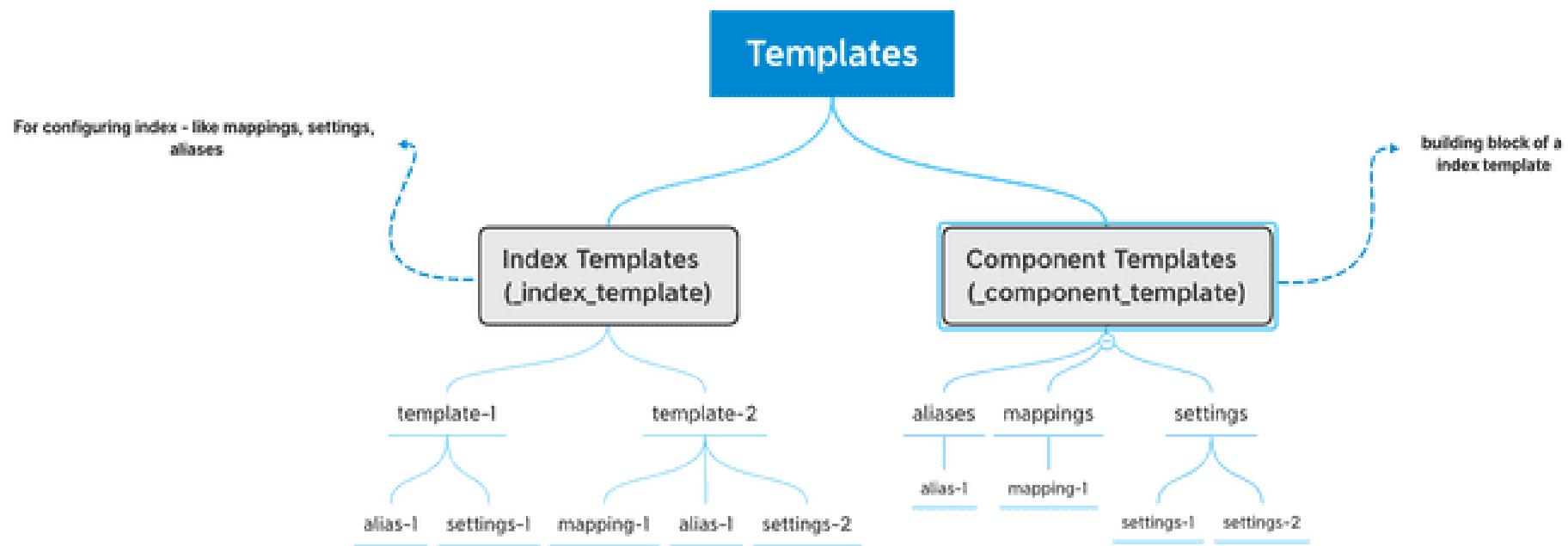




# Templates





# Templates

---

- Index templates contain settings like how many shards and replicas the index should initialize with, what mapping settings, aliases to use. One can also assign priority to the index template.
- 100 is the default priority.
- Component templates are nothing but building blocks for index templates.
- Create an alias component template, settings component template, or a mapping component template and use them via the "composed\_of" parameter.
- Elastic has built-in index templates which maps to index/datasream patterns logs-\*-\* ,metrics-\*-\* , synthetics-\*-\* , which have default priority of 100. To create index template's which overrides the built-in index templates but use the same patterns, assign a priority that is above 100.
- To disable the built-in index and component templates, set the stack.templates.enabled to false.



# Index Management

---

- Data in Elasticsearch is stored in one or more indices.
- Elasticsearch typically deal with large volumes of data, data in an index is partitioned across shards to make storage more manageable.
- An index may be too large to fit on a single disk, but shards are smaller and can be allocated across different nodes as needed.
- Another benefit of proper sharding is that searches can be run across different shards in parallel, speeding up query processing.
- The number of shards in an index is decided upon index creation and cannot be changed later.



# Index Management

---

- Sharding an index is useful, still only a single copy of each document in the index, which means there is no protection against data loss.
- We can set up replication.
- Each shard may have several replicas, which are configured upon index creation and may be changed later.
- The primary shard is the main shard that handles the indexing of documents and can also handle processing of queries.
- The replica shards process queries but do not index documents directly.
- They are always allocated to a different node from the primary shard, and, in the event of the primary shard failing, a replica shard can be promoted to take its place.



# Index Management

---

- While more replicas provide higher levels of availability in case of failures, it is also important not to have too many replicas.
- Each shard has a state that needs to be kept in memory for fast access.
- The more shards we use, the more overhead can build up and affect resource usage and performance.



# Index Lifecycle Management

---

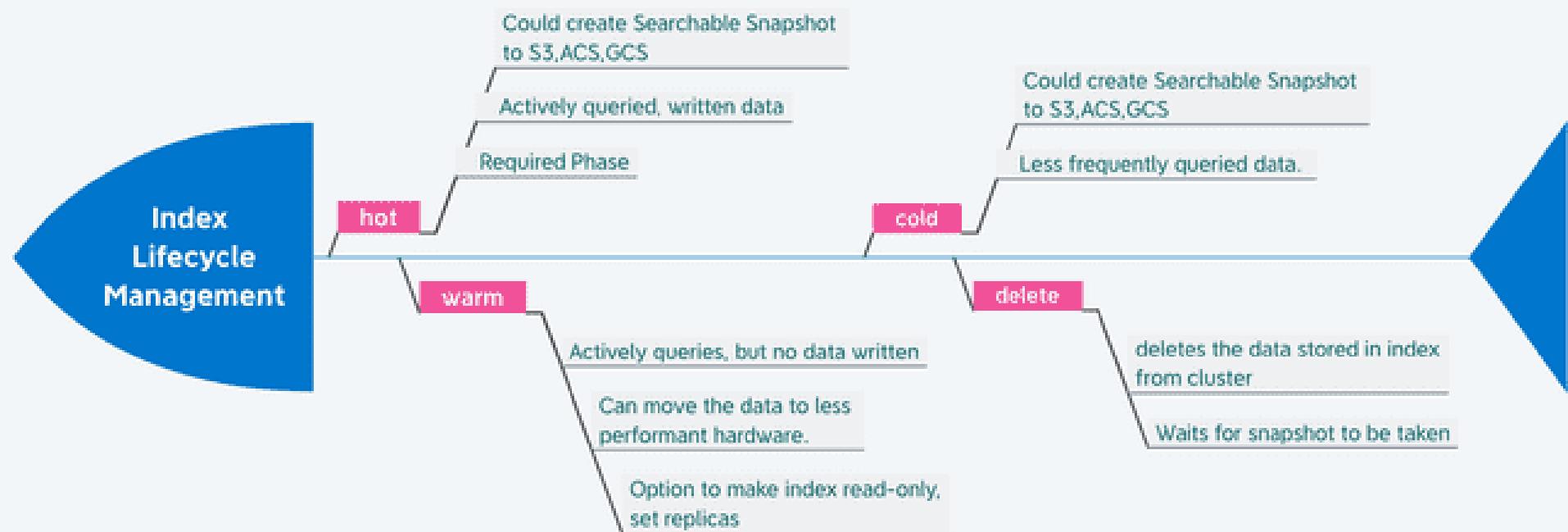
- Index Lifecycle management helps manage all data in the indices, leverage the underlying hardware capabilities.
- Time-series Data has different tenures, and only a sizeable chunk needs to be actively queried all the time



# Index Lifecycle Management

## Index Lifecycle Phases

1. Phases transition based on age/# of days set.
2. Actions designated are executed in each phase before moving to another.
3. Example Actions: Unfollow, Rollover, Shrink, Force-merge, Allocate Migrate, Freeze, Searchable Snapshot, Delete





# Index Lifecycle Management

---

- ILM has policies that are instructions configured to execute on indices containing data.
- These instructions are like triggers that create, rollover, deletes data based on set conditions.
- There can be multiple ILM policies with different priorities set.
- For example, if we are ingesting 1 GB, logs, metrics, and traces data from the apps and infrastructure.
- By the end of the month, it becomes 30 GB, quarter: 90 GB, An year: 360+ GB.



# Index Lifecycle Management

---

- We might not need entire raw data to gather insights, as it does not change with time, but you might be interested in comparing trends.
- With ILM, you could gather critical insights and rollover the raw data to less expensive hardware-based nodes or delete the cluster's unneeded data.
- Besides, using the data tiers (hot, warm, cold), you could move the index data to designated nodes and have faster hardware for faster performance.
- ILM automates this entire process of moving data to different types of nodes.



# Split API on Index

- The split index API allows you to split an existing index into a new index, where each original primary shard is split into two or more primary shards in the new index.
- The number of times the index can be split (and the number of shards that each original shard can be split into) is determined by the `index.number_of_routing_shards` setting.
- The number of routing shards specifies the hashing space that is used internally to distribute documents across shards with consistent hashing.
- For instance, a 5 shard index with `number_of_routing_shards` set to 30 ( $5 \times 2 \times 3$ ) could be split by a factor of 2 or 3.
- In other words, it could be split as follows:
  - $5 \rightarrow 10 \rightarrow 30$  (split by 2, then by 3)
  - $5 \rightarrow 15 \rightarrow 30$  (split by 3, then by 2)
  - $5 \rightarrow 30$  (split by 6)



# How Splitting Works

---

- A split operation:
- Creates a new target index with the same definition as the source index, but with a larger number of primary shards.
- Hard-links segments from the source index into the target index.
- Hashes all documents again, after low level files are created, to delete documents that belong to a different shard.
- Recovers the target index as though it were a closed index which had just been re-opened.



# Alias Indexing

---

- Add an alias
- The following request adds the alias1 alias to the test1 index.
- POST /\_aliases
- {
- "actions" : [
- { "add" : { "index" : "test1", "alias" : "alias1" } }
- ]
- }



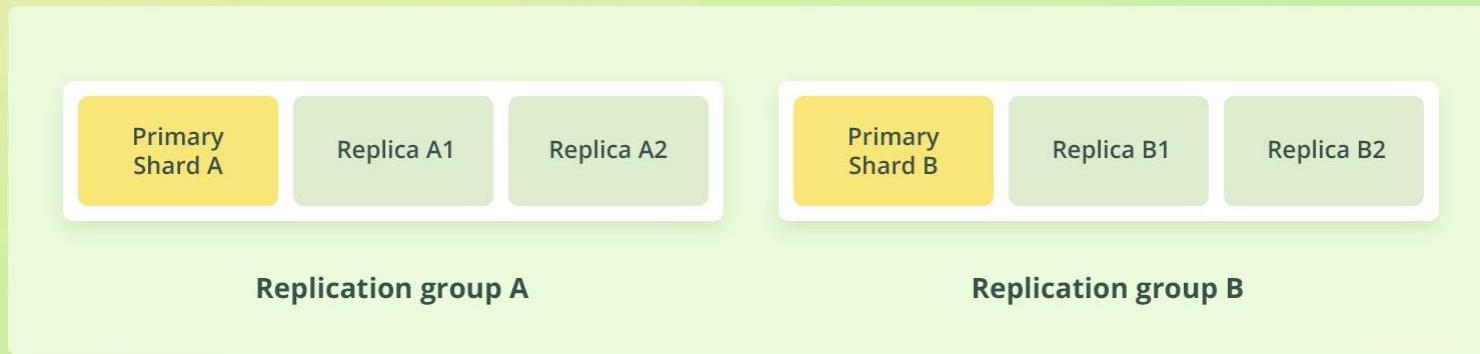
# How does replication work

- Replication is configured at the index level
- Replication works by creating copies of shards, referred to as *replica shards*
- A shard that has been replicated, is called a *primary shard*
- A primary shard and its replica shards are referred to as a *replication group*
- Replica shards are a complete copy of a shard
- A replica shard can serve search requests, exactly like its primary shard
- The number of replicas can be configured at index creation



# How does replication work

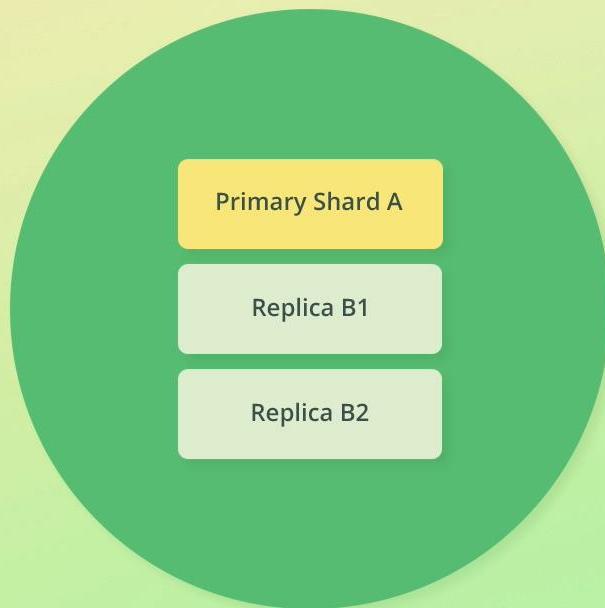
## Example Index



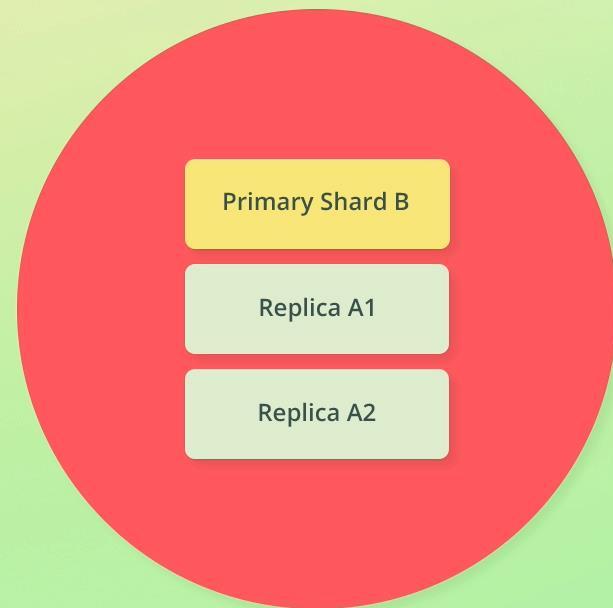


# How does replication work

**Node A**



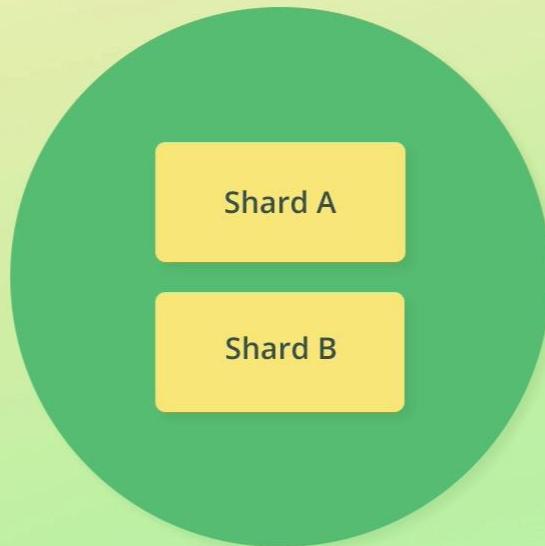
**Node B**



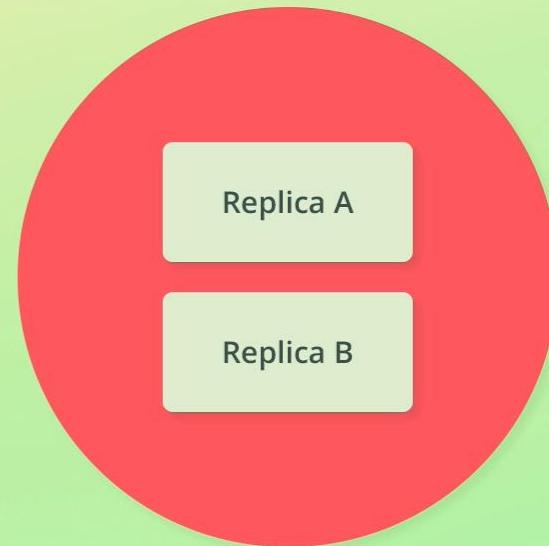


# How does replication work

**Node A**



**Node B**





# Choosing number of replica shards

- How many replica shards are ideal, depends on the use case
- E.g. is the data stored elsewhere, such as in a RDBMS?
- Is it OK for data to be unavailable while you restore it?
- For mission critical systems, downtime is not acceptable
- Replicate shards *once* if data loss is not a disaster
- For critical systems, data should be replicated *at least* twice



# Shards

---

- Elasticsearch supports taking snapshots as backups
- Snapshots can be used to restore to a given point in time
- Snapshots can be taken at the index level, or for the entire cluster
- Use snapshots for backups, and replication for high availability (and performance)



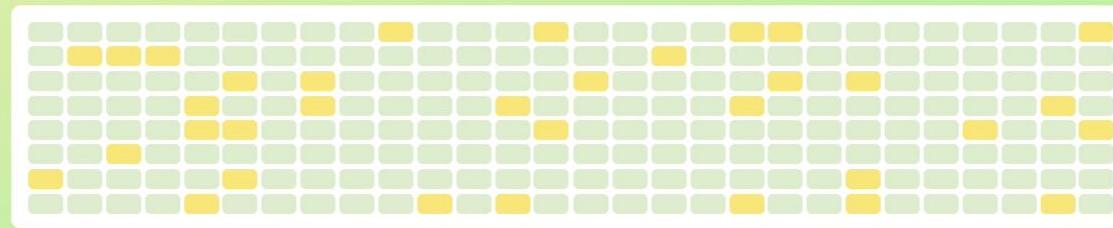
# Index



Take  
snapshot



Rollback to  
snapshot



## Index

with 1 million documents

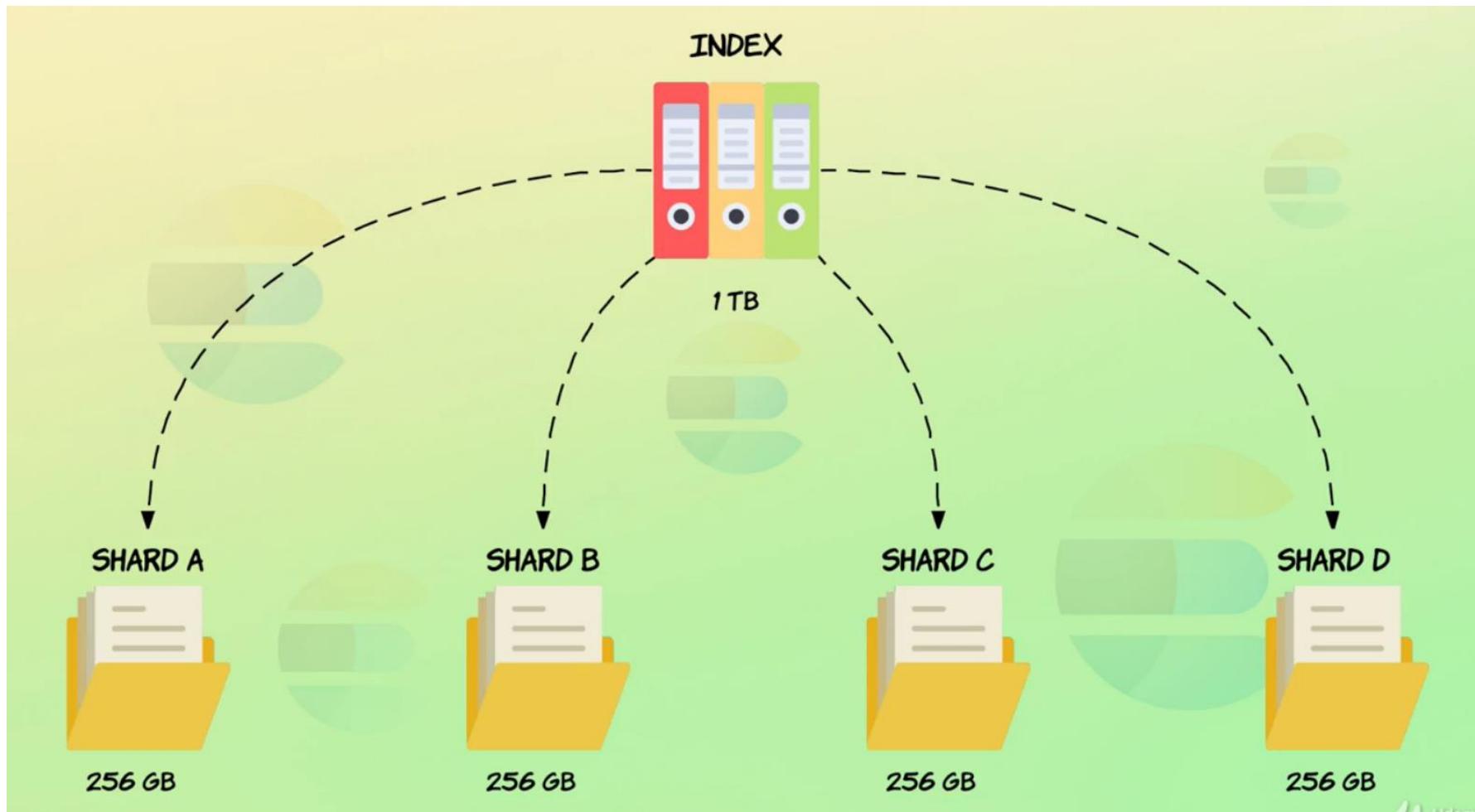


# Increasing query throughput with replication

- Replica shards of a replication group can serve different search requests simultaneously
  - This increases the number of requests that can be handled at the same time
- Elasticsearch intelligently routes requests to the *best* shard (more on that later)
- CPU parallelization improves performance if multiple replica shards are stored on the same node

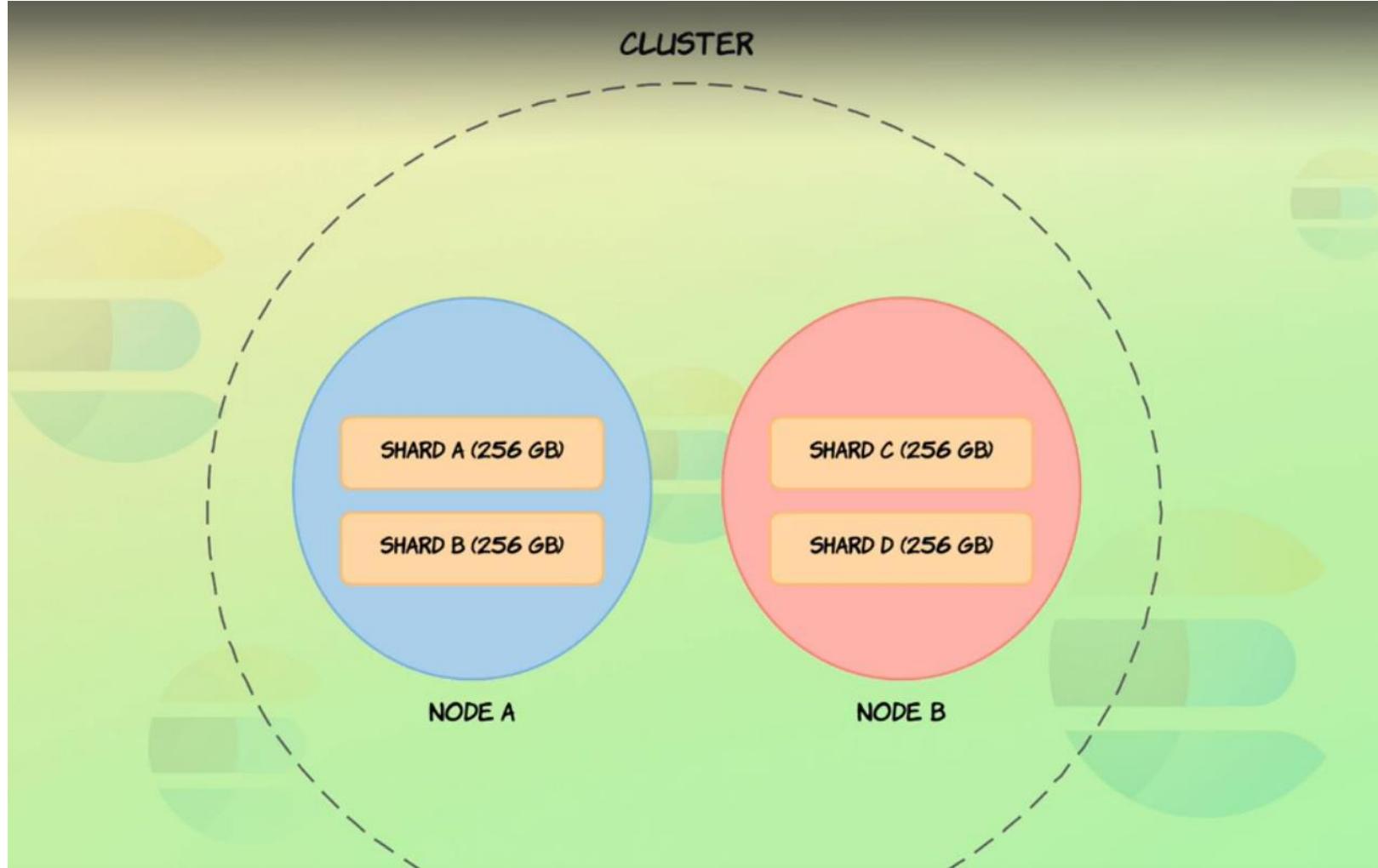


# Sharding



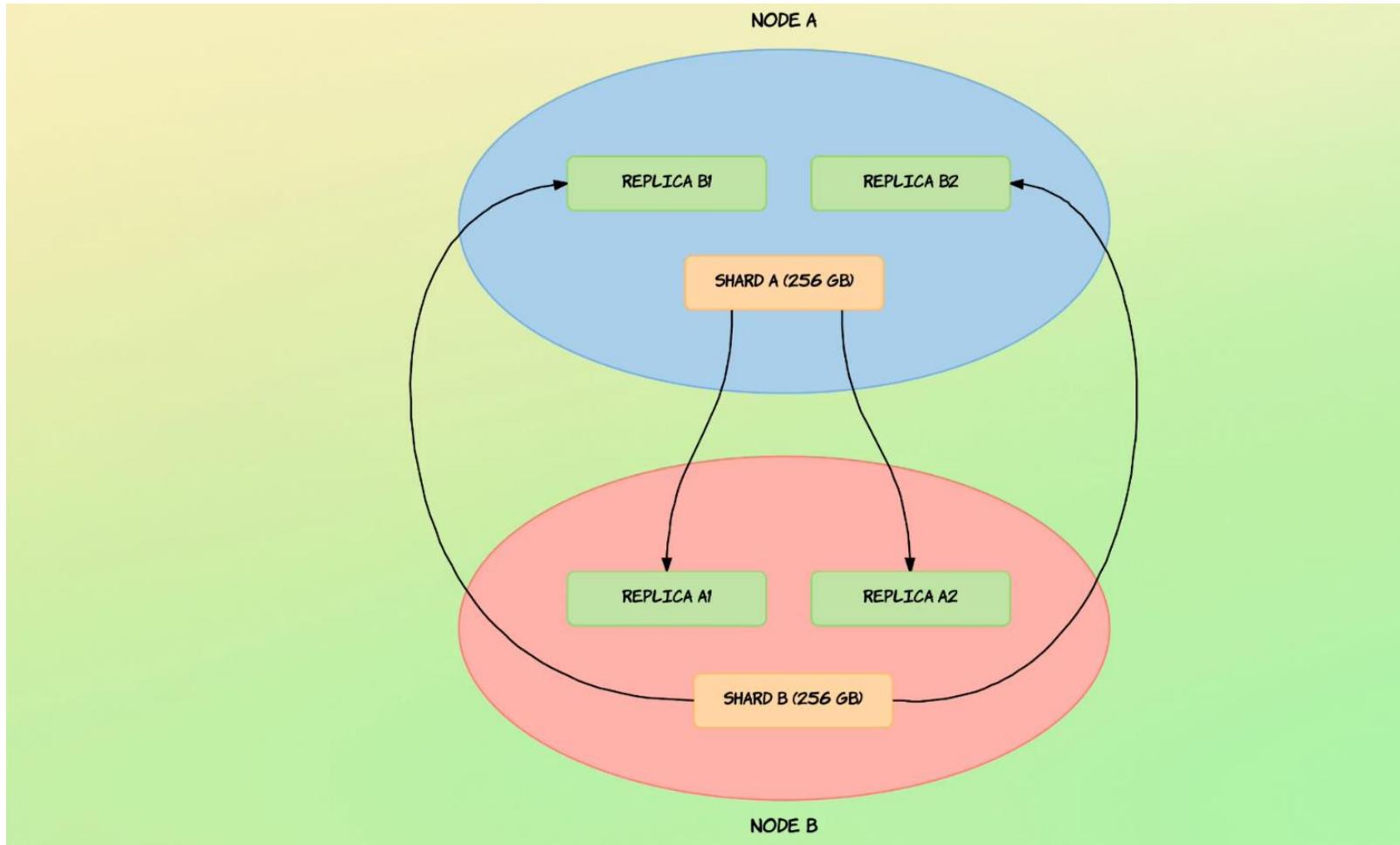


# Sharding



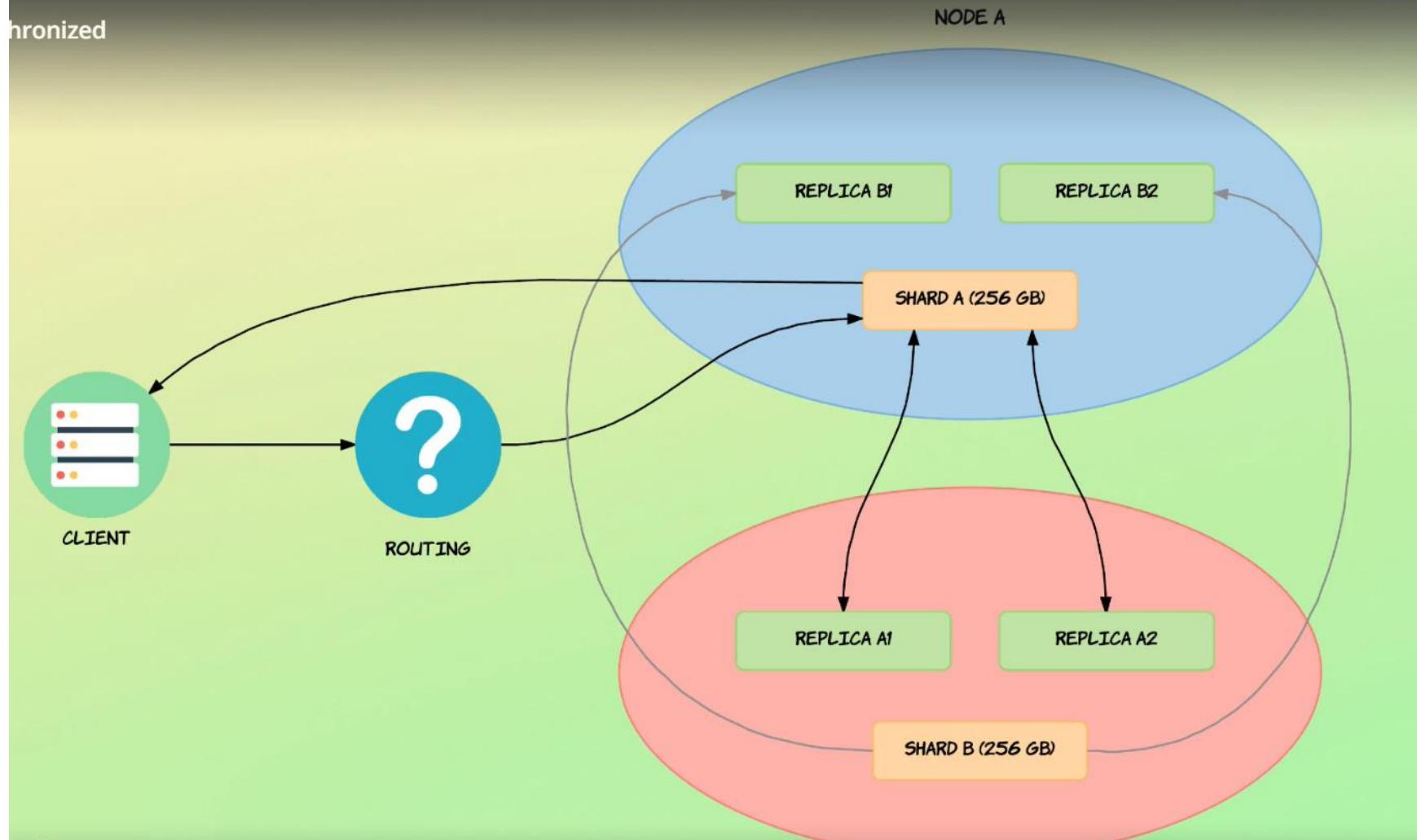


# Replicas





# Replicas and Shards





## Elastic Health

---

- **Red:** Some or all primary shards are not ready to receive the requests.
- **Yellow:** All primary shards are allocated but some or all of the replicas are unallocated. For single node clusters, the normal status should be yellow because no other node is available for replication.
- **Green:** All shards and their replicas are well allocated and the cluster is working fine.



# Using Indices



## RESTful API

Elasticsearch fundamentally works via HTTP requests and JSON data. Any language or tool that can handle HTTP can use Elasticsearch.



## Client API's

Most languages have specialized Elasticsearch libraries to make it even easier.



## Analytic Tools

Web-based graphical UI's such as Kibana let you interact with your indices and explore them without writing code.



# Connecting To Your Cluster

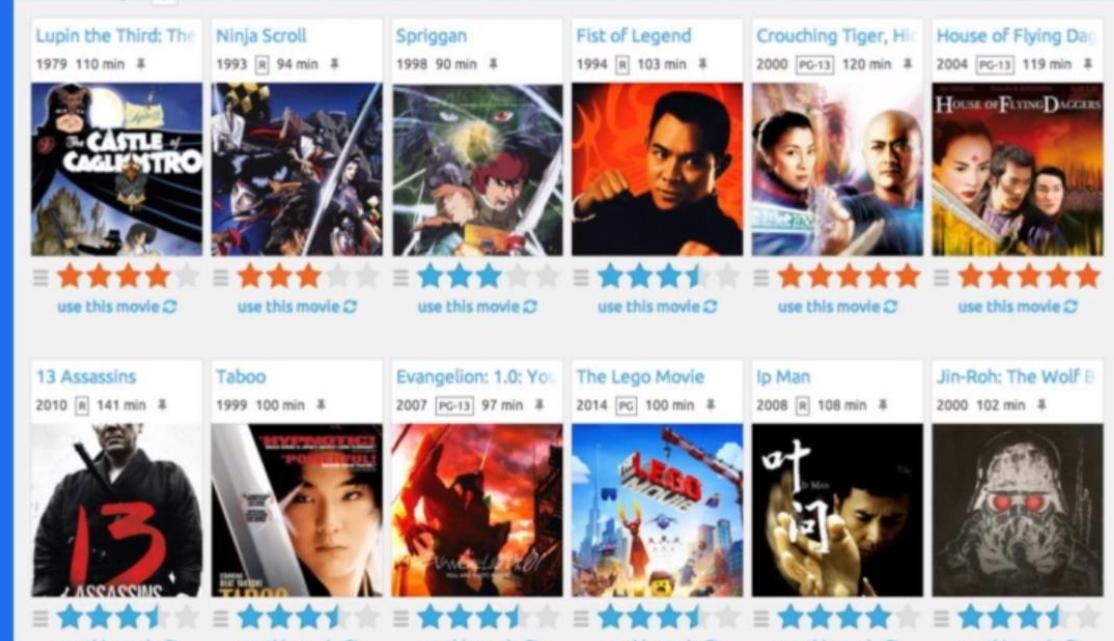


# movielens

Movielens is a free dataset of movie ratings gathered from movielens.org.

It contains user ratings, movie metadata, and user metadata.

Let's download and examine the data files from movielens.org





Complete Guide to Elasticsearch | localhost:9200 | Elasticsearch 7 and the Elastic Stack | MovieLens | GroupLens

groupLens.org/datasets/movielens/

62,000 movies by 162,000 users. Includes tag genome data with 15 million relevance scores across 1,129 tags.  
Released 12/2019

- [README.txt](#)
- [ml-25m.zip](#) (size: 250 MB, [checksum](#))

Permalink: <https://grouplens.org/datasets/movielens/25m/>

[Personality\\_2018](#)

[Learning from Sets of Items 2019](#)

---

## recommended for education and development

### MovieLens Latest Datasets

These datasets will change over time, and are not appropriate for reporting research results. We will keep the download links stable for automated downloads. We will not archive or make available previously released versions.

*Small*: 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. Last updated 9/2018.

- [README.html](#)
- [ml-latest-small.zip](#) (size: 1 MB)

*Full*: 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users. Includes tag genome data with 14 million relevance scores across 1,100 tags. Last updated 9/2018.

- [README.html](#)
- [ml-latest.zip](#) (size: 265 MB)

Permalink: <https://grouplens.org/datasets/movielens/latest/>

---

## synthetic datasets





# Search Query





```
curl -XGET "http://localhost:9200/.kibana/_search" -H 'Content-Type: application/json' -d'{ "query":{ "match_all":{ }} }'
```

Complete Guide to Elasticsearch x Elastic Kibana x

localhost:5601/app/kibana#/dev\_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

Dev Tools Dismiss

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 134 ms

Copy as cURL

Open documentation

Auto indent

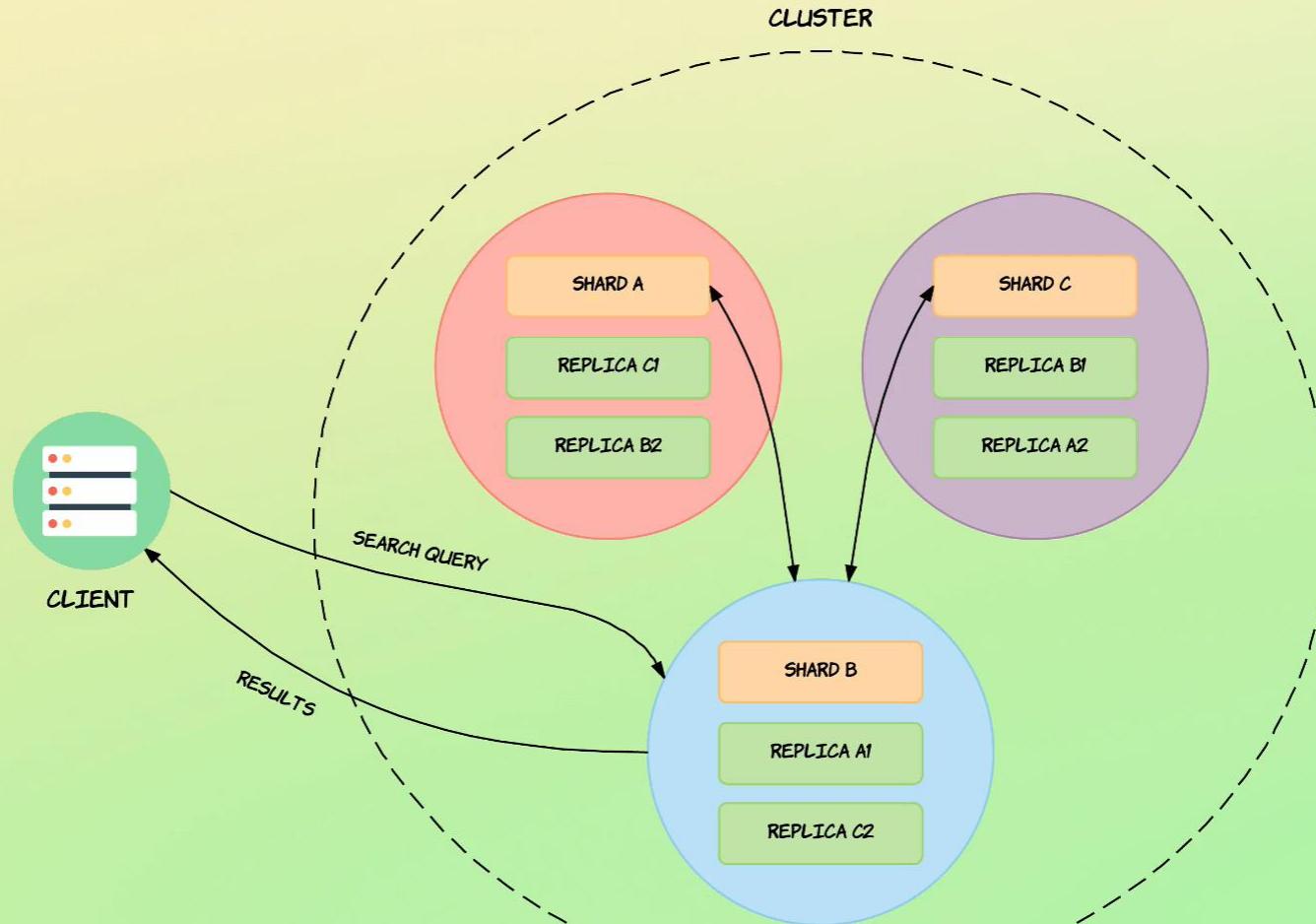
Request copied as cURL

```
1 GET /.kibana/_search
2 {
3
4   "query":{
5     "match_all":{
6
7   }
8 }
9 }
```

```
1 {
2   "took" : 41,
3   "out" : false,
4   "shards" : [
5     {
6       "index" : 1,
7       "successful" : 1,
8       "failed" : 0,
9       "unindexed" : 0
10    }
11   ],
12   "total" : {
13     "value" : 70,
14     "relation" : "eq"
15   },
16   "max_score" : 1.0,
17   "hits" : [
18     {
19       "_index" : ".kibana_1",
20       "_type" : "_doc",
21       "_id" : "space:default",
22       "_score" : 1.0,
23       "_source" : {
24         "space" : {
25           "id" : "space:default"
26         }
27       }
28     }
29   ]
30 }
```

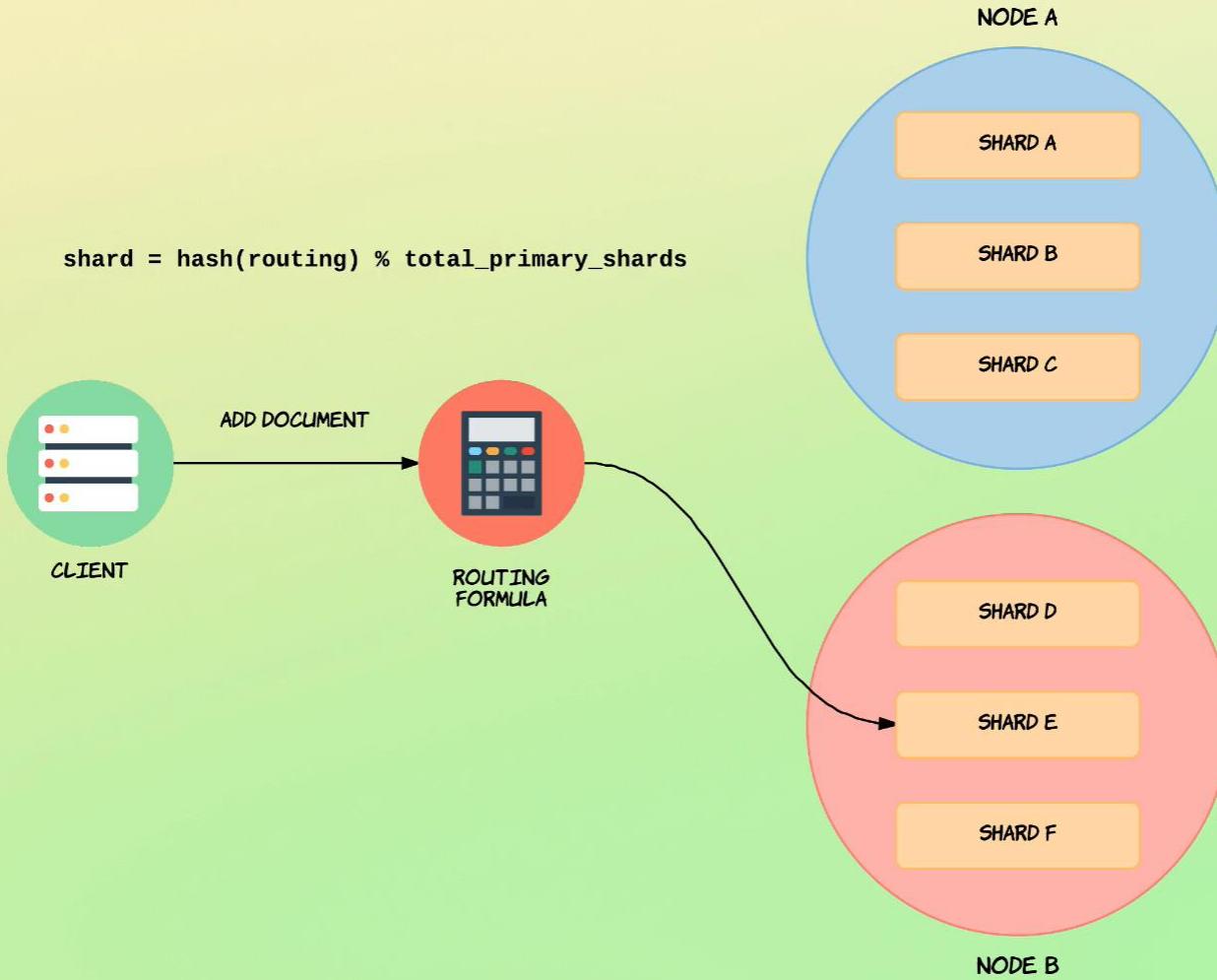


# Search Query



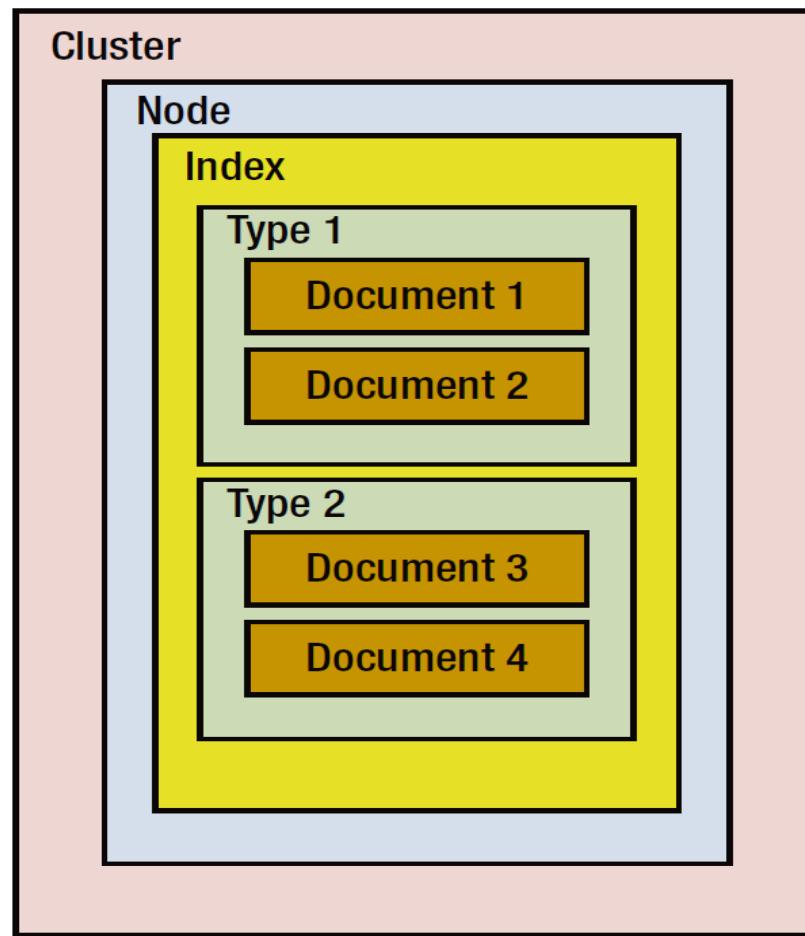


# Search Query





# Search Query





# Elastic Cloud



Clusters Help

Your free trial ends in 14 days

Account

Sign out

## Summary

Platform	Amazon Web Services
Region	US East (N. Virginia)
Memory	4 GB
Storage	96 GB
SSD	Yes
High availability	Yes
Hourly rate	\$0.3125
Monthly rate	\$228

Create

## Get started with Elastic Cloud

During your free trial period, you can provision a single cluster with up to 4GB memory, 96GB storage, and high availability (HA) across two zones for 14 days. You also get a chance to explore [Shield](#), [Watcher](#), [Marvel](#), and [Kibana](#).

[Adding a credit card](#) will prevent your cluster from being stopped after 14 days, and let you create additional and larger clusters. A 4GB cluster with HA will remain free for 14 days. Additional and/or larger clusters will accrue charges.

Should you have any technical questions, email us at [support@elastic.co](mailto:support@elastic.co). We also offer world class, SLA-based [premium support](#).

## Cluster Size

Choose a cluster size. Cluster size can be changed later without downtime.



SSD — Selected for improved storage performance.

Need a larger cluster? [Contact us](#).

## Cloud Platform

Pick your cloud:



Choose a region near you:

US East (N. Virginia)

US West (N. California)

US West (Oregon)

EU (Ireland)

Asia Pacific (Singapore)

Asia Pacific (Tokyo)

South America East

Asia Pacific (Sydney)

EU (Frankfurt)





# Managing Documents – Create Index

localhost:5601/app/kibana#/dev\_tools/console

Dismiss

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 2710 ms

PUT /pages

```
1 {  
2   "acknowledged" : true,  
3   "shards_acknowledged" : true,  
4   "index" : "pages"  
5 }  
6
```



# Managing Documents – Create Index

A screenshot of the Kibana Dev Tools interface showing the results of a search for indices. A blue arrow points from the text "Index created" in the previous slide to the "yellow" status column in the table below.

The Kibana Dev Tools interface includes:

- Dev Tools sidebar with various icons.
- Header bar with tabs: Console (selected), Search Profiler, Grok Debugger, Painless Lab (BETA).
- History, Settings, Help menu.
- Search bar at the bottom.
- Table results:

ID	Status	Open	Type	Index	Shards	Primary	Replicas	Size	Index Size	docs	Size
1	yellow	open	filebeat-data	YRwag81eSWO_jnC44JzifA	1	1	125	0	245kb	245kb	
2	yellow	open	pages	qhzP3nTVQBqzYSC7oov8ja	1	1	0	0	208b	208b	
3	green	open	.apm-custom-link	X_8ja4QjTCKPTGUiuKQZ8A	1	0	0	0	208b	208b	
4	green	open	.kibana_task_manager_1	dYdBqltzTSuibsoSncnAtg	1	0	5	4	39.8kb	39.8kb	
5	yellow	open	logstash-2020.05.31-000001	HIM2I-K4Rm6H59G1irUeSQ	1	1	5	0	38.4kb	38.4kb	
6	yellow	open	nginx-test	3NSaFtCIRXWtg2gjQ9e9CA	1	1	2	0	16.1kb	16.1kb	
7	green	open	.apm-agent-configuration	BAmQ09WhSN-JFQEe1j5t1Q	1	0	0	0	208b	208b	
8	yellow	open	tmax115-data	JHDez_SuRA-pBtXiu69gZQ	1	1	9	0	28.6kb	28.6kb	
9	yellow	open	tomcat-data	72arsNi1Rhm571JKbw1YfQ	1	1	44	0	71.3kb	71.3kb	
10	green	open	.kibana_1	SP4_xG0MTE6_VsiU1ABE5A	1	0	80	3	67.1kb	67.1kb	
11	yellow	open	tomcat-pipeline-data	oDZS9Ko0TV-AeSAgrvwhSQ	1	1	44	0	70.5kb	70.5kb	
12											

# Managing Documents – Create Index



The screenshot shows the Elasticsearch Dev Tools Console in Kibana. The top navigation bar includes tabs for 'Console' (selected), 'Search Profiler', 'Grok Debugger', and 'Painless Lab (BETA)'. Below the tabs, there are links for 'History', 'Settings', and 'Help'. The main area displays a table of index statistics:

	Index	Type	Size	IP	Node		
1	nginx-test	p	STARTED	2	16.1kb	127.0.0.1	DESKTOP-55AGIOI
2	nginx-test	r	UNASSIGNED				
3	tomcat-pipeline-data	p	STARTED	44	70.5kb	127.0.0.1	DESKTOP-55AGIOI
4	tomcat-pipeline-data	r	UNASSIGNED				
5	.apm-agent-configuration	p	STARTED	0	208b	127.0.0.1	DESKTOP-55AGIOI
6	tmax115-data	p	STARTED	9	28.6kb	127.0.0.1	DESKTOP-55AGIOI
7	tmax115-data	r	UNASSIGNED				
8	pages	p	STARTED	0	208b	127.0.0.1	DESKTOP-55AGIOI
9	pages	r	UNASSIGNED				
10	.apm-custom-link	p	STARTED	0	208b	127.0.0.1	DESKTOP-55AGIOI
11	.kibana_1	p	STARTED	81	77.3kb	127.0.0.1	DESKTOP-55AGIOI
12	tomcat-data	p	STARTED	44	71.3kb	127.0.0.1	DESKTOP-55AGIOI
13	tomcat-data	r	UNASSIGNED				
14	.kibana_task_manager_1	p	STARTED	5	39.8kb	127.0.0.1	DESKTOP-55AGIOI
15	logstash-2020.05.31-000001	p	STARTED	5	38.4kb	127.0.0.1	DESKTOP-55AGIOI
16	logstash-2020.05.31-000001	r	UNASSIGNED				
17	filebeat-data	p	STARTED	125	245kb	127.0.0.1	DESKTOP-55AGIOI
18	filebeat-data	r	UNASSIGNED				
19							



# Creating and Deleting indices

Screenshot of the Elastic Kibana Dev Tools Console tab showing a DELETE request to the /pages index.

The Dev Tools sidebar is visible on the left, and the top navigation bar shows "localhost:5601/app/kibana#/dev\_tools/console".

The console output:

```
1 DELETE /pages
2   "acknowledged" : true
3 }
```

Response status: 200 - OK | 830 ms

Bottom taskbar icons include: Start button, Search bar, File, Task View, Mail (64), Microsoft Store, Edge, Chrome, File Explorer, Microsoft Word, Microsoft Powerpoint, Microsoft Excel, Microsoft Teams, and Microsoft Edge.



# Creating and Deleting indices

Saved

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 2066 ms

1 `DELETE /pages` `PUT /products` `▶ 🔍`

2 `PUT /products`

3 `{`

4 `"settings":{`

5 `"number_of_shards":2,`

6 `"number_of_replicas":2`

7 `}`

8 `}`

9

1 `{`

2 `"acknowledged" : true,`

3 `"shards_acknowledged" : true,`

4 `"index" : "products"`

5 `}`

6

||

Type here to search O 64 Microsoft Edge Chrome File WS P 00:33 ENG IN 02/06/2020 21

```
1 DELETE /pages
2 PUT /products
3 {
4   "settings":{
5     "number_of_shards":2,
6     "number_of_replicas":2
7   }
8 }
9
10
11 {
12   "acknowledged" : true,
13   "shards_acknowledged" : true,
14   "index" : "products"
15 }
```



# Create Indexing documents

Screenshot of the Elastic Kibana Dev Tools Console tab showing a successful POST request to index a document.

The URL in the browser bar is `localhost:5601/app/kibana#/dev_tools/console`.

The Dev Tools interface shows the following:

- Console Tab:** Active tab.
- Search Profiler Tab:**
- Grok Debugger Tab:**
- Painless Lab Tab:** Labeled "BETA".

**History:** Shows the recent operation:

```
1 POST /products/_doc
2 {
3   "productId":4974,
4   "name":"flask",
5   "qty":89
6 }
7 }
```

**Response:** 201 - Created | 755 ms

```
1 { "_index": "products",
2   "_type": "_doc",
3   "_id": "dGJNcXIBxt9H8GTT2c0_",
4   "_version": 1,
5   "result": "created",
6   "_shards": {
7     "total": 3,
8     "successful": 1,
9     "failed": 0
10   },
11   "_seq_no": 0,
12   "_primary_term": 1
13 }
14
15
```

Two blue arrows point from the right side of the slide towards the response JSON object in the Kibana console.



# Create Indexing documents

Complete Guide to Elasticsearch x Elastic Kibana x

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

201 - Created 142 ms

```
1 POST /products/_doc
2 {
3   "productId":4974,
4   "name":"flask",
5   "qty":89
6 }
7 ^
8 POST /products/_doc/123
9 {
10   "productId":4975,
11   "name":"Heater",
12   "qty":89
13 }
14 ^
15 PUT /products/_doc/156
16 {
17   "productId":4974,
18   "name":"flask",
19   "qty":89
20 }
21 ^
```

1 {  
2 "\_index" : "products",  
3 "\_type" : "\_doc",  
4 "\_id" : "156",  
5 "\_version" : 1,  
6 "result" : "created",  
7 "\_shards" : {  
8 "total" : 3,  
9 "successful" : 1,  
10 "failed" : 0  
11 },  
12 "\_seq\_no" : 1,  
13 "\_primary\_term" : 1  
14 }  
15

Type here to search ENG IN 00:50 02/06/2020 [21]



# Create Indexing documents

Complete Guide to Elasticsearch x Elastic Kibana x

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

201 - Created 142 ms

```
1 POST /products/_doc
2 {
3   "productId":4974,
4   "name":"flask",
5   "qty":89
6 }
7 ^
8 POST /products/_doc/123
9 {
10   "productId":4975,
11   "name":"Heater",
12   "qty":89
13 }
14 ^
15 PUT /products/_doc/156
16 {
17   "productId":4974,
18   "name":"flask",
19   "qty":89
20 }
21 ^
```

```
1 {
2   "_index" : "products",
3   "_type" : "_doc",
4   "_id" : "156",
5   "_version" : 1,
6   "result" : "created",
7   "_shards" : {
8     "total" : 3,
9     "successful" : 1,
10    "failed" : 0
11  },
12   "_seq_no" : 1,
13   "_primary_term" : 1
14 }
15
```

Type here to search

00:50 02/06/2020



# Retrieving Documents

Complete Guide to Elasticsearch x Elastic Kibana x +

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 50 ms

1 GET /products/\_doc/156

```
1 {  
2   "_index": "products",  
3   "_type": "_doc",  
4   "_id": "156",  
5   "_version": 1,  
6   "_seq_no": 1,  
7   "_primary_term": 1,  
8   "found": true,  
9   "source": {  
10    "productId": 4974,  
11    "name": "flask",  
12    "qty": 89  
13  }  
14 }  
15
```

Type here to search

00:53 02/06/2020



# Retrieving Documents

Complete Guide to Elasticsearch    Elastic Kibana    ×    +

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 46 ms

1 GET /products/\_doc/123

```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "123",  
5   "_version" : 1,  
6   "_seq_no" : 0,  
7   "_primary_term" : 1,  
8   "found" : true,  
9   "_source" : {  
10     "productId" : 4975,  
11     "name" : "Heater",  
12     "qty" : 89  
13   }  
14 }  
15
```

Type here to search



# Updating the Document

Complete Guide to Elasticsearch x Elastic Kibana x

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 46 ms

```
1 POST /products/_update/123
2 {
3   "doc":{
4     "qty":43
5   }
6 }
7 }
8 }
9 }
10 }
11 GET /products/_doc/123
12
13
```

The screenshot shows the Elasticsearch Dev Tools interface in Kibana. On the left, there's a sidebar with various icons. The main area has tabs for Console, Search Profiler, Grok Debugger, and Painless Lab (labeled BETA). The Console tab is active. It displays a sequence of API requests and their responses. Request 1 is a POST to update document 123 with a new value of 43. Request 11 is a GET to retrieve the updated document. The response for the GET request is shown on the right, detailing the document's fields: \_index: products, \_type: \_doc, \_id: 123, \_version: 2, \_seq\_no: 1, \_primary\_term: 1, found: true, and \_source: { productId: 4975, name: "Heater", qty: 43 }. The status bar at the bottom indicates a 200 - OK response and a duration of 46 ms.



# Scripting the document

The screenshot shows the Elasticsearch Dev Tools Console in Kibana. The left panel displays the history of requests and their responses. A blue arrow points from the number 1 in the history list to the first line of the response JSON.

**History:**

- 1 POST /products/\_update/123
- 2 {
- 3    "script":{
- 4     | "source":"ctx.\_source.qty--"
- 5    }
- 6 }
- 7 }
- 8
- 9 GET /products/\_doc/123
- 10
- 11

**Response (Line 1):**

```
1 {  
2   "_index" : "products",  
3   "_type" : "_doc",  
4   "_id" : "123",  
5   "_version" : 5,  
6   "_seq_no" : 4,  
7   "_primary_term" : 1,  
8   "found" : true,  
9   " _source" : {  
10     "productId" : 4975,  
11     "name" : "Heater",  
12     "qty" : 42  
13   }  
14 }  
15
```



# 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

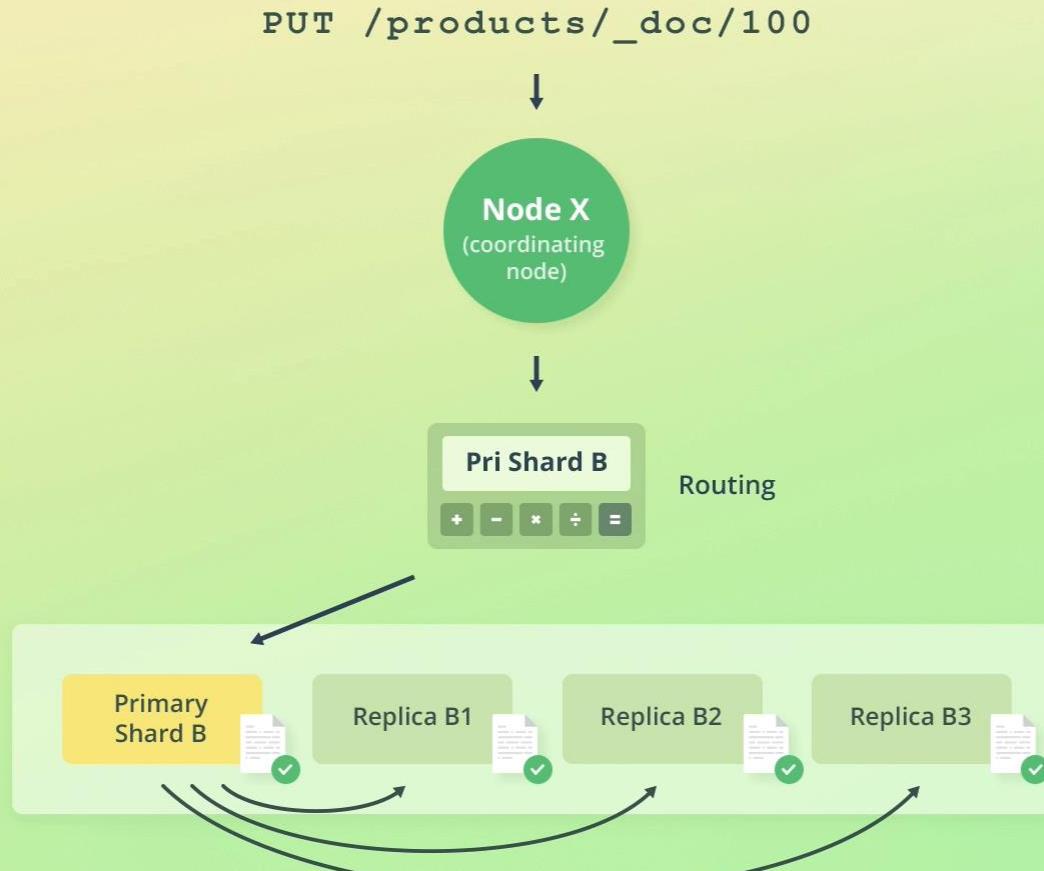
```
1 | POST /product/default
2 | {
3 |   "name": "Processing Events with Logstash",
4 |   "instructor": {
5 |     "firstName": "Bo",
6 |     "lastName": "Andersen"
7 |   }
8 |
9 |
10 | PUT /product/default/_1
11 | {
12 |   "name": "Complete Guide to Elasticsearch",
13 |   "instructor": {
14 |     "firstName":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

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

```
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

Primary  
Shard A

Replica A1

Replica A2

Replica A3

Primary  
Shard B

Replica B1

Replica B2

Replica B3

Primary  
Shard C

Replica C1

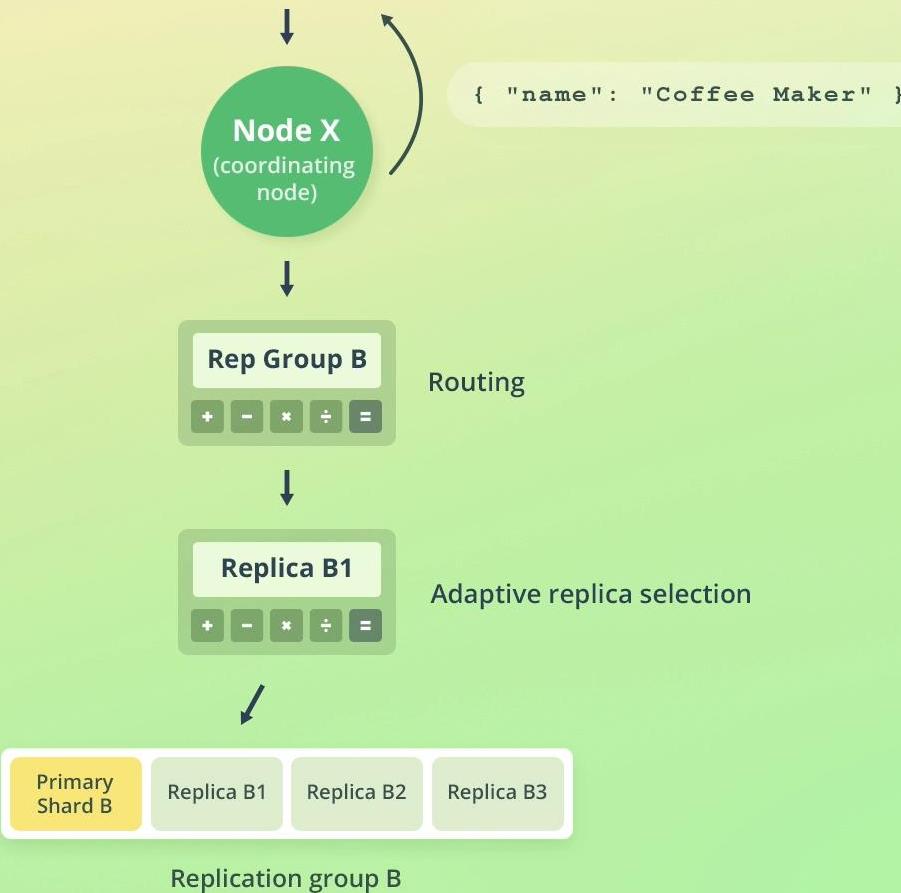
Replica C2

Replica C3



# 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 }
```

...

Discover Visualize Dashboard Timelion Machine Learning Graph Dev Tools Management

Udemy 56



# 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

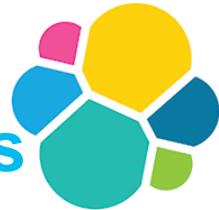
```
1  DELETE /member/default/1
2
3  POST /member/default/1/_update
4  {
5    "script": "ctx._source.fees+=500",
6    "upsert": {
7      "fees": 2000
8    }
9  }
10
11 GET /member/default/1| ➡ 🔒 ⚙️ ⋮
12
13 POST /member/default/1/_update
14 {
15   "script": "ctx._source.fees+=500",
16   "upsert": {
17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

History Settings Help

Discover Visualize Dashboard Timelion APM Dev Tools Monitoring Management

1 \_index: "member", \_type: "default", \_id: "1", \_version: 7, found: true, \_source: { fees: 2500 }

# Managing Documents – Upserting Documents



Gmail x MUST READ-CLASS RE x Launch Meeting - Zoom x localhost:9200/account x Elastic Kibana x eswaribalab/kibana#/\_dev\_tools/console x Transforming data with x +

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

Kumarswamy N B

POST /bookdb\_index/book/3/\_update

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

GET /bookdb\_index/book/3

#! Deprecation: [types removal] Specifying types in document get requests is deprecated, use the /{index}/\_doc/{id} endpoint instead.

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

Type here to search



# 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, Microsoft Edge, Google Chrome, Microsoft Word, Microsoft Powerpoint, Microsoft Word, Microsoft Excel, Microsoft Word, 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
```



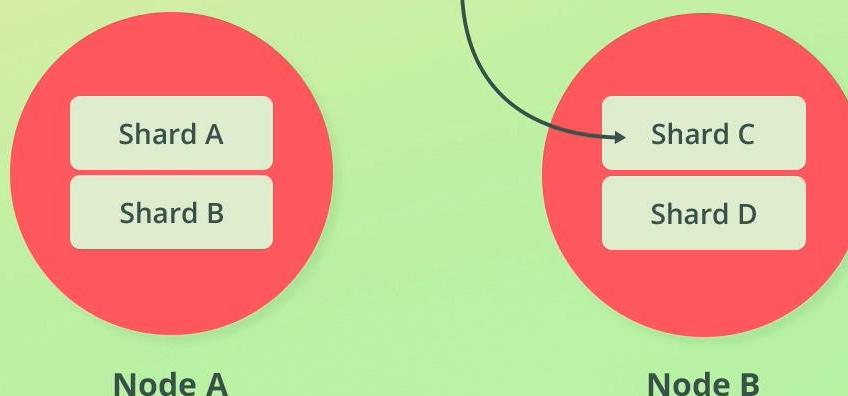
# Introduction to Routing

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



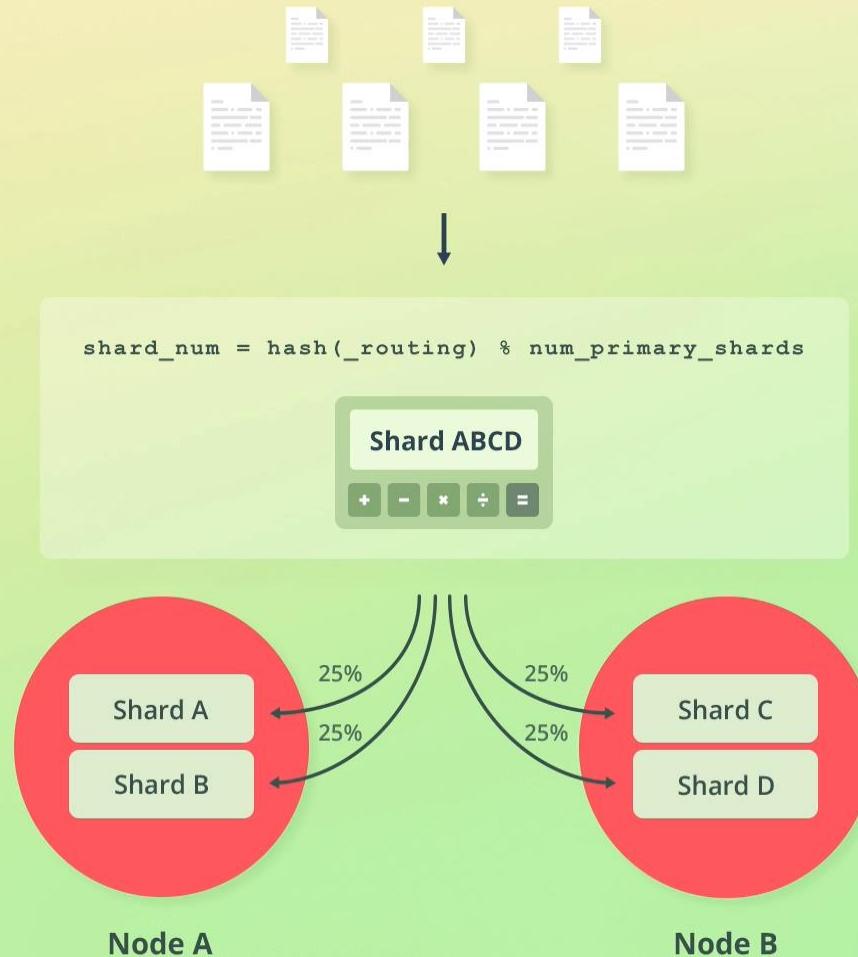
```
shard_num = hash(_routing) % num_primary_shards
```

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
```

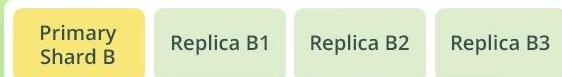
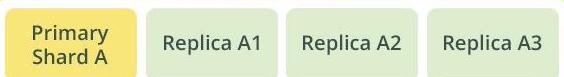
↓  
**Node X**  
(coordinating node)

↓  
**Rep Group B**  
Routing  
+ - × ÷ =

Replication group A

Replication group B

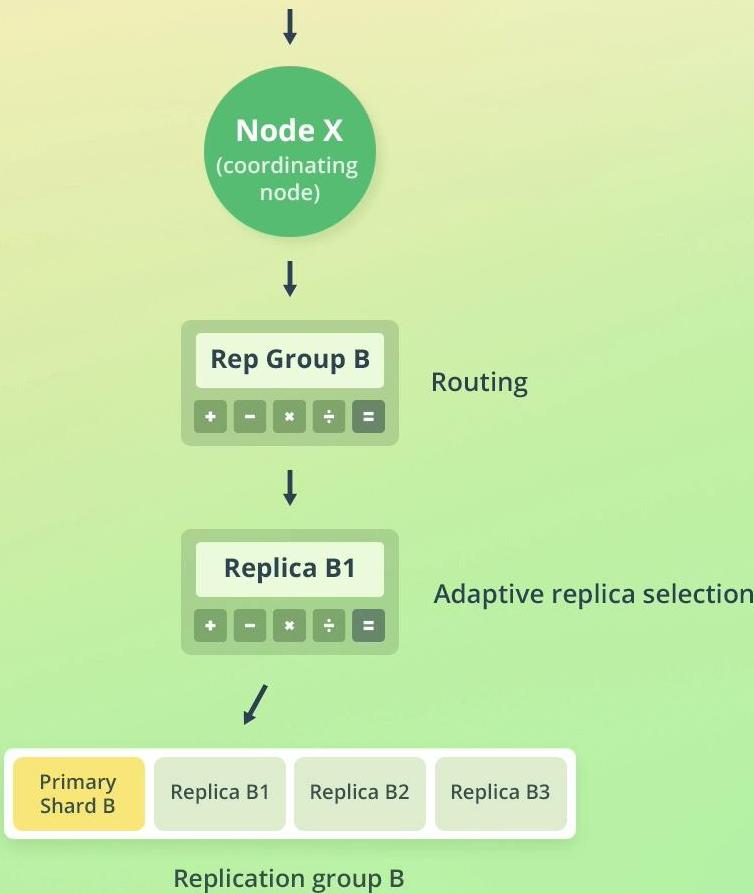
Replication group C





# Introduction to Read Document

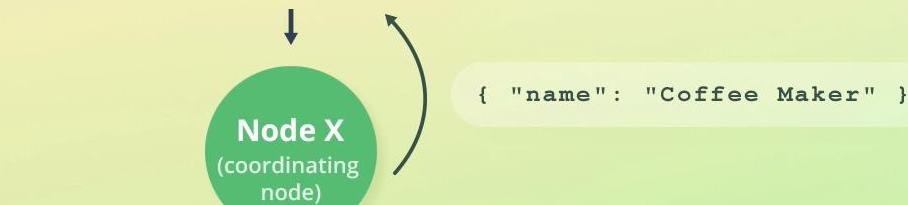
GET /products/\_doc/100



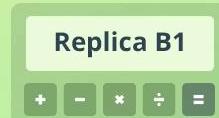


# Introduction to Read Document

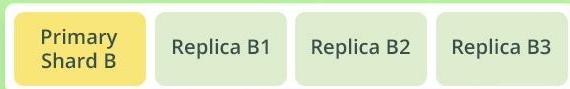
GET /products/\_doc/100



Routing



Adaptive replica selection



Replication group B



# 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

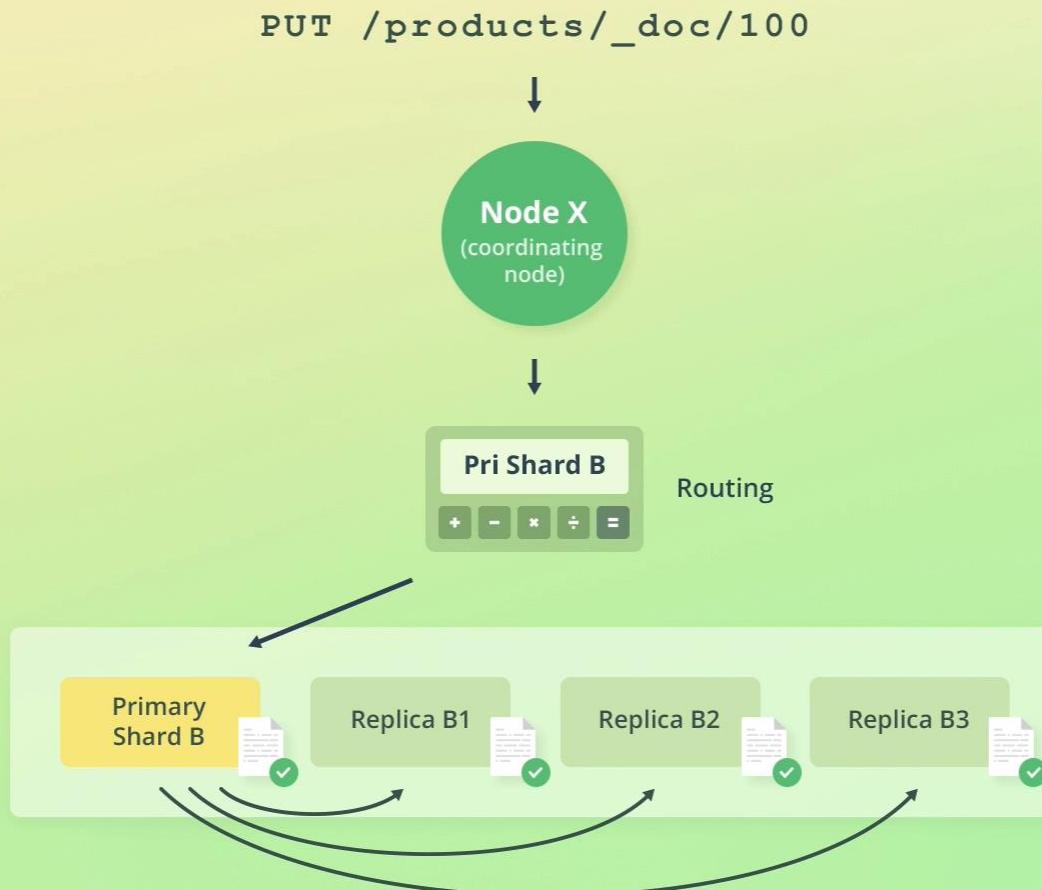
Replica B1

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



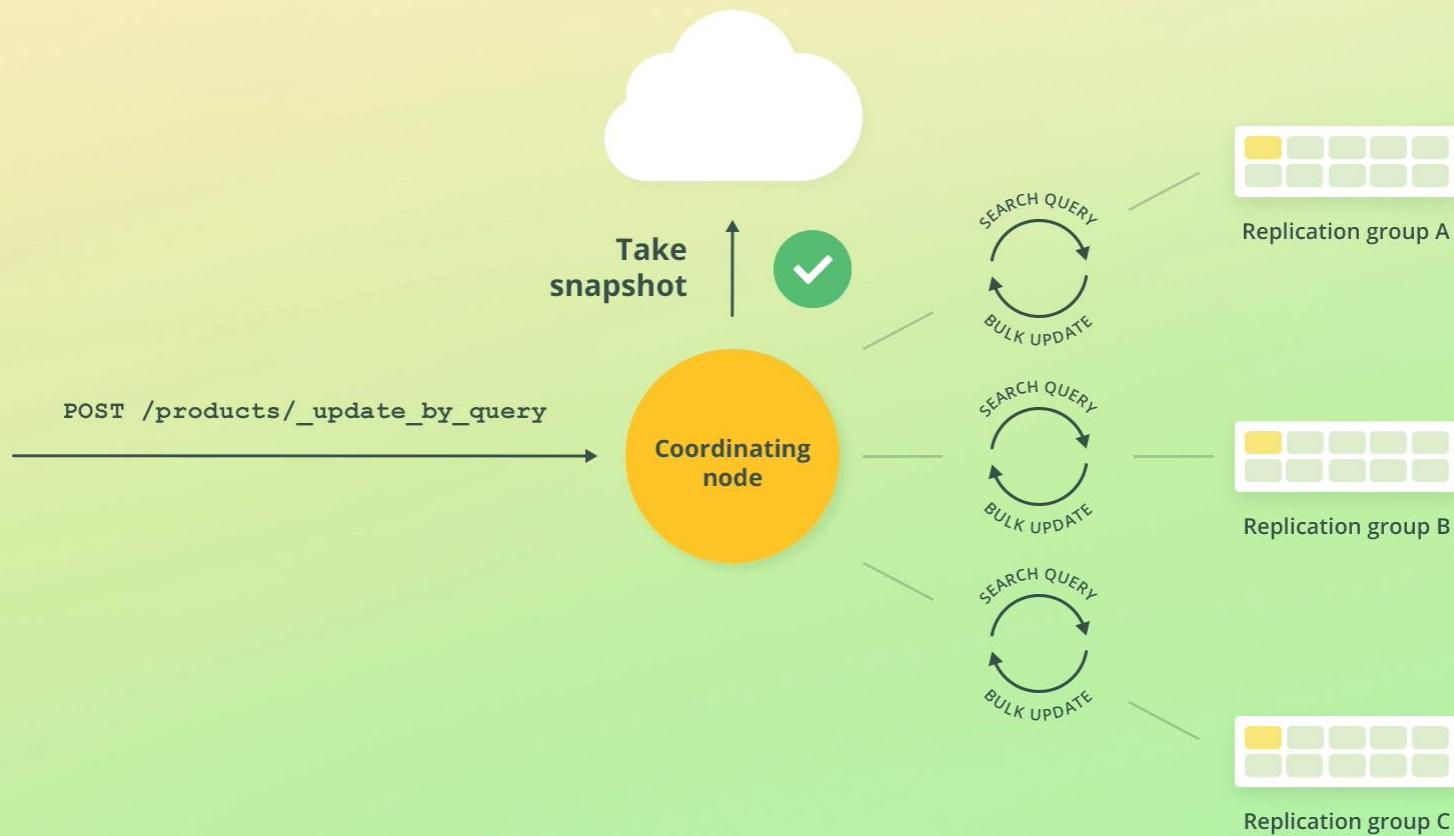
# Multiple Update

```
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"
      }
    }
  ]
}
```

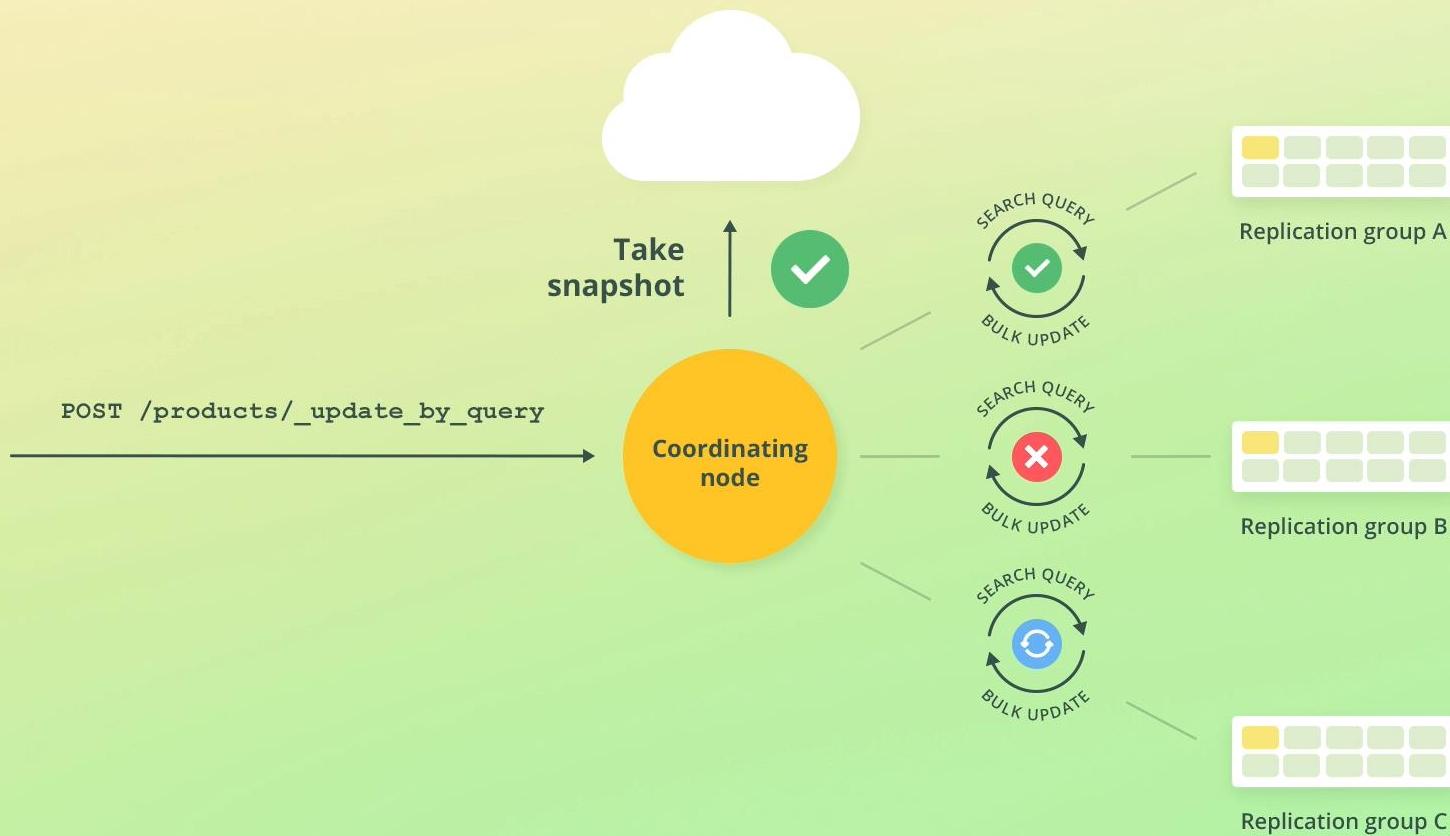


# Multiple update(Scroll API)





# 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

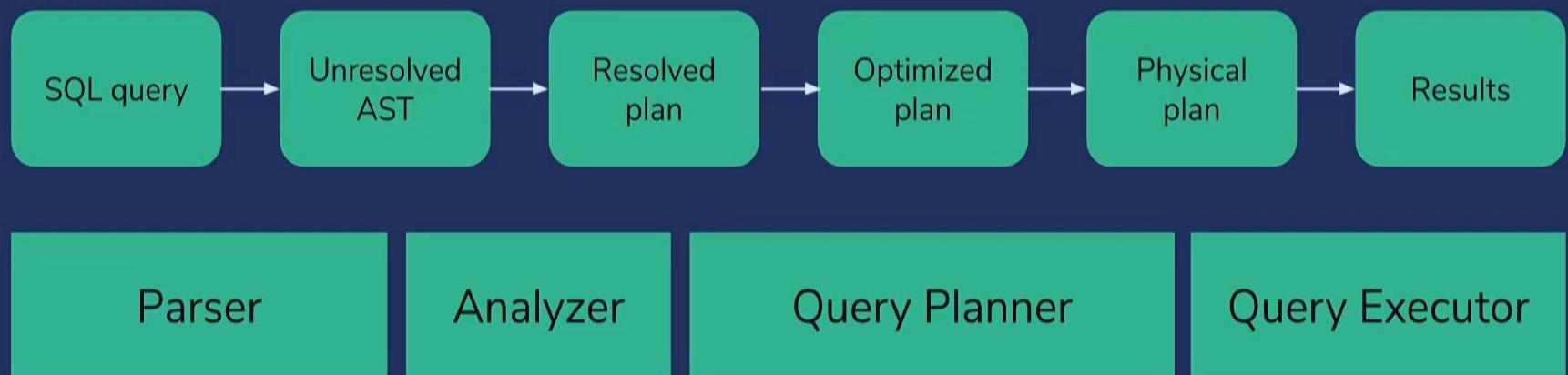
---

- 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. At the top, it says 'Dev Tools' and has links for 'History', 'Settings', and 'Help'. Below that are tabs for 'Console', 'Search Profiler', and 'Grok Debugger', with 'Console' being the active tab. The main area shows a code editor with two sections of JSON. The first section is a PUT request to '/reviews/\_mapping' with a single property 'created\_at' of type 'date'. The second section is a GET request to the same endpoint, which returns a complex mapping for the 'reviews' index. This mapping includes properties for 'author', 'content', 'last\_name', 'first\_name', 'product\_id', 'rating', and 'created\_at', each with specific types like 'keyword', 'text', 'integer', and 'float'. A cursor is visible over the GET request line.

```
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 }
```



# 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 }
```

```
11   "version": 5,
12   "_primary_term": 1
13 }
14
15
```



# 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



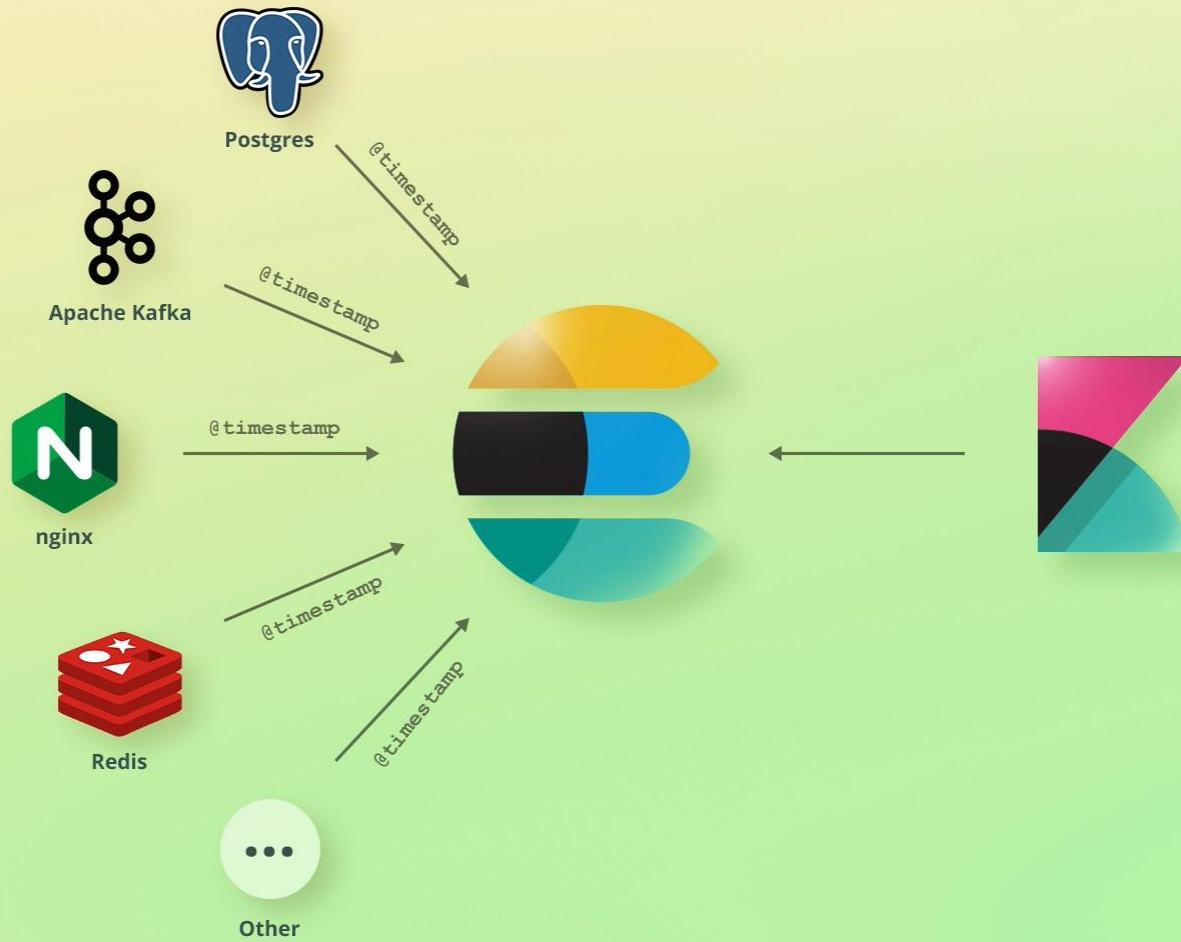
# Limitation to Update Mappings

---

- 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  
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 monitoring data from Elastic Kibana.

The browser tabs are:

- Gmail
- Launch Meeting - Zoom
- localhost:9200/\_cat/shards/heart
- Elastic Kibana

The Elastic Kibana interface shows the following data:

- Uptime** dashboard:
  - 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 (value 3), except for 11:12 which is light blue (value 1).
- Monitor status** table:

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
      }
    }
  }
}
```



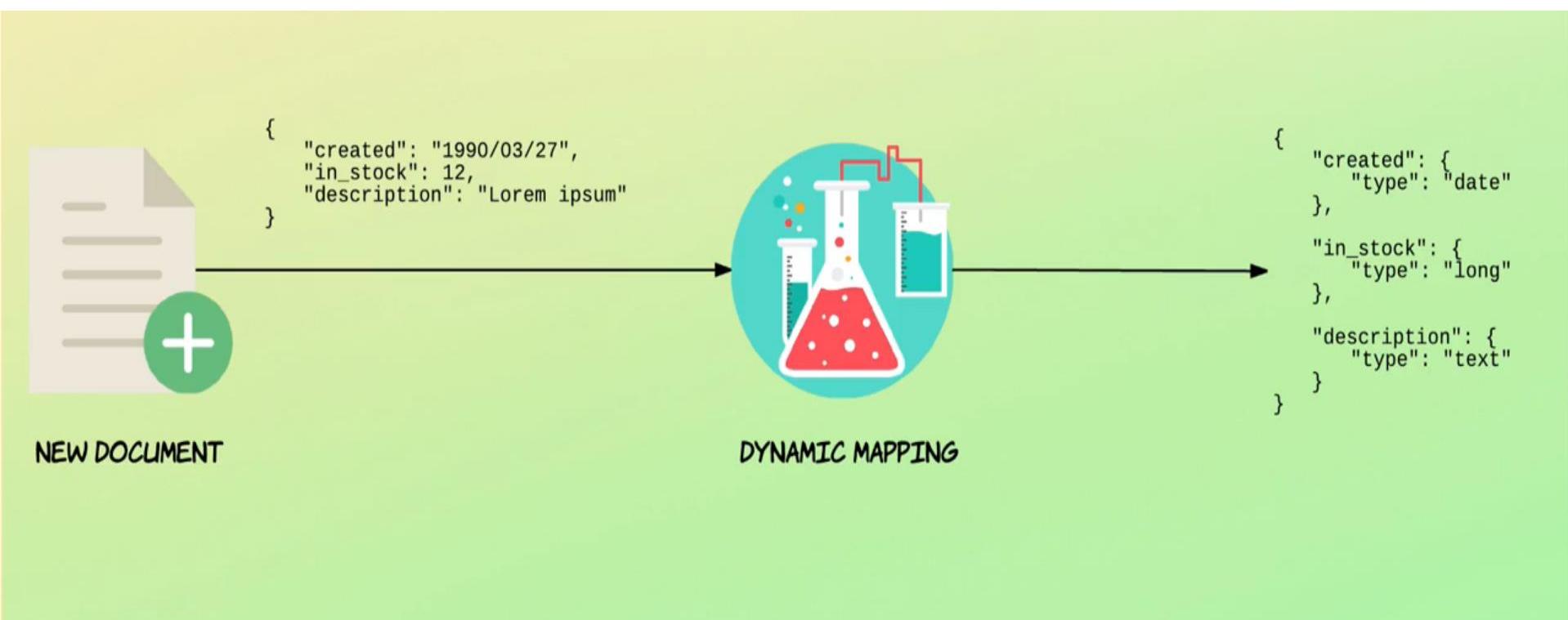
# Dynamic Mapping

JSON	ELASTICSEARCH
string	One of the following: <ul style="list-style-type: none"><li>• <code>text</code> field with <code>keyword</code> mapping</li><li>• <code>date</code> field</li><li>• (<code>float</code> or <code>long</code> field)</li></ul>
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

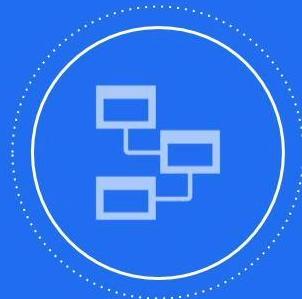
```
PUT /product/default/_mapping
{
  "properties": {
    "discount": {
      "type": "integer"
    }
  }
}

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

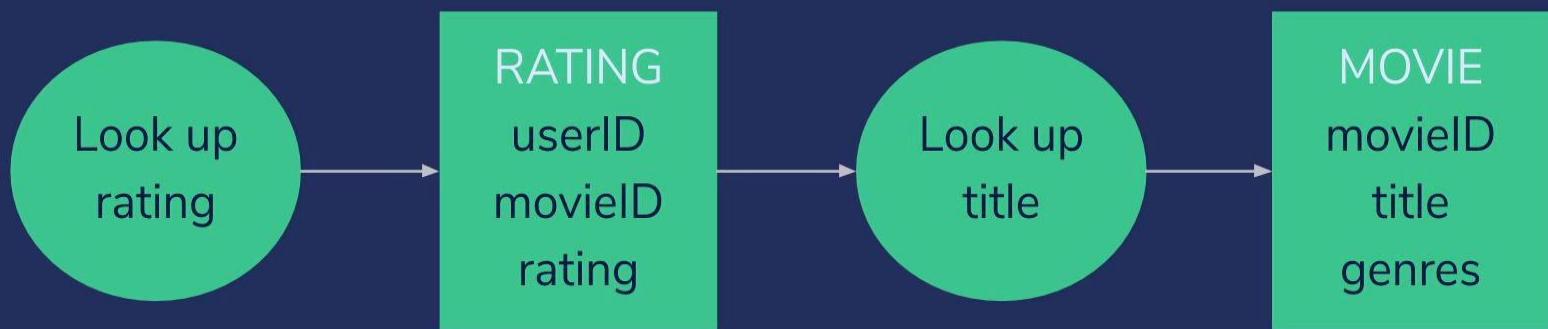


# Data Modeling



## Strategies For Relational Data

Normalized data

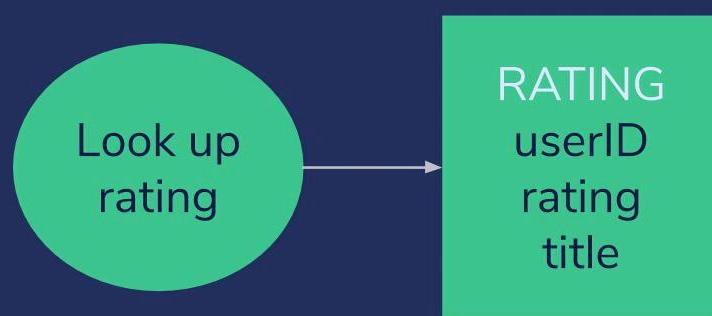


Minimizes storage space, makes it easy to change titles  
But requires two queries, and storage is cheap!



## 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

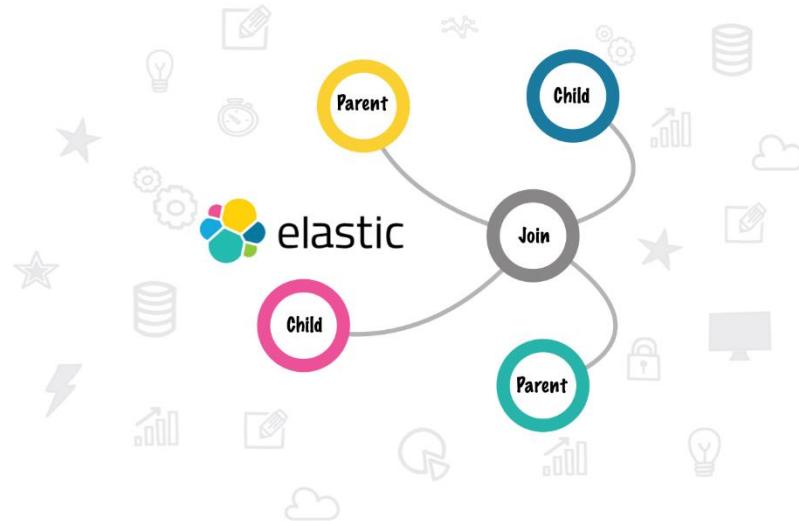
Return of the  
Jedi

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](https://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": {} }
- }
- }
- }



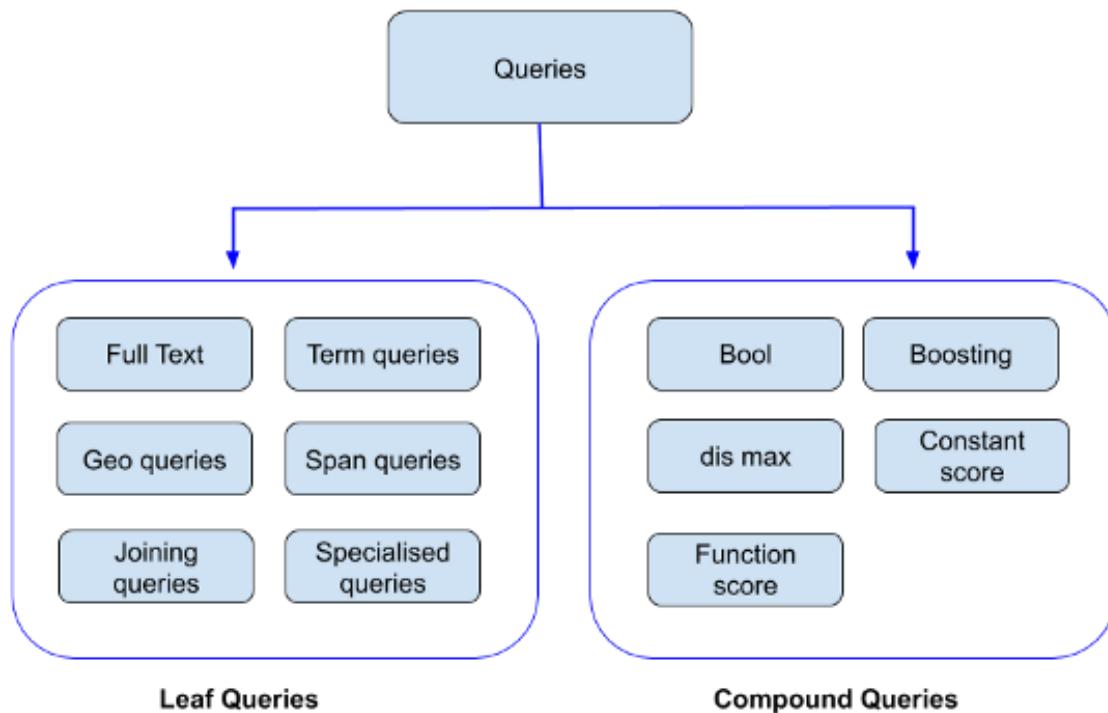
# Parent child Relationship

---

- 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

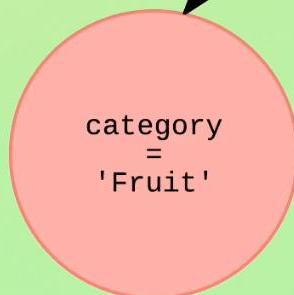
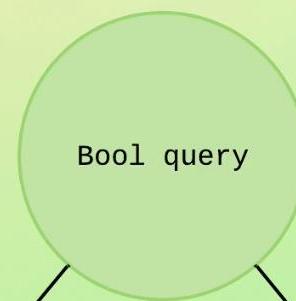
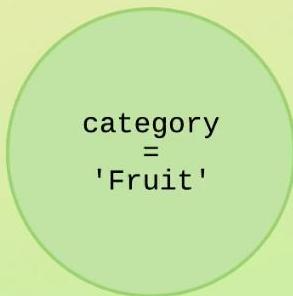


# Introducing the Query DSL

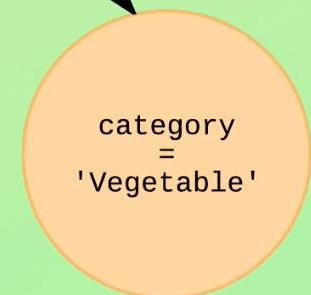


## COMPOUND QUERY

### LEAF QUERY



OR





# 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
25             search engine using Apache Solr",
26           "publisher" : "manning",
27           "num_reviews" : 63,
           "title" : "Solr in Action",
         }
       }
     ]
   }
 }
```

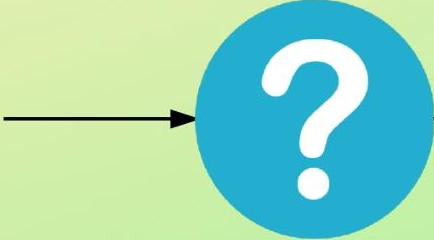


## Understanding relevance scores





GET /product/default/\_search?q=pizza



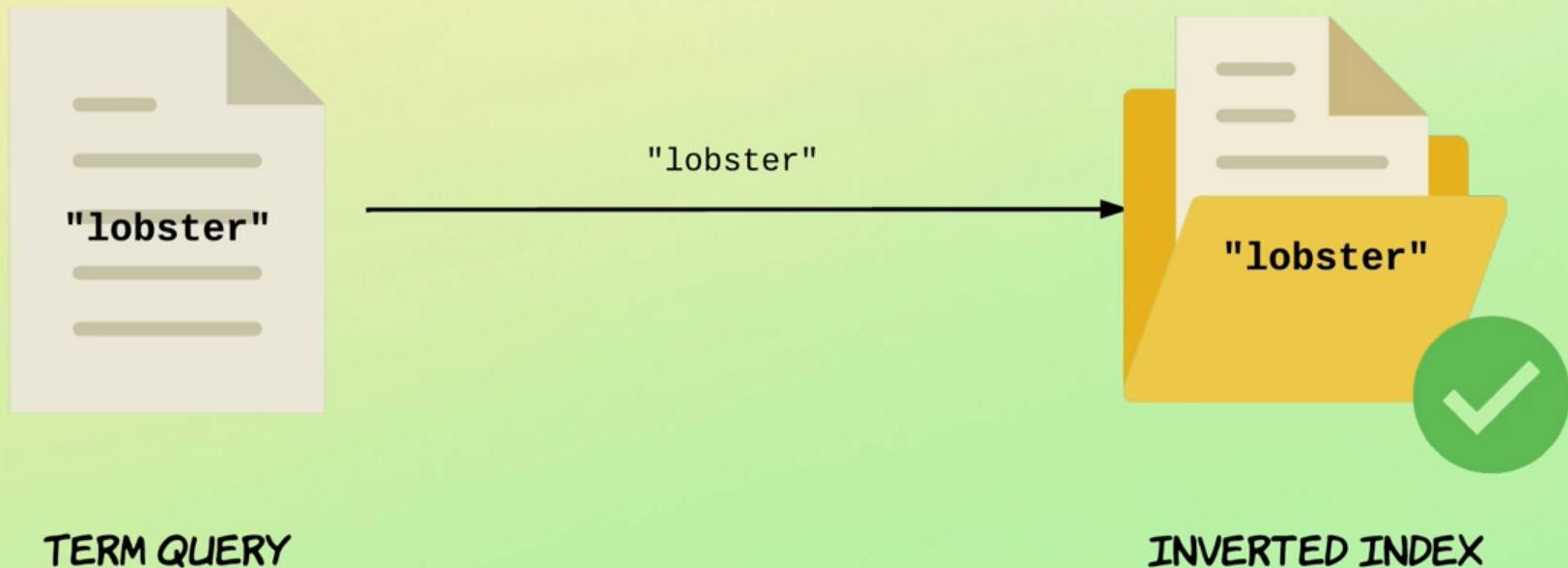
Find documents  
matching the query

- 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  
x",  
16 "\_type": "doc",  
17 "\_id": "cr1dxWcBjynLt7N6KjB3  
",  
18 "\_score": 1,  
19 "\_source": {  
20 "path": "D:/logstash  
configurations/loggly\_events\_2018-12  
10 07 01 10 451570.json"  
21 }  
22 }  
23 ]  
24 }  
25 }



# Retrieving Documents By Id

Kibana Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

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

1 {  
2 "took": 189,  
3 "timed\_out": false,  
4 "\_shards": {  
5 "total": 5,  
6 "successful": 5,  
7 "skipped": 0,  
8 "failed": 0  
9 },  
10 "hits": {  
11 "total": 2,  
12 "max\_score": 1,  
13 "hits": [  
14 {  
15 "\_index": "syslog-logglyindex",  
16 "\_type": "doc",  
17 "\_id": "cr1dxWcBjynLt7N6KjB3"  
18 },  
19 {  
20 "\_score": 1,  
21 "\_source": {  
22 "path": "D:/logstash/configurations/loggly\_events\_2018-12-01.log",  
23 "version": 1  
24 }  
25 }  
26 ]  
27 }  
28}  
29



# 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  }
```

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 History Settings Help

```
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 }
```

```
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         "_score": 1,
19         "_source": {
20           "Low": "903.4",
21           "High": "950.0",
22           "Close": "917.05879",
23           "Open": "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 , "message": "2012-08-12T11:5102,11.765,11.45,11.62399,30340
.953643,353092.674073,11.6374942669"
,
"timestamp": "2018-12-17T15:12:32.976Z",
"Date": "2012-08-12"
}
},
{
"\_index": "bitcoin-prices",
"\_type": "doc",
"\_id": "\_lu5vGcB1da7Pa9b1b0m
",
"\_score": 1,
"\_source": {
"Low": "8.8183",
"High": "9.23355",
"Close": "8.86999",
"host": "DESKTOP-55AGI0I",
"path": "D:/logstash
configurations/BCHARTS-MTGOXUSD.csv"
},

History Settings Help



# Searching with wild cards

Kibana Dev Tools

Console Search Profiler Grok Debugger

```
1 GET bitcoin-prices/doc/_search
2 {
3   "query":{
4     "wildcard":{
5       "message":"20?2"
6     }
7   }
8 }
9
10
11
12
13
14 GET syslog-logglyindex/doc/_search
```

History Settings Help

```
1 {
2   "took": 4,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10   "hits": {
11     "total": 732,
12     "max_score": 1,
13     "hits": [
14       {
15         "_index": "bitcoin-prices",
16         "_type": "doc",
17         "_id": "-1u5vGcB1da7Pa9b1b0m",
18         "_score": 1,
19         "_source": {
20           "Low": "11.45",
21           "High": "11.765",
22           "Close": "11.62399",
23           ...
24         }
25       }
26     ]
27   }
28 }
```



# Searching with Regular Expressions

Kibana Dev Tools

Console Search Profiler Grok Debugger

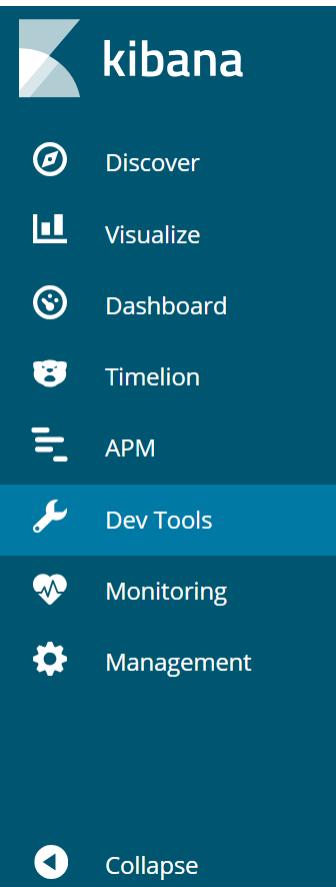
```
1 GET bitcoin-prices/doc/_search
2 {
3   "query":{
4     "regexp":{
5       "message": "[0-9]+"
6     }
7   }
8 }
9 }
10 }
11 }
12 }
13
14 GET syslog-logglyindex/doc/_search
```

▶ 🔒

```
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
```

```
{ "receipt": { "mappings": { "doc": { "properties": { "created": { "type": "date", "format": "yyyy/MM/dd HH:mm:ss||yyyy/MM/dd||epoch_millis" }, "description": { "type": "text", "fields": { "keyword": { "type": "keyword", "ignore_above": 256 } } }, "ingredients": { "properties": { "name": { "type": "text", "fielddata": true } } } } } } }
```



# 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

Discover Visualize Dashboard Timelion APM Dev Tools Monitoring Management

collapse

```
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 - }
```

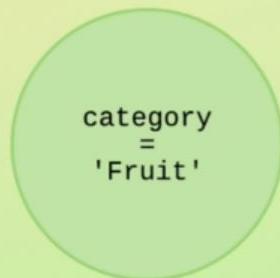
1, ]  
2, "created": "2002/01/04",  
3, "ratings": [  
4, 3,  
5, 2.5,  
6, 4,  
7, 4.5,  
8, 3.5,  
9, 4  
10, ]  
11, }  
12, {  
13, "\_index": "recipe",  
14, "\_type": "default",  
15, "\_id": "12",  
16, "\_score": 1.3220012,  
17, "\_source": {  
18, "title": "Penne With Melted-Vegetable Sauce",  
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

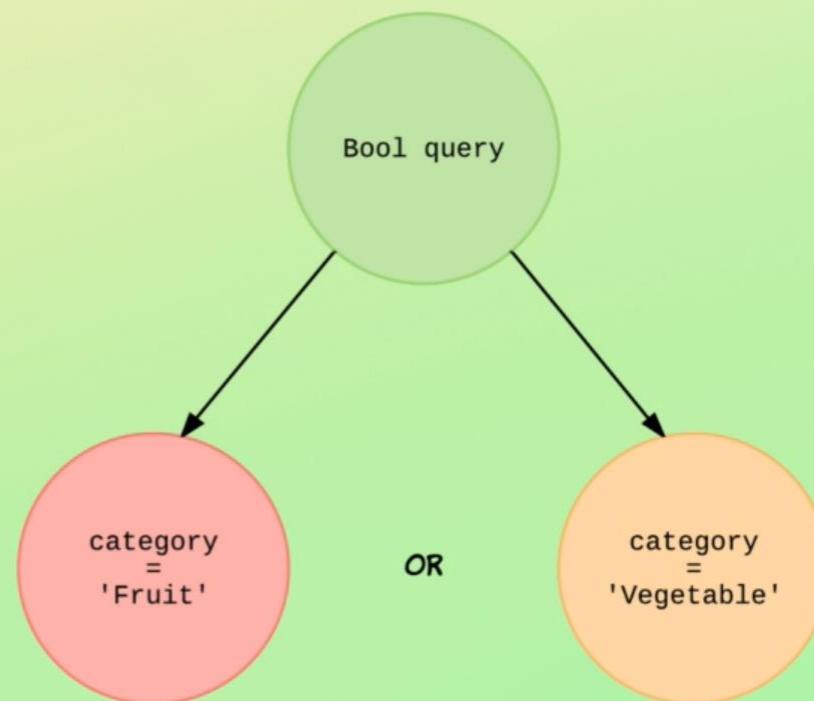


# Compound Queries

LEAF QUERY

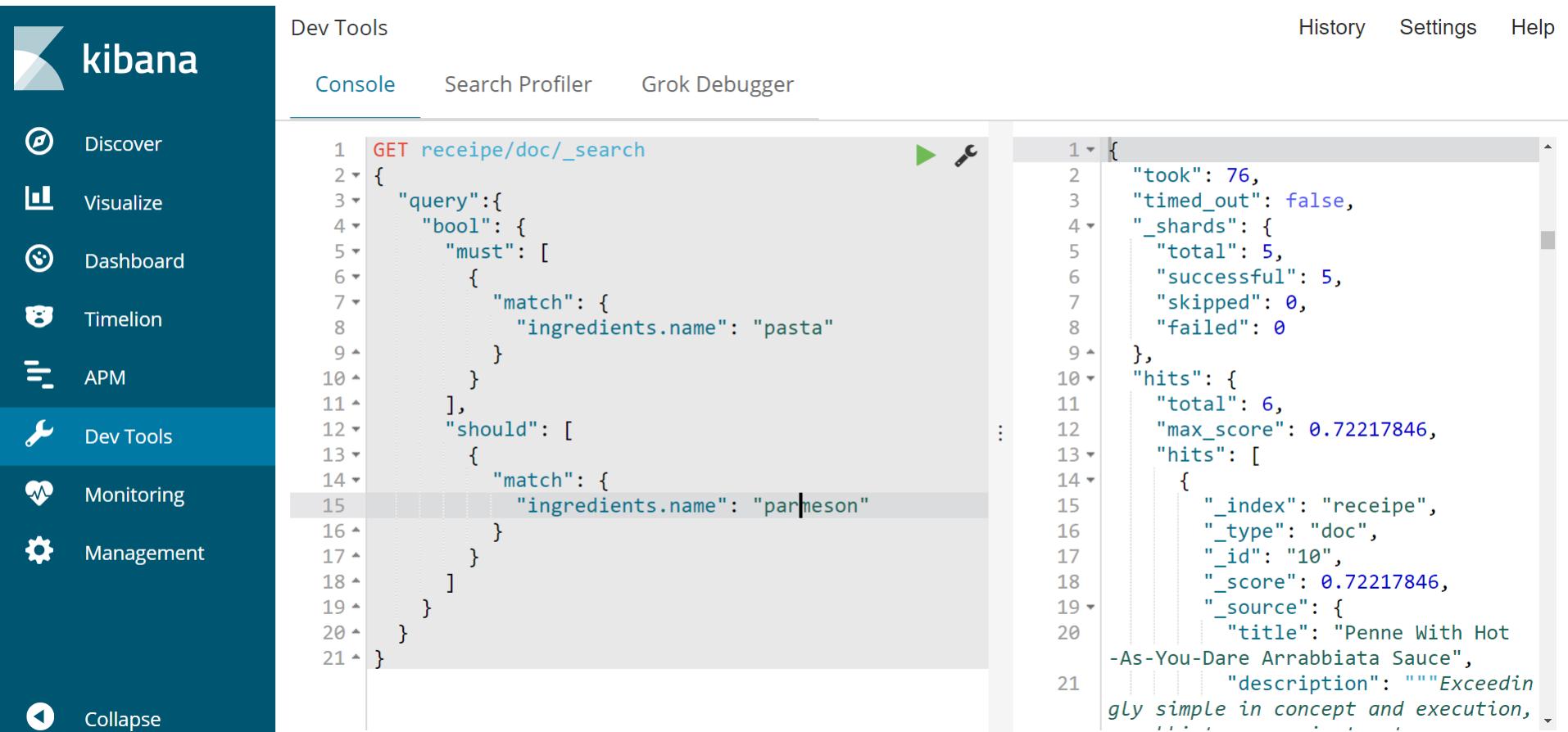


COMPOUND QUERY





# Compound Queries





# 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](https://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

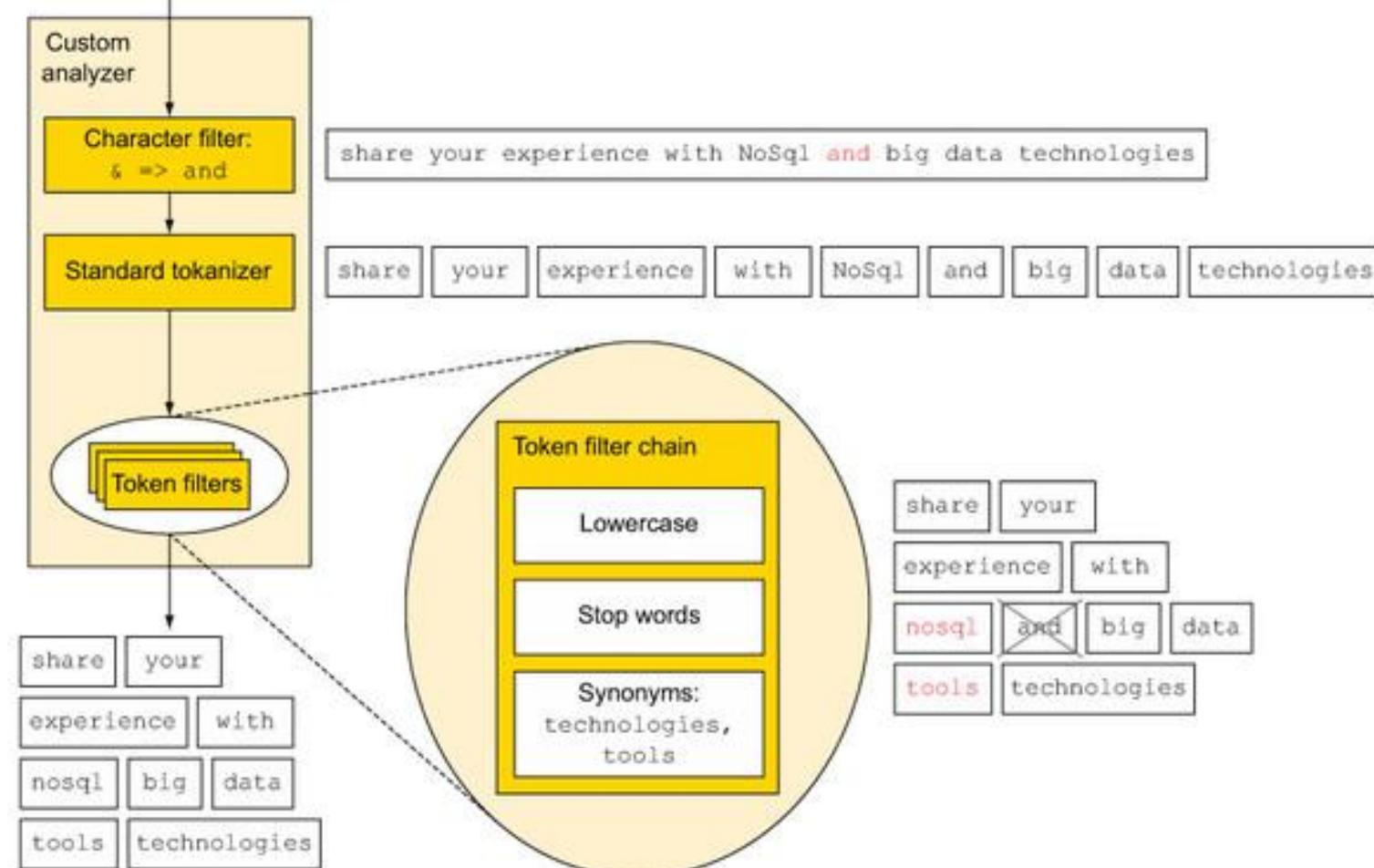


## Analyzers and search queries



Original text

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





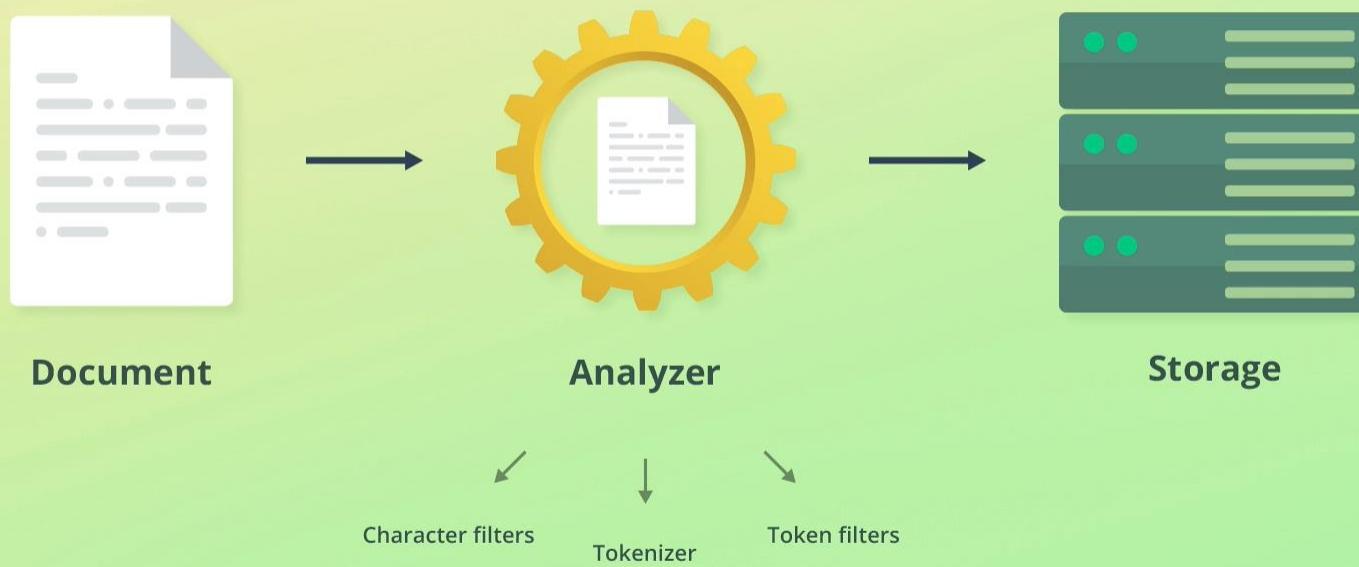
# Analyzer

---

- 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 <em>good</em> mood&nbs;-&nbs; and I <strong>love</strong> 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 ▶ 🔍

```
1 POST _analyze
2 {
3   "text": "Tom and Jerry always fights",
4   "analyzer": "standard"
5 }
```

```
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

02:39 ENG IN 02/06/2020 [21]

# Filters



The screenshot shows the Elasticsearch Dev Tools Console interface. The left panel displays a code editor with a history of requests. The current request is a POST \_analyze with the following body:

```
1 POST _analyze
2 {
3     "text": "Tom and Jerry always fights",
4     "analyzer": "standard"
5 }
```

The right panel shows the response, which is a JSON object representing tokens:

```
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": 19
28        }
29    ]
30 }
```



# 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



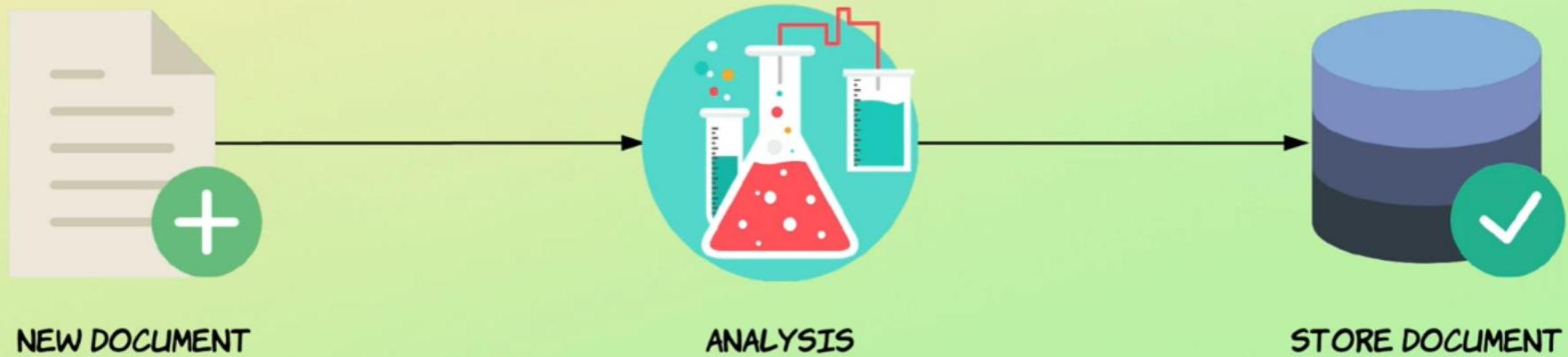
# Inverted Indices

---

- 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

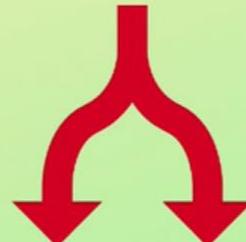
<strong>Two</strong> words!



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."
}
```



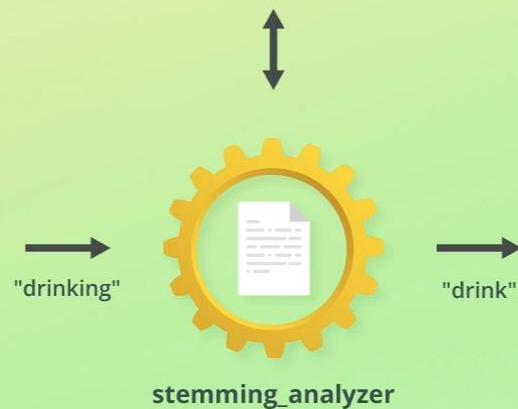
```
["i", "love", "drink", "bottl",
 "of", "wine", "on", "last",
 "year", "vacat"]
```



# Stemming Analyzer

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

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



TERM	DOCUMENT #1	...
i	X	
love	X	
drink	X	
bottl	X	
of	X	
wine	X	
on	X	
last	X	
year	X	
vacat	X	



## Built-in analyzers



# Standard Analyzer

- Splits text at word boundaries and removes punctuation
  - Done by the `standard tokenizer`
- Lowercases letters with the `lowercase token filter`
- Contains the `stop token filter` (disabled by default)



# Standard Analyzer

"Is that Peter's cute-looking dog?"



["is", "that", "peter's", "cute", "looking", "dog"]



# Simple Analyzer

---

- Similar to the standard analyzer
  - Splits into tokens when encountering *anything else* than letters
- Lowercases letters with the lowercase tokenizer
  - Unusual and a performance hack



# Simple Analyzer

"Is that Peter's cute-looking dog?"



```
["is", "that", "peter", "s", "cute", "looking", "dog"]
```



# Whitespace Analyzer

- Splits text into tokens by whitespace
- Does *not* lowercase letters

"Is that Peter's cute-looking dog?"



[ "Is", "that", "Peter's", "cute-looking", "dog?"]



# Keyword Analyzer

---

- No-op analyzer that leaves the input text intact
  - It simply outputs it as a single token
- Used for `keyword` fields by default
  - Used for exact matching



# Keyword Analyzer

"Is that Peter's cute-looking dog?"



["Is that Peter's cute-looking dog?"]



# Pattern Analyzer

- A regular expression is used to match token separators
  - It should match whatever should split the text into tokens
- This analyzer is very flexible
- The default pattern matches all non-word characters (`\w+`)
- Lowercases letters by default



# Pattern Analyzer(Default Configuration)

"Is that Peter's cute-looking dog?"



["is", "that", "peter", "s", "cute", "looking", "dog"]



[« Specify an analyzer](#)

[Fingerprint Analyzer »](#)

## Built-in analyzer reference



Elasticsearch ships with a wide range of built-in analyzers, which can be used in any index without further configuration:

### Standard Analyzer

The standard analyzer divides text into terms on word boundaries, as defined by the Unicode Text Segmentation algorithm. It removes most punctuation, lowercases terms, and supports removing stop words.

### Simple Analyzer

The simple analyzer divides text into terms whenever it encounters a character which is not a letter. It lowerscases all terms.

### Whitespace Analyzer

The whitespace analyzer divides text into terms whenever it encounters any whitespace character. It does not lowercase terms.

### Stop Analyzer

The stop analyzer is like the simple analyzer, but also supports removal of stop words.

### Keyword Analyzer

The keyword analyzer is a “noop” analyzer that accepts whatever text it is given and outputs the exact same text as a single term.

### Pattern Analyzer

The pattern analyzer uses a regular expression to split the text into terms. It supports lower-casing and stop words.

### Language Analyzers

Elasticsearch provides many language-specific analyzers like english or french.

### Fingerprint Analyzer

The fingerprint analyzer is a specialist analyzer which creates a fingerprint which can be used for duplicate detection.

## Custom analyzers



If you do not find an analyzer suitable for your needs, you can create a [custom](#) analyzer which combines the appropriate [character filters](#), [tokenizer](#), and [token filters](#).

## Elasticsearch Video

Optimizing Elastic for Search at McQueen Solutions.

[Watch now](#)

+ Elasticsearch Reference: master

+ Elasticsearch introduction

+ Getting started with Elasticsearch

+ Set up Elasticsearch

+ Upgrade Elasticsearch

+ Aggregations

+ Query DSL

Search across clusters

Long-running searches

+ Scripting

+ Mapping

- Text analysis

Overview

+ Concepts

+ Configure text analysis

- Built-in analyzer reference

Fingerprint Analyzer

Keyword Analyzer

Language Analyzers

Pattern Analyzer

Simple Analyzer



```
PUT /english_example
{
  "settings": {
    "analysis": {
      "filter": {
        "english_stop": {
          "type": "stop",
          "stopwords": "_english_" ①
        },
        "english_keywords": {
          "type": "keyword_marker",
          "keywords": ["example"] ②
        },
        "english_stemmer": {
          "type": "stemmer",
          "language": "english"
        },
        "english_possessive_stemmer": {
          "type": "stemmer",
          "language": "possessive_english"
        }
      },
      "analyzer": {
        "rebuilt_english": {
          "tokenizer": "standard",
          "filter": [
            "english_possessive_stemmer",
            "lowercase",
            "english_stop",
            "english_keywords",
            "english_stemmer"
          ]
        }
      }
    }
  }
}
```



```
PUT /products
{
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "analyzer": "english"
      }
    }
  }
}
```

```
POST /products/_doc
{
  "description": "Is that
  Peter's cute-looking dog?"
}
```



```
["peter", "cute", "look", "dog"]
```



# Standard Analyzer

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 47 ms

```
1 POST _analyze
2 {
3   "analyzer": "standard",
4   "text": "The 2 QUICK Brown-Foxes jumped over the lazy
      dog's bone."
5 }
```

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

The screenshot shows the Kibana Dev Tools interface, specifically the Console tab. A POST request is being made to the '\_analyze' endpoint. The request body contains an analyzer configuration with 'analyzer': 'standard' and a text string: 'The 2 QUICK Brown-Foxes jumped over the lazy dog's bone.'. The response is a JSON object containing an array of tokens. Each token object has properties: 'token' (the word), 'start\_offset' (the character index where it begins), 'end\_offset' (the character index where it ends), 'type' (determined by the character set), and 'position' (its position in the original text). The tokens extracted are 'the', '2', 'quick', 'brown', and several tokens for punctuation and hyphenation.



# Stop Analyzer

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 54 ms

```
POST _analyze
{
  "analyzer": "stop",
  "text": "The 2 QUICK Brown-Foxes jumped over the lazy
          dog's bone."
}
```

```
1  {
2    "tokens" : [
3      {
4        "token" : "quick",
5        "start_offset" : 6,
6        "end_offset" : 11,
7        "type" : "word",
8        "position" : 1
9      },
10     {
11       "token" : "brown",
12       "start_offset" : 12,
13       "end_offset" : 17,
14       "type" : "word",
15       "position" : 2
16     },
17     {
18       "token" : "foxes",
19       "start_offset" : 18,
20       "end_offset" : 23,
21       "type" : "word",
22       "position" : 3
23     },
24     {
25       "token" : "jumped",
26       "start_offset" : 24,
27       "end_offset" : 29
28     }
29   }
30 }
```



# Html\_strip Analyzer

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 50 ms

```
1 POST _analyze
2 {
3   "tokenizer": "keyword",
4   "char_filter": [ "html_strip" ],
5   "text": "<p>I'm so <b>happy</b>!</p>"
6 }
```

1 {
2 "tokens" : [
3 {
4 "token" : ""
5 I'm so happy!
6 },
7 "start\_offset" : 0,
8 "end\_offset" : 32,
9 "type" : "word",
10 "position" : 0
11 ]
12 }
13 }
14



# Lowercase Tokenizer

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 191 ms

```
1 POST _analyze
2 {
3   "tokenizer": "lowercase",
4   "text": "It Was a Beautiful Weather 5 Days ago."
5 }
```

```
1 {
2   "tokens": [
3     {
4       "token": "it",
5       "start_offset": 0,
6       "end_offset": 2,
7       "type": "word",
8       "position": 0
9     },
10    {
11      "token": "was",
12      "start_offset": 3,
13      "end_offset": 6,
14      "type": "word",
15      "position": 1
16    },
17    {
18      "token": "a",
19      "start_offset": 7,
20      "end_offset": 8,
21      "type": "word",
22      "position": 2
23    },
24    {
25      "token": "beautiful",
26      "start_offset": 9,
27      "end_offset": 19
28    }
29  ]
30}
```



# Pattern Replace Tokenizer

Complete Guide to Elasticsearch x localhost:9200/\_cat/indices x Elastic Kibana x Pattern Replace Char Filter | Elasti x +

localhost:5601/app/kibana#/dev\_tools/console

Apps Projects Gmail YouTube Maps Pluralsight

D Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 1163 ms

```
PUT pattern_replace_analyzer
{
  "settings": {
    "analysis": {
      "analyzer": {
        "vm_pattern_analyzer": {
          "tokenizer": "standard",
          "char_filter": [
            "my_char_filter"
          ]
        },
        "char_filter": {
          "my_char_filter": {
            "type": "pattern_replace",
            "pattern": "(\\d+)-(?=\\d)",
            "replacement": "$1_"
          }
        }
      }
    }
  }
}
```

Type here to search

22:30 ENG 03/06/2020 19



Source  
Text

The two <em>lazy</em> dogs were slower than the less  
lazy <em>dog</em>, Rover.



html\_strip  
Char Filter

The two lazy dogs were slower than the less lazy dog,  
Rover.



standard  
Tokenizer

The<sup>1</sup> two<sup>2</sup> lazy<sup>3</sup> dogs<sup>4</sup> than<sup>5</sup> the<sup>6</sup>  
less<sup>7</sup> lazy<sup>8</sup> dog<sup>9</sup> Rover<sup>1</sup>



lowercase  
Token Filter

the<sup>1</sup> two<sup>2</sup> lazy<sup>3</sup> dogs<sup>4</sup> than<sup>5</sup> the<sup>6</sup>  
less<sup>7</sup> lazy<sup>8</sup> dog<sup>9</sup> rover<sup>10</sup>



stop  
Token Filter

two<sup>2</sup> lazy<sup>3</sup> dogs<sup>4</sup> than<sup>5</sup>  
less<sup>7</sup> lazy<sup>8</sup> dog<sup>9</sup> rover<sup>10</sup>



snowball  
Token Filter

two<sup>2</sup> lazi<sup>3</sup> dog<sup>4</sup> than<sup>5</sup>  
less<sup>7</sup> lazi<sup>8</sup> dog<sup>9</sup> rover<sup>10</sup>



# Pattern Replace Tokenizer

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 54 ms

```
1 | POST pattern_replace_analyzer/_analyze
2 { "analyzer": "vm_pattern_analyzer",
3   "text": "My credit card is 123-456-789"
4 }
```

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



# Custom Analyzer

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 1252 ms

PUT custom\_analyzer\_test

```
1 PUT custom_analyzer_test
2 {
3   "settings": {
4     "analysis": {
5       "analyzer": {
6         "mycustom_analyzer": {
7           "type": "custom",
8           "char_filter": ["html_strip"],
9           "tokenizer": "standard",
10          "filter": [
11            "lowercase", "stop", "asciifolding"
12          ]
13        }
14      }
15    }
16  }
17 }
```

```
1 {
2   "acknowledged": true,
3   "shards_acknowledged": true,
4   "index": "custom_analyzer_test"
5 }
```

241



# Custom Analyzer

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 85 ms

```
1 POST /custom_analyzer_test/_analyze
2 {
3   "analyzer": "mycustom_analyzer",
4   "text": ["Tom and Jerry fights always"]
5 }
```

```
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": "jerry",
12      "start_offset": 8,
13      "end_offset": 13,
14      "type": "<ALPHANUM>",
15      "position": 2
16    },
17    {
18      "token": "fights",
19      "start_offset": 14,
20      "end_offset": 20,
21      "type": "<ALPHANUM>",
22      "position": 3
23    },
24    {
25      "token": "always",
26      "start_offset": 21,
27      "end_offset": 27
28    }
29  ]
30}
```



# Custom Analyzer

Gmail Launch Meeting - Zoom localhost:9200/\_cat/shards/ elastic Kibana You are screen sharing Stop Share +

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 74 ms

```
55 GET near_deal-./.0-2020.06.03-000001/_search?pretty
34
35 PUT custom_analyzer_test_v2
36 {
37   "settings": {
38     "analysis": {
39       "analyzer": {
40         "mycustom_analyzer": {
41           "type": "custom",
42           "char_filter": ["html_strip", "my_char_filter"],
43           "tokenizer": "whitespace",
44           "filter": [
45             "lowercase", "stop", "asciifolding"
46           ]
47         }
48       },
49     },
50
51     "char_filter": {
52       "my_char_filter": {
53         "type": "pattern_replace",
54         "pattern": "(\\d+)-(?=\\d+)",
55         "replacement": "$1-"
56       }
57     }
58   }
59 }
```

```
5
6   "start_offset" : 4,
7   "end_offset" : 6,
8   "type" : "word",
9   "position" : 0
10  },
11  {
12    "token" : "credit",
13    "start_offset" : 15,
14    "end_offset" : 21,
15    "type" : "word",
16    "position" : 1
17  },
18  {
19    "token" : "card",
20    "start_offset" : 22,
21    "end_offset" : 26,
22    "type" : "word",
23    "position" : 2
24  },
25  {
26    "token" : "type",
27    "start_offset" : 34,
28    "end_offset" : 38,
29    "type" : "word",
30    "position" : 4
31  },
```

Type here to search

14:43 ENG 04/06/2020 22



# Custom Analyzer

Gmail Launch Meeting - Zoom localhost:9200/\_cat/shards/ elastic Kibana You are screen sharing Stop Share +

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 140 ms

48 }  
49 },  
50 |  
51 "char\_filter": {  
52 "my\_char\_filter": {  
53 "type": "pattern\_replace",  
54 "pattern": "(\\d+)-(\\d+)",  
55 "replacement": "\$1-\$"  
56 }  
57 }  
58 }  
59 }  
60 }  
61 }  
62 }  
63 POST custom\_analyzer\_test\_v2/\_analyze  
64 {  
65 "analyzer": "mycustom\_analyzer",  
66 "text": "<h1>My</h1> <p>credit card</p> of type visa with value 123  
-456-789"  
67 }  
68  
69  
70  
71  
72  
73

1 {  
2 "tokens": [  
3 {  
4 "token": "my",  
5 "start\_offset": 4,  
6 "end\_offset": 6,  
7 "type": "word",  
8 "position": 0  
9 },  
10 {  
11 "token": "credit",  
12 "start\_offset": 15,  
13 "end\_offset": 21,  
14 "type": "word",  
15 "position": 1  
16 },  
17 {  
18 "token": "card",  
19 "start\_offset": 22,  
20 "end\_offset": 26,  
21 "type": "word",  
22 "position": 2  
23 },  
24 {  
25 "token": "type",  
26 "start\_offset": 34,  
27 "end\_offset": 38  
28 }

Type here to search 14:43 04/06/2020



# Inverted Index

Term	Document #1	Document #2
best	X	
carbonara		X
delicious		X
pasta	X	X
pesto	X	
recipe	X	X
the	X	
with	X	



# Inverted Index

Term	Document #1	Document #2
best	X	
carbonara		X
delicious		X
pasta	X	X
pesto	X	
recipe	X	X
the	X	
with	X	



# Inverted Index

Term	Document #1	Document #2
best	X	
carbonara		X
delicious		X
pasta	X	X
pesto	X	
recipe	X	X
the	X	
with	X	



# Character Filters – 3 Types



## Mapping Character Filter (mapping)



Replaces values based on a map of keys and values.

*"I broke my leg\_sad\_ But luckily I have some red wine  
around\_happy\_"*



"I broke my leg :-( But luckily I have some red wine  
around :)"



# Character Filters – 3 Types

---

## Pattern Replace (pattern\_replace)



Uses a regular expression to match characters and replaces them with the specified replacement. Capture groups may be used.

**Pattern:** ([a-zA-Z0-9]+)(-?)

**Replacement:** \$1



"9c559866-d0a5-4dee-84fe-54d878fe1800"

"9c559866d0a54dee84fe54d878fe1800"