**ElasticSearch**

**Version 6.4.2**

Notice of Proprietary Property

The information contained herein is the property of Girmiti Software Private Limited the possessor agrees to maintain this document in Confidence, not to reproduce, copy, reveal or publish it in whole or in part.

Table of Contents

[1 Introduction 3](#_Toc527535621)

[2 Component 3](#_Toc527535622)

[2.1 How it helps – Features and Scenarios to use 3](#_Toc527535623)

[2.2 Advantages 3](#_Toc527535624)

[2.3 Setup and Execution 3](#_Toc527535625)

[2.4 Disadvantages 3](#_Toc527535626)

[2.5 Equivalent OR Alternative Component 3](#_Toc527535627)

[3 Dependencies 3](#_Toc527535628)

[4 References 3](#_Toc527535629)

# 

# Introduction

**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.

You can send data in the form of JSON documents to Elasticsearch using the API,Elasticsearch automatically stores the original document and adds a searchable reference to the document in the cluster’s index. You can then search and retrieve the document using the Elasticsearch API. You can also use [Kibana](https://aws.amazon.com/elasticsearch-service/kibana/), an open-source visualization tool, with Elasticsearch to visualize your data and build interactive dashboards.

# Components

# **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.
* **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.
* **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.
* **Shards** - A shard is a subset of documents of an index. An index can be divided into many shards.

## How it helps – Features and Scenarios to use

**1 – Logging and Log Analysis**

For anyone familiar with Elasticsearch, this one should be no surprise. The ecosystem built up around Elasticsearch has made it one of the easiest to implement and scale logging solutions. Many of the the users on our platform are no different and have taken advantage of this to either add logging to their main use case, or are using us purely for logging. From Beats, to Logstash, to Ingest Nodes, Elasticsearch gives you plenty of options for grabbing data wherever it lives and getting it indexed. From there, tools like Kibana give you the ability to create rich dashboards and analysis, while Curator allows you to put the retention period on autopilot.

**2 – Scraping and Combining Public Data**

Like log data, the Elastic Stack has plenty of tools to make grabbing and indexing remote data easy. Also, like most document stores, the lack of a strict schema gives Elasticsearch the flexibility to take in multiple different sources of data and still keep it all manageable and searchable. A cool example of this that you can check out is our Twitter connector, which allows you to set up hashtags to watch on Twitter and then grab all tweets with those hashtags and analyze them in Kibana. We built that product on core Elastic Stack components and added some additional pieces to help it scale.

**3 – Full Text Search**

It’s also no surprise that full text search, as the core capability of Elasticsearch, is high on this list. The surprising part is the applications of this among our customer set, which go well beyond traditional Enterprise search or E-commerce. From fraud detection/security to collaboration and beyond, our users have shown that Elasticsearch’s search capabilities are powerful, flexible, and include a great number of tools to make search easier; Elasticsearch has its own query DSL as well as built in capabilities for auto-complete, “Did you mean” responses, and more.

**4 – Event Data and Metrics**

Elasticsearch also operates really well on time-series data like metrics and application events. This is another area where the huge Beats ecosystem allows you to easily grab data for common applications. Whatever technologies you use, there’s a pretty good chance that Elasticsearch has the components to grab metrics and events out of the box… and in the rare case that it can’t, adding that capability is really easy.

**5 – Visualizing Data**

With tons of charting options, a tile service for geo-data, and TimeLion for time-series data, Kibana is an amazingly powerful and easy to use visualization tool. For every use case above there is some visual component handled by Kibana. Once you’re comfortable with the various data ingest tools, you’ll find that Elasticsearch + Kibana will become your go-to tool for visualizing data that you’re trying to wrap your head around

## Advantages

* **FAST TIME TO VALUE**

Elasticsearch offers simple REST based APIs, a simple HTTP interface, and uses schema-free JSON documents, making it easy to get started and quickly build applications for a variety of use-cases.

* **HIGH PERFORMANCE**

The distributed nature of Elasticsearch enables it to process large volumes of data in parallel, quickly finding the best matches for your queries.

* **NEAR REAL-TIME OPERATIONS**

Elasticsearch operations such as reading or writing data usually take less than a second to complete. This lets you use Elasticsearch for near real-time use cases such as application monitoring and anomaly detection.

* **EASY APPLICATION DEVELOPMENT**

Elasticsearch provides support for various languages including Java, Python, PHP, JavaScript, Node.js, Ruby, and many more.

* **BACKUP**

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

## Setup and Execution

**2.3.1 Installing Elasticsearch**

* Download latest version of elasticsearch from https://www.elastic.co/downloads/elasticsearch
* Install Deb file and , go to elasticsearch6.4.2/bin > openterminal and run elasticsearch.sh file as *./elasticsearch*
* then elasticsearch server will up in 9200 port
* check in browser as http://localhost: 9200

ElasticSearch version : 6.4.2

**2.3.2 Installing Kibana**

* Download kibana deb file

wget https://artifacts.elastic.co/downloads/kibana/kibana-6.4.2-amd64.deb and install it

* To start kibana service : service kibana start
* open kibana in browser : localhost:5601
* Kibana version : 6.4.2

**2.3.3 Import Project**

* import project into eclipse
* Run as spring boot application

**2.3.4 Running the project**

* import Demo project into the Eclipse
* Run as the spingboot project *spring-boot:run*, then application will start In default port 8080 and elastic search is running in 9200

**save data in elastic search :**

* open postman

**Req :** POST localhost:8080/product/addProduct

{

"id" :1,

"productName" : "Motorola",

"price" : "100"

}

**Resp :** {

"id": 1,

"productName": "Motorola",

"price": "100"

}

**Get all data :**

**Req :** GET localhost:8080/product/getAllProducts

**Resp :** [{

"id": 7,

"productName": "iphone6",

"price": "100"

},

{

"id": 1,

"productName": "Motorola",

"price": "100"

}

}]

**Get data by ID :**

**Req :** GET localhost:8080/product/1

**Resp :** {

"id": 1,

"productName": "Motorola",

"price": "100"

}

**Update data by ID :**

**Req :** PUT localhost:8080/product/1

{

"id" :1,

"productName" : "Moto",

"price" : "100"

}

**Resp :** {

"id": 1,

"productName": "Motorola",

"price": "100"

}

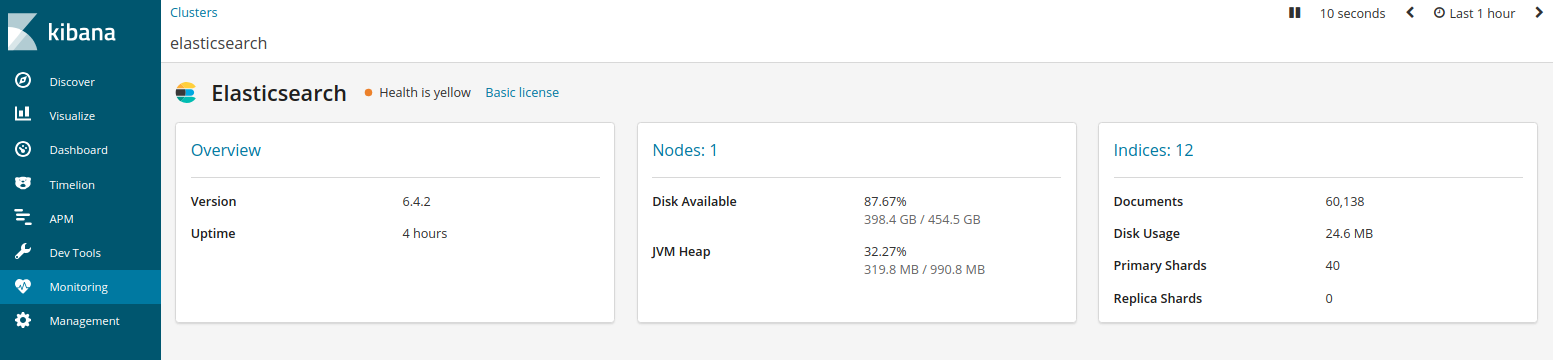
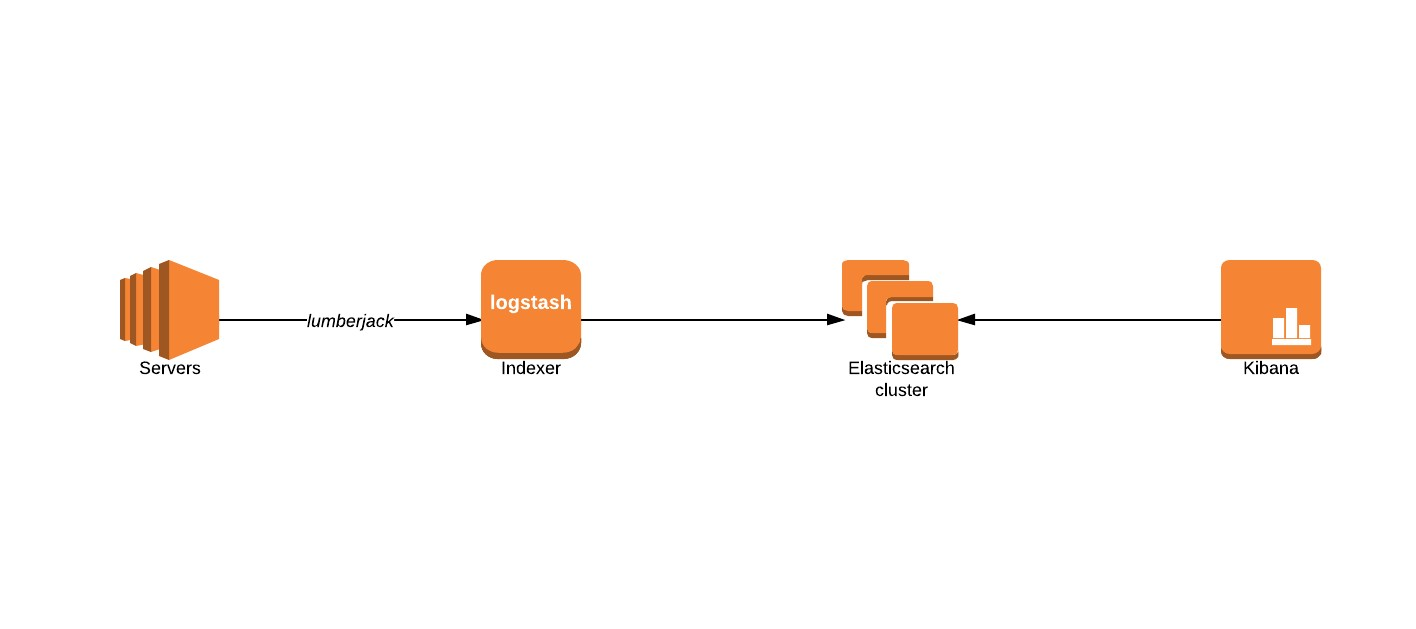
**Delete data by ID :**

**Req :** DEL localhost:8080/product/1

**Resp :** Product 1 Document Deleted

**2.3.5 Log Analysis**

* Elasticsearch is vastly used as a centralized location for storing logs.
* For the purpose of indexing and searching logs, there is a bundled solution offered at the Elasticsearch page - ELK stack, which stands for elasticsearch, logstash and kibana.





**2.3.6**  **Basic urls and commands**

* To check elasicservice properties : localhost:9200
* To Create Index : PUT localhost:9200/index\_name
* To check Indexes present in elasticsearch cluster : localhost:9200/\_cat/indices
* To check Data inside Index : localhost:9200/\_search?q=\*&index=index\_name
* To delete index : DELETE /index\_name (in Kibana > Console)

## 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.

## Equivalent OR Alternative Component

**Elasticsearch VS RDBMS**

* Elasticsearch is a No sql Database.
* it has no relations, no constraints, no joins, no transactional behaviour.
* Easier to scale as compared to a relational Database.

|  |  |
| --- | --- |
| **Relational DB** | **Elasticsearch** |
| database | Index |
| Table | Type |
| Row/Record | Document |
| Column Name | Field |

**Usecases where Relational Databases are not suitable**

* Relevance based searching
* Searching when entered spelling of search term is wrong
* Full text search
* Synonym search
* Phonetic search
* Log analysis

**Wrong Spelling searching in Elastic search by query**

**Query**

{ "query":{

"match":{

"name":{

"query": "shrt",

"fuzziness": 2,

"prefix\_length": 0

}

}

}

}

**Result**

{

"\_index": "test",

"\_type": "product",

"\_id": "AV0iKKplJJfvpLB9dSHk",

"\_score": 0.21576157,

"\_source": {

"id": 1,

}

"name": "Shirt"

}

}

# Dependencies

* Need to add “spring-boot-starter-data-elasticsearch” Dependency in pom.xml

<!-- https://mvnrepository.com/artifact/org.springframework.data/spring-data-elasticsearch -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-elasticsearch</artifactId>

<version>3.1.0.RELEASE</version>

</dependency>

* spring boot dependencies – spring-boot-starter-parent ,spring-boot-starter-web
* Need to add “spring.data.elasticsearch.cluster-nodes=localhost:9300”

property in application.properties

<https://www.elastic.co/guide/en/elasticsearch>

<https://projects.spring.io/spring-data-elasticsearch/>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-oubuntu16-04>

<https://www.journaldev.com/18148/spring-boot-elasticsearch>

<https://www.tutorialspoint.com/elasticsearch/elasticsearch_basic_concepts.htm>

<https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/>

<https://www.slideshare.net/richu100/elasticsearch-vs-relational-database>

<https://dzone.com/articles/what-is-elasticsearch-and-how-it-can-be-useful>

<https://en.wikipedia.org/wiki/Elasticsearch>

**4 References**

https://www.elastic.co/guide/en/elasticsearch

https://projects.spring.io/spring-data-elasticsearch/

https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-oubuntu16-04

https://www.journaldev.com/18148/spring-boot-elasticsearch

https://www.tutorialspoint.com/elasticsearch/elasticsearch\_basic\_concepts.htm

https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/

https://www.slideshare.net/richu100/elasticsearch-vs-relational-database

https://dzone.com/articles/what-is-elasticsearch-and-how-it-can-be-useful

<https://en.wikipedia.org/wiki/Elasticsearch>

https://www.objectrocket.com/blog/elasticsearch/top-elasticsearch-use-cases/

<https://www.elastic.co/guide/en/elasticsearch>

<https://projects.spring.io/spring-data-elasticsearch/>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-oubuntu16-04>

<https://www.journaldev.com/18148/spring-boot-elasticsearch>

<https://www.tutorialspoint.com/elasticsearch/elasticsearch_basic_concepts.htm>

<https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/>

<https://www.slideshare.net/richu100/elasticsearch-vs-relational-database>

<https://dzone.com/articles/what-is-elasticsearch-and-how-it-can-be-useful>

<https://en.wikipedia.org/wiki/Elasticsearch>

<https://www.elastic.co/guide/en/elasticsearch>

<https://projects.spring.io/spring-data-elasticsearch/>

<https://projects.spring.io/spring-data-elasticsearch/>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-oubuntu16-04>

<https://www.journaldev.com/18148/spring-boot-elasticsearch>

<https://www.tutorialspoint.com/elasticsearch/elasticsearch_basic_concepts.htm>

<https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/>

<https://www.slideshare.net/richu100/elasticsearch-vs-relational-database>

<https://dzone.com/articles/what-is-elasticsearch-and-how-it-can-be-useful>

<https://en.wikipedia.org/wiki/Elasticsearch>

<https://projects.spring.io/spring-data-elasticsearch/>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-oubuntu16-04>

<https://www.journaldev.com/18148/spring-boot-elasticsearch>

<https://www.tutorialspoint.com/elasticsearch/elasticsearch_basic_concepts.htm>

<https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/>

<https://www.slideshare.net/richu100/elasticsearch-vs-relational-database>

<https://dzone.com/articles/what-is-elasticsearch-and-how-it-can-be-useful>

<https://en.wikipedia.org/wiki/Elasticsearch>