic punch and then some."
30   ,
31   "preparation_time_minutes" : 15,
32   "servings" : {
33     "min" : 2,
34     "max" : 3
35   },
36   "steps" : [
```



Relevance

- Precision
- Recall



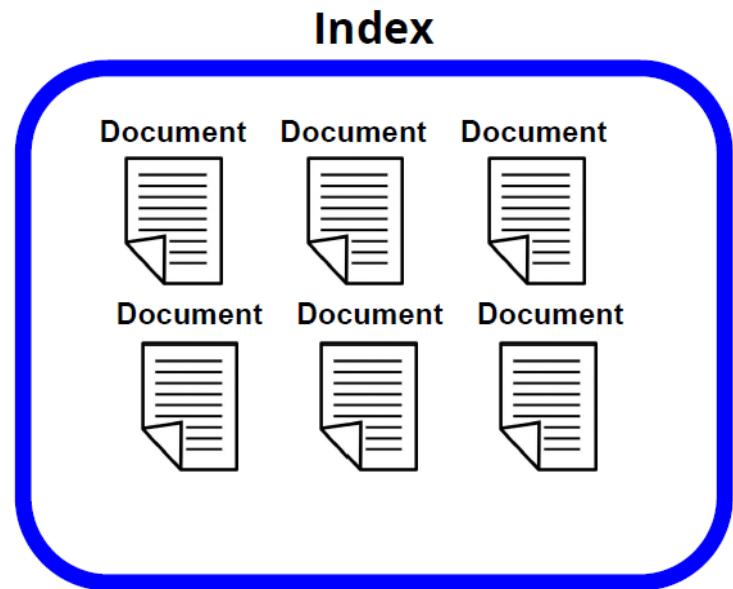
Relevance



Elasticsearch

Store | Search | Analyze

I store data as documents!



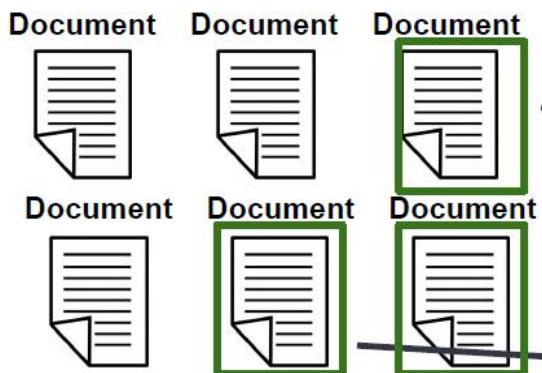
Documents with similar traits are grouped into an index!



Relevance

When search query is sent, Elasticsearch retrieves relevant documents and presents the documents as search results.

Index



The screenshot shows the Elasticsearch Dev Tools interface with the "Search Results" tab selected. The search query is `GET kibana_sample_data_ecommerce/_search`. The results are displayed as a JSON object:

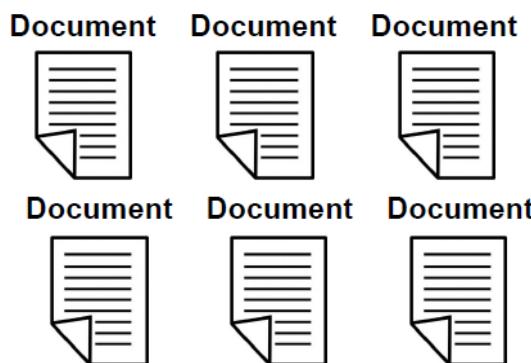
```
10. "hits" : {  
11.   "total" : {  
12.     "value" : 4675,  
13.     "relation" : "eq"  
14.   },  
15.   "max_score" : 1.0,  
16.   "hits" : [  
17.     {  
18.       "_index" :  
19.         "kibana_sample_data_ecommerce",  
20.       "_type" : "_doc",  
21.       "_id" : "79WD1XYBy9gvFwLxZogX",  
22.       "_score" : 1.0,  
23.       "_source" : {  
24.         "category" : [  
25.           "Men's Clothing"  
26.         ],  
27.         "currency" : "EUR",  
28.         "customer_first_name" : "Eddie",  
          "customer_full_name" : "Eddie Underwood".  
        }  
      }  
    ]  
  }  
}
```



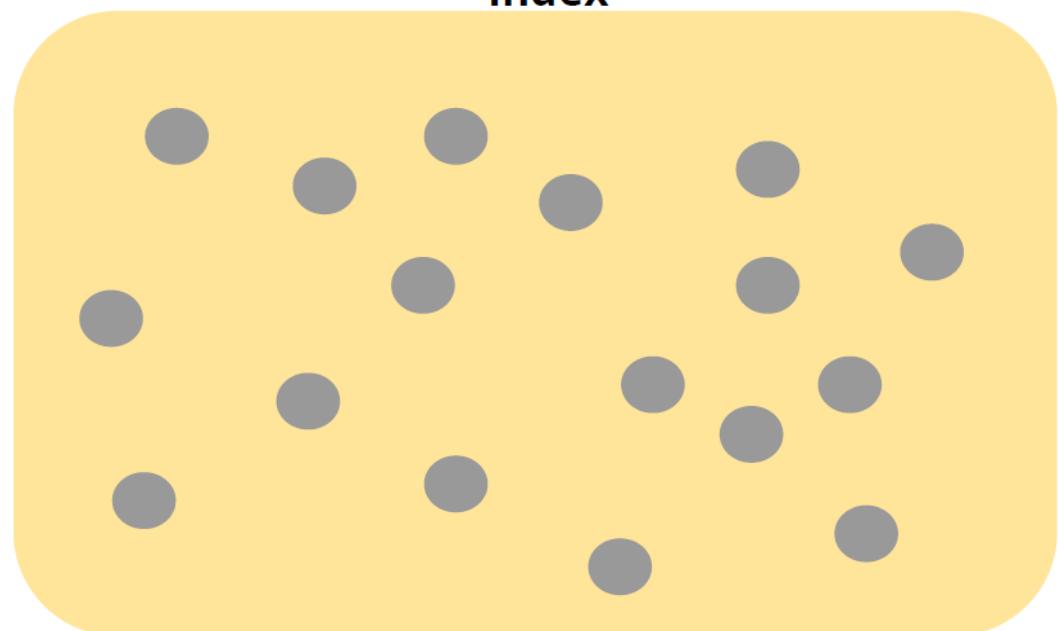
Relevance

These two diagrams depict the same thing!

Index



Index

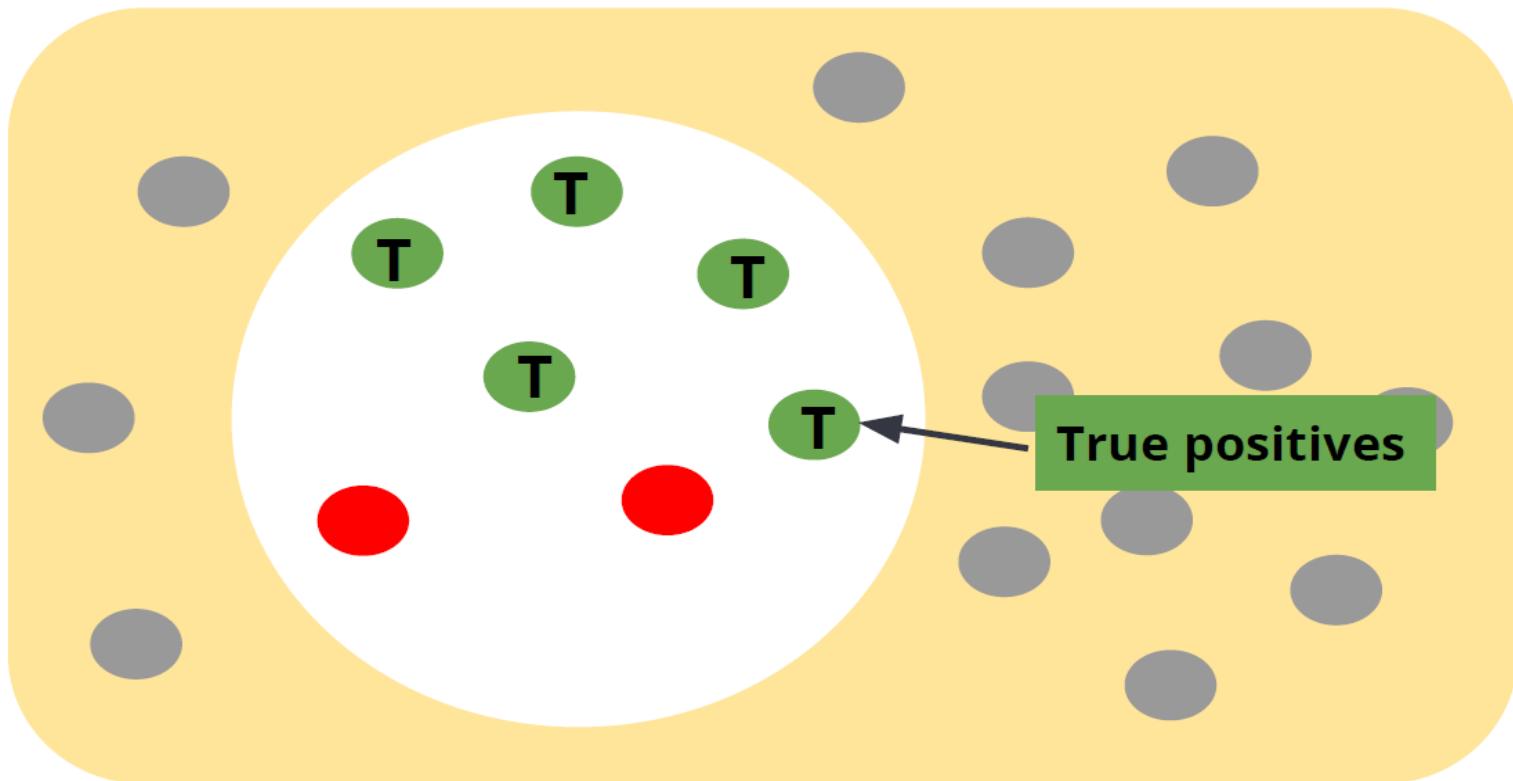




Relevance

True positives are relevant documents that are returned to the user.

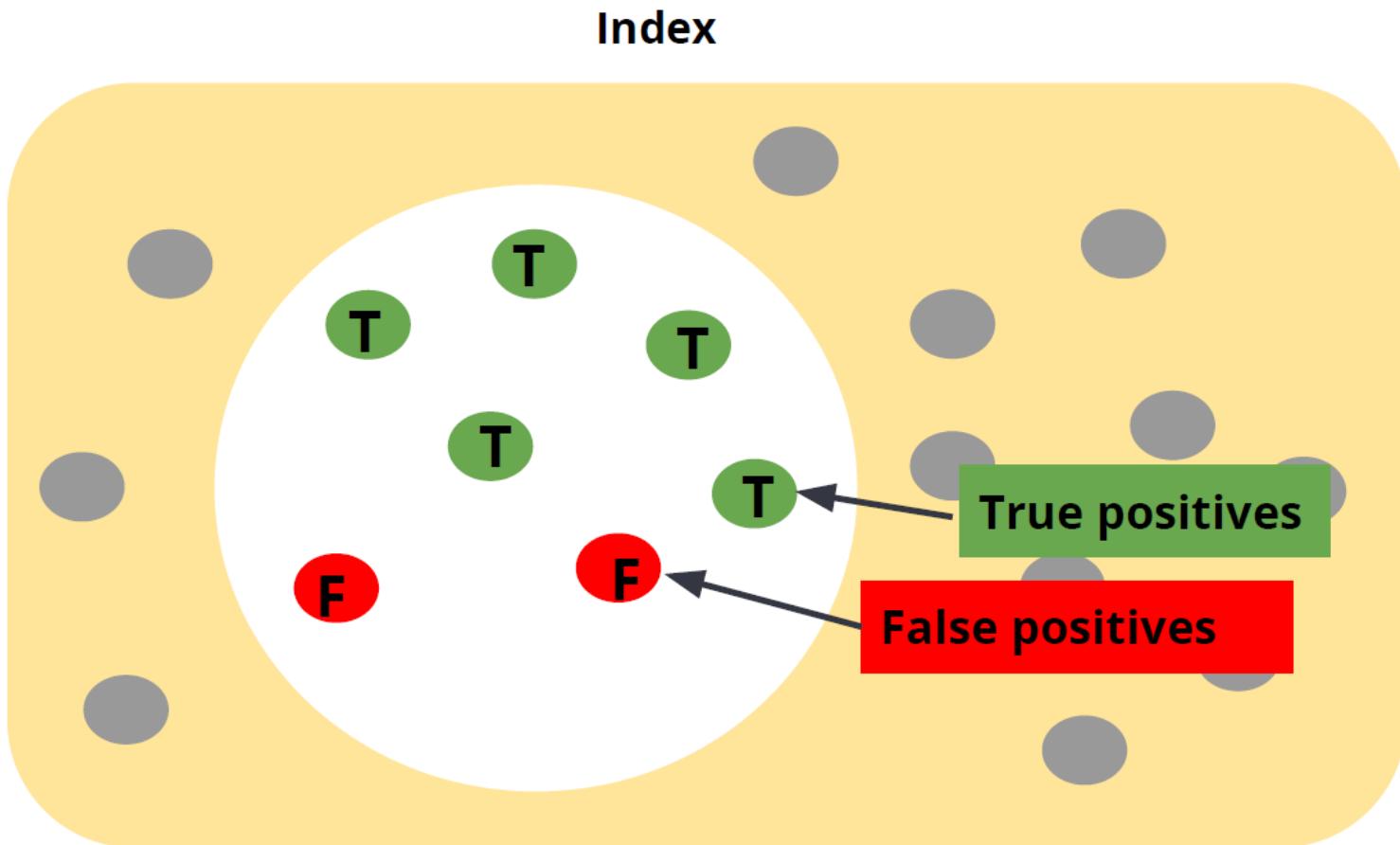
Index





Relevance

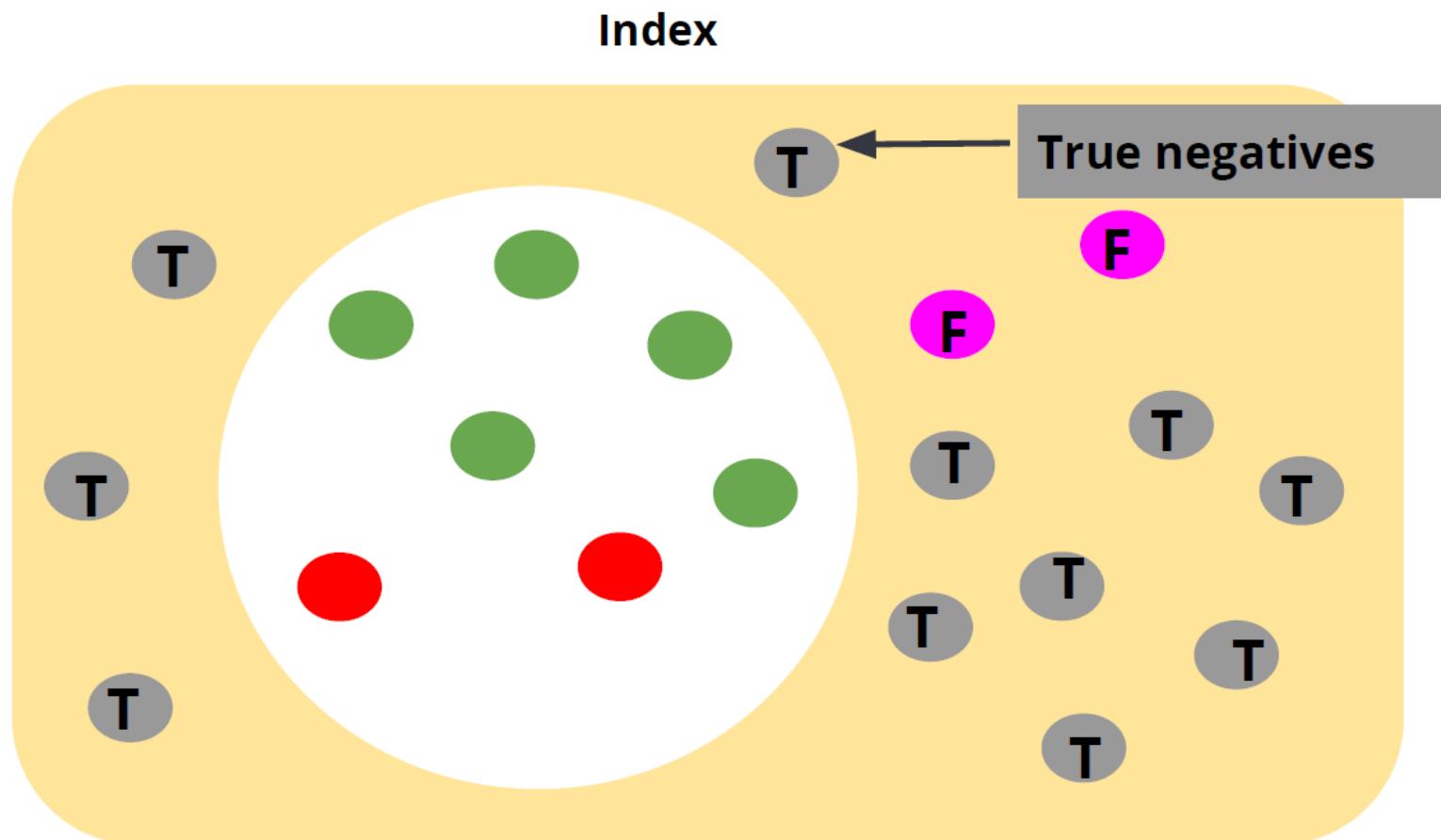
False positives are irrelevant documents that are returned to the user.





Relevance

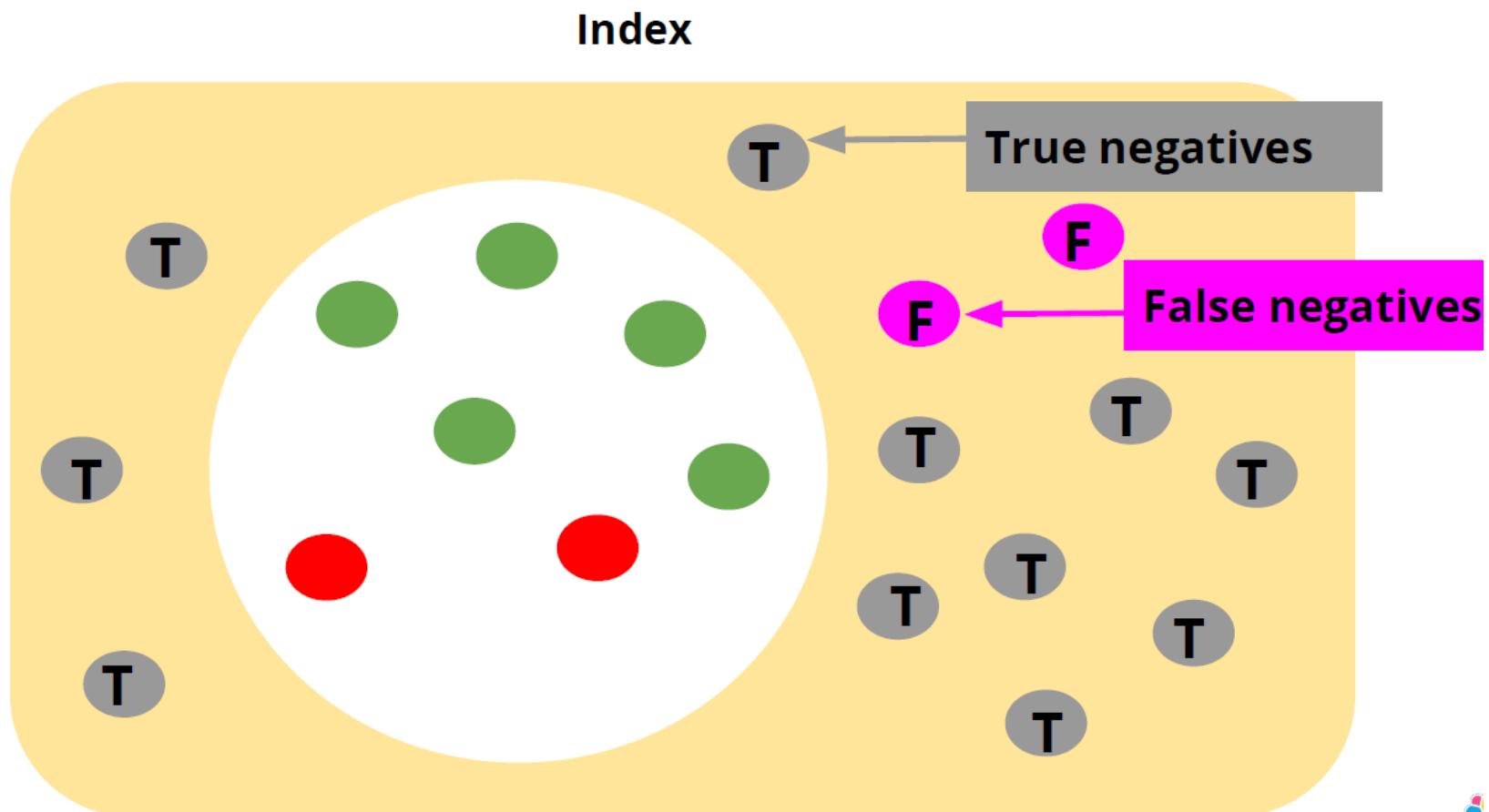
True negatives are irrelevant documents that are not returned to the user.





Relevance

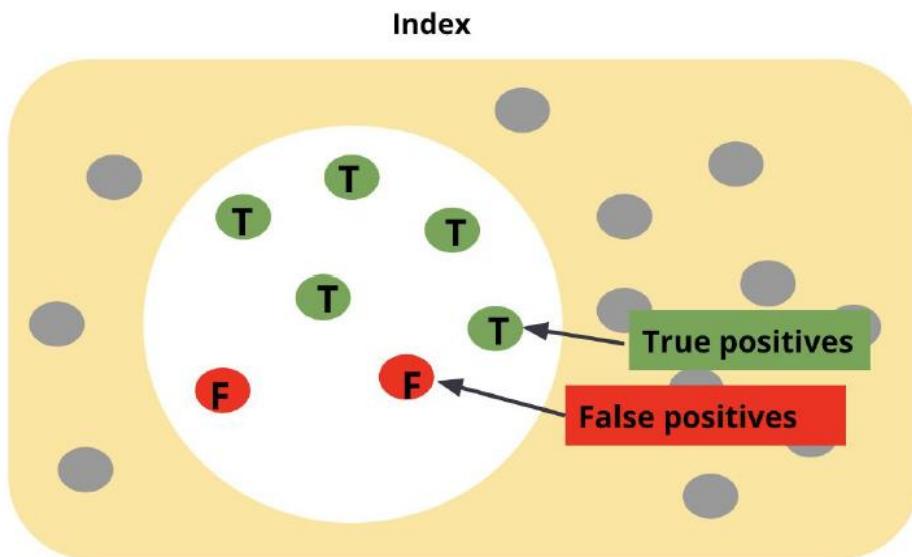
False negatives are relevant documents that were not returned to the user.





Relevance

What is precision?



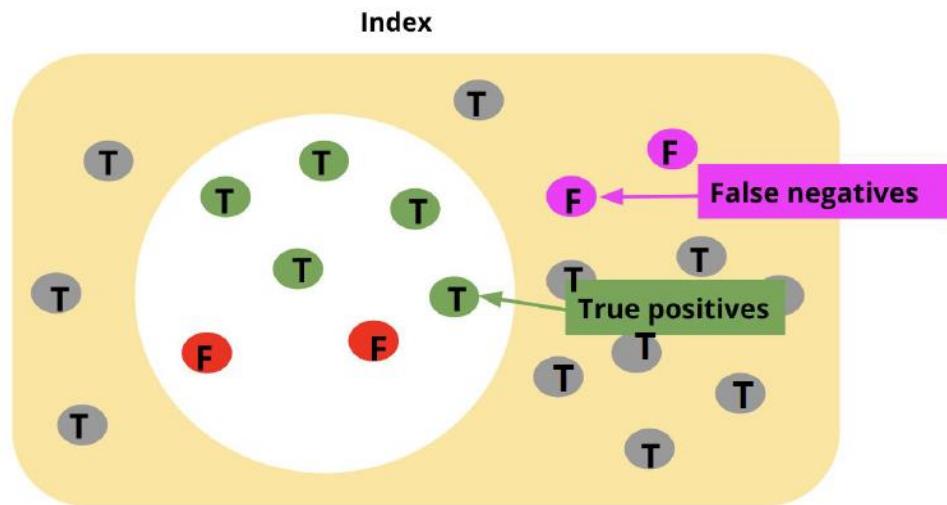
$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

What portion of the retrieved data is actually relevant to the search query?



Relevance

What is recall?



Recall =

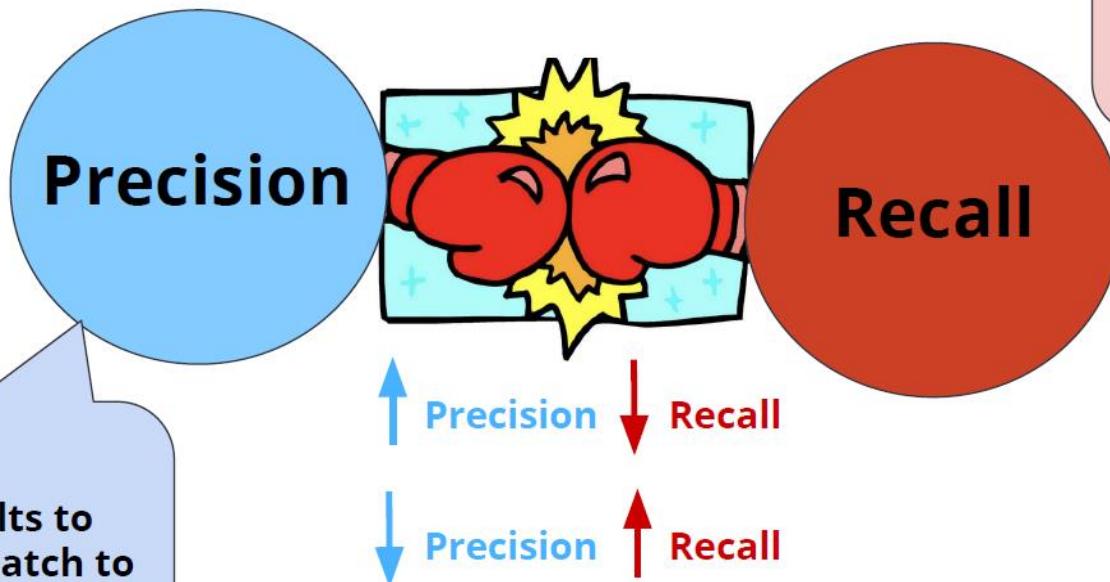
$$\frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

What portion of relevant data is being returned as search results?



Relevance

Precision and Recall are inversely related



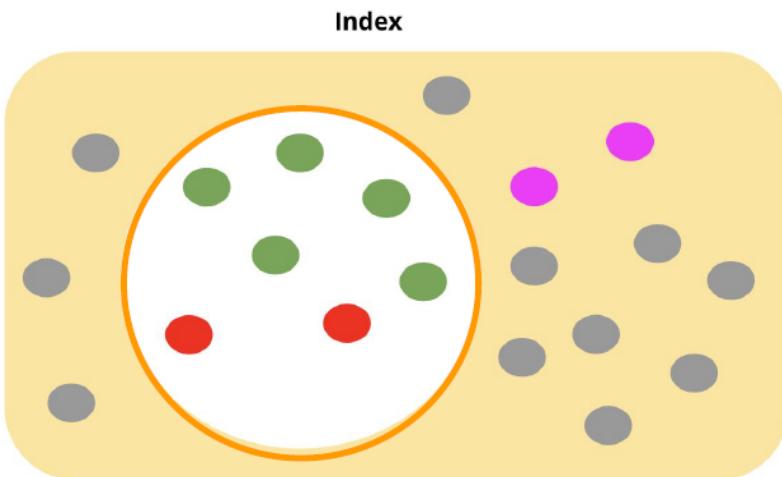
I want all the retrieved results to be a perfect match to the query, even if it means returning less documents.

I want to retrieve more results even if documents may not be a perfect match to the query.



Relevance

Precision and recall determine which documents are included in the search results.



Elastic Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

```
1 GET kibana_sample_data_ecommerce/_search
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

200 - success 76 ms

```
"hits" : {
  "total" : {
    "value" : 4675,
    "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
    {
      "_index" :
        "kibana_sample_data_ecommerce",
      "_type" :
        "doc",
      "_id" :
        "79WD1XYBy9gvFwLxZogX",
      "_score" :
        1.0,
      "_source" :
        {
          "category" : [
            "Men's Clothing"
          ],
          "currency" : "EUR",
          "customer_first_name" : "Eddie",
          "customer_full_name" : "Eddie Underwood".
        }
    }
  ]
}
```

Precision and recall do not determine which of the returned documents are more relevant compared to the other!



Relevance

Ranking refers to ordering of the results (from most relevant results at the top, to least relevant at the bottom).

The screenshot shows a search interface with a search bar containing the query "How to form good habits". Below the search bar, the results are ordered from "Most Relevant" at the top to "Least Relevant" at the bottom. The results are represented by ellipses (...).

Relevance Category	Score Description
Most Relevant	(Highest Score)
Less Relevant	
Least Relevant	(Lowest Score)



Score

- The score is a value that represents how relevant a document is to that specific query
- A score is computed for each document that is a hit



Score

- Term Frequency(TF)
- Inverse Document Frequency(IDF)



Score

The screenshot shows a search interface with a toolbar at the top featuring a magnifying glass icon, a home icon, and three colored dots (green, yellow, red). The search bar contains the text "How to form good habits" followed by a red curly brace and the text "Search Query". Below the search bar, the word "Hits" is centered. A large blue rounded rectangle highlights the search results, which are organized into three sections: "Most Relevant (Highest Score)", "Less Relevant", and "Least Relevant (Lowest Score)". Each section has three ellipsis (...) entries.

Hits	
Most Relevant	(Highest Score)
...	
...	
...	
Less Relevant	
...	
...	
...	
Least Relevant	(Lowest Score)



Score

Term Frequency(TF) determines how many times each search term appears in a document.

Search Terms

How to form good **habits** } Search Query

```
{  
  "title": "Atomic Habits",  
  "author": "James Clear",  
  "category": "self-help",  
  "description": "No matter your goals, Atomic Habits offers a proven  
    framework for improving every day. James clear, ... habits...habits  
    ...habits" TF= 4  
}  
  
{  
  "title": "The Mental Toughness Handbook",  
  "author": "Damon Zahariades",  
  "category": "self-help",  
  "description": "Imagine boldly facing any challenge that comes your way... 5  
    daily habits you must embrace to strengthen your mind and harden your  
    resolve. Why willpower and motivation are unreliable..." TF= 1  
}
```

If search terms are found in high frequency in a document, the document is considered more relevant to the search query.



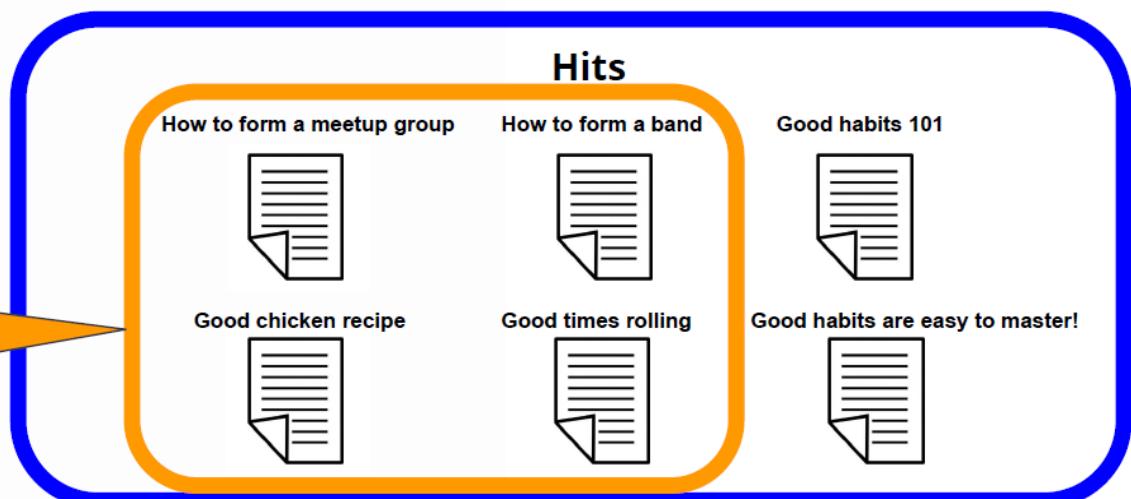
Score

What is Inverse Document Frequency(IDF)?

How to form good habits

We may contain some of the search terms but we have nothing to do with forming good habits!

IDF diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely!





Function Score Query, using Field Value Factor

- Predefined functions supported by Function Score:
 - weight: Apply a single enlargement to each document without the enlargement being normalized.
 - field_value_factor: Use the value of a field in the document to change the _score.
 - random_score: Use consistent random score to rank results differently for each user.
 - decay functions: Score a document with a function that decays depending on the distance of a numeric field value in the document from a user-determined source.
 - script_score: Use a custom script to take full control of the scoring logic.



Function Score Query, using Field Value Factor

- max_boost: limit the maximum effect the function can have.
- score_mode: Combines the result of the used functions. Score mode has the following parameters:
 - multiply: Function results are multiplied together (default).
 - sum: Function results are added up.
 - avg: The average of all the function results.
 - max: The highest function result is used.
 - min: The lowest function result is used.
 - first: Uses only the result from the first function that either doesn't have a filter or that has a filter matching the document.



Function Score Query, using Field Value Factor

- boost_mode: Combines the _score result of the query with the _score of the functions. Boost mode accepts the following parameters:
 - multiply: Multiply the _score with the function result.
 - sum: Add the function result to the _score.
 - min: The lower of the _score and the function result
 - max: The higher of the _score and the function result.
- - replace: Replace the _score with the function result



Function Score

- Start Searching for Apartments
- In the first query, we are searching for apartments that match our first criteria:
 - We need at least three rooms, but we would favor apartments with four rooms.
 - Apartments with four rooms get a boost of 1.

Refer github.com/eswaribala/rpsvelkjan2023/functionscore.txt



Function Score

rank	rooms	city	score
1	4.5	Biel	2.0
2	4.5	Brügg	2.0
3	4.5	Lyss	2.0
4	4.5	Bern	2.0
5	4.5	Grenchen	2.0
6	5.5	Schüpfen	2.0
7	4.5	zollikofen	2.0
8	3.5	Moosseedorf	1.0
9	3.5	Ostermundigen	1.0



Decay Functions

- This decay separates our results into three neat compartments with a very natural transition between all three parts.
- Apartments, that farther it gets from the origin price, receive a minor penalty or decay.
- Moderately more expensive apartments get a more substantial score penalty, increasing more and more within the scale.
- Apartments outside our preferred range get a severe punishment.
- Decay functions score a document with a function that decays depending on the distance of a numeric field value of the document from a user given origin.



Decay Functions

rank	price in CHF	rooms	city	score
1	1790	4.5	Brügg	2.8185682
2	1795	4.5	Zollikofen	2.7985601
3	1980	4.5	Lyss	1.6401769
4	1980	5.5	Grenchen	1.6401769
5	1990	4.5	Biel	1.5697317
6	2070	3.5	Ostermundigen	0.34841746
7	2100	3.5	Moosseedorf	0.29163226
8	2350	4.5	Schüpfen	0.115865104
9	2390	4.5	Bern	0.07667832



Combine Function Scores

- Let us assume I have switched my working location.
- Now I want to find a new apartment based on my new working site in a 20 km radius.
- Now we add another influence to our ranking: the location.
- My new origin is Lyss, a very family-friendly city, and my scale is 5 kilometres.
- We do a function score on a geo_point field and our scale is defined as 5 km.
- The new gauss function on the location is similar to the geo_distance query.
- We want to multiply the scores to get a ranking based on price and location.



Combine Function Scores

rank	price in CHF	rooms	city	distance	score
1	1980	4.5	Lyss	0 km	1.0934513
2	1790	4.5	Brügg	7 km	0.7212604
3	1990	4.5	Biel	10 km	0.117894106
4	2350	4.5	Schüpfen	8 km	0.022560354
5	1795	4.5	Zollikofen	16 km	0.009281355
6	2540	5.5	Grenchen	17 km	0.0023489068
7	2100	3.5	Moosseedorf	16 km	6.728557E-4
8	2070	3.5	Ostermundigen	21.5 km	2.0915837E-5
9	2390	4.5	Bern	20.5 km	9.355606E-6

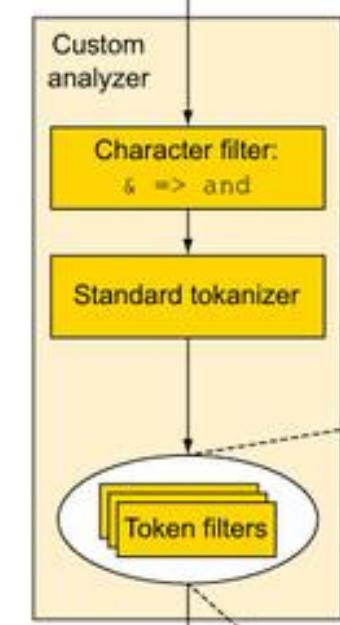


Analyzers and search queries



Original text

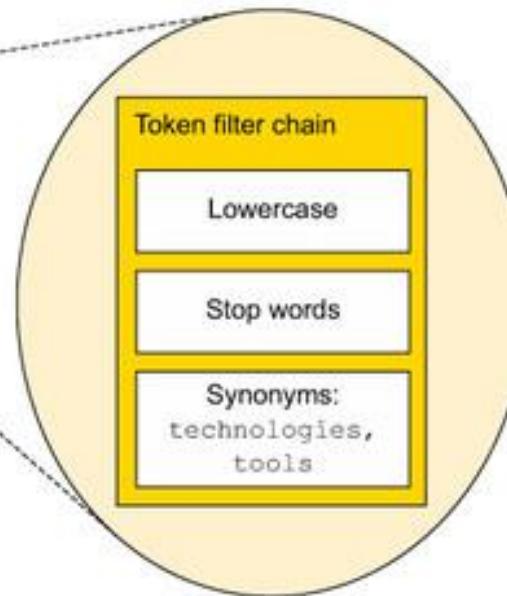
```
"share your  
experience with NoSql &  
big data technologies"
```



Refer tokenizer.txt

```
share your experience with NoSql and big data technologies
```

```
share your experience with NoSql and big data technologies
```



```
share  
your  
experience  
with  
nosql  
and  
big  
data  
tools  
technologies
```

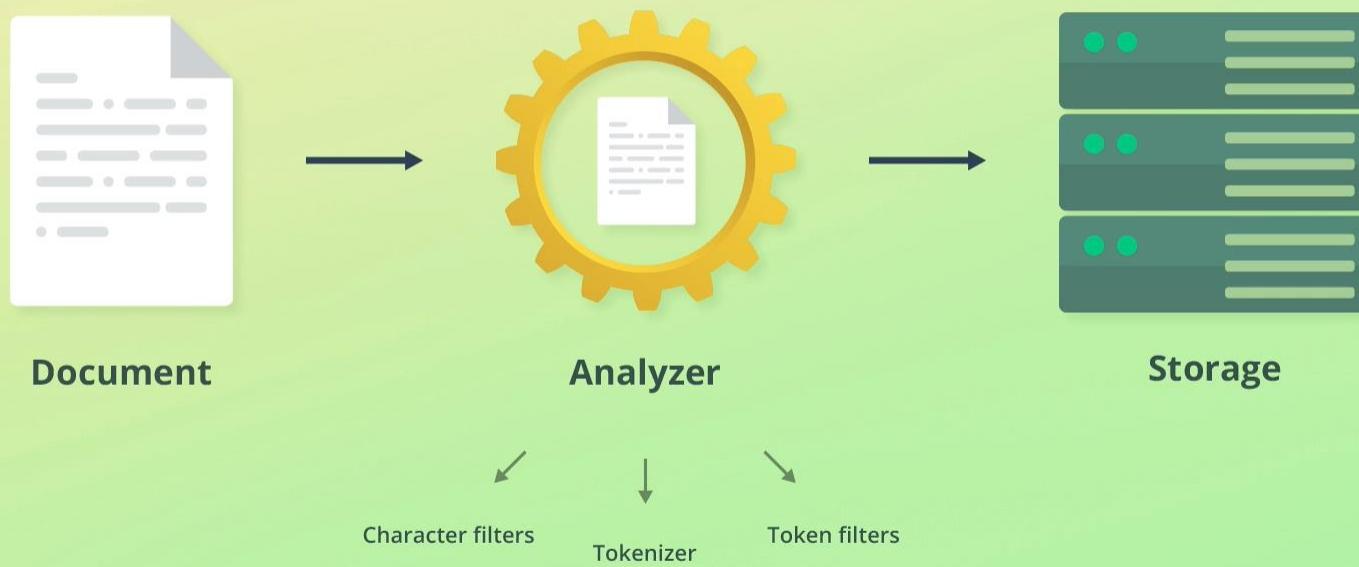


Analyzer

- Sometimes referred to as *text analysis*
- Applicable to text fields/values
- Text values are analyzed when indexing documents
- The result is stored in data structures that are efficient for searching etc.
- The `_source` object is **not** used when searching for documents
 - It contains the exact values specified when indexing a document



Analyzer





Character Filters

- Adds, removes, or changes characters
- Analyzers contain zero or more character filters
- Character filters are applied in the order in which they are specified
- Example (html_strip filter)
 - **Input:** "I'm in a good mood&nbs;-&nbs; and I love açai!"
 - **Output:** "I'm in a good mood - and I love açai!"



Tokenizers

- An analyzer contains **one** tokenizer
- Tokenizes a string, i.e. splits it into tokens
- Characters may be stripped as part of the tokenization
- Example
 - **Input:** "I REALLY like beer!"
 - **Output:** ["I", "REALLY", "like", "beer"]



Token Filters

- Receive the output of the tokenizer as input (i.e. the tokens)
- A token filter can add, remove, or modify tokens
- An analyzer contains zero or more token filters
- Token filters are applied in the order in which they are specified
- Example (lowercase filter)
 - **Input:** ["I", "REALLY", "like", "beer"]
 - **Output:** ["i", "really", "like", "beer"]



Filters

Character filters

(none)

Input: "I REALLY like beer!"
Output: "I REALLY like beer!"

Tokenizer

standard

Input: "I REALLY like beer!"
Output: ["I", "REALLY", "like", "beer"]

Token filters

["lowercase"]

Input: "I", "REALLY", "like", "beer"
Output: "i", "really", "like", "beer"



STANDARD ANALYZER



Filters

localhost:5601/app/kibana#/dev_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 537 ms

1 POST _analyze ▶ 🔍

2 {

3 "text": "Tom and Jerry always fights", ▶ 🔍

4 "analyzer": "standard"

5 }

6

7

1 "tokens": [

2 {

3 "token": "tom",

4 "start_offset": 0,

5 "end_offset": 3,

6 "type": "<ALPHANUM>",

7 "position": 0

8 },

9 {

10 "token": "and",

11 "start_offset": 4,

12 "end_offset": 7,

13 "type": "<ALPHANUM>",

14 "position": 1

15 },

16 {

17 "token": "jerry",

18 "start_offset": 8,

19 "end_offset": 13,

20 "type": "<ALPHANUM>",

21 "position": 2

22 },

23 {

24 "token": "always",

25 "start_offset": 14,

26 "end_offset": 20

27 }

Type here to search

02:39 ENG IN 02/06/2020 21



Filters

Complete Guide to Elasticsearch ✎ Elastic Kibana

localhost:5601/app/kibana#/dev_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

200 - OK 478 ms

```
1 POST _analyze
2 {
3   "text": "Tom and Jerry always fights",
4   "analyzer": "standard"
5 }
6
7 POST _analyze
8 {
9   "text": "Tom and Jerry always fights",
10  "char_filter": [],
11  "tokenizer": "standard",
12  "filter": ["lowercase"]
13 }
14
```

```
1 {
2   "tokens": [
3     {
4       "token": "tom",
5       "start_offset": 0,
6       "end_offset": 3,
7       "type": "<ALPHANUM>",
8       "position": 0
9     },
10    {
11      "token": "and",
12      "start_offset": 4,
13      "end_offset": 7,
14      "type": "<ALPHANUM>",
15      "position": 1
16    },
17    {
18      "token": "jerry",
19      "start_offset": 8,
20      "end_offset": 13,
21      "type": "<ALPHANUM>",
22      "position": 2
23    },
24    {
25      "token": "always",
26      "start_offset": 14,
27      "end_offset": 20
28    }
29  ]
30}
```

Type here to search

Windows taskbar icons: File Explorer, Edge, Google Chrome, Microsoft Edge, Microsoft Store, Microsoft Word, Microsoft Powerpoint, Microsoft Excel, Microsoft Access, Microsoft OneDrive, Microsoft Outlook, Microsoft Project, Microsoft Visio, Microsoft Publisher, Microsoft Word, Microsoft Powerpoint, Microsoft Excel, Microsoft Access, Microsoft OneDrive, Microsoft Outlook, Microsoft Project, Microsoft Visio, Microsoft Publisher.

System tray: ENG IN 02/06/2020 [21]



Tokens

Sentence → Tokens

"2 guys walk into a bar, but the third... DUCKS! :-)"



["2", "guys", "walk", "into", "a", "bar", "but", "the", "third", "ducks"]

TERM	DOCUMENT #1
2	X
a	X
bar	X
but	X
ducks	X
guys	X
into	X
the	X
third	X
walk	X



Sentence and Tokens

Sentence → Tokens

"2 guys walk into a bar, but the third... DUCKS! :-)"



["2", "guys", "walk", "into", "a", "bar", "but", "the", "third", "ducks"]

"2 guys went into a bar"



["2", "guys", "went", "into", "a", "bar"]

"2 ducks walk around the lake"



["2", "ducks", "walk", "around", "the", "lake"]

TERM	DOCUMENT #1	DOCUMENT #2	DOCUMENT #3
2	X	X	X
a	X	X	
around			X
bar	X	X	
but	X		
ducks	X		X
guys	X	X	
into	X	X	
lake			X
the	X		X
third	X		
walk	X		X
went		X	



Inverted Indices

- Mapping between terms and which documents contain them
- Outside the context of analyzers, we use the terminology “terms”
- Terms are sorted alphabetically
- Inverted indices contain more than just terms and document IDs
 - E.g. information for relevance scoring



Inverted Indices

```
{  
  "name": "Coffee Maker",  
  "description": "Makes coffee super fast!",  
  "price": 64,  
  "in_stock": 10,  
  "created_at": "2009-11-08T14:21:51Z"  
}
```

```
{  
  "name": "Toaster",  
  "description": "Makes delicious toasts...",  
  "price": 49,  
  "in_stock": 4,  
  "created_at": "2007-01-29T09:44:15Z"  
}
```

name field

TERM	DOCUMENT #1	DOCUMENT #2
coffee	X	
maker	X	
toaster		X

description field

TERM	DOCUMENT #1	DOCUMENT #2
coffee		X
delicious		X
fast		X
makes	X	X
super		X
toasts	X	



Inverted Indices

```
POST /stemming_test/_doc
{
  "description": "I loved drinking
  bottles of wine on last year's
  vacation."
}
```



```
GET /stemming_test/_search
{
  "query": {
    "match": {
      "description": "loves"
    }
  }
}
```

no matches

TERM	DOCUMENT #1	...
i	X	
loved	X	
drinking	X	
bottles	X	
of	X	
wine	X	
on	X	
last	X	
year's	X	
vacation	X	



Introduction to Stop Words

- Words that are filtered out during text analysis
 - Common words such as "a", "the", "at", "of", "on", etc.
- They provide little to no value for relevance scoring
- Fairly common to remove such words
 - Less common in Elasticsearch today than in the past
 - The relevance algorithm has been improved significantly
- Not removed by default, and I generally don't recommend doing so



Inverted Indices

- Mapping between terms and which documents contain them
- Outside the context of analyzers, we use the terminology “terms”
- Terms are sorted alphabetically
- Inverted indices contain more than just terms and document IDs
 - E.g. information for relevance scoring
- One inverted index per text field
- Other data types use BKD trees, for instance



Inverted Indices

- Values for a text field are analyzed and the results are stored within an inverted index
- Each field has a dedicated inverted index
- An inverted index is a mapping between terms and which documents contain them
- Terms are sorted alphabetically for performance reasons
- Created and maintained by Apache Lucene, *not* Elasticsearch

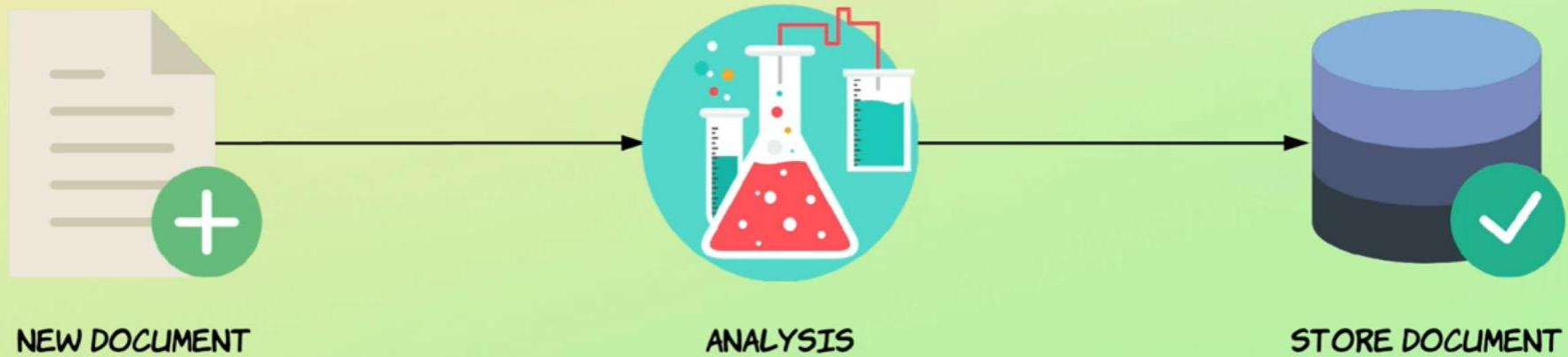


Inverted Indices

- Inverted indices enable **fast** searches
- Inverted indices contain other data as well
 - E.g. things used for relevance scoring (covered later)
- Elasticsearch (technically, Apache Lucene) uses other data structures as well
 - E.g. BKD trees for numeric values, dates, and geospatial data



Analysis Process





Analysis Process

CHARACTER FILTERS

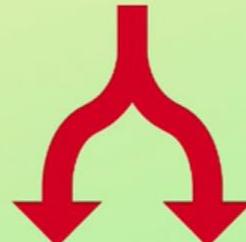
Two words!



Two words!

TOKENIZER

Two words!



[Two, words]

TOKEN FILTERS

[Two, words]



[two, words]



Analysis Process

CHARACTER FILTERS



TOKENIZER

I'm in the mood for drinking
semi-dry red wine!



[I'm, in, the, mood, for, drinking,
semi, dry, red, wine]

TOKEN FILTERS



[I'm, in, the, mood, for, drinking,
semi, dry, red, wine]

[i'm, in, the, mood, for, drinking,
semi, dry, red, wine]



Standard Analyzer

"I loved drinking bottles of
wine on last year's vacation."



["i", "love", "drink", "bottl",
"of", "wine", "on", "last",
"year", "vacat"]

```
GET /stemming_test/_search
{
  "query": {
    "match": {
      "description": "drinking"
    }
  }
}
```



Stemming Analyzer

```
{
  "properties": {
    "description": {
      "type": "text",
      "analyzer": "stemming_analyzer"
    }
  }
}
```

```
POST /stemming_test/_doc
{
  "description": "I loved drinking
  bottles of wine on last year's
  vacation."
}
```



```
["i", "love", "drink", "bottl",
 "of", "wine", "on", "last",
 "year", "vacat"]
```