elasticsearch

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Elasticsearch is a real-time distributed and open source full-text search and analytics engine. It is used in Single Page Application (SPA) projects. Elasticsearch is open source developed in Java and used by many big organizations around the world. It is licensed under the Apache license version 2.0.

In this brief tutorial, we will be explaining the basics of Elasticsearch and its features.

Audience

This tutorial is designed for software professionals who want to learn the basics of Elasticsearch and its programming concepts in simple and easy steps. It describes the components of Elasticsearch with suitable examples.

Prerequisites

You should have a basic understanding of Java, JSON, search engines, and web technologies. The interaction with Elasticsearch is through RESTful API; therefore, it is always recommended to have knowledge of RESTful API.

Disclaimer & Copyright

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial i

Audience i

Prerequisites i

Disclaimer & Copyright i

Table of Contents ii

1. ELASTICSEARCH — BASIC CONCEPTS 1

Elasticsearch – General Features 1

Elasticsearch – Key Concepts 1

Elasticsearch – Advantages 2

Elasticsearch – Disadvantages 2

Comparison between Elasticsearch and RDBMS 3

2. ELASTICSEARCH — INSTALLATION 4

3. ELASTICSEARCH – POPULATE ELASTICSEARCH 8

Create Index 8

Create Mapping and Add data 8

Add another Index 9

Create Mapping and Add Data 9

4. ELASTICSEARCH — MIGRATION BETWEEN VERSIONS 11

Snapshot and Restore Module 11

Full Cluster Restart 12

Rolling Upgrades 14

5. ELASTICSEARCH — API CONVENTIONS 15

Multiple Indices 15

Comma Separated Notation 15

_all keyword for all indices 15

Wildcards (* , + , –) 16

allow_no_indices 18

expand_wildcards 19

Date Math Support in Index Names 19

Pretty Results 20

Human Readable Output 21

Response Filtering 21

6. ELASTICSEARCH — DOCUMENT APIS 22

Index API 22

Automatic Index Creation 22

Versioning 23

Operation Type 23

Automatic ID generation 24

Parents and Children 24

Timeout 25

Get API 25

Delete API 26

Update API 27

Multi Get API 28

Bulk API 28

7. ELASTICSEARCH — SEARCH APIS 29

Multi-Index 29

Multi-Type 29

URI Search 30

Request Body Search 30

8. ELASTICSEARCH — AGGREGATIONS 32

Metrics Aggregations 32

Avg Aggregation 32

Cardinality Aggregation 33

Extended Stats Aggregation 34

Max Aggregation 35

Min Aggregation 35

Sum Aggregation 36

Bucket Aggregations 36

Aggregation Metadata 37

9. ELASTICSEARCH — INDEX APIS 39

Create Index 39

Delete Index 41

Get Index 41

Index Exist 41

Open / Close Index API 42

Index Aliases 42

Index Settings 43

Analyze 43

Index Templates 44

Index Stats 44

Flush 45

Refresh..45

10. ELASTICSEARCH — CLUSTER APIS 46

Cluster Health 46

Cluster State..46

Cluster Stats 47

Pending Cluster Tasks..47

Cluster Reroute 47

Cluster Update Settings..48

Node Stats..48

Nodes hot_threads..49

11. ELASTICSEARCH — QUERY DSL 50

Match All Query 50

Full Text Queries 51

Match query..51

multi_match query..52

Query String Query 52

Term Level Queries 53

Range Query 53

Compound Queries 55

Joining Queries..56

Geo Queries 57

12. ELASTICSEARCH — MAPPING 59

Field Data Types..59

Mapping Types....61

Dynamic Mapping....61

Mapping Parameters....62

13. ELASTICSEARCH — ANALYSIS....63

Analyzers....64

Tokenizers....65

Token Filters....66

Character Filters....66

14. ELASTICSEARCH — MODULES....67

Cluster-Level Routing and Shard Allocation....67

Discovery....69

Gateway....70

HTTP....70

Indices....71

Node....72

15. ELASTICSEARCH — TESTING....73

Unit Testing....73

Integration Testing....73

Randomized Testing....75

Assertions....75

1. Elasticsearch — Basic Concepts

Elasticsearch is an Apache Lucene-based search server. It was developed by Shay Banon and published in 2010. It is now maintained by Elasticsearch BV. Its latest version is 2.1.0.

Elasticsearch is a real-time distributed and open source full-text search and analytics engine. It is accessible from RESTful web service interface and uses schema less JSON (JavaScript Object Notation) documents to store data. It is built on Java programming language, which enables Elasticsearch to run on different platforms. It enables users to explore very large amount of data at very high speed.

Elasticsearch – General Features

The general features of Elasticsearch are as follows:

- Elasticsearch is scalable up to petabytes of structured and unstructured data.
- Elasticsearch can be used as a replacement of document stores like MongoDB and RavenDB.
- Elasticsearch uses denormalization to improve the search performance.
- Elasticsearch is one of the popular enterprise search engines, which is currently being used by many big organizations like Wikipedia, The Guardian, StakOverflow, GitHub etc.
- Elasticsearch is open source and available under the Apache license version 2.0.

Elasticsearch – Key Concepts

The key concepts of Elasticsearch are as follows:

- **Node:** It refers to a single running instance of Elasticsearch. Single physical and virtual server accommodates multiple nodes depending upon the capabilities of their physical resources like RAM, storage and processing power.
- **Cluster:** It is a collection of one or more nodes. Cluster provides collective indexing and search capabilities across all the nodes for entire data.

- **Index:** It is a collection of different type of documents and document properties. Index also uses the concept of shards to improve the performance. For example, a set of document contains data of a social networking application.

- **Type/Mapping:** It is a collection of documents sharing a set of common fields present in the same index. For example, an Index contains data of a social networking application, and then there can be a specific type for user profile data, another type for messaging data and another for comments data.

- **Document:** It is a collection of fields in a specific manner defined in JSON format. Every document belongs to a type and resides inside an index. Every document is associated with a unique identifier, called the UID.

- **Shard:** Indexes are horizontally subdivided into shards. This means each shard contains all the properties of document, but contains less number of JSON objects than index. The horizontal separation makes shard an independent node, which can be store in any node. Primary shard is the original horizontal part of an index and then these primary shards are replicated into replica shards.

- **Replicas:** Elasticsearch allows a user to create replicas of their indexes and shards. Replication not only helps in increasing the availability of data in case of failure, but also improves the performance of searching by carrying out a parallel search operation in these replicas.

Elasticsearch – Advantages

- Elasticsearch is developed on Java, which makes it compatible on almost every platform.

- Elasticsearch is real time, in other words after one second the added document is searchable in this engine.

- Elasticsearch is distributed, which makes it easy to scale and integrate in any big organization.

- Creating full backups are easy by using the concept of gateway, which is present in Elasticsearch.

- Handling multi-tenancy is very easy in Elasticsearch when compared to Apache Solr.

- Elasticsearch uses JSON objects as responses, which makes it possible to invoke the Elasticsearch server with a large number of different programming languages.

- Elasticsearch supports almost every document type except those that do not support text rendering.

Elasticsearch – Disadvantages

- Elasticsearch does not have multi-language support in terms of handling request and response data (only possible in JSON) unlike in Apache Solr, where it is possible in CSV, XML and JSON formats.

- Elasticsearch also have a problem of Split brain situations, but in rare cases.

Comparison between Elasticsearch and RDBMS

In Elasticsearch, index is a collection of type just as database is a collection of tables in RDBMS (Relation Database Management System). Every table is a collection of rows just as every mapping is a collection of JSON objects Elasticsearch.

Elasticsearch	**RDBMS**
Index	Database
Shard	Shard
Mapping	Table
Field	Field
JSON Object	Tuple

2. Elasticsearch — Installation

The steps for installation of Elasticsearch are as follows:

Step 1: Check the minimum version of your java in installed your computer, it should be java 7 or more updated version. You can check by doing the following:

In Windows Operating System (OS) (using command prompt):

```
> java -version
```

In UNIX OS (Using Terminal):

```
$ echo $JAVA_HOME
```

Step 2: Download Elasticsearch from https://www.elastic.co/downloads/elasticsearch

- For windows OS download ZIP file.
- For UNIX OS download TAR file.
- For Debian OS download DEB file.
- For Red Hat and other Linux distributions download RPN file.
- APT and Yum utilities can also be used to install Elasticsearch in many Linux distributions.

Step 3: Installation process for Elasticsearch is very easy and described below for different OS:

- **Windows OS:** Unzip the zip package and the Elasticsearch is installed.
- **UNIX OS:** Extract tar file in any location and the Elasticsearch is installed.

  ```
  $tar -xvf elasticsearch-2.1.0.tar.gz
  ```

- **Using APT utility for Linux OS:**
  - Download and install the Public Signing Key:

```
$ wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch |
sudo     apt-key add -
```

- Save the repository definition:

```
$ echo "deb http://packages.elastic.co/elasticsearch/2.x/debian
stable main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch-
2.x.list
```

- Run update:

```
$ sudo apt-get update
```

- Now you can install by using the following command:

```
$ sudo apt-get install elasticsearch
```

- **Using YUM utility for Debian Linux OS**:
  - Download and install the Public Signing Key:

```
$ rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch
```

- ADD the below text in the file with .repo suffix in your "/etc/yum.repos.d/" directory. For example, **elasticsearch.repo**

```
[elasticsearch-2.x]
name=Elasticsearch repository for2.x packages
baseurl=http://packages.elastic.co/elasticsearch/2.x/centos
gpgcheck=1
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

- You can now install Elasticsearch by using the following command:

```
$ yum install elasticsearch
```

Step 4: Go to the Elasticsearch home directory and inside the bin folder. Run the elasticsearch.bat file in case of windows or you can do the same using command prompt and through terminal in case of UNIX rum Elasticsearch file.

In Windows:

```
> cd elasticsearch-2.1.0/bin
> elasticsearch
```

In Linux:

```
$ cd elasticsearch-2.1.0/bin
$ ./elasticsearch
```

Note: in case of windows, you might get error stating JAVA_HOME is not set, please set it in environment variables to "C:\Program Files\Java\jre1.8.0_31" or the location where you installed java.

Step 5: Default port for Elasticsearch web interface is 9200 or you can change it by changing http.port inside elasticsearch.yml file present in bin directory. You can check if the server is up and running by browsing http://localhost:9200. It will return a JSON object, which contains the information about the installed Elasticsearch in the following way:

```
{
  "name" : "Brain-Child",
  "cluster_name" : "elasticsearch",
  "version" : {
    "number" : "2.1.0",
    "build_hash" : "72cd1f1a3eee09505e036106146dc1949dc5dc87",
    "build_timestamp" : "2015-11-18T22:40:03Z",
    "build_snapshot" : false,
    "lucene_version" : "5.3.1"
  },
  "tagline" : "You Know, for Search"
}
```

Step 6: You can install fiddler2 from http://www.telerik.com/download/fiddler as a front end for your Elasticsearch.

- In the configure window of fiddler2, you can hit the address of Elasticsearch adding an index and if you want, then the type/mapping also using HTTP POST method, for example:

  Address bar

  ```
  http://localhost:9200/schools/school
  ```

  Request body

  You can add JSON object, which will get store into that index.

- You can use the same for searching anything by just adding "_search" keyword at the end of URL and sent a query in request body for example:

  Address bar

  ```
  POST http://localhost:9200/city/schools/_search
  ```

  Request body

  ```
  {
    "query":{
              "match_all":{}
            }
  }
  ```

This query will return everything from that index, which belongs to that particular type.

- You can delete a particular index or type by just putting the URL of the same in address bar and hit it with HTTP DELETE method.

3. Elasticsearch – Populate Elasticsearch

In this section, we will add some index, mapping and data to Elasticsearch. This data will be used in the examples explained in this tutorial.

Create Index

```
POST http://localhost:9200/schools
```

Request Body

It can contain index specific settings, but for now, it is empty for default settings.

Response

```
{"acknowledged": true}
```

(This means index is created)

Create Mapping and Add data

Elasticsearch will auto-create the mapping according to the data provided in request body, we will use its bulk functionality to add more than one JSON object in this index.

```
POST http://localhost:9200/schools/_bulk
```

Request Body

```
{"index":{"_index":"schools", "_type":"school", "_id":"1"}}
{"name":"Central School", "description":"CBSE Affiliation", "street":"Nagan",
"city":"paprola", "state":"HP", "zip":"176115",
"location":[31.8955385,76.8380405], "fees":2000,
"tags":["Senior Secondary", "beautiful campus"],"rating":"3.5"}


{"index":{"_index":"schools", "_type":"school", "_id":"2"}}
{"name":"Saint Paul School", "description":"ICSE
Afiliation","street":"Dawarka","city":"Delhi","state":"Delhi","zip":"110075","loca
tion":[28.5733056,77.0122136],"fees":5000,
"tags":["Good Faculty", "Great Sports"],"rating":"4.5"}


{"index":{"_index":"schools", "_type":"school", "_id":"3"}}

{"name":"Crescent School", "description":"State Board Affiliation", "street":"Tonk
Road", "city":"Jaipur", "state":"RJ",

"zip":"176114","location":[26.8535922,75.7923988],"fees":2500,"tags":["Well
equipped labs"],"rating":"4.5"}
```

Response

```
{"took":328,"errors":false,"items":[{"index":{"_index":"schools","_type":"school",
"_id":"1","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":20
1}},{"index":{"_index":"schools","_type":"school","_id":"2","_version":1,"_shards"
:{"total":2,"successful":1,"failed":0},"status":201}},{"index":{"_index":"schools"
,"_type":"school","_id":"3","_version":1,"_shards":{"total":2,"successful":1,"fail
ed":0},"status":201}}]}
```

Add another Index

Create Index

```
POST http://localhost:9200/schools_gov
```

Request Body

It can contain index specific settings, but for now it’s empty for default settings.

Response

```
{"acknowledged": true} (This means index is created)
```

Create Mapping and Add Data

```
POST http://localhost:9200/schools_gov/_bulk
```

Request Body

```
{"index":{"_index":"schools_gov", "_type":"school", "_id":"1"}}
{"name":"Model School", "description":"CBSE Affiliation",
"street":"silk city", "city":"Hyderabad", "state":"AP", "zip":"500030",
"location":[17.3903703,78.4752129],  "fees":200,
 "tags":["Senior Secondary", "beautiful campus"],"rating":"3"}


{"index":{"_index":"schools_gov", "_type":"school", "_id":"2"}}
{"name":"Government School", "description":"State Board Affiliation",
"street":"Hinjewadi", "city":"Pune", "state":"MH", "zip":"411057",
"location": [18.599752, 73.6821995],"fees":500,"tags":["Great
Sports"],"rating":"4"}
```

Response

```
{"took":179,"errors":false,"items":[{"index":{"_index":"schools_gov","_type":"scho
ol","_id":"1","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status
":201}},{"index":{"_index":"schools_gov","_type":"school","_id":"2","_version":1,"
_shards":{"total":2,"successful":1,"failed":0},"status":201}}]}
```