

# AN INTRODUCTION TO ELASTICSEARCH

Contributors:  
Arnab Bhowmick

**HCL Technologies**  
Enterprise Productivity Service Line – Modern Apps

Version No	Date	Prepared by	Reviewed by
1.0	19 <sup>th</sup> Jan 2017	Arnab Bhowmick	Arup Kumar Pal

## Contents

Overview .....	4
Basic Architecture of Elasticsearch .....	5
Cluster .....	5
Node.....	5
Master Node .....	5
Data Node .....	5
Client Node.....	6
Index.....	6
Shard .....	6
Replica.....	6
Type.....	6
Document.....	6
Installing Elasticsearch .....	7
Talking to Elasticsearch.....	8
Installing Kibana .....	8
Scaling Out Elasticsearch .....	9
Creating Index .....	9
CRUD Operation on Elasticsearch.....	11
Add a document to an index.....	11
Read, Update and Delete a document.....	12
Example: An Elasticsearch application on .NET .....	12
NEST .....	12
Logstash .....	14
Input.....	14
Filter .....	14
Output.....	14
Kibana .....	15
Use case of ELK stack .....	16
References .....	17

## Figures

Figure 1: Elastic Stack.....	4
Figure 2: An index with 3 shards and one replica on 3 nodes architecture.....	7
Figure 3: A Running Elasticsearch Node.....	8
Figure 4: A default instance of Kibana running on http://localhost:5601.....	9
Figure 5: Create Elasticsearch Index.....	9
Figure 6: _Cluster/health response.....	10
Figure 7: _Cluster/health after adding one more node.....	11
Figure 8: Add a document to an index.....	11
Figure 9: Read, update or delete a document from index.....	12
Figure 10: Auto complete with Elasticsearch.....	13
Figure 11: Search result page.....	13
Figure 12: Logstash high level architecture.....	14
Figure 13: Kibana default installation tabs.....	15
Figure 14: Visualize tab in Kibana.....	15
Figure 15: A sample Kibana dashboard.....	16

## Overview

Elasticsearch is a real-time distributed search and analytics engine. It is an open-source search engine built on top of Apache Lucene™, a full-text search-engine library. Elasticsearch allows us to explore data at a speed and at a scale that was not possible before. It is used for full-text search, structured search, analytics, and all three in combination.

- Wikipedia uses Elasticsearch to provide full-text search with highlighted search snippets, and search-as-you-type and did-you-mean suggestions.
- The Guardian uses Elasticsearch to combine visitor logs with social -network data to provide real-time feedback to its editors about the public’s response to new articles.
- Stack Overflow combines full-text search with geolocation queries and uses more-like-this to find related questions and answers.
- GitHub uses Elasticsearch to query 130 billion lines of code.

Elasticsearch is developed in Java and is released as open source under the terms of the Apache License. Elasticsearch is the most popular enterprise search engine today. Please refer to the [reference](#) section for more Elasticsearch use cases from various enterprises.

It is developed alongside a data collection and log parsing engine called "Logstash", and an analytics and visualization platform called "Kibana". These three products together called "ELK stack".

Though the basic Elasticsearch search engine and other two products (Logstash & Kibana) are free, but it doesn't contain any security components or connectors for any document source. These need to be built separately or some paid components need to be used. Elastic as an organization themselves have some X-Pack components that can be used. There are various third-party components also available. Companies like “Search Technologies” or “BA Insights” Elasticsearch connectors for various data sources. Few open source components are also available on GitHub. Please refer to the [reference](#) section for URL.



Figure 1: Elastic Stack

## Basic Architecture of Elasticsearch

Elasticsearch is built to be always available, and to scale with needs. Scale can come from buying bigger servers (vertical scale, or scaling up) or from buying more servers (horizontal scale, or scaling out).

An Elasticsearch setup is identified by a Cluster. A Cluster can have one or more nodes.

### Cluster

- A cluster is a collection of nodes (servers).
- Consists of one or more nodes depending on scale.
  - It can contain as many node as we need.
- Together, these nodes contain all data.
- A cluster provides indexing and searching capabilities across all nodes.
- A cluster is identified by a unique name (default installation has the name "elasticsearch").

### Node

- A node is a single server that is part of a cluster.
- It stores searchable data.
  - Stores all data if there is only one node present in the cluster, or part of the data in case of multiple nodes in the cluster.
- It is identified by a node name.
- By default, a node joins to a cluster called "elasticsearch".

Nodes in Elasticsearch cluster can be configured in three different ways.

### Master Node

- It controls the Elasticsearch cluster and is responsible for all cluster wide operations like creating/deleting an index, keeping track of which nodes are part of the cluster and assigning shards to nodes. The master node processes one cluster state at a time and broadcasts the state to all the other nodes which respond with confirmation to the master node.
- A node can be configured to be eligible to become a master node by setting the node.master property to be true (default) in elasticsearch.yml.
- For large production clusters, it's recommended to have a dedicated master node to just control the cluster and not serve any user requests.

### Data Node

- It holds the data and the inverted index. By default, every node is configured to be a data node and the property node.data is set to true in elasticsearch.yml. If we want to have a dedicated master node, then change the node.data property to false.

## Client Node

- If both node.master and node.data are set to false, then the node gets configured as a client node and acts a load balancer routing incoming requests to different nodes in the cluster.
- A client node is also called as coordinating node. The coordinating node routes the client requests to the appropriate shard in the cluster. For read requests, the coordinating node selects a different shard every time to serve the request in order to balance the load.

## Index

- An Elasticsearch index is a logical namespace to organize your data (like a database).
- An Elasticsearch index has one or more shards (default is 5).
- An Elasticsearch index also has “types” (like tables in a database) which allow you to logically partition your data in an index.
- All documents in a given “type” in an Elasticsearch index have the same properties (like schema for a table).
- An Elasticsearch cluster can have as many indices as require.

## Shard

- A shard is a Lucene index which actually stores the data and is a search engine in itself.
- Each shard can have zero or more replicas (default is 1).
- A shard can be stored in any data node of an Elasticsearch cluster.
- No. of shards can be specified during index creation.
- It allows to distribute and parallelize operations across shards, which increases performance.

## Replica

- A replica is a copy of a shard.
- Each shard can have zero or more replicas (default is 1).
- It provides high availability in case a shard or node fails.
  - A replica never resides on the same node as shard.
- Allows scaling search volume, because search queries can be executed on all replicas in parallel.

## Type

- A type represents a class or category of similar documents.
- A type is like a table in a database. It consists a name and a mapping.
- An index can have one or more type defined, each with their own mapping.

## Document

- Document is a basic unit of information that can be indexed.
- Consists of fields which are key-value pair.
  - A value can be a string, date, object, etc.
- Documents are expressed in JSON.

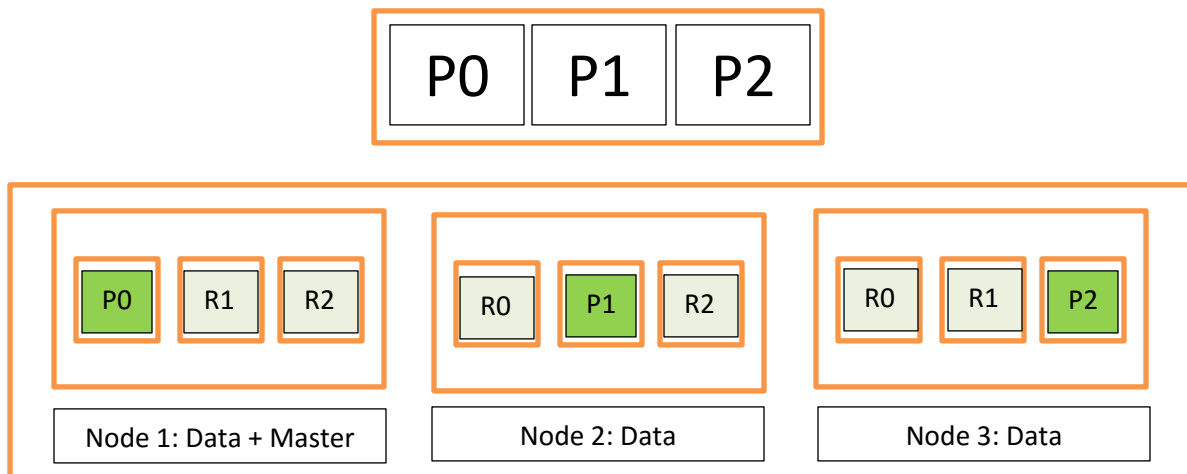
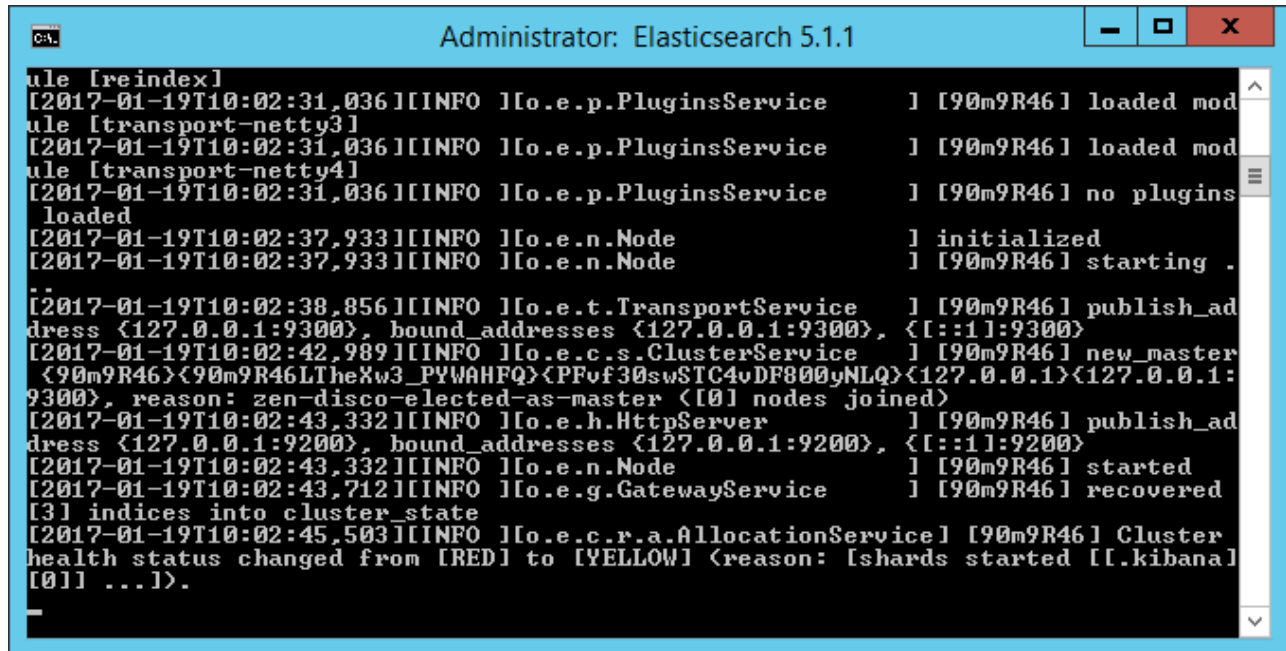


Figure 2: An index with 3 shards and one replica on 3 nodes architecture

## Installing Elasticsearch

- Download Elasticsearch latest version zip file from Elasticsearch site.
- Unzip the file.
- Install latest Java VM on the server if not already installed.
- Set JAVA\_HOME environment variable.
- Open a command prompt and navigate to <Elasticsearch unzipped folder>/bin.
- Run "Elasticsearch-service.bat" if want to install Elasticsearch as windows service.
- To simply run Elasticsearch, run "Elasticsearch.bat" file. It will start Elasticsearch cluster with one node.
- To verify, if the search has started perfectly- Go to browser, type "<http://localhost:9200>". It should either download a JSON file or should open the content directly.



```

Administrator: Elasticsearch 5.1.1
ule [reindex]
[2017-01-19T10:02:31,036][INFO ][o.e.p.PluginsService] 1 [90m9R46] loaded mod
ule [transport-netty3]
[2017-01-19T10:02:31,036][INFO ][o.e.p.PluginsService] 1 [90m9R46] loaded mod
ule [transport-netty4]
[2017-01-19T10:02:31,036][INFO ][o.e.p.PluginsService] 1 [90m9R46] no plugins
loaded
[2017-01-19T10:02:37,933][INFO ][o.e.n.Node] 1 initialized
[2017-01-19T10:02:37,933][INFO ][o.e.n.Node] 1 [90m9R46] starting .
...
[2017-01-19T10:02:38,856][INFO ][o.e.t.TransportService] 1 [90m9R46] publish_ad
dress <127.0.0.1:9300>, bound_addresses <127.0.0.1:9300>, <[::1]:9300>
[2017-01-19T10:02:42,989][INFO ][o.e.c.s.ClusterService] 1 [90m9R46] new_master
<90m9R46><90m9R46LTheXw3_PYWAhFQ><PFvf30swSIC4vDF800yNLQ><127.0.0.1><127.0.0.1:
9300>, reason: zen-disco-elected-as-master <[0] nodes joined>
[2017-01-19T10:02:43,332][INFO ][o.e.h.HttpServer] 1 [90m9R46] publish_ad
dress <127.0.0.1:9200>, bound_addresses <127.0.0.1:9200>, <[::1]:9200>
[2017-01-19T10:02:43,332][INFO ][o.e.n.Node] 1 [90m9R46] started
[2017-01-19T10:02:43,712][INFO ][o.e.g.GatewayService] 1 [90m9R46] recovered
[3] indices into cluster_state
[2017-01-19T10:02:45,503][INFO ][o.e.c.r.a.AllocationService] [90m9R46] Cluster
health status changed from [RED] to [YELLOW] (reason: [shards started [[.kibana]
[0]] ...]).

```

Figure 3: A Running Elasticsearch Node

## Talking to Elasticsearch

Communication to Elasticsearch depends on whether we are using Java or not.

- Elasticsearch comes with two build-in clients for Java.
  - Node client &
  - Transport client
- For non-Java usage, it uses RESTful API with JSON over http.

In this article we will use REST API for all examples.

We can use REST API from any programming language (including C#, PHP, JavaScript, Curl, PowerShell, etc.) for creating Elasticsearch index and doing all CRUD & search operations. For this article, we will use Query DSL from Kibana developer tool.

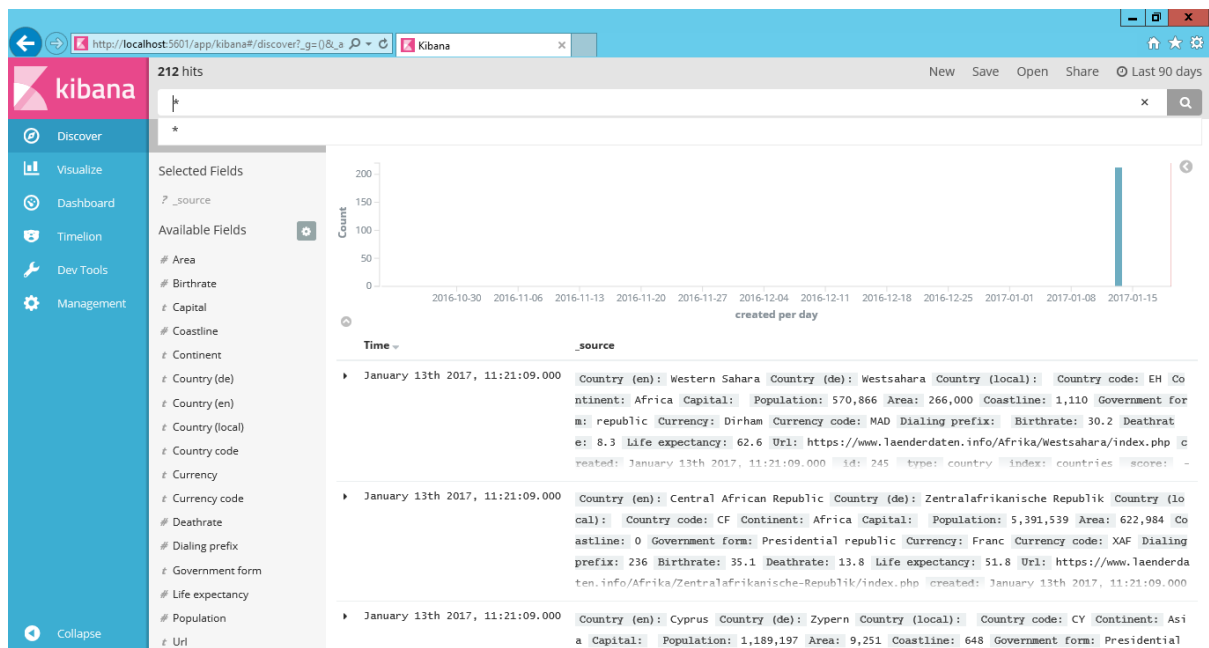
## Installing Kibana

Kibana can be installed same way as Elasticsearch.

- Download Kibana latest version zip file from Elasticsearch site.
- Unzip the file.
- Open a command prompt and navigate to <Kibana unzipped folder>/bin.



- Run "Kibana.bat" file.
- To verify, if the Kibana has started perfectly- Go to browser, type "<http://localhost:5601>". It should open Kibana.



**Figure 4: A default instance of Kibana running on <http://localhost:5601>**

## Scaling Out Elasticsearch

Elasticsearch, by nature is distributed. It is very easy to scale out to multiple nodes.

Let's start with creating an index with 3 primary shards and 1 replica.

### Creating Index

Following REST API will create an index named "employee" with 3 primary shards and one replica.

```
PUT /employee
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 1
  }
}
```

**Figure 5: Create Elasticsearch Index**

At present, we have only one node running for this Elasticsearch cluster. To check cluster health, we can use

GET /\_cluster/health command. It will return result like below:

```
{
  "cluster_name": "ESDemoHCL-Cluster",
  "status": "yellow",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "active_primary_shards": 4,
  "active_shards": 4,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 4,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 50
}
```

Figure 6: \_Cluster/health response

Cluster status is showing “yellow” in the above screenshot as there is only one data node. Primary shards are kept on this node. But, there are unassigned shards as replicas couldn’t be assigned due to non-availability of other data nodes.

**\*Note:** We have created the index with 3 primary shards. But, the above screenshot is showing 4 primary shards as Kibana is using one shard.

Now, we will start one more node keeping the same cluster name. This node will be automatically attached to the existing Elasticsearch cluster. Also, shards will be automatically distributed among the nodes. After starting one more node, if we check cluster health again, we will get the following response.

```
{
  "cluster_name": "ESDemoHCL-Cluster",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 2,
  "number_of_data_nodes": 2,
  "active_primary_shards": 4,
  "active_shards": 8,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100
}
```

Figure 7: \_Cluster/health after adding one more node

Above screenshot shows, now cluster health status changes to “green”. Also, total no. of data nodes is now 2. 4 replica nodes are now assigned to some nodes, making total shards no. 8.

This way, Elasticsearch can be scaled out to hundreds of nodes depending on the data volume.

## CRUD Operation on Elasticsearch

In this section we will discuss how we can add a document to an index, retrieve a document, update or delete a document.

### Add a document to an index

PUT verb can be used to insert a document for indexing. A type may or may not be defined before inserting document. If, a type is not defined, it will be automatically created during insertion of document. Following command will insert one employee information with id 1 to “employees” index, “employee” type.

```
PUT /employees/employee/1
{
  "first-name": "john",
  "last-name": "smith",
  "age": 35
}
```

Figure 8: Add a document to an index

## Read, Update and Delete a document

Following screenshot containing 3 commands shows how to read, update or delete a document respectively.

```
GET /employees/employee/1

PUT /employees/employee/1
{
  "doc":{
    .....
    "age": 40
  }
}

DELETE /employees/employee/1
```

**Figure 9: Read, update or delete a document from index**

## Example: An Elasticsearch application on .NET

Following example shows how an Elasticsearch based application can be developed on .NET using Visual Studio. This is an open source example available on GitHub. One can download and setup this example project on his/her own environment and compile it to see it running.

### NEST

To develop Elasticsearch connector on .Net, we need some .Net client. NEST is one of the two official .Net Client for Elasticsearch. NEST can be included and managed using nugget package manager from Visual Studio.

This example application uses a single page application and Angular JS with MVVM framework. The application searches like a web search and provides auto complete functionality. Following screenshots shows the application in running.

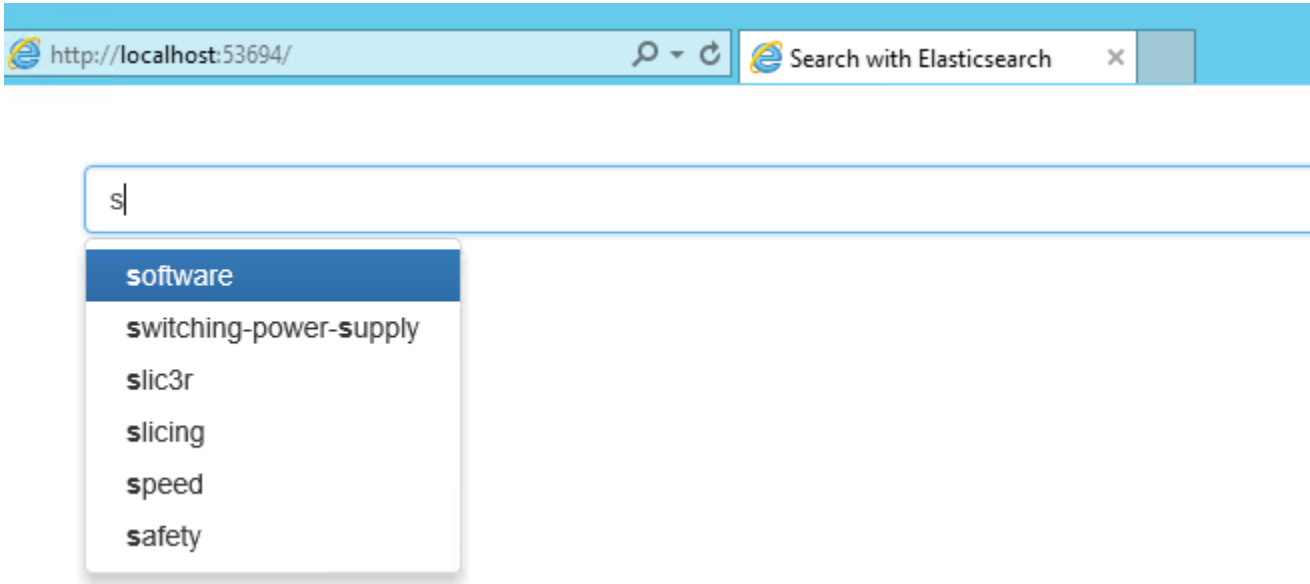


Figure 10: Auto complete with Elasticsearch

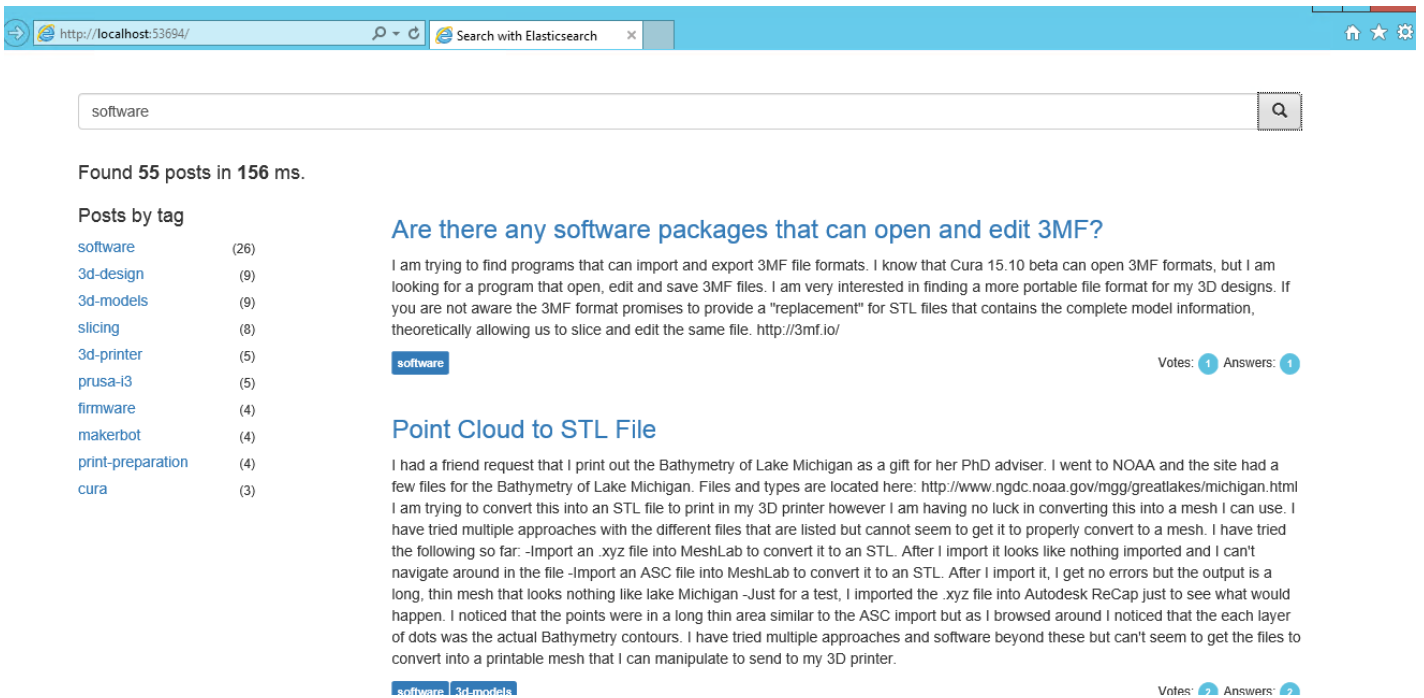


Figure 11: Search result page

## Logstash

Apart from Elasticsearch, one of the other component of ELK stack is Logstash. It is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to different types of “stash.”

The Logstash event processing pipeline has three stages: inputs → filters → outputs. Inputs generate events, filters modify them, and outputs ship them elsewhere. Inputs and outputs support codecs that enable you to encode or decode the data as it enters or exits the pipeline without having to use a separate filter.

### Input

It ingests data of all shapes, sizes & sources. Data is often scattered or siloed across many systems in many formats. Logstash supports a variety of inputs that pull in events from a multitude of common sources, all at the same time. Easily ingest from logs, metrics, web applications, data stores, and various AWS services, all in continuous, streaming fashion.

### Filter

It parses and transform input data on the fly. As data travels from source to store, Logstash filters parse each event, identify named fields to build structure, and transform them to converge on a common format for easier, accelerated analysis and business value.

### Output

It is the final phase of the Logstash pipeline. After transformation, Logstash can sends those data to various types of “Stash” including “Elasticsearch”.

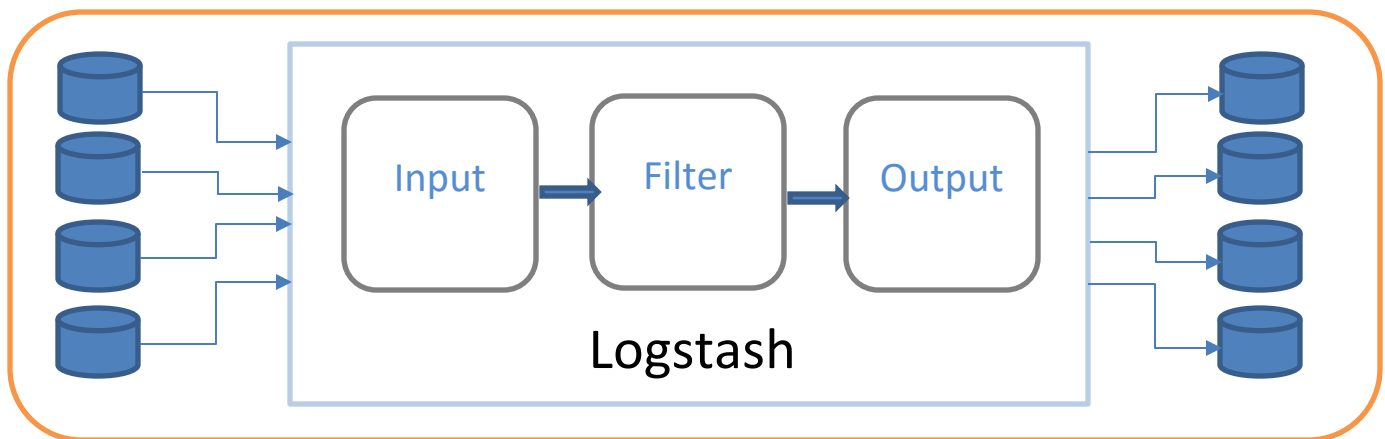
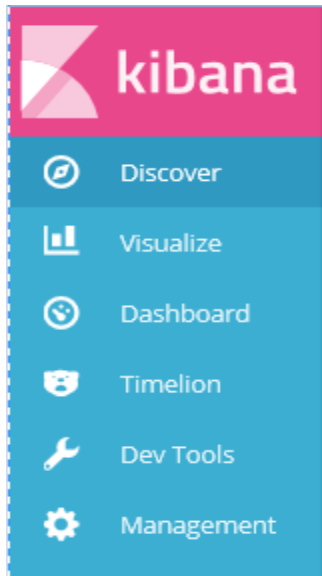


Figure 12: Logstash high level architecture

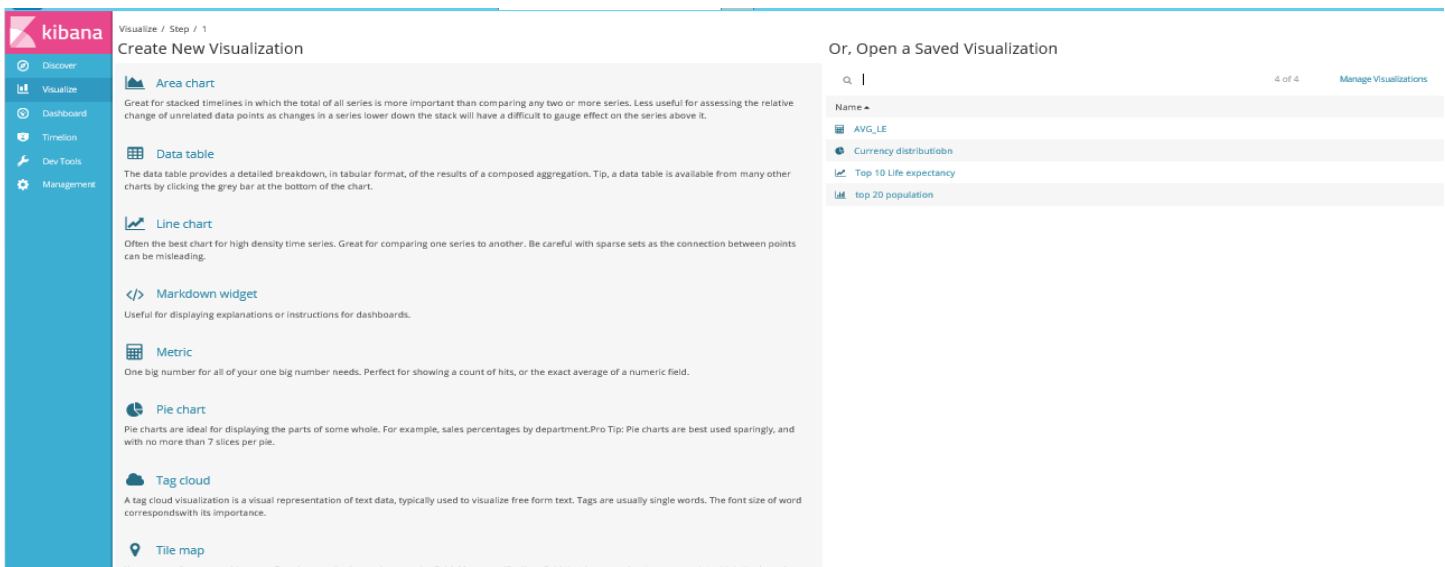
## Kibana

Kibana is the other product of ELK stack which is used visualize Elasticsearch data. Default, free version of Kibana installation has the following tabs.



**Figure 13: Kibana default installation tabs**

Different visualize components can be created from “Visualize” tab. Those can be stored and used to develop a dashboard.



**Figure 14: Visualize tab in Kibana**

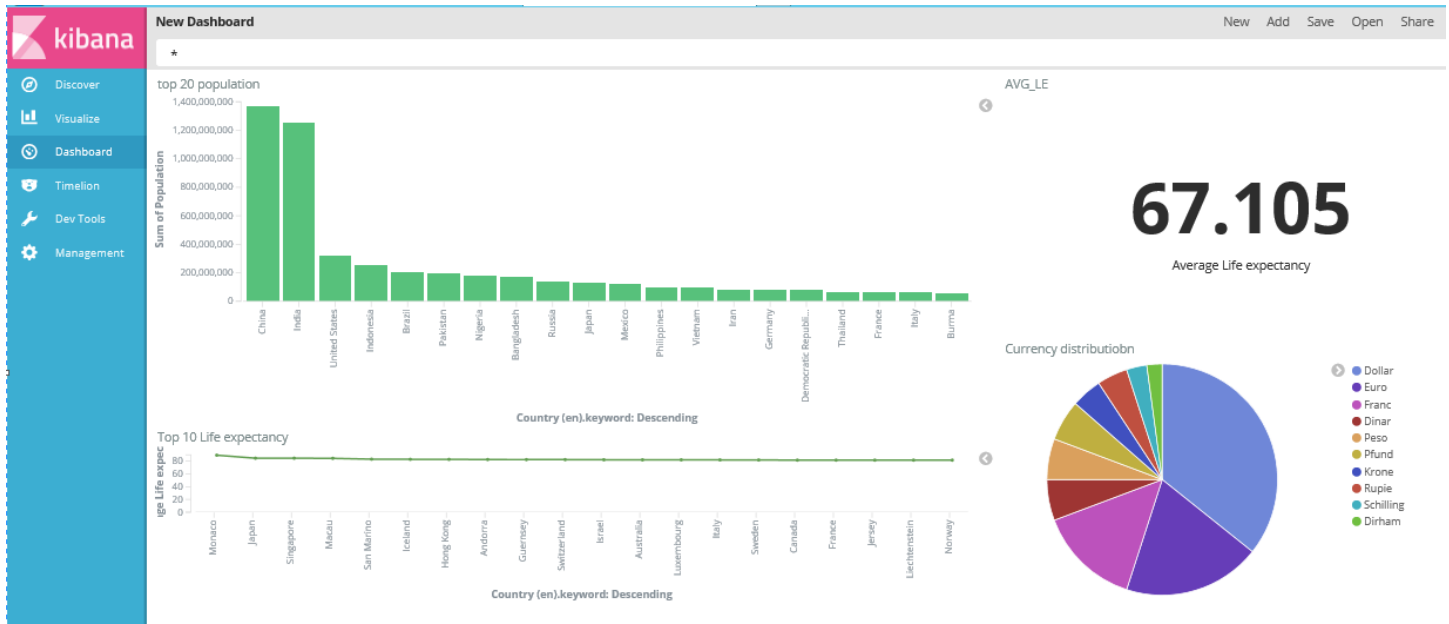


Figure 15: A sample Kibana dashboard

### Use case of ELK stack

While Elasticsearch is a flexible, distributed search engine which can be used for any search based application, ELK stack is mainly used for log parsing and to get actionable insights from those logs. Logs could be various applications logs from different sources or business data capable of providing insights to take business decision. This stack of 3 applications together provides great flexibility to collect logs from different sources including File system, system logs, Web server logs, SQL logs, system metrics and various other sources and transform those logs to meaningful structures, index them and great visualization of insights through real-time dashboard.

In today’s enterprises, analyzing business data and getting insights from them with great visualization is one of top most requirements. ELK stack can be a primary choice because of easy-to-install and configure nature, free & open source characteristics.



## References

- <https://www.elastic.co/products/elasticsearch>
- <https://www.elastic.co/products/kibana>
- <https://www.elastic.co/products/logstash>
- <https://www.elastic.co/guide/en/logstash/current/input-plugins.html>
- <https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>
- <https://www.elastic.co/guide/en/logstash/current/output-plugins.html>
- <https://github.com/elastic/examples>
- <https://www.elastic.co/guide/en/elasticsearch/guide/current/index.html>
- <https://www.simple-talk.com/dotnet/net-development/how-to-build-a-search-page-with-elasticsearch-and-net/>
- <https://www.elastic.co/use-cases>