# ELATSIC/ ELK STACK

- **E – ElasticSearch**
- **L- Logstash**
- **K- Kibana**

**ElasticSearch:**

- It is open-source **Search & Analytics Engine** and can also serve as a **NOSQL Database** which will store data in the form of **Json** and uses **RESTFUL API** to store and retrieve data.
- It works based on Apache Lucene which is also known as "**Heart of Elasticsearch"**

**Logstash:**

It is used to **read, write, filter and modify data** from various sources and store it in Elasticsearch.

**Kibana:**

- It is a web-interface which is used to **Discover, Analyze, Monitor and visualize** the data from Elasticsearch.
- It also used to apply **Machine Learning** Algorithms on the data from Elasticsearch to get insights of **data Anomaly** and **future trends**.

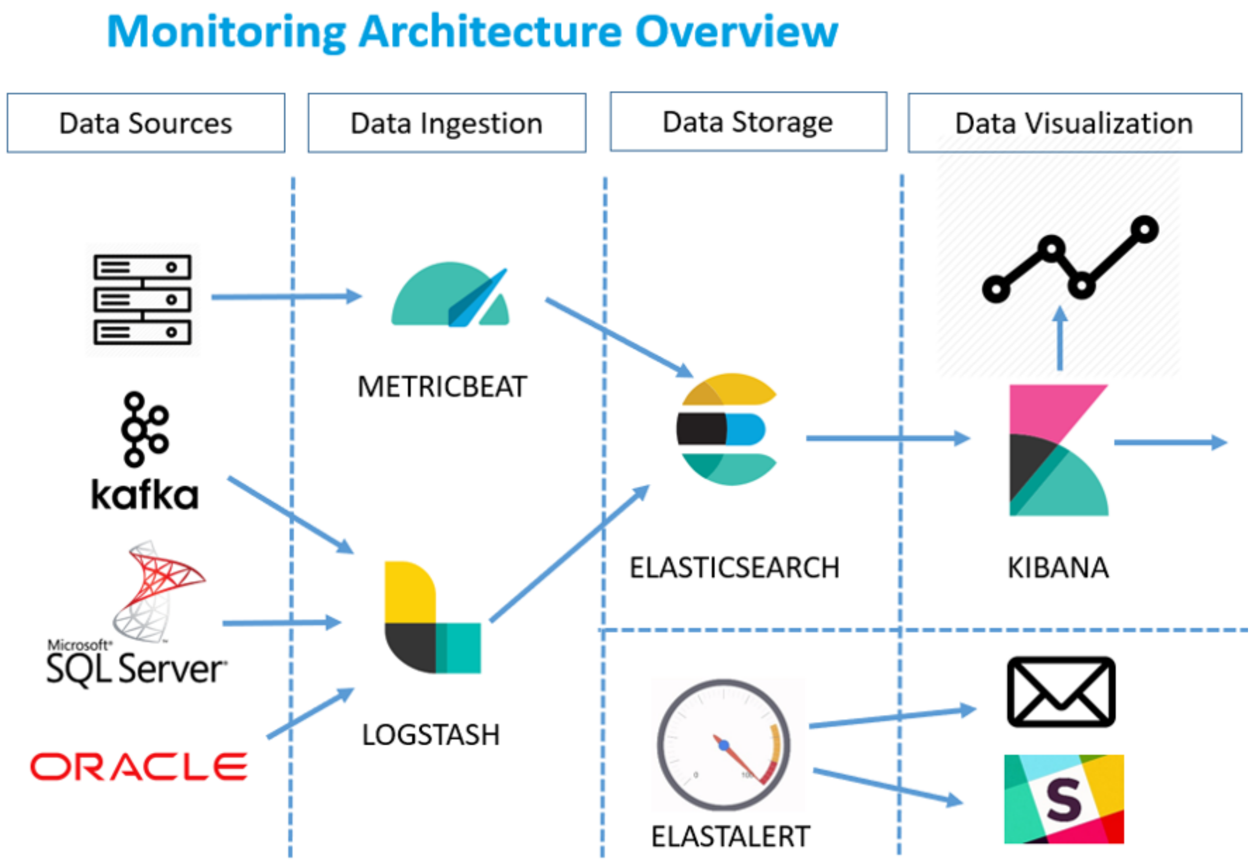
**Beats:**

They are Light-weight data shippers which is used to ship data from data source to ElasticSearch.

**Types:**

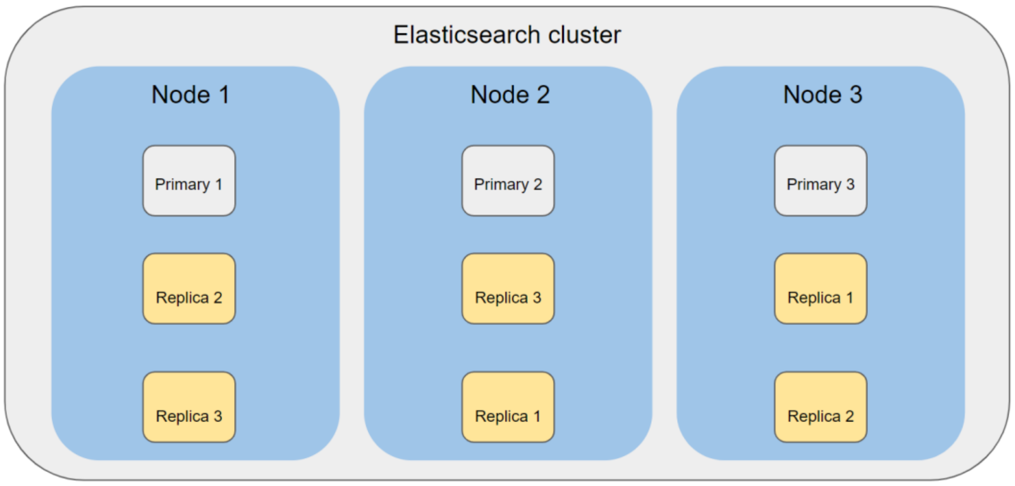
| | |
|---|---|
| File Beats | Log files |
| Metric Beats | Metrics(CPU,Memory) |
| Packet Beats | Network Data |
| Win log Beats | Windows Event Logs |
| Audit Beats | Audit (OS files) |
| Heart Beats | Uptime Monitor |
| Function Beats | Serverless Shipper |

**<u>Architecture Of Elastic Stack:</u>**



**<u>Terminology in Elastic Stack:</u>**

| Elastic terms | RDBMS terms | Refers to |
|---|---|---|
| Fields | Columns | Key-value Pair (JSON Objects) |
| Documents | Rows | Collection of Fields |
| Index | Table | Collection of Documents |
| Cluster | Database | Collection of Index |
| Shards | - | Horizontal partitioning of Index |
| Replica | - | Copy of Shards |



Primary Shards and its Replicas in Elasticsearch cluster

**Types of Nodes in Cluster:**

| | |
|---|---|
| Master Node | • Responsible for creation or Deletion of Index.<br>• Tracks the other nodes.<br>• Determines the location of shards. |
| Data Node | Responsible for performing CRUD, Search and Aggregation functions. |
| Ingest Node | Responsible for processing a document before indexing them. |
| Co-Ordinating Node | • Performs Routing<br>• Aids for Search Reduction Phase<br>• Responsible for Distributing the works via BULK Indexing. |

**Installation and Set Up Procedure:**

**Server:**

- Install Oracle VM.
- Install Ubuntu. (Server)
- Download and install:
  - wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
  - sudo apt-get install apt-transport-https
  - echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
  - sudo apt-get update && sudo apt-get install elasticsearch
  - sudo nano /etc/elasticsearch/elasticsearch.yml

    After this enable the following fields in the .yml file:

    ```
    node.name:  node-1
    ```

```
network.host: 0.0.0.0
discovery.seed.hosts: ["127.0.0.1"]
xpack.security.enabled: false
cluster.initial_master_nodes: ["node-1"]
```

- sudo chmod 755 -R /var/log/elasticsearch/

- Configure Elasticsearch to start automatically when the system boots up
    - sudo /bin/systemctl daemon-reload
    - sudo /bin/systemctl enable elasticsearch.service
- Elasticsearch can be started as follows
    - sudo /bin/systemctl start elasticsearch.service
    - sudo /bin/systemctl status elasticsearch.service
- Install curl
    - sudo apt-get install curl
- Go to Settings ->Networks ->Port Forwarding
  and add the following Network Configuration:
    - **Elasticsearch-127.0.0.1-9200**
    - **Kibana 127.0.0.1 5601**
    - **SSL 127.0.0.1 22**
- Open Terminal and Type the following Commands:
    - Sudo apt-get install openssh-server
    - Sudo systemctl enable ssh
    - Sudo adduser username
    - Sudo usermod -aG sudo username

**Client:**

- Download Putty. (Client)
- Configure the ElasticSearch in 127.0.0.1 in port 9200 and load it and open it.
- Login as username and enter the password.
- Then load the dataset into Elasticsearch using the Command:
    - Dataset: http://media.sundog-soft.com/es8/movies.json
- Create the index and post the data using the command:
    - curl -X PUT "localhost:9200/movies?pretty"
    - curl -XPOST "localhost:9200/movies/_bulk?pretty" --data-binary @movies.json

**CRUD OPERATIONS IN INDEX:**

- Command to know the mappings of the index movies:
    - curl -XGET "127.0.0.1:9200/movies/_mappings?pretty" (Note: To reduce the mapping area of any index we can define the datatype as "Flattened")
- Command to add document to the index movies:
    - curl -XPUT 127.0.0.1:9200/movies/_doc/
- Command to delete document from the index movies:
    - curl -XDELETE 127.0.0.1:9200/movies/_doc/1234567 //1234567-id
- Command to Update a document to the index movies:
    - Curl -XPUT 127.0.0.1:9200/movies/_doc/1234567

- Command to do OPTIMISTIC CONCURRENCY CONTROL to update a document is:
  - ➢ Curl -XPUT 127.0.0.1:9200/movies/_doc/1234567?if seq_no=7&if_primary_term=1

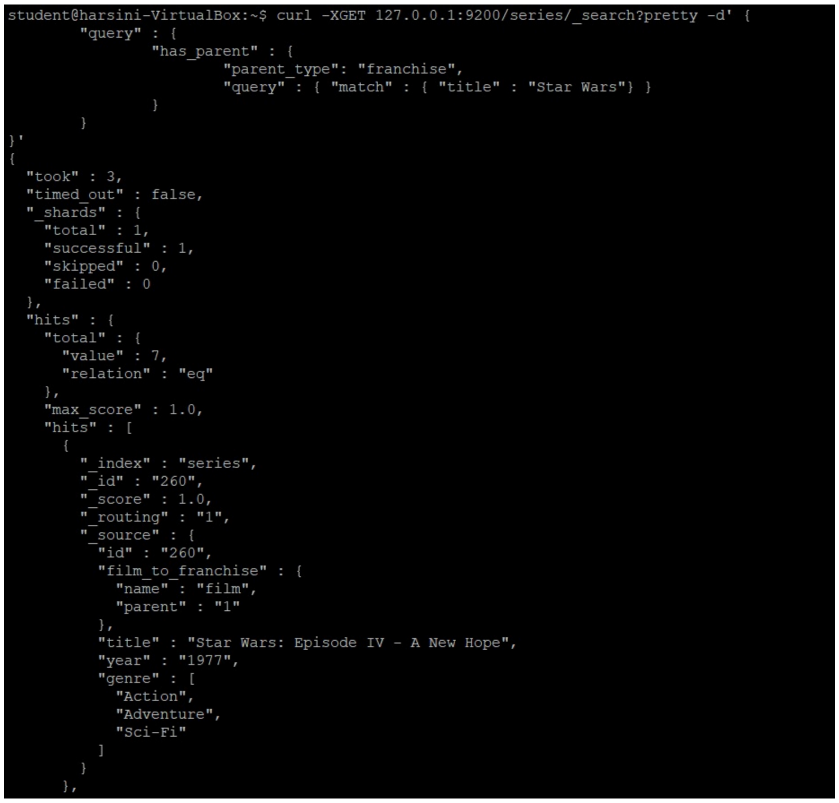
**REALTIONSHIP BETWEEN DOCUMENTS IN A INDEX:**

**Command to establish a parent child relationship between franchise and film:**

```
        Curl -XPUT 127.0.0.1:9200/Series -d ‘ {
        “mappings” : {
                “properties” : {
                        “film_to_franchise” : {
                                “type” : “join” ,
                                “relations”: { “franchise” , “flim”}        //franchise -parent
                                }                                           //flim-child
                        }
        }
}’
```

**Command to find the child who has “franchise” as parent:**

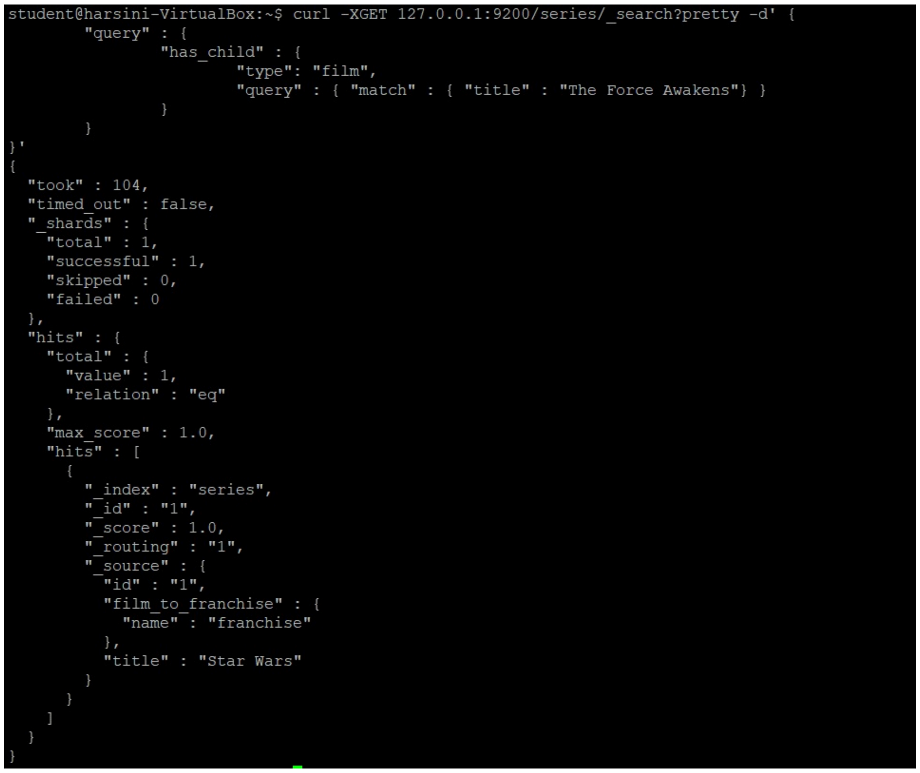
```
        Curl -XGET 127.0.0.1:9200/series/_search?pretty -d’ {
        “query” : {
                “has_parent” : {
                        “parent_type”: “franchise”,
                        “query” : { “match” : { “title” : “Star Wars”} }
                }
        }
}’
```



```
student@harsini-VirtualBox:~$ curl -XGET 127.0.0.1:9200/series/_search?pretty -d' {
        "query" : {
                "has_parent" : {
                        "parent_type": "franchise",
                        "query" : { "match" : { "title" : "Star Wars"} }
                }
        }
}'
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 7,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "series",
        "_id" : "260",
        "_score" : 1.0,
        "_routing" : "1",
        "_source" : {
          "id" : "260",
          "film_to_franchise" : {
            "name" : "film",
            "parent" : "1"
          },
          "title" : "Star Wars: Episode IV - A New Hope",
          "year" : "1977",
          "genre" : [
            "Action",
            "Adventure",
            "Sci-Fi"
          ]
        }
      },
```

**<u>Command to find the parent who has “The Force Awakens” as child:</u>**

```
Curl -XGET 127.0.0.1:9200/series/_search?pretty -d’ {
        “query” : {
                “has_child” : {
                        “type”: “film”,
                        “query” : { “match” : { “title” : “The Force Awakens”} }
                }
        }
}’
```



```
student@harsini-VirtualBox:~$ curl -XGET 127.0.0.1:9200/series/_search?pretty -d' {
        "query" : {
                "has_child" : {
                        "type": "film",
                        "query" : { "match" : { "title" : "The Force Awakens"} }
                }
        }
}'
{
  "took" : 104,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "series",
        "_id" : "1",
        "_score" : 1.0,
        "_routing" : "1",
        "_source" : {
          "id" : "1",
          "film_to_franchise" : {
            "name" : "franchise"
          },
          "title" : "Star Wars"
        }
      }
    ]
  }
}
```

**SEARCH IN ELASTICSEARCH:**

**Query Line Search**

The query is given directly as a parameter

Example: Query to get the details of the movie which is released after the year 2010 and has the word "trek" in the title.

- Curl -XGET 'http://localhost:9200/movies/_search?q=+year:>2010+title:trek'

**Request Body Search**

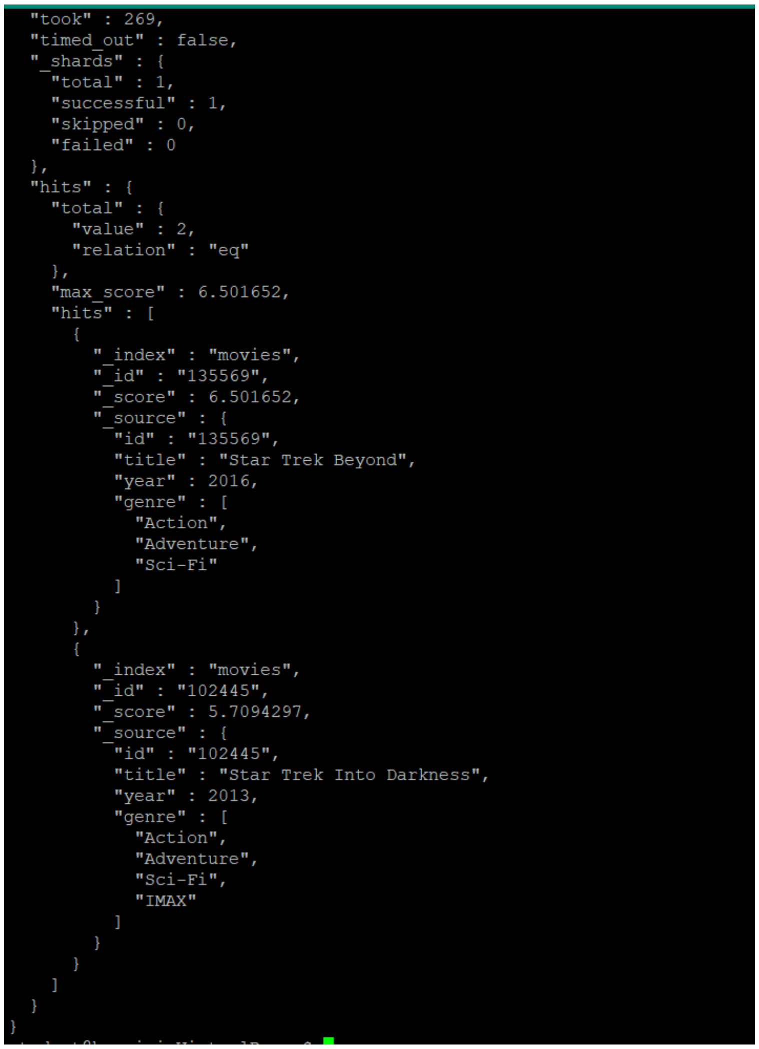
The query is given as a request body

Example: Query to get the details of the movie which is released after the year 2010 and has the word "trek" in the title.

- Curl -XGET 'http://localhost:9200/movies/_search?pretty' -d '{

```
        "query" : {
                "bool" : {
                "must" : { "term" : { "title" : "trek"}},
                "filter" : { "range" : { "year" : { "gte" : 2010}}}
                }
        }
```

}'



```
  "took" : 269,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : 6.501652,
    "hits" : [
      {
        "_index" : "movies",
        "_id" : "135569",
        "_score" : 6.501652,
        "_source" : {
          "id" : "135569",
          "title" : "Star Trek Beyond",
          "year" : 2016,
          "genre" : [
            "Action",
            "Adventure",
            "Sci-Fi"
          ]
        }
      },
      {
        "_index" : "movies",
        "_id" : "102445",
        "_score" : 5.7094297,
        "_source" : {
          "id" : "102445",
          "title" : "Star Trek Into Darkness",
          "year" : 2013,
          "genre" : [
            "Action",
            "Adventure",
            "Sci-Fi",
            "IMAX"
          ]
        }
      }
    ]
  }
}
```

**<u>Difference between match and match_phrase</u>**

 **Term Matching:**

- **match:** Breaks down the input text into individual terms and matches any of them.
- **match_phrase:** Searches for the exact sequence of terms as a phrase.

**Order and Proximity:**

- **match:** Ignores the order of terms and proximity.
- **match_phrase:** Considers the order and ensures the terms appear close to each other as specified.

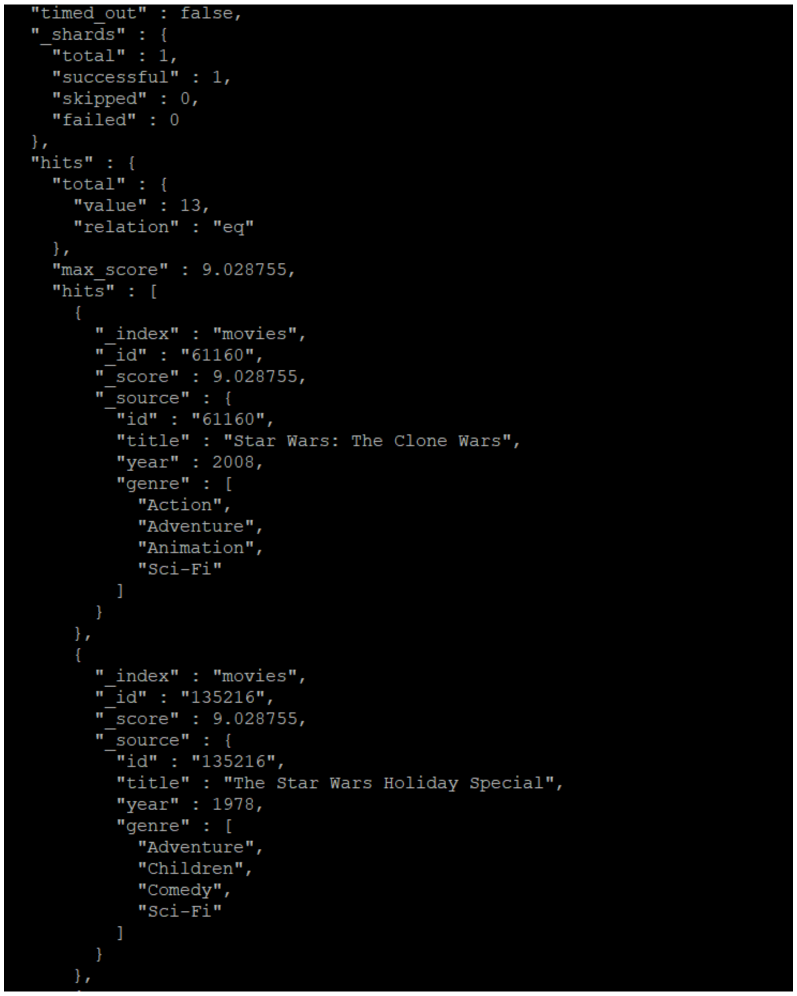
**<u>Match_Phrase Search</u>**

Search for exach phrases given in the query

➢ curl -XGET 'http://localhost:9200/movies/_search?pretty' -d '{

"query": {

```
"match_phrase": {
        "title": {
                "query": "Star Wars",
                "slop": 2
                }
        }
}}'
```



```
"timed_out" : false,
"_shards" : {
  "total" : 1,
  "successful" : 1,
  "skipped" : 0,
  "failed" : 0
},
"hits" : {
  "total" : {
    "value" : 13,
    "relation" : "eq"
  },
  "max_score" : 9.028755,
  "hits" : [
    {
      "_index" : "movies",
      "_id" : "61160",
      "_score" : 9.028755,
      "_source" : {
        "id" : "61160",
        "title" : "Star Wars: The Clone Wars",
        "year" : 2008,
        "genre" : [
          "Action",
          "Adventure",
          "Animation",
          "Sci-Fi"
        ]
      }
    },
    {
      "_index" : "movies",
      "_id" : "135216",
      "_score" : 9.028755,
      "_source" : {
        "id" : "135216",
        "title" : "The Star Wars Holiday Special",
        "year" : 1978,
        "genre" : [
          "Adventure",
          "Children",
          "Comedy",
          "Sci-Fi"
        ]
      }
    },
```

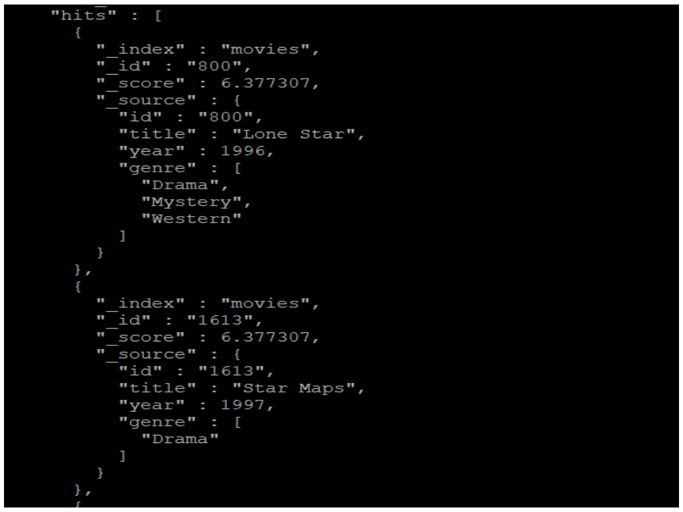
## <u>Pagination</u>

While searching for the query we can do pagination by defining from and size keywords.

- From – specifies the starting point
- Size – specifies the number of results to be retrieved

➢ curl -XGET 'http://localhost:9200/movies/_search?pretty' -d '{

```
"query": {
    "match": {
        "title": "Star"
    }
},
"from": 0,
"size": 10
}'
```



```
  "hits" : [
    {
      "_index" : "movies",
      "_id" : "800",
      "_score" : 6.377307,
      "_source" : {
        "id" : "800",
        "title" : "Lone Star",
        "year" : 1996,
        "genre" : [
          "Drama",
          "Mystery",
          "Western"
        ]
      }
    },
    {
      "_index" : "movies",
      "_id" : "1613",
      "_score" : 6.377307,
      "_source" : {
        "id" : "1613",
        "title" : "Star Maps",
        "year" : 1997,
        "genre" : [
          "Drama"
        ]
      }
    },
```

**Sorting**

It is used to sort the result which is fetched using search query

➢ Curl -XGET 127.0.0.1:9200/movies/_search?sort=year&pretty’

In order to sort based on text value we need to define them as keyword in the raw data format

➢ Curl -XPUT 127.0.0.1:9200/movies/ -d ‘ {

```
“mappings” : {
    “properties” : {
        “title” : {
            “type” : “text” ,
            “fields” : { “raw” : { “type” : “keyword” } }
        }
    }
```

```
            }
        }’
```

**Difference between Text and Keyword fields**

**Text Fields**

- **Purpose:** Text fields are used for full-text search. They are analyzed, meaning the text is processed and broken down into individual terms (tokens) using an analyzer.

**Keyword Fields**

- **Purpose:** Keyword fields are used for exact matching, sorting, and aggregations. They are not analyzed, meaning the text is indexed as a single token.

**Why Have a Keyword Field for a Text Field?**

1. **Exact Matching:** When you need to perform exact match queries on a field, such as finding all documents where the title is exactly "Star Wars." Analyzing the text would break it down into individual terms, making it unsuitable for exact matches.
2. **Sorting:** Sorting requires the exact values of the field. Analyzed text fields cannot be sorted properly because they are broken down into multiple terms.
3. **Aggregations:** Aggregations, like counting unique values, require the exact terms. Text fields, which are analyzed, cannot be used for accurate aggregations.

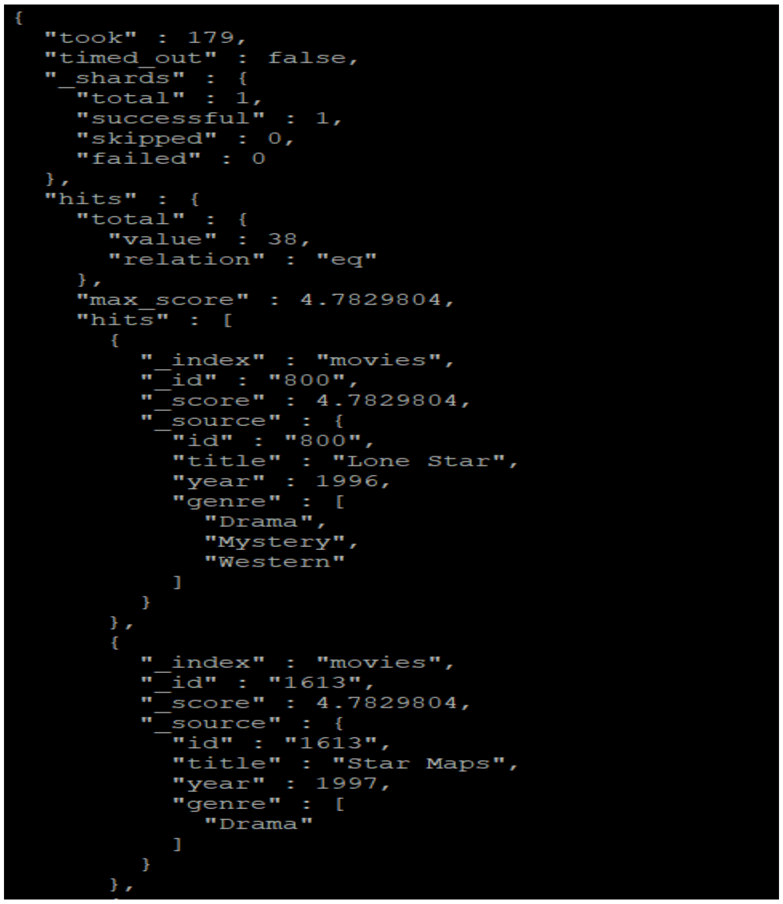
**Fuzzy Queries**

Fuzzy queries are designed to handle search terms that may contain misspellings or typographical errors. They can identify similar terms within a certain edit distance, allowing for more flexible searches. Fuzzy queries support:

- Substitution   Eg: Apple -> Appla
- Insertion        Eg: Apple -> Applea
- Deletion        Eg: Apple->Aple

- curl -XGET 'http://localhost:9200/movies/_search?pretty' -d '{

```
            "query": {
                "fuzzy": {
                    "title": {“value”:"Ster, “fuzziness”:1}
                }
            }
        }’
```


```
{
  "took" : 179,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 38,
      "relation" : "eq"
    },
    "max_score" : 4.7829804,
    "hits" : [
      {
        "_index" : "movies",
        "_id" : "800",
        "_score" : 4.7829804,
        "_source" : {
          "id" : "800",
          "title" : "Lone Star",
          "year" : 1996,
          "genre" : [
            "Drama",
            "Mystery",
            "Western"
          ]
        }
      },
      {
        "_index" : "movies",
        "_id" : "1613",
        "_score" : 4.7829804,
        "_source" : {
          "id" : "1613",
          "title" : "Star Maps",
          "year" : 1997,
          "genre" : [
            "Drama"
          ]
        }
      },
```

**Partial Matching**

It is used to perform partial matching for search

**Prefix query**

```
➢ curl -XGET 'http://localhost:9200/movies/_search?pretty' -d '{
          "query": {
                    "prefix": {
                              “year”:"201”
                    }
          }
  }’
```

**Wildcard query**

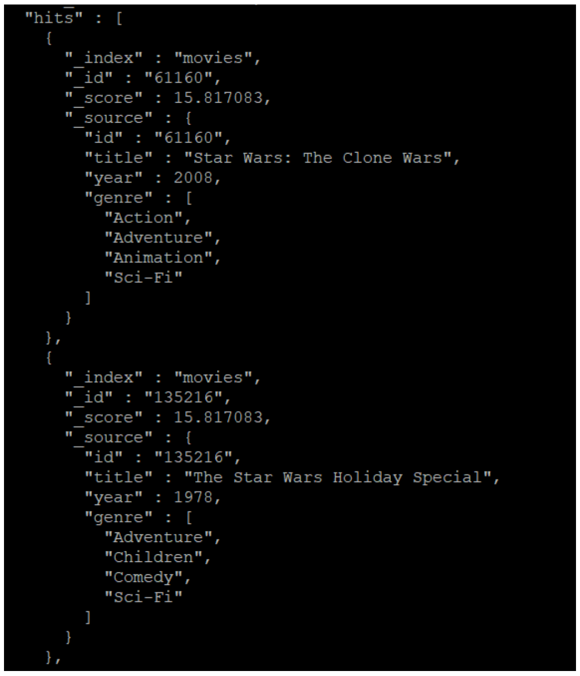
```
➢ curl -XGET 'http://localhost:9200/movies/_search?pretty' -d '{
          "query": {
                    "wildcard": {
                              “year”:"19*”
                    }
            }
    }’
```

**<u>Search-as-you-type</u>**

As the name suggests, it will perform search as you type

- curl -XGET 'http://localhost:9200/movies/_search?pretty'  -d '{

```
     "query": {
            "match_phrase_prefix": {
                    "title": {
                            "query": "Star Wars",
                            "slop": 10
                            }
                    }
            }
    }'
```



```
"hits" : [
  {
    "_index" : "movies",
    "_id" : "61160",
    "_score" : 15.817083,
    "_source" : {
      "id" : "61160",
      "title" : "Star Wars: The Clone Wars",
      "year" : 2008,
      "genre" : [
        "Action",
        "Adventure",
        "Animation",
        "Sci-Fi"
      ]
    }
  },
  {
    "_index" : "movies",
    "_id" : "135216",
    "_score" : 15.817083,
    "_source" : {
      "id" : "135216",
      "title" : "The Star Wars Holiday Special",
      "year" : 1978,
      "genre" : [
        "Adventure",
        "Children",
        "Comedy",
        "Sci-Fi"
      ]
    }
  },
```

**EXCEPTION HANDLING FOR SEARCH:**

**To ignore the exception which is throw by datatype:**

Example: When we try to perform search operation using the keyword datatype:

```
Curl –location –XPUT 127.0.0.1:9200/microservice-logs/_settings \
--data-raw ‘ {
        “index.mapping.ignore_malformed” : true
}
```

**To ignore the exception which is throw because we are exceeding the default limit(1000):**

```
Curl –location –XPUT 127.0.0.1:9200/big_Objects/_settings \
--data-raw ‘ {
        “index.mapping.total.fields.limit” : 1005
}
```

**Importing Data from different sources to Elasticsearch**

- Java- Elastic.co
- Python – Elasticsearch.package
- Ruby-Elasticsearch.ruby
- Perl-Elasticsearch.pm

**Importing Data using Python:**

Create and Run IndexRatings.py file

**IndexRatings.py file**:

```
import csv
from collections import deque
import elasticsearch
from elasticsearch import helpers

def readMovies():
    csvfile = open('ml-latest-small/movies.csv', 'r', encoding="utf8")

    reader = csv.DictReader( csvfile )

    titleLookup = {}

    for movie in reader:
            titleLookup[movie['movieId']] = movie['title']

    return titleLookup

def readRatings():
    csvfile = open('ml-latest-small/ratings.csv', 'r', encoding="utf8")
    titleLookup = readMovies()

    reader = csv.DictReader( csvfile )
    for line in reader:
        rating = {}
        rating['user_id'] = int(line['userId'])
        rating['movie_id'] = int(line['movieId'])
        rating['title'] = titleLookup[line['movieId']]
        rating['rating'] = float(line['rating'])
        rating['timestamp'] = int(line['timestamp'])
        yield rating


es = elasticsearch.Elasticsearch(["http://127.0.0.1:9200"])

#es.indices.delete(index="ratings",ignore=404)
deque(helpers.parallel_bulk(es,readRatings(),index="ratings", request_timeout=300), maxlen=0)
es.indices.refresh()
```

**Run the command**

```
curl -XGET 127.0.0.1:9200/ratings/_search?pretty
```

**Importing Data from MySql:**

**Install Mysql Connector:**

- sudo apt-get install mysql-server
- wget http://files.grouplens.org/datasets/movielens/m1-100k.zip
- unzip ml-100k.zip
- sudo mysql --local-infile=1 -u root -p
- CREATE DATABASE movielens;CREATE TABLE movielens.movies (

  movieID IN PRIMARY KEY NOT NULL,

  title TEXT,releaseDate DATE

);

**Update the mysql.conf file**

```
sudo cat /etc/logstash/conf.d/mysql.conf
```

**Mysql.conf**

```
input{
  jdbc{
   jdbc_connection_string => "jdbc:mysql://localhost:3306/movielens"
  jdbc_user => "student"
  jdbc_password => " *****"
  jdbc_driver_library => "home/student/usr/share/logstash/mysql-connector-java-8.0.16/mysql-connector-java-8.0.16.jar"
  jdbc_driver_class => "com.mysql.jdbc.Driver"
  statement => "SELECT * from movies"
}
}
output{
 stdout { codec => json_lines }
elasticsearch{
 hosts => ["localhost:9200"]
 index => "movielens-sql"
}
}
```

**Run the commands:**

```
sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/mysql.conf
 curl -XGET 127.0.0.1:9200/movielens-sql/_search?pretty
```

**Importing Data from .Csv:**

```
sudo cat /etc/logstash/conf.d/csv-read-drop.conf
```

**csv-read-drop.conf:**

```
input {
  file {
    path => "/home/student/csv-data/csv-schema-short-numerical.csv"
    start_position => "beginning”
  }
}
filter {
  csv {
      separator => ","
      skip_header => "true"
      columns =>
["id","timestamp","paymentType","name","gender","ip_address","purpose","country","age"]
  }
}
output {
   elasticsearch {
     hosts => "http://localhost:9200"
     index => "demo-csv"
  }

stdout {}
```

**Run the Commands:**

```
sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/demo-csv.conf
 curl -XGET 127.0.0.1:9200/demo-csv/_search?pretty
```

**Importing Data from .Json/.Log:**

- cd /etc/logstash/conf.d/
- sudo vi json-read.conf

**json-read.conf: (using filter)**

```
input {
        file {
                start_position => "beginning"
                path=> "/home/student/json-data/sample-json.log"
        }
}
filter {
        json {
                source => "message"
        }
}
```

```
output {
        elasticsearch {
                hosts => "http://localhost:9200"
                index=>"demo-json"
        }
        stdout {}
}
```

**Run the commands:**

```
sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/demo-json.conf
 curl -XGET 127.0.0.1/demo-json/_search?pretty
```

**<u>demo-json-drop.conf: (using mutate and removing unwanted fields)</u>**

```
input {
        file {
                start_position => "beginning"
                path=> "/home/student/json-data/sample-json.log"
        }
}
filter {
        json {
                source => "message"
        }
        if [paymentType]  == "Mastercard" {
                drop{}
        }
        mutate{
                remove_field =>["message","@timestamp","path","host","@version"]
        }
}
output {
        elasticsearch {
                hosts => "http://localhost:9200"
                index=>"demo-json-drop"
        }
        stdout {}
}
```

**Run the Commands**:

```
sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/json-drop.conf
curl -XGET 'http://localhost:9200/demo-json-drop/_search?pretty'
```

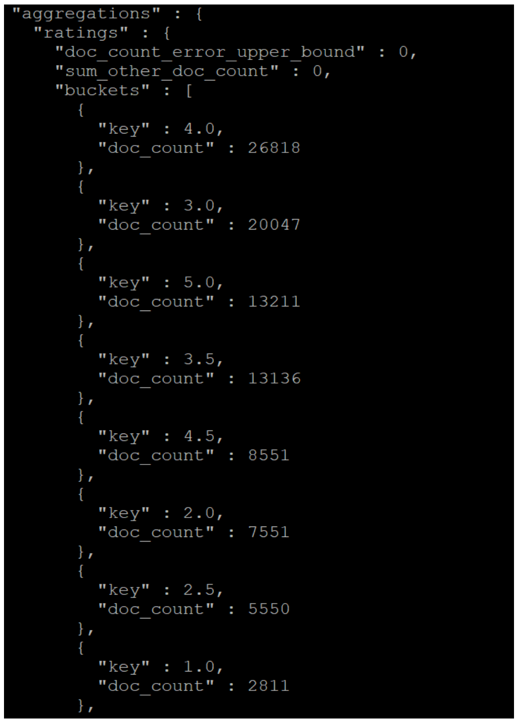
## AGGREGATION

- Metrics - Avg,Min,Max
- Buckets - Histogram,Piechart

**Metrics:**

1. **Aggregation on 'Ratings' index**

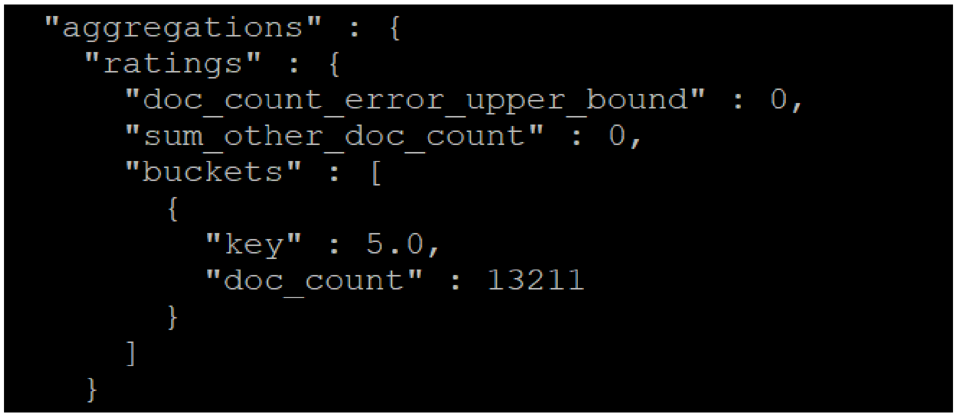
```
curl -XGET '127.0.0.1:9200/ratings/_search?pretty' -d '
{ "aggs" : {
        "ratings" : {
                "terms" : {
                        "field":"rating"
 }}}}'
```



```
"aggregations" : {
  "ratings" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : 4.0,
        "doc_count" : 26818
      },
      {
        "key" : 3.0,
        "doc_count" : 20047
      },
      {
        "key" : 5.0,
        "doc_count" : 13211
      },
      {
        "key" : 3.5,
        "doc_count" : 13136
      },
      {
        "key" : 4.5,
        "doc_count" : 8551
      },
      {
        "key" : 2.0,
        "doc_count" : 7551
      },
      {
        "key" : 2.5,
        "doc_count" : 5550
      },
      {
        "key" : 1.0,
        "doc_count" : 2811
      },
```

2. **Using match and aggregation**

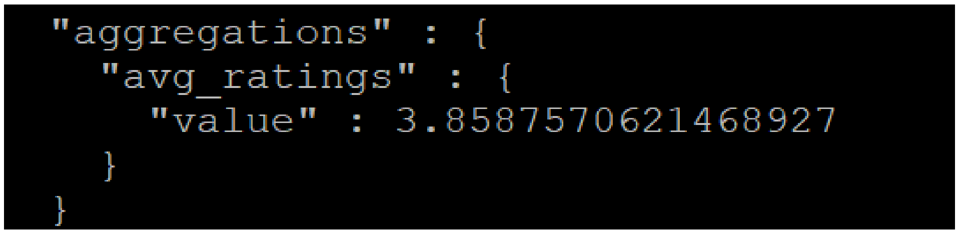
```
curl -XGET '127.0.0.1:9200/ratings/_search?pretty' -d '
{
        "query":
                {"match":{"rating":5.0}},
        "aggs" : {
                "ratings" : {
                        "terms" : {
                                "field":"rating"
}}}}'
```



```
  "aggregations" : {
    "ratings" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 0,
      "buckets" : [
        {
          "key" : 5.0,
          "doc_count" : 13211
        }
      ]
    }
```

3. **Using match_phrase and avg_aggregation**

```
curl -XGET '127.0.0.1:9200/ratings/_search?pretty' -d '
{
        "query":{
                "match_phrase":{"title":"Star Wars"}},
        "aggs" : {
                "avg_ratings" : {
                        "avg" : {
                                "field":"rating"
}}}}'
```

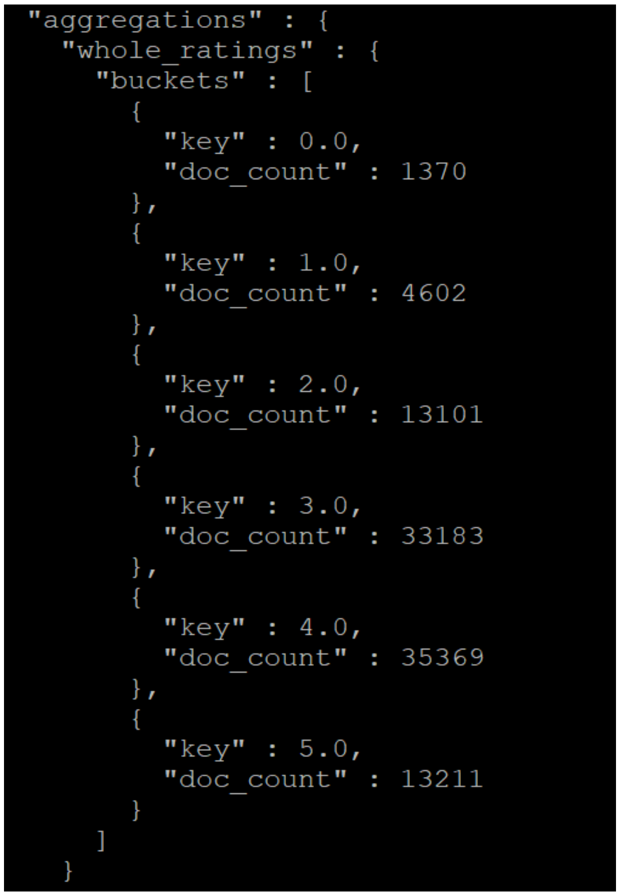


```
  "aggregations" : {
    "avg_ratings" : {
      "value" : 3.8587570621468927
    }
  }
```

**Buckets-Histogram:**

1. **Histogram on the field ratings at interval 1.0**

```
curl -XGET '127.0.0.1:9200/ratings/_search?pretty' -d '
{
        "aggs" : {
                "whole_ratings" : {
                        "histogram" : {
                                "field":"rating", "interval" : 1.0
}}}}'
```

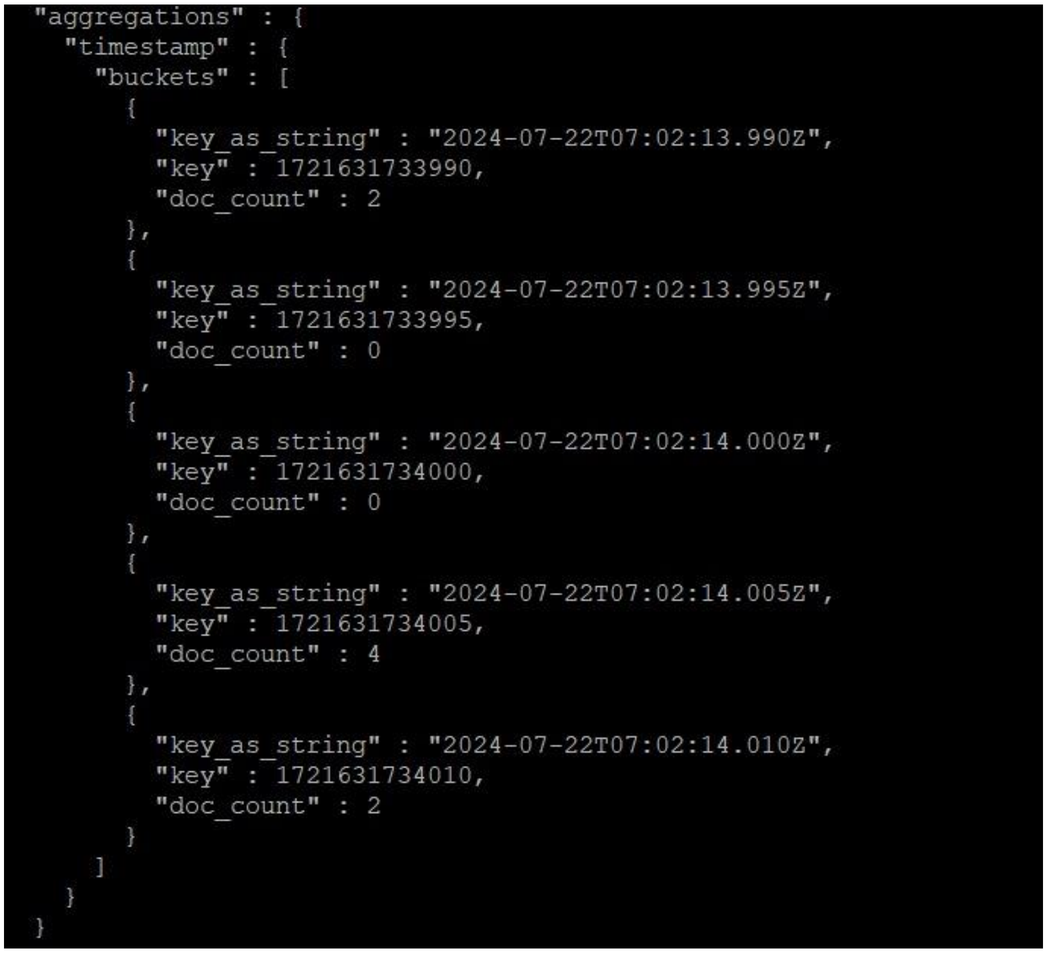


```
"aggregations" : {
  "whole_ratings" : {
    "buckets" : [
      {
        "key" : 0.0,
        "doc_count" : 1370
      },
      {
        "key" : 1.0,
        "doc_count" : 4602
      },
      {
        "key" : 2.0,
        "doc_count" : 13101
      },
      {
        "key" : 3.0,
        "doc_count" : 33183
      },
      {
        "key" : 4.0,
        "doc_count" : 35369
      },
      {
        "key" : 5.0,
        "doc_count" : 13211
      }
    ]
  }
```

**Buckets-Time Series**

1. Histogram for time series data(calendar interval or fixed interval)

```
curl -XGET '127.0.0.1:9200/demo-grok/_search?pretty' -d '

{

        "aggs" : {

                "timestamp" : {

                "date_histogram" : {
                    "field":"@timestamp", "fixed_interval" : "5ms"
        }}}}'
```



```
"aggregations" : {
  "timestamp" : {
    "buckets" : [
      {
        "key_as_string" : "2024-07-22T07:02:13.990Z",
        "key" : 1721631733990,
        "doc_count" : 2
      },
      {
        "key_as_string" : "2024-07-22T07:02:13.995Z",
        "key" : 1721631733995,
        "doc_count" : 0
      },
      {
        "key_as_string" : "2024-07-22T07:02:14.000Z",
        "key" : 1721631734000,
        "doc_count" : 0
      },
      {
        "key_as_string" : "2024-07-22T07:02:14.005Z",
        "key" : 1721631734005,
        "doc_count" : 4
      },
      {
        "key_as_string" : "2024-07-22T07:02:14.010Z",
        "key" : 1721631734010,
        "doc_count" : 2
      }
    ]
  }
}
```

**NESTED AGGREGATION:**

Average movie ratings that contain word 'Star' in its title

```
curl -XGET '127.0.0.1:9200/ratings/_search?pretty' -d '
{
        "query":
                {
                "match_phrase" : { "title" : "Star Wars" }},
                        "aggs" :
                        { "titles":
                                {
```

```
                    "terms" : { "field" : "title.raw" },
                    "aggs" :
                            { "avg_ratings" :
                                    { "avg" :
                                            { "field" : "rating" }
}}}}}'
```

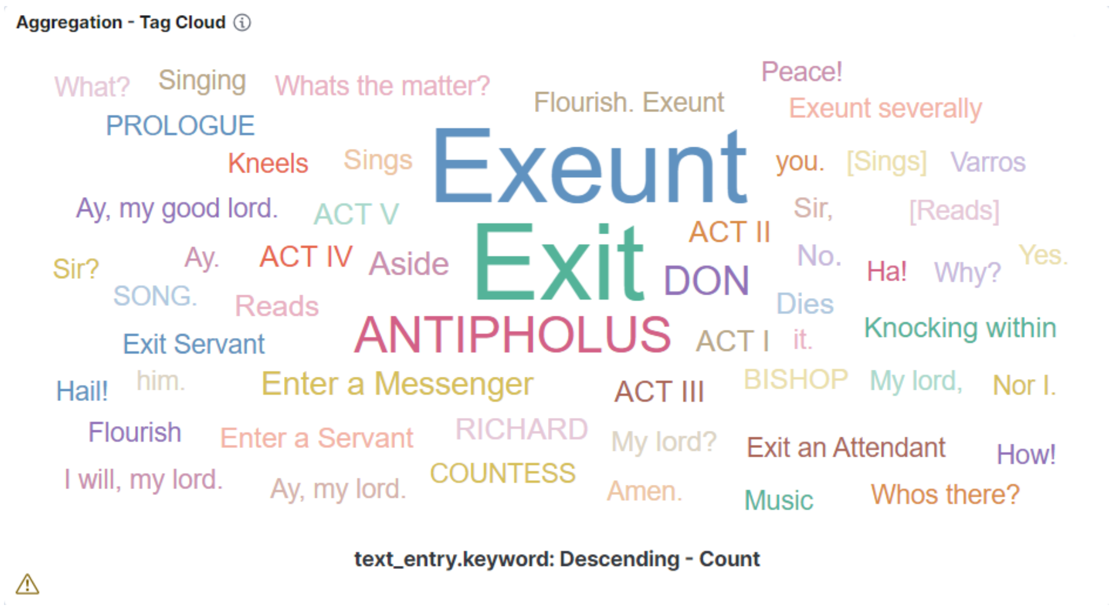
## KIBANA

**Install and enable Kibana:**

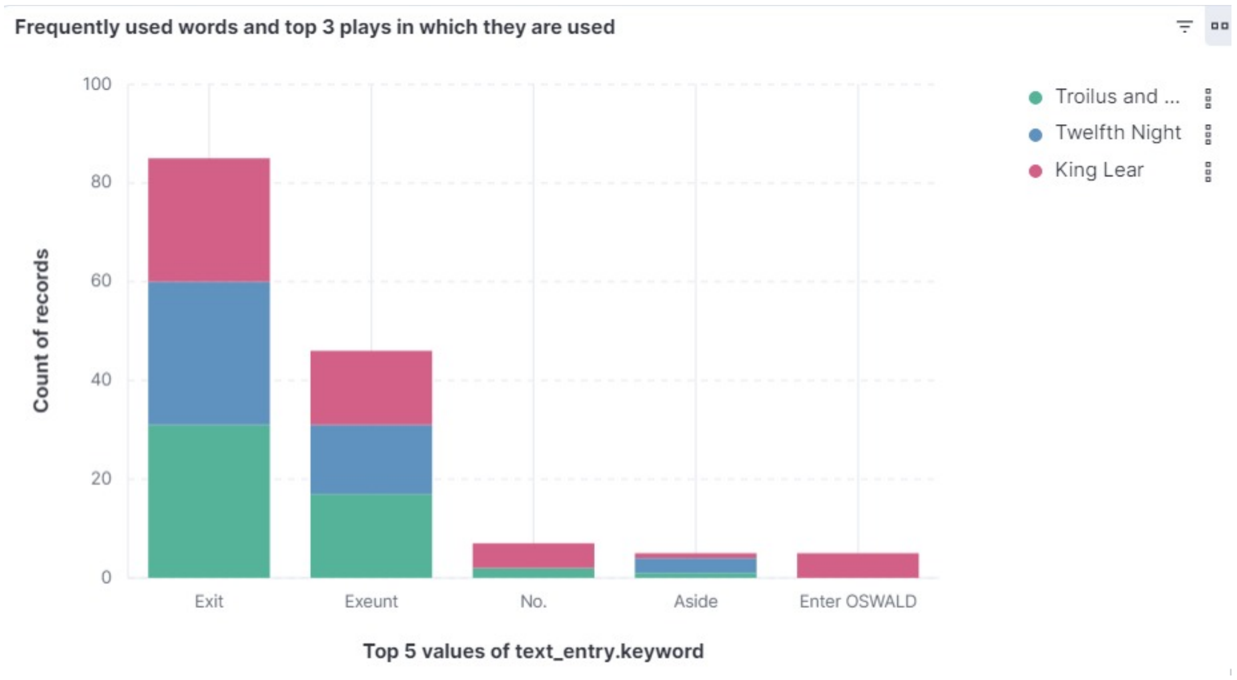
- sudo apt list kibana
- sudo apt-get install kibana=8.14.3
- sudo nano /etc/kibana/kibana.yml
- sudo /bin/systemctl enable kibana.service
- sudo /bin/systemctl start kibana.service
- Open kibana service from port localhost:5601

**Workouts and Dashboard:**

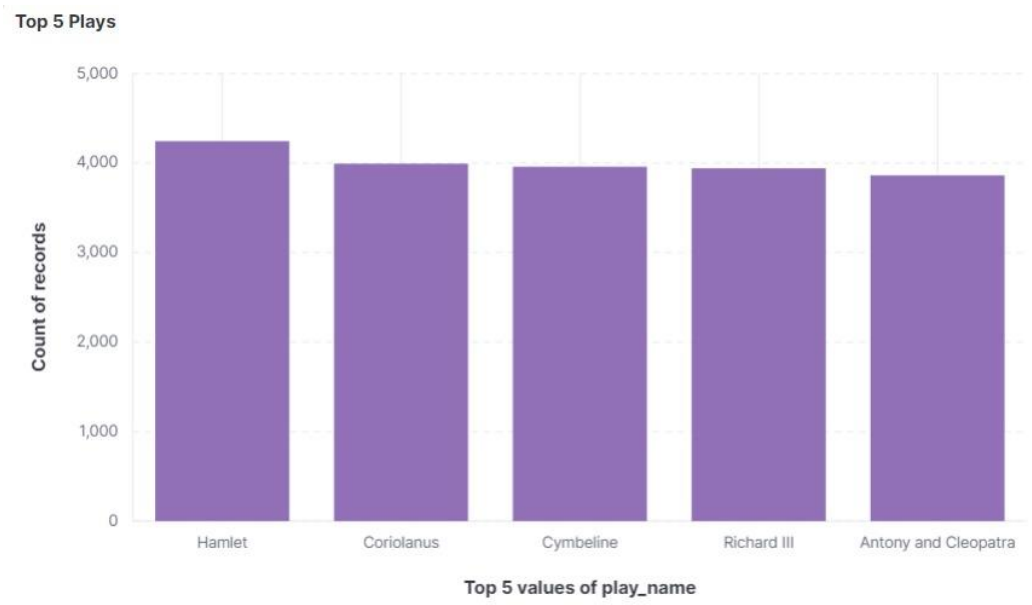
**Displaying the most frequently used words in the Shakespear's works using Tag Cloud in Aggregation**



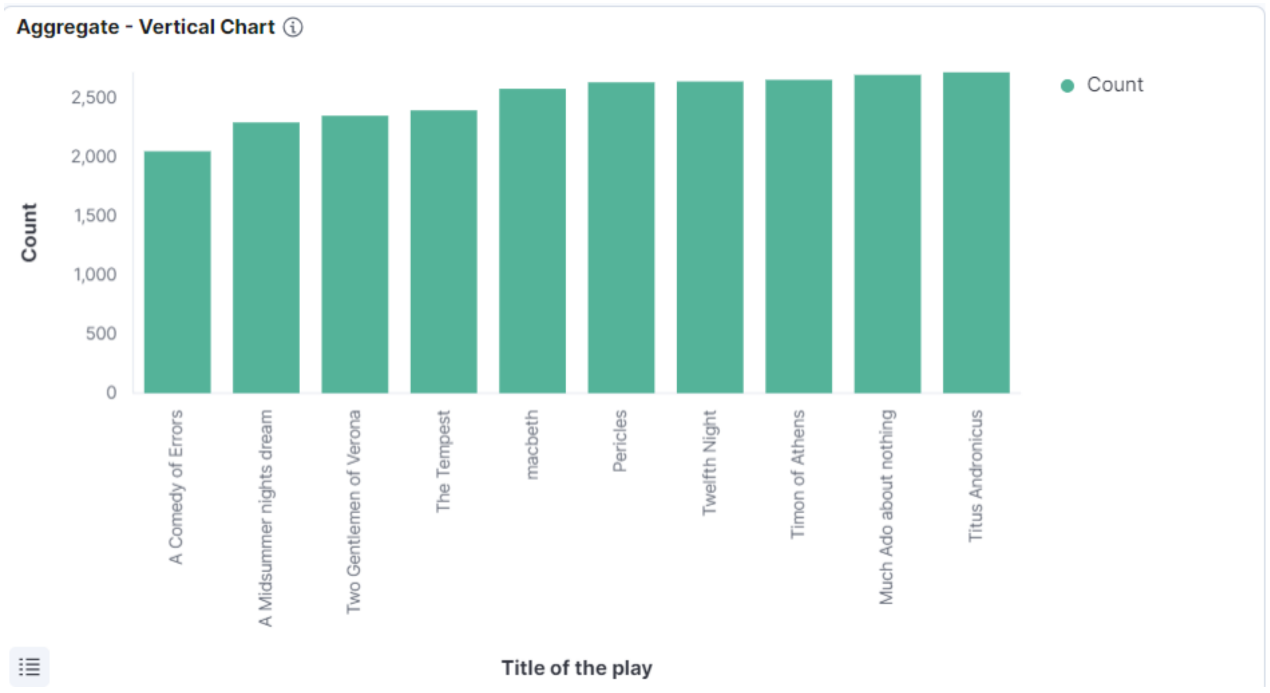
**Top 5 frequently used words and the plays in which they are used.**



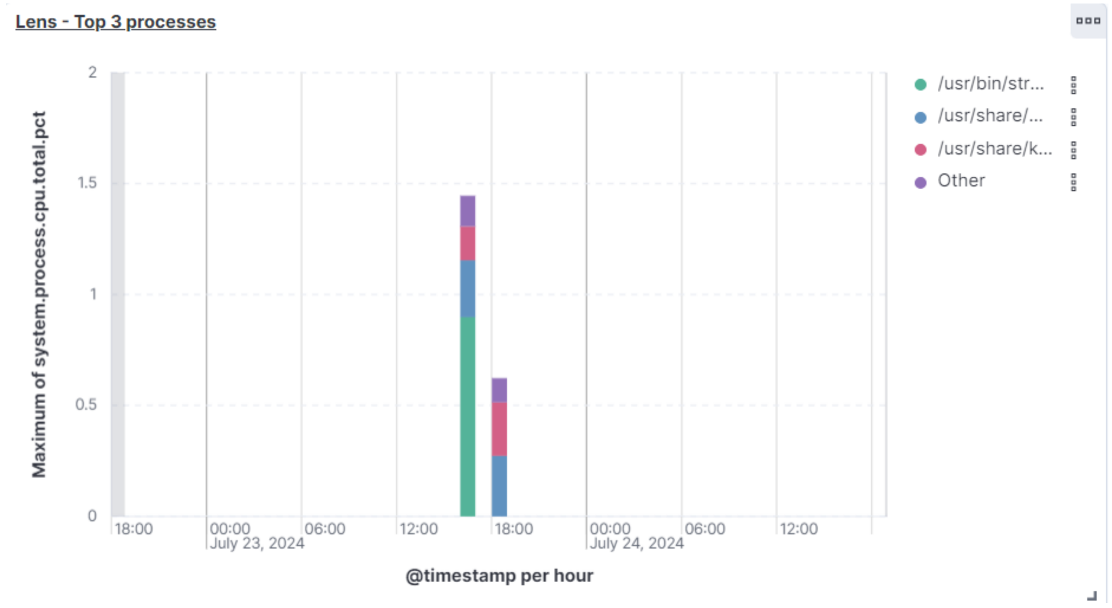
**Vertical Bar chart to display the top 5 plays (having highest documents).**



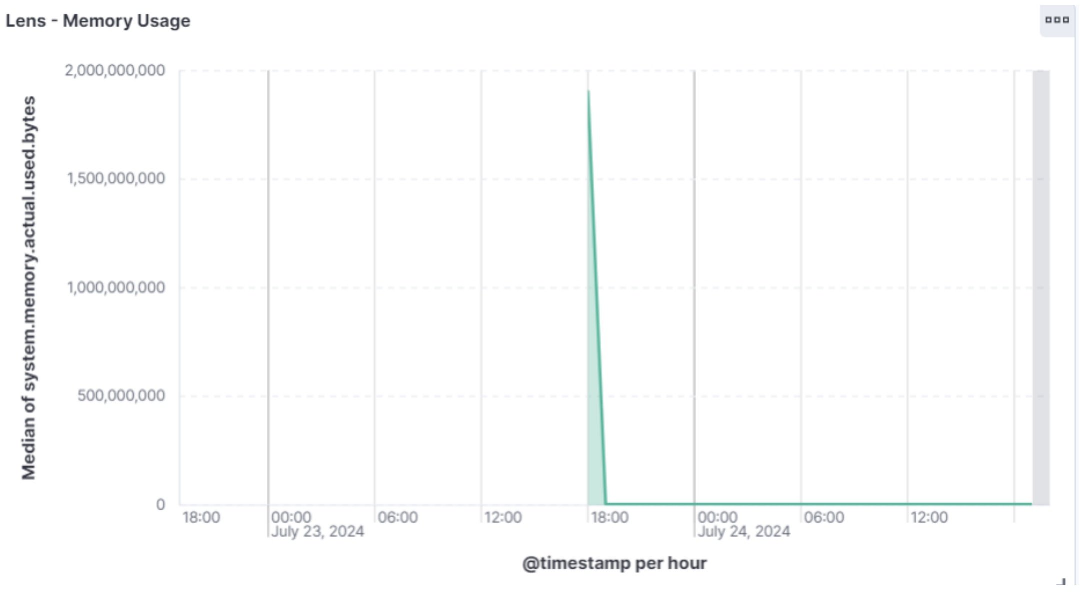
**Vertical Bar chart displaying number of documents in last 10 plays.**



**Top 3 processes**

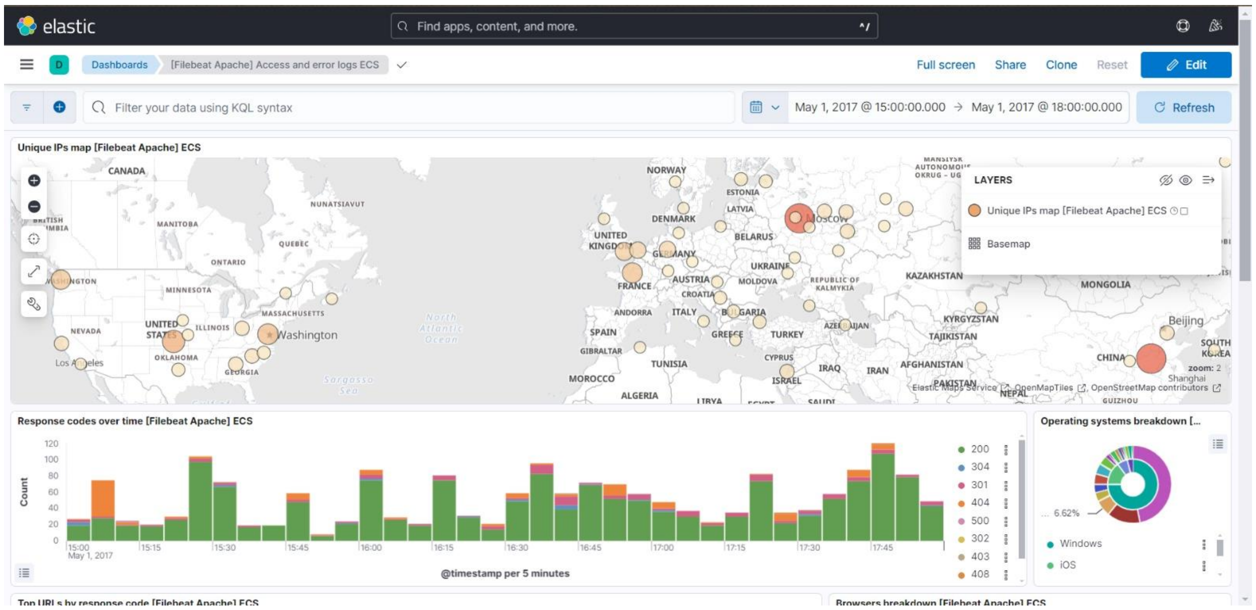


## Memory Usage



## LOG ANALYSIS IN KIBANA

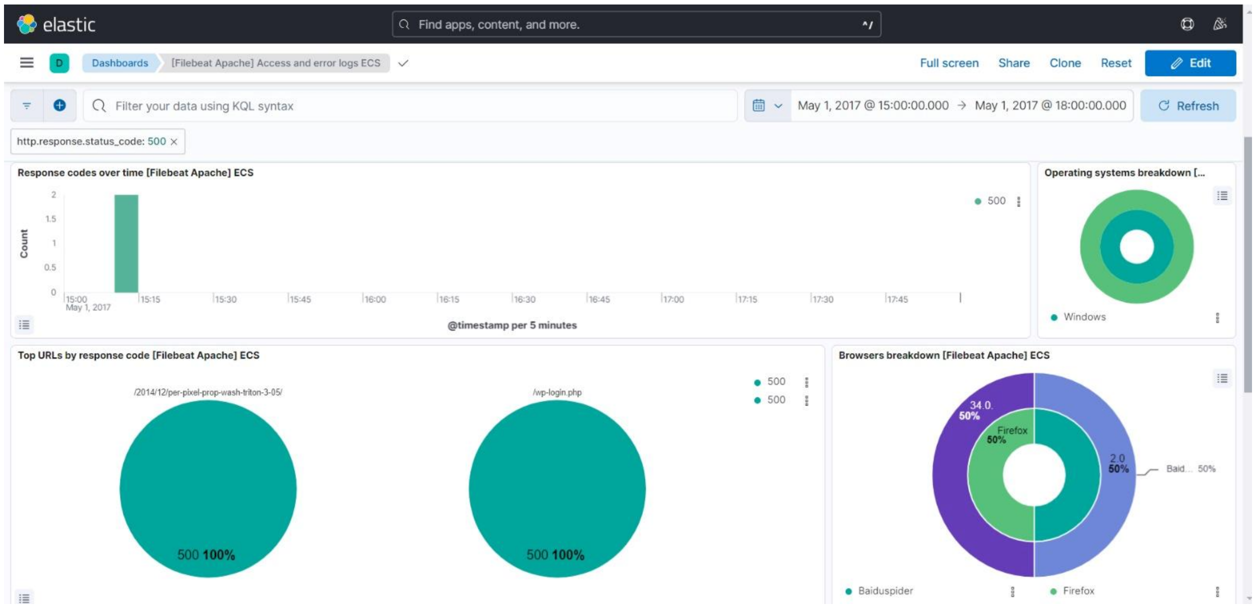
**The following dashboard displays the entire logs details of specific web**



elastic
Find apps, content, and more.
Dashboards
[Filebeat Apache] Access and error logs ECS
Full screen
Share
Clone
Reset
Edit
Filter your data using KQL syntax
May 1, 2017 @ 15:00:00.000 → May 1, 2017 @ 18:00:00.000
Refresh
Unique IPs map [Filebeat Apache] ECS
LAYERS
Unique IPs map [Filebeat Apache] ECS
Basemap
Response codes over time [Filebeat Apache] ECS
Count
@timestamp per 5 minutes
200
304
301
404
500
302
403
408
Operating systems breakdown [...
6.62%
Windows
iOS
Top URLs by response code [Filebeat Apache] ECS
Browsers breakdown [Filebeat Apache] ECS

**The following dashboard displays the logs details of specific web where it faced internal server error**

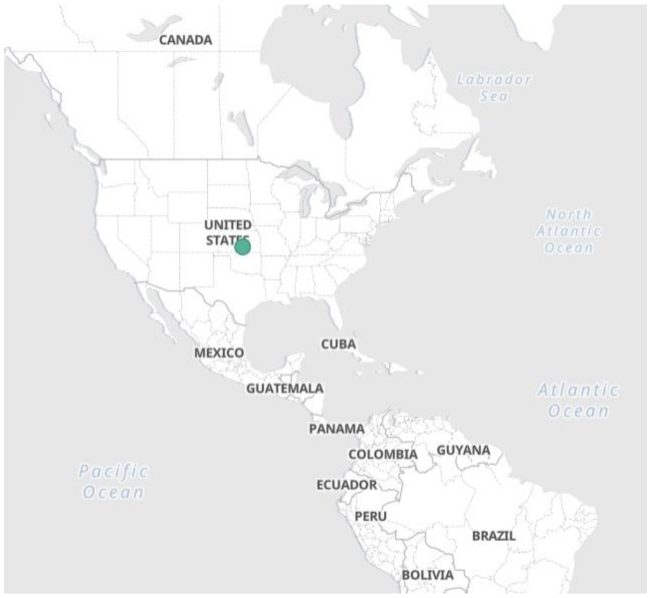
**STATUS CODE: 500**

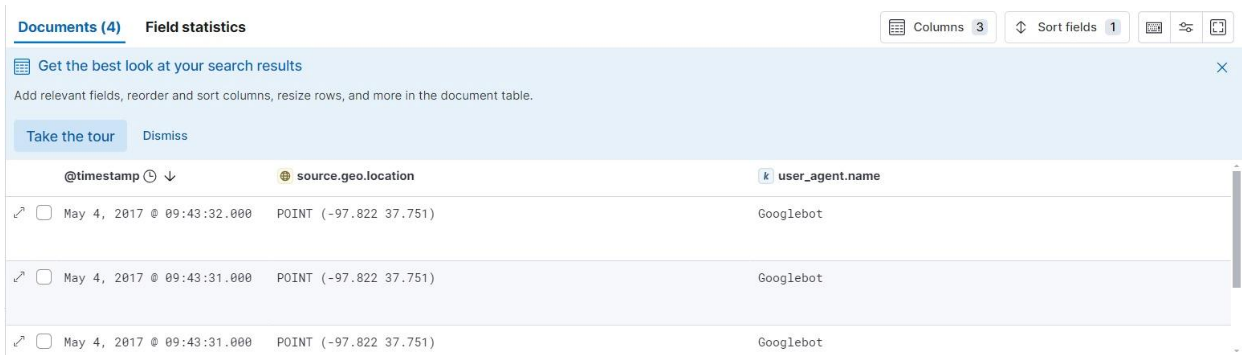

elastic
Find apps, content, and more.
Dashboards
[Filebeat Apache] Access and error logs ECS
Full screen
Share
Clone
Reset
Edit
Refresh
Response codes over time [Filebeat Apache] ECS
Count
@timestamp per 5 minutes
Operating systems breakdown [...
Windows
Top URLs by response code [Filebeat Apache] ECS
/2014/12/per-pixel-prop-wash-triton-3-05/
/wp-login.php
500 100%
500 100%
Browsers breakdown [Filebeat Apache] ECS
34.0. 50%
Firefox 50%
2.0 50%
Baid... 50%
Baiduspider
Firefox

Filter your data using KQL syntax

May 1, 2017 @ 15:00:00.000 → May 1, 2017 @ 18:00:00.000

http.response.status_code: 500





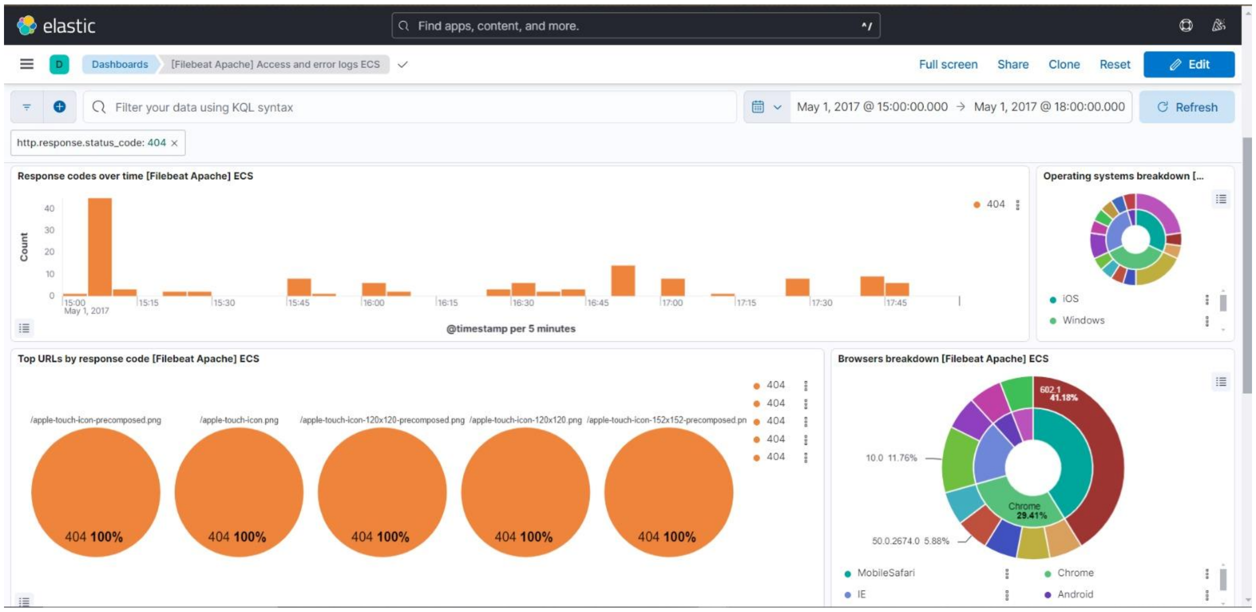
Get the best look at your search results

Add relevant fields, reorder and sort columns, resize rows, and more in the document table.

| @timestamp | source.geo.location | user_agent.name |
|---|---|---|
| May 4, 2017 @ 09:43:32.000 | POINT (-97.822 37.751) | Googlebot |
| May 4, 2017 @ 09:43:31.000 | POINT (-97.822 37.751) | Googlebot |
| May 4, 2017 @ 09:43:31.000 | POINT (-97.822 37.751) | Googlebot |

**The following dashboard displays the logs details of specific web where it faced resource not found error**

**STATUS CODE: 404**

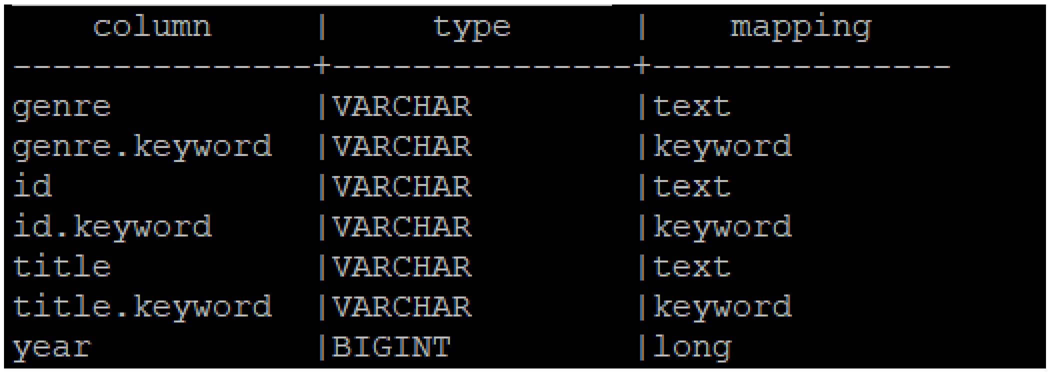

elastic
Find apps, content, and more.
Dashboards
[Filebeat Apache] Access and error logs ECS
Full screen
Share
Clone
Reset
Edit
Filter your data using KQL syntax
May 1, 2017 @ 15:00:00.000 → May 1, 2017 @ 18:00:00.000
Refresh
http.response.status_code: 404
Response codes over time [Filebeat Apache] ECS
Count
@timestamp per 5 minutes
Operating systems breakdown [...
iOS
Windows
Top URLs by response code [Filebeat Apache] ECS
/apple-touch-icon-precomposed.png
/apple-touch-icon.png
/apple-touch-icon-120x120-precomposed.png
/apple-touch-icon-120x120.png
/apple-touch-icon-152x152-precomposed.pn
404 100%
Browsers breakdown [Filebeat Apache] ECS
Chrome 29.41%
MobileSafari
IE
Chrome
Android

## ELASTICSEARCH and SQL

1. **To get the type mappings**

➢ curl -XPOST 127.0.0.1:9200/_sql? Format=txt -d '

{"query": "DESCRIBE movies”}'

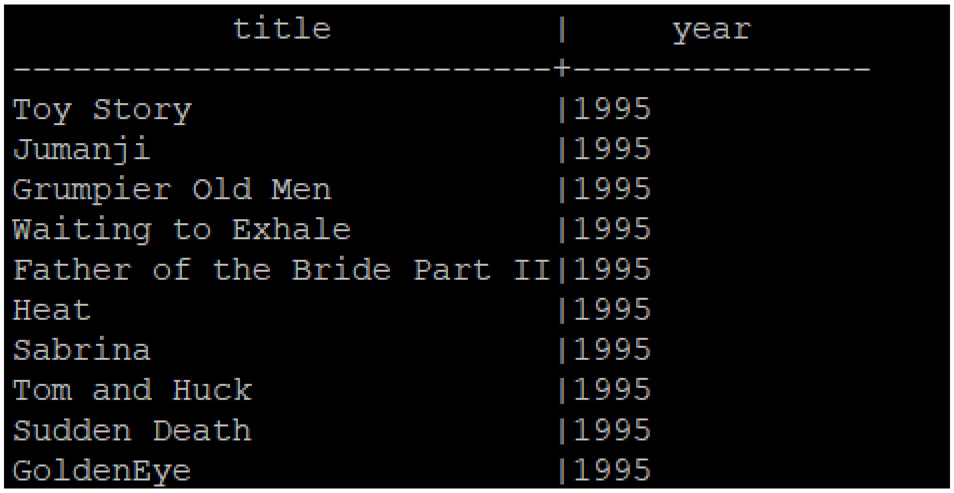


```
    column      |     type      |    mapping
----------------+---------------+---------------
genre           |VARCHAR        |text
genre.keyword   |VARCHAR        |keyword
id              |VARCHAR        |text
id.keyword      |VARCHAR        |keyword
title           |VARCHAR        |text
title.keyword   |VARCHAR        |keyword
year            |BIGINT         |long
```

2. **To get the movies with year field less than 2000 and limit results to 10**

➢ curl -XPOST 127.0.0.1:9200/_sql?format=txt -d '

{"query": "SELECT title, year from movies where year < 2000 limit 10" }'



```
           title           |     year
---------------------------+---------------
Toy Story                  |1995
Jumanji                    |1995
Grumpier Old Men           |1995
Waiting to Exhale          |1995
Father of the Bride Part II|1995
Heat                       |1995
Sabrina                    |1995
Tom and Huck               |1995
Sudden Death               |1995
GoldenEye                  |1995
```

## <u>CANVAS AND SQL</u>

- Firstly, we must create a work pad which can consist of single or multiple pages.
- Each page can consist of elements like charts, graphs, maps, etc.…

**Four elements**:

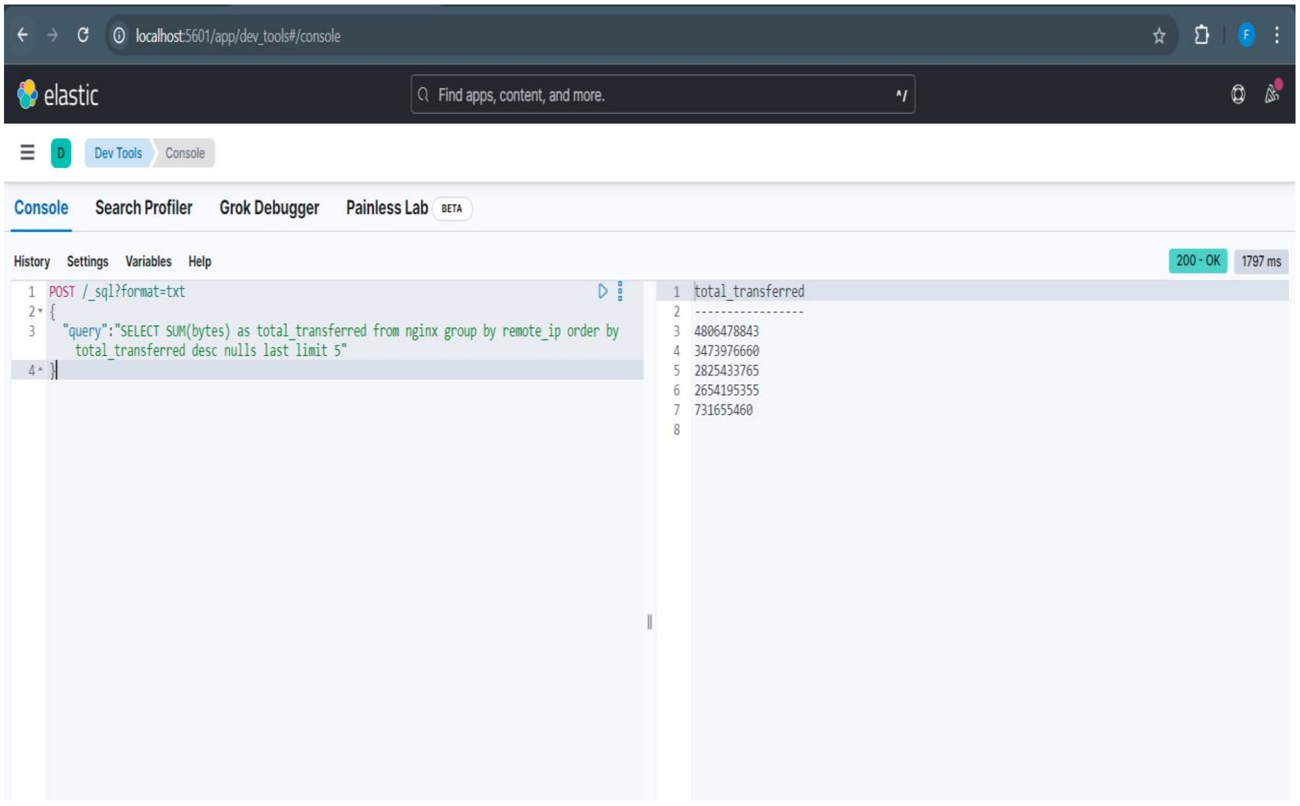
- Charts – bar chart, pie chart, doughnut, area, line, etc.…
- Shapes – text boxes
- Images – can have no. of images varied based on the live data from elastic search
- Supporting elements – dropdown, filter options

**Canvas Data Sources**

Elasticsearch SQL queries

**Steps to create a canvas in Kibana**

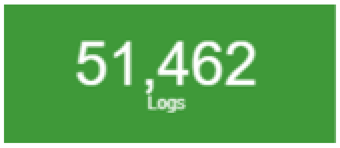
1. Inject Log data into Elastic search index - For examples: **nginx** in our canvas
2. Select the Kibana space in which we want to work
3. Click on Kibana dev tools and check if the SQL queries work fine on the index, we will be using to create canvas metrics



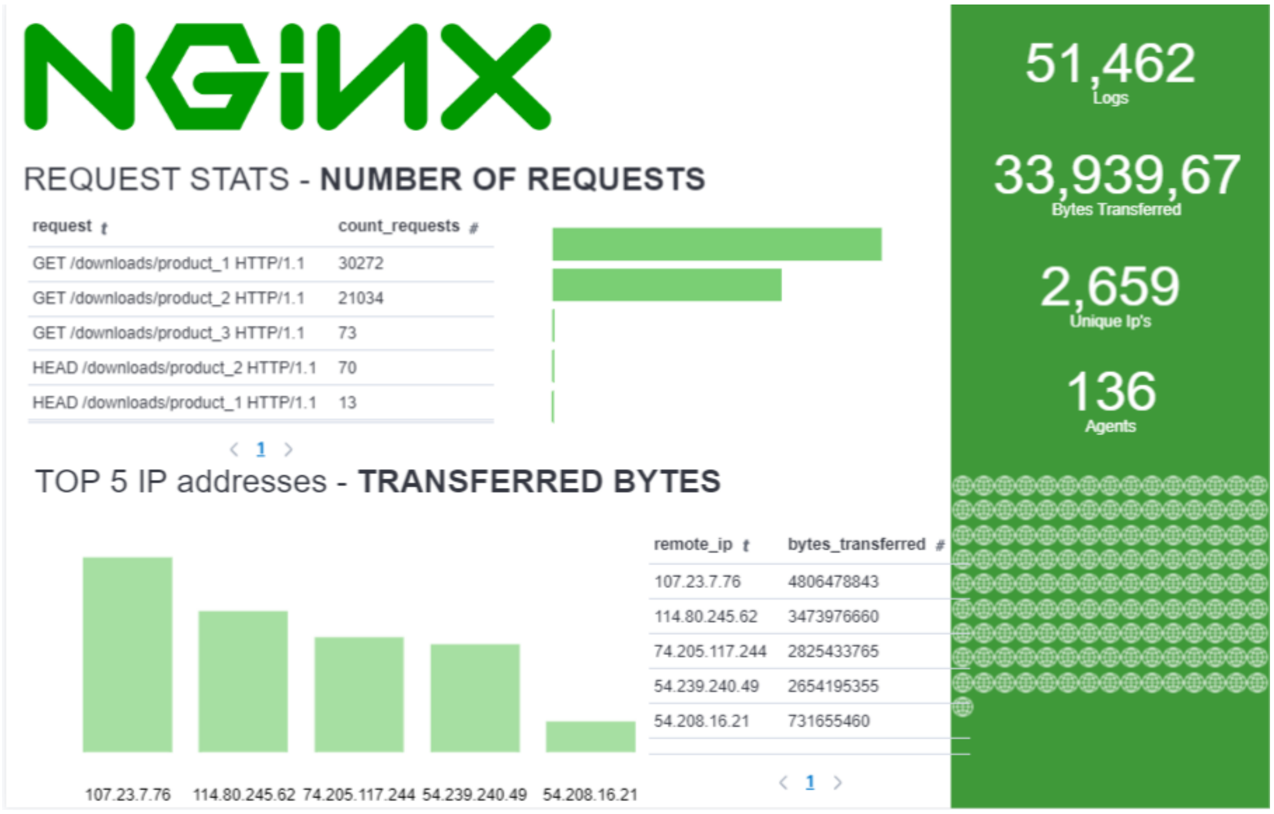
```
POST /_sql?format=txt
{
  "query":"SELECT SUM(bytes) as total_transferred from nginx group by remote_ip order by
    total_transferred desc nulls last limit 5"
}
```

```
total_transferred
-----------------
4806478843
3473976660
2825433765
2654195355
731655460
```

4. Navigate to Analytics -> Canvas -> Work pad
5. After creating new work pad, start adding elements, for example, metric element inside chart



6. In **display tab**, we can change the font properties of the metric and in **data source** using Elasticsearch SQL
   - **SELECT Count (*) AS count_document FROM nginx.**
7. In display we will use the value **count_document** to display the total logs in nginx index.
8. For inserting tables, insert the table element from charts and will get the data from elasticsearch SQL
   - **SELECT request, count (*) as count_requests FROM ngix GROUP BY request ORDER BY count_requests DESC**
9. Insert chart -> Bar chart element. Change the data source (will be same as table)
   In display, we will change the **x-axis to count_requests** and **y-axis to request.**



REQUEST STATS - **NUMBER OF REQUESTS**

| request | count_requests |
|---|---|
| GET /downloads/product_1 HTTP/1.1 | 30272 |
| GET /downloads/product_2 HTTP/1.1 | 21034 |
| GET /downloads/product_3 HTTP/1.1 | 73 |
| HEAD /downloads/product_2 HTTP/1.1 | 70 |
| HEAD /downloads/product_1 HTTP/1.1 | 13 |

TOP 5 IP addresses - **TRANSFERRED BYTES**

| remote_ip | bytes_transferred |
|---|---|
| 107.23.7.76 | 4806478843 |
| 114.80.245.62 | 3473976660 |
| 74.205.117.244 | 2825433765 |
| 54.239.240.49 | 2654195355 |
| 54.208.16.21 | 731655460 |

51,462
Logs

33,939,67
Bytes Transferred

2,659
Unique Ip's

136
Agents

## BACKUP AND TROUBLE SHOOTING

### Categories

- Node setup
- Discovery and cluster formation
- Indexing data and sharding
- Searching
- Backing up data

### Steps to perform Back up in Elasticsearch:

- sudo nano /etc/elasticsearch/elasticsearch.yml
- Now add path.repo : ["/home/student/backups"] after the path.logs in elasticsearch.yml
- sudo cp /etc/elasticsearch/elasticsearch.yml ~/
- sudo mkdir -p /home/student/backups
- sudo chgrp elasticsearch /home/student/backups
- sudo chmod g+w /home/student/backups/
- sudo /bin/systemctl stop elasticsearch.service
- sudo /bin/systemctl start elasticsearch.service
- curl --request PUT localhost:9200/_snapshot/backup-repo \

```
--data-raw ' {
"type" : "fs",
"settings": {
        "location":"/home/student/backups/backup-repo"
}
}'
```

- curl --request PUT localhost:9200/_snapshot/backup-repo/snapshot-1
- curl --request GET localhost:9200/_snapshot/backup-repo/snapshot-1? Pretty

**Potential Issues and Trouble Shooting in Elasticsearch:**

**Open a new PUTTY window – Terminal 2**

sudo visudo

In the bottom of the file add:username ALL=(elasticsearch) NOPASSWD: ALL

sudo -su elasticsearch

cd /var/log/elasticsearch/

tail -n 500 elastisearch.log | grep ERROR

cat Elasticsearch.log | grep Bootstrap --context=3

**MEMORY LOCK ISSUE:**

**In Terminal-2:**

- sudo nano /etc/elasticsearch/elasticsearch.yml
- Uncomment the line bootstrap.memory_lock:true
- sudo systemctl stop elasticsearch.service
- sudo systemctl start elasticsearch.service

**It will throw you an error: It will show memory is locked error

**Go back to Terminal-1**

- sudo systemctl edit elasticsearch.service
- Add the following in the file to resolve the above error:
  ```
  [Service]
  LimitMEMLOCK=infinity
  ```
- sudo systemctl start elasticsearch.service

## HEAP MEMORY ALLOCATION ISSUE:

**In Terminal-2:**

- sudo nano /etc/elasticsearch/jvm.options
- Comment out both -Xmslg and Xmxlg in the jvm file
- And add:
  -Xms500m
  -Xmslg
- sudo systemctl stop elasticsearch.service
- sudo systemctl start elasticsearch.service

**It will throw error initial heap size not equal to the initial allocation error

- sudo nano /etc/elasticsearch/jvm.options
- Uncomment out both -Xmslg and Xmxlg
- And remove:
  -Xms500m
  -Xmslg
- sudo systemctl stop elasticsearch.service
- sudo systemctl start elasticsearch.service

## NODE SETUP ISSUES:

**In Terminal-2**

- sudo cat /usr/lib/system/system/elasticsearch.service
- sudo nano /etc/elasticsearch/elasticsearch.yml
- Comment out:
  discovery.seed_hosts: ["127.0.0.1"] and
  cluster.initial_master_nodes: ["node-1"]

**It will throw master not found exception

- sudo systemctl stop elasticsearch.service

**In Terminal-1**

- rm -rf /var/lib/elasticsearch/*
- Go back to terminal 2 and do
- sudo vim /etc/elasticsearch/elasticsearch.yml
- Uncomment and change:
  cluster-name:lecture-cluster
  discovery.seed_hosts: ["127.0.0.1:9301"]
  sudo systemctl start elasticsearch.service

**The cluster_uuid will be na which means the cluster is not formed.

sudo systemctl stop elasticsearch.service

**Reasons for not forming cluster:**

**It may be due to network issues where nodes within cluster might be unable to communicated with each other.**

## INDEX SETUP ISSUES

**Creating index with 1 shard and 1 replica**:

- Curl –request PUT localhost:9200/test \
  ```
  --data-raw '{
  "settings":
  {
  "number_of_shards":1,
  "number_of_replicas":1}
  }'
  ```

- To check about the shard's status using:
  - ➢ Curl localhost:9200/_cat/shards? V

    It will return that the status as started or unassigned.

- Cluster allocation API to explain why shards aren't allocated
  - ➢ Curl localhost:9200/_cluster/allocation/explain? Pretty

    Reason – replica to the same node is not allowed.
  - ➢ How to overcome?

    Add a new node and take replica to the new node.

**Steps to setup 2nd node:**

- Sudo nano /etc/elasticsearch-node2/elasticsearch.yml

  Node.name : node-2

  Master.nodes will be node-1 and node-2
- Start the 2nd node on the same VM

  Sudo systemctl start elasticsearch-node2

## INDEX DESIGN CHANGES (SPLITTING, SHRINKING)

**Index settings:**

➔ Dynamic – can be changed after index creation

- Number_of_replicas
- Refresh intervals
- Blocks – disabling readability/writability of index
- Pipeline – preprocessing pipeline for every documents

➔ Static – can’t be changed after index creation
  - Number_of_shards

**Sharding goals:**

- High availability - working uninterrupted for a long time
- High resiliency – resist errors (using replicas)

**Increase/decrease shards**

**To decrease**

- POST /{source_index}/_shrink/{target_index-name}

**To increase**

- POST /{source_index}/_split/{target_index-name}

# IMPLEMATATION OF ELASTICSTACK IN CAPSTONE PROJECT

## 1. Setting up Logger and ELK in Spring boot:

In the **pom.xml file** add the two dependencies:

```
<!-- Logback for logging -->
            <dependency>
               <groupId>ch.qos.logback</groupId>
               <artifactId>logback-classic</artifactId>
            </dependency>
<!-- Logstash Logback Encoder to send logs to Elasticsearch -->
            <dependency>
               <groupId>net.logstash.logback</groupId>
               <artifactId>logstash-logback-encoder</artifactId>
               <version>7.2</version>
            </dependency>
```

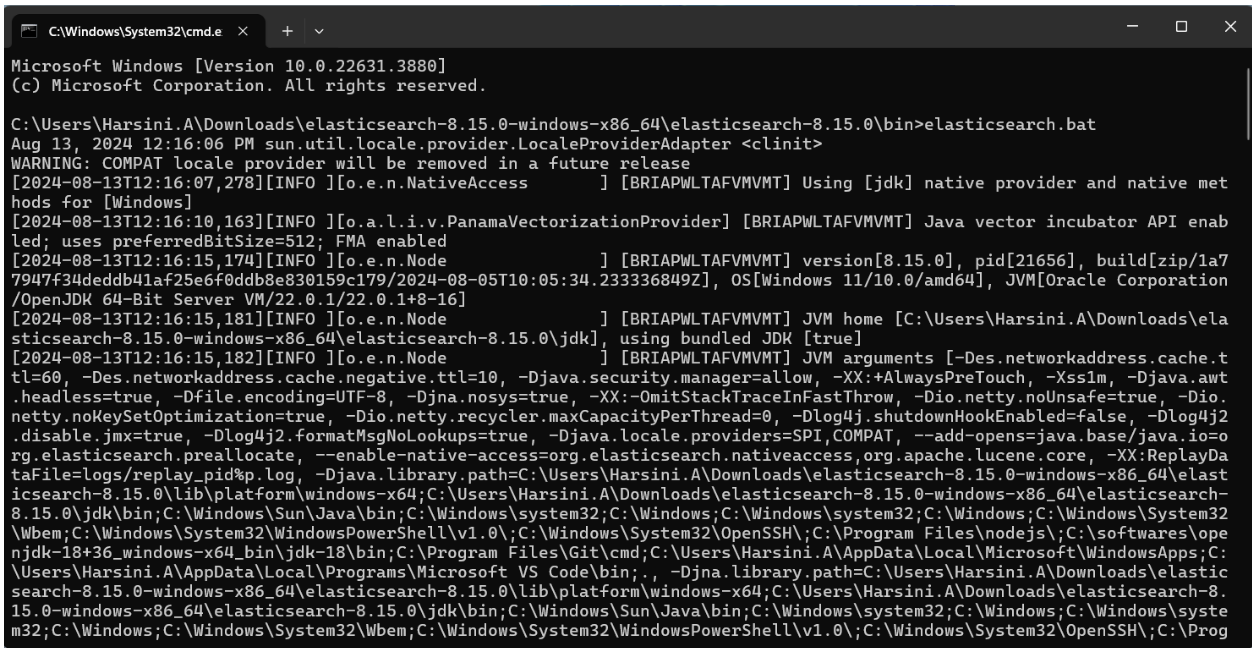
In the **application.yml** file add:

*Here give the absolute or relative path of the file where you want to store your logs.

```
logging:
  file:
    path: C:/Users/Devatharshini.S/OneDrive - Brillio/Desktop/logs/customer
```

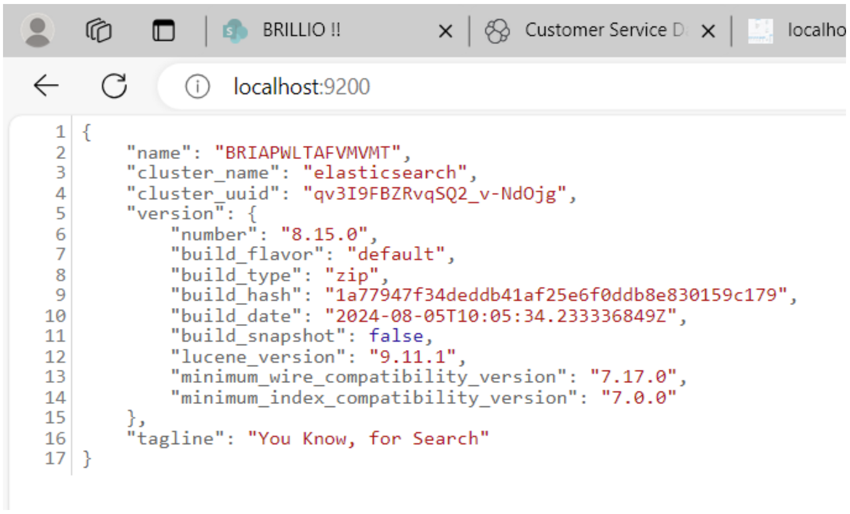
## 2. Setting Up Elasticsearch on Windows

- Download the latest Elasticsearch version's(version-8.15.0) zip and extract the files.
- Change the configuration: Set the X-**Pack security features to false** else it **won't allow Elasticsearch to run** locally due to security conflicts.
- In the Command Prompt, navigate to the bin directory of Elasticsearch and run the **elasticsearch.bat** file.



```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Harsini.A\Downloads\elasticsearch-8.15.0-windows-x86_64\elasticsearch-8.15.0\bin>elasticsearch.bat
Aug 13, 2024 12:16:06 PM sun.util.locale.provider.LocaleProviderAdapter <clinit>
WARNING: COMPAT locale provider will be removed in a future release
[2024-08-13T12:16:07,278][INFO ][o.e.n.NativeAccess       ] [BRIAPWLTAFVMVMT] Using [jdk] native provider and native met
hods for [Windows]
[2024-08-13T12:16:10,163][INFO ][o.a.l.i.v.PanamaVectorizationProvider] [BRIAPWLTAFVMVMT] Java vector incubator API enab
led; uses preferredBitSize=512; FMA enabled
[2024-08-13T12:16:15,174][INFO ][o.e.n.Node               ] [BRIAPWLTAFVMVMT] version[8.15.0], pid[21656], build[zip/1a7
7947f34deddb41af25e6f0ddb8e830159c179/2024-08-05T10:05:34.233336849Z], OS[Windows 11/10.0/amd64], JVM[Oracle Corporation
/OpenJDK 64-Bit Server VM/22.0.1/22.0.1+8-16]
[2024-08-13T12:16:15,181][INFO ][o.e.n.Node               ] [BRIAPWLTAFVMVMT] JVM home [C:\Users\Harsini.A\Downloads\ela
sticsearch-8.15.0-windows-x86_64\elasticsearch-8.15.0\jdk], using bundled JDK [true]
[2024-08-13T12:16:15,182][INFO ][o.e.n.Node               ] [BRIAPWLTAFVMVMT] JVM arguments [-Des.networkaddress.cache.t
tl=60, -Des.networkaddress.cache.negative.ttl=10, -Djava.security.manager=allow, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt
.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -XX:-OmitStackTraceInFastThrow, -Dio.netty.noUnsafe=true, -Dio.
netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2
.disable.jmx=true, -Dlog4j2.formatMsgNoLookups=true, -Djava.locale.providers=SPI,COMPAT, --add-opens=java.base/java.io=o
rg.elasticsearch.preallocate, --enable-native-access=org.elasticsearch.nativeaccess,org.apache.lucene.core, -XX:ReplayDa
taFile=logs/replay_pid%p.log, -Djava.library.path=C:\Users\Harsini.A\Downloads\elasticsearch-8.15.0-windows-x86_64\elast
icsearch-8.15.0\lib\platform\windows-x64;C:\Users\Harsini.A\Downloads\elasticsearch-8.15.0-windows-x86_64\elasticsearch-
8.15.0\jdk\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\system32;C:\Windows;C:\Windows\System32
\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\nodejs\;C:\softwares\ope
njdk-18+36_windows-x64_bin\jdk-18\bin;C:\Program Files\Git\cmd;C:\Users\Harsini.A\AppData\Local\Microsoft\WindowsApps;C:
\Users\Harsini.A\AppData\Local\Programs\Microsoft VS Code\bin;., -Djna.library.path=C:\Users\Harsini.A\Downloads\elastic
search-8.15.0-windows-x86_64\elasticsearch-8.15.0\lib\platform\windows-x64;C:\Users\Harsini.A\Downloads\elasticsearch-8.
15.0-windows-x86_64\elasticsearch-8.15.0\jdk\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\syste
m32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Prog
```

- If we go to the web browser and enter the URL http://localhost:9200, we can see a JSON response indicating the status of your Elasticsearch node, including details like version and cluster name.



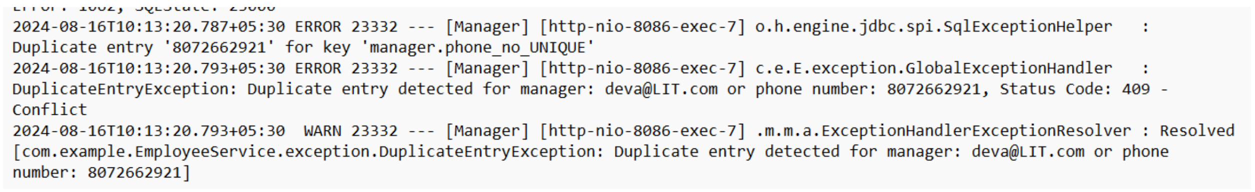
```
{
    "name": "BRIAPWLTAFVMVMT",
    "cluster_name": "elasticsearch",
    "cluster_uuid": "qv3I9FBZRvqSQ2_v-NdOjg",
    "version": {
        "number": "8.15.0",
        "build_flavor": "default",
        "build_type": "zip",
        "build_hash": "1a77947f34deddb41af25e6f0ddb8e830159c179",
        "build_date": "2024-08-05T10:05:34.233336849Z",
        "build_snapshot": false,
        "lucene_version": "9.11.1",
        "minimum_wire_compatibility_version": "7.17.0",
        "minimum_index_compatibility_version": "7.0.0"
    },
    "tagline": "You Know, for Search"
}
```

## 3. Setting Up Logstash on Windows

- Download the latest Logstash version's (version: 8.15.0) zip and extract the files.
- In the Command Prompt, navigate to the bin directory of Logstash and we can run Logstash with our configuration directly in the command line using the -e option

**Application.log** → File containing logs from Employee Micro service

**Example of logs in the Application.log file**



```
2024-08-16T10:13:20.787+05:30 ERROR 23332 --- [Manager] [http-nio-8086-exec-7] o.h.engine.jdbc.spi.SqlExceptionHelper   :
Duplicate entry '8072662921' for key 'manager.phone_no_UNIQUE'
2024-08-16T10:13:20.793+05:30 ERROR 23332 --- [Manager] [http-nio-8086-exec-7] c.e.E.exception.GlobalExceptionHandler   :
DuplicateEntryException: Duplicate entry detected for manager: deva@LIT.com or phone number: 8072662921, Status Code: 409 -
Conflict
2024-08-16T10:13:20.793+05:30  WARN 23332 --- [Manager] [http-nio-8086-exec-7] .m.m.a.ExceptionHandlerExceptionResolver : Resolved
[com.example.EmployeeService.exception.DuplicateEntryException: Duplicate entry detected for manager: deva@LIT.com or phone
number: 8072662921]
```

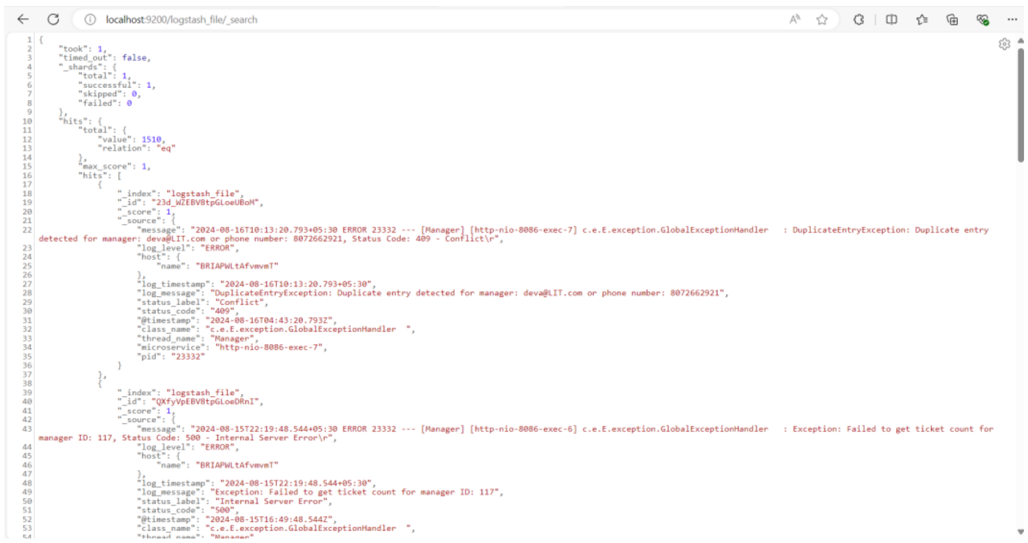
**Employee Microservice logs Logstash configuration**

The configuration file is used to push the employee microservice log data which is stored in the application.log file to the Elasticsearch and the index is named as ‘**logstash_file’**

```
logstash -e "

        input {
                file {
                        path => 'C:/Users/Harsini.A/OneDrive -
                        Brillio/Documents/logs/application.log'
                        start_position => 'beginning'
                }

        }

        filter {

            grok {

                match => {

                        'message' => '%{TIMESTAMP_ISO8601:log_timestamp}
                        %{LOGLEVEL:log_level} %{NUMBER:pid} ---
                        \[%{DATA:thread_name}\] \[%{DATA:microservice}\]
                        %{GREEDYDATA:class_name} : %{GREEDYDATA:log_message},
                        Status Code: %{NUMBER:status_code} -
                        %{GREEDYDATA:status_label}\r' }

                }

                date {

                        match => ['log_timestamp', 'ISO8601']

                        target => '@timestamp'

                }

                mutate {

                         remove_field => [ 'event', 'path', 'log', '@version' ]

                        }

                }

        output {

                stdout {

                        codec => rubydebug

                        }

                elasticsearch {
                        hosts => ['localhost:9200']

                        index => 'logstash_file'

                        }

                }"
```

**Logstash_file index:**

You can view the documents in the logstash_file index by navigating to [http://localhost/logstash_file/_search](http://localhost/logstash_file/_search)



localhost:9200/logstash_file/_search

```
{
    "took": 1,
    "timed_out": false,
    "_shards": {
        "total": 1,
        "successful": 1,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": {
            "value": 1510,
            "relation": "eq"
        },
        "max_score": 1,
        "hits": [
            {
                "_index": "logstash_file",
                "_id": "23d_WZEBV8tpGLoeUBoM",
                "_score": 1,
                "_source": {
                    "message": "2024-08-16T10:13:20.793+05:30 ERROR 23332 --- [Manager] [http-nio-8086-exec-7] c.e.E.exception.GlobalExceptionHandler   : DuplicateEntryException: Duplicate entry detected for manager: deva@LIT.com or phone number: 8072662921, Status Code: 409 - Conflict\r",
                    "log_level": "ERROR",
                    "host": {
                        "name": "BRIAPWLtAfvmvmT"
                    },
                    "log_timestamp": "2024-08-16T10:13:20.793+05:30",
                    "log_message": "DuplicateEntryException: Duplicate entry detected for manager: deva@LIT.com or phone number: 8072662921",
                    "status_label": "Conflict",
                    "status_code": "409",
                    "@timestamp": "2024-08-16T04:43:20.793Z",
                    "class_name": "c.e.E.exception.GlobalExceptionHandler  ",
                    "thread_name": "Manager",
                    "microservice": "http-nio-8086-exec-7",
                    "pid": "23332"
                }
            },
            {
                "_index": "logstash_file",
                "_id": "QXfyVpEBV8tpGLoeDRnI",
                "_score": 1,
                "_source": {
                    "message": "2024-08-15T22:19:48.544+05:30 ERROR 23332 --- [Manager] [http-nio-8086-exec-6] c.e.E.exception.GlobalExceptionHandler   : Exception: Failed to get ticket count for manager ID: 117, Status Code: 500 - Internal Server Error\r",
                    "log_level": "ERROR",
                    "host": {
                        "name": "BRIAPWLtAfvmvmT"
                    },
                    "log_timestamp": "2024-08-15T22:19:48.544+05:30",
                    "log_message": "Exception: Failed to get ticket count for manager ID: 117",
                    "status_label": "Internal Server Error",
                    "status_code": "500",
                    "@timestamp": "2024-08-15T16:49:48.544Z",
                    "class_name": "c.e.E.exception.GlobalExceptionHandler  ",
                    "thread_name": "Manager",
```

**Customer-application.log** → File containing logs from Customer Micro service

**Example of logs in the customer-application.log file**:

```
2024-08-16T10:16:58.984+05:30  INFO 656 --- [Customer] [http-nio-8088-exec-9] c.e.C.controller.CustomerController      :
Calculating resolution average for managerId: 117
2024-08-16T10:16:58.984+05:30  INFO 656 --- [Customer] [http-nio-8088-exec-9] c.example.Customer.dao.CustomerdaoImpl   :
Calculating top 5 representative-wise average resolution time for manager ID: 117
2024-08-16T10:16:58.984+05:30  INFO 656 --- [Customer] [http-nio-8088-exec-9] c.example.Customer.dao.CustomerdaoImpl   : Fetching
tickets for manager ID: 117
2024-08-16T10:16:58.998+05:30  INFO 656 --- [Customer] [http-nio-8088-exec-9] c.example.Customer.dao.CustomerdaoImpl   :
Successfully fetched 27 tickets for manager ID: 117
2024-08-16T10:16:59.012+05:30  INFO 656 --- [Customer] [http-nio-8088-exec-9] c.example.Customer.dao.CustomerdaoImpl   : Top 5
representative-wise average resolution time for manager ID: 117: {178=4.0, 171=2.0, 43=2.25, 188=2.5, 174=1.0}
2024-08-16T10:16:59.012+05:30  INFO 656 --- [Customer] [http-nio-8088-exec-9] c.e.C.controller.CustomerController      :
Resolution average result for managerId 117: {178=4.0, 171=2.0, 43=2.25, 188=2.5, 174=1.0}
```

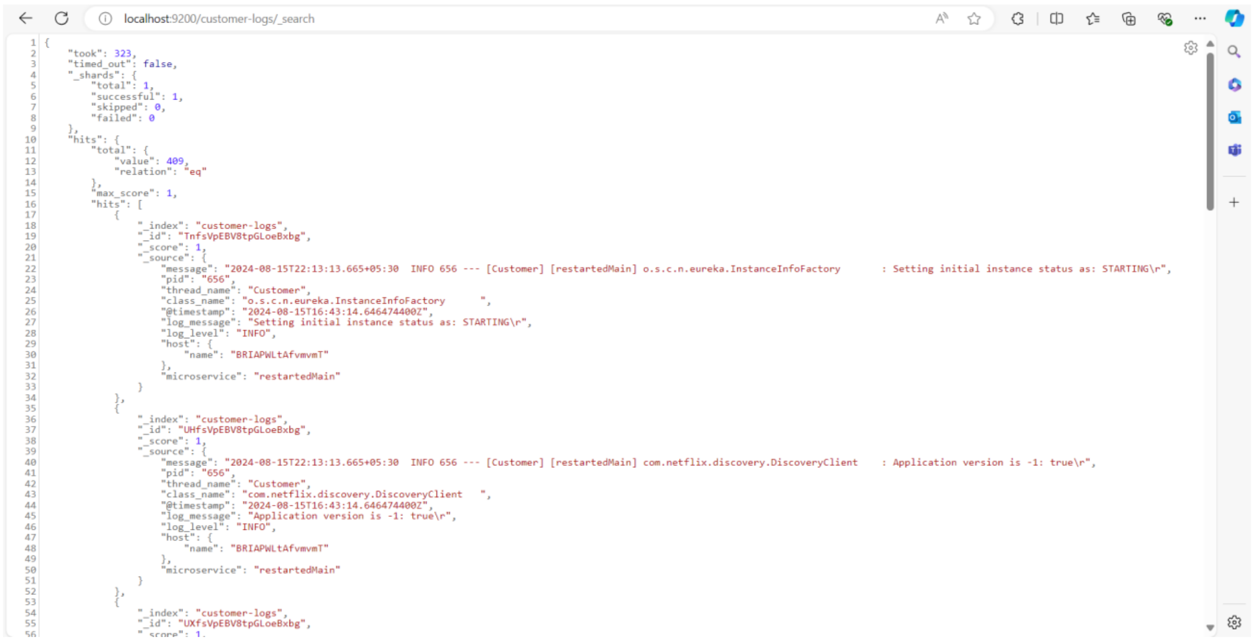
**Customer Microservice logs logstash configuration**

The configuration file is used to push the employee microservice log data which is stored in the application.log file to the Elasticsearch and the index is named as '**customer_logs'**

```
logstash -e "
    input {
        file {
            path => 'C:/Users/Harsini.A/OneDrive - Brillio/Documents/logs/customer-
            application.log'
            start_position => 'beginning'
            }
        }
    filter {
            grok {
                    match =>
                     { 'message' => '%{LOGLEVEL:log_level} %{NUMBER:pid} ---
            \[%{DATA:thread_name}\] \[%{DATA:microservice}\]
            %{GREEDYDATA:class_name} : %{GREEDYDATA:log_message}' }
                    }
                    mutate {
                    remove_field => [ 'event', 'path', 'log', '@version' ]
                    } }
    output {
            stdout {
                    codec => rubydebug }
            elasticsearch {
                    hosts => ['localhost:9200']
                    index => 'customer-logs' }
    }"
```

**Customer-logs index**

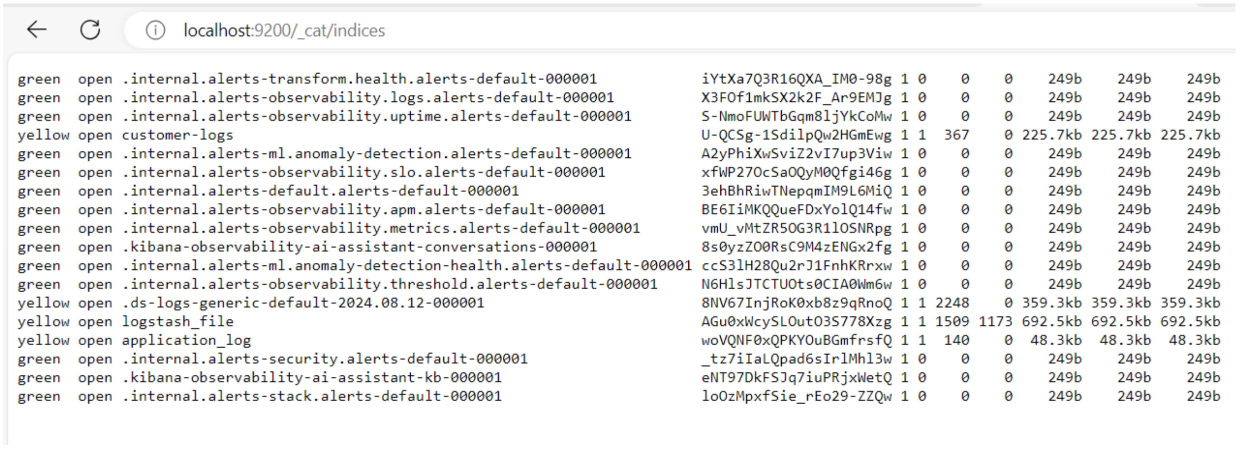
You can view the documents in the customer-logs index by navigating to http://localhost/customer-logs/_search



localhost:9200/customer-logs/_search

```
{
    "took": 323,
    "timed_out": false,
    "_shards": {
        "total": 1,
        "successful": 1,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": {
            "value": 409,
            "relation": "eq"
        },
        "max_score": 1,
        "hits": [
            {
                "_index": "customer-logs",
                "_id": "TnfsVpEBV8tpGLoeBxbg",
                "_score": 1,
                "_source": {
                    "message": "2024-08-15T22:13:13.665+05:30  INFO 656 --- [Customer] [restartedMain] o.s.c.n.eureka.InstanceInfoFactory       : Setting initial instance status as: STARTING\r",
                    "pid": "656",
                    "thread_name": "Customer",
                    "class_name": "o.s.c.n.eureka.InstanceInfoFactory      ",
                    "@timestamp": "2024-08-15T16:43:14.646474400Z",
                    "log_message": "Setting initial instance status as: STARTING\r",
                    "log_level": "INFO",
                    "host": {
                        "name": "BRIAPWLtAfvmvmT"
                    },
                    "microservice": "restartedMain"
                }
            },
            {
                "_index": "customer-logs",
                "_id": "UHfsVpEBV8tpGLoeBxbg",
                "_score": 1,
                "_source": {
                    "message": "2024-08-15T22:13:13.665+05:30  INFO 656 --- [Customer] [restartedMain] com.netflix.discovery.DiscoveryClient    : Application version is -1: true\r",
                    "pid": "656",
                    "thread_name": "Customer",
                    "class_name": "com.netflix.discovery.DiscoveryClient   ",
                    "@timestamp": "2024-08-15T16:43:14.646474400Z",
                    "log_message": "Application version is -1: true\r",
                    "log_level": "INFO",
                    "host": {
                        "name": "BRIAPWLtAfvmvmT"
                    },
                    "microservice": "restartedMain"
                }
            },
            {
                "_index": "customer-logs",
                "_id": "UXfsVpEBV8tpGLoeBxbg",
                "_score": 1,
```

- In the below picture you can see the indices are created successfully bu navigating to **http://localhost/_cat/_indices**.
- **Customer-logs and logstash_file** are the indices which we created, and the rest are the default indices present in elasticsearch.



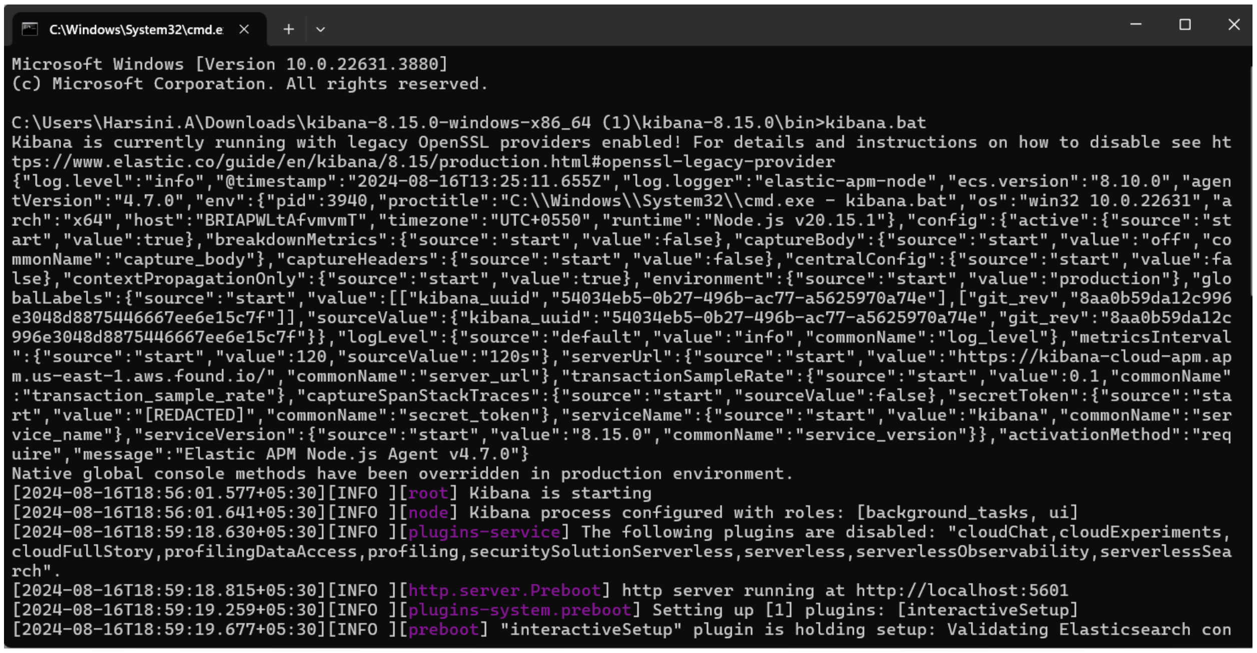
localhost:9200/_cat/indices

```
green  open .internal.alerts-transform.health.alerts-default-000001         iYtXa7Q3R16QXA_IM0-98g 1 0    0    0    249b    249b    249b
green  open .internal.alerts-observability.logs.alerts-default-000001       X3FOf1mkSX2k2F_Ar9EMJg 1 0    0    0    249b    249b    249b
green  open .internal.alerts-observability.uptime.alerts-default-000001     S-NmoFUWTbGqm8ljYkCoMw 1 0    0    0    249b    249b    249b
yellow open customer-logs                                                   U-QCSg-1SdilpQw2HGmEwg 1 1  367    0 225.7kb 225.7kb 225.7kb
green  open .internal.alerts-ml.anomaly-detection.alerts-default-000001     A2yPhiXwSviZ2vI7up3Viw 1 0    0    0    249b    249b    249b
green  open .internal.alerts-observability.slo.alerts-default-000001        xfWP27OcSaOQyM0Qfgi46g 1 0    0    0    249b    249b    249b
green  open .internal.alerts-default.alerts-default-000001                  3ehBhRiwTNepqmIM9L6MiQ 1 0    0    0    249b    249b    249b
green  open .internal.alerts-observability.apm.alerts-default-000001        BE6IiMKQQueFDxYolQ14fw 1 0    0    0    249b    249b    249b
green  open .internal.alerts-observability.metrics.alerts-default-000001    vmU_vMtZR5OG3R1lOSNRpg 1 0    0    0    249b    249b    249b
green  open .kibana-observability-ai-assistant-conversations-000001         8s0yzZO0RsC9M4zENGx2fg 1 0    0    0    249b    249b    249b
green  open .internal.alerts-ml.anomaly-detection-health.alerts-default-000001 ccS3lH28Qu2rJ1FnhKRrxw 1 0    0    0    249b    249b    249b
green  open .internal.alerts-observability.threshold.alerts-default-000001  N6HlsJTCTUOts0CIA0Wm6w 1 0    0    0    249b    249b    249b
yellow open .ds-logs-generic-default-2024.08.12-000001                      8NV67InjRoK0xb8z9qRnoQ 1 1 2248    0 359.3kb 359.3kb 359.3kb
yellow open logstash_file                                                   AGu0xWcySLOutO3S778Xzg 1 1 1509 1173 692.5kb 692.5kb 692.5kb
yellow open application_log                                                 woVQNF0xQPKYOuBGmfrsfQ 1 1  140    0  48.3kb  48.3kb  48.3kb
green  open .internal.alerts-security.alerts-default-000001                 _tz7iIaLQpad6sIrlMhl3w 1 0    0    0    249b    249b    249b
green  open .kibana-observability-ai-assistant-kb-000001                    eNT97DkFSJq7iuPRjxWetQ 1 0    0    0    249b    249b    249b
green  open .internal.alerts-stack.alerts-default-000001                    loOzMpxfSie_rEo29-ZZQw 1 0    0    0    249b    249b    249b
```

**KIBANA**

## 4. Setting Up Kibana on Windows

- Download the latest Kibana version's (version:8.15.0) zip and extract the files.
- Make changes in the Kibana configuration files which is inside the directory called **config and named as kibana.yml** like establishing connection with the Elasticsearch through the **port 9200**
- In the Command Prompt, navigate to the bin directory of Kibana and run the **kibana.bat** file.
- In the web browser, type the URL http://localhost:5601/

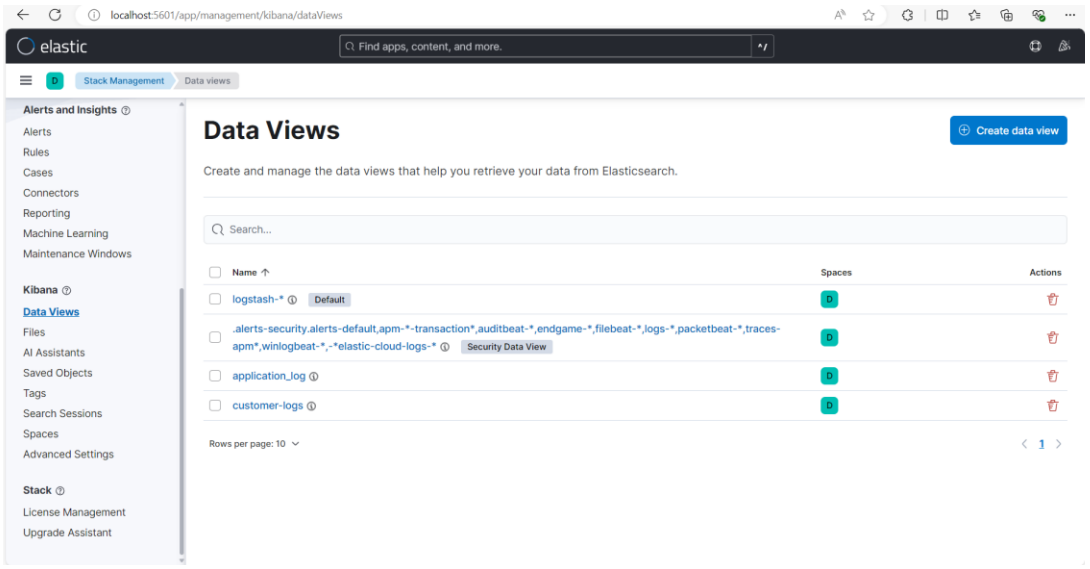


```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Harsini.A\Downloads\kibana-8.15.0-windows-x86_64 (1)\kibana-8.15.0\bin>kibana.bat
Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on how to disable see ht
tps://www.elastic.co/guide/en/kibana/8.15/production.html#openssl-legacy-provider
{"log.level":"info","@timestamp":"2024-08-16T13:25:11.655Z","log.logger":"elastic-apm-node","ecs.version":"8.10.0","agen
tVersion":"4.7.0","env":{"pid":3940,"proctitle":"C:\\Windows\\System32\\cmd.exe - kibana.bat","os":"win32 10.0.22631","a
rch":"x64","host":"BRIAPWLtAfvmvmT","timezone":"UTC+0550","runtime":"Node.js v20.15.1"},"config":{"active":{"source":"st
art","value":true},"breakdownMetrics":{"source":"start","value":false},"captureBody":{"source":"start","value":"off","co
mmonName":"capture_body"},"captureHeaders":{"source":"start","value":false},"centralConfig":{"source":"start","value":fa
lse},"contextPropagationOnly":{"source":"start","value":true},"environment":{"source":"start","value":"production"},"glo
balLabels":{"source":"start","value":[["kibana_uuid","54034eb5-0b27-496b-ac77-a5625970a74e"],["git_rev","8aa0b59da12c996
e3048d8875446667ee6e15c7f"]],"sourceValue":{"kibana_uuid":"54034eb5-0b27-496b-ac77-a5625970a74e","git_rev":"8aa0b59da12c
996e3048d8875446667ee6e15c7f"}},"logLevel":{"source":"default","value":"info","commonName":"log_level"},"metricsInterval
":{"source":"start","value":120,"sourceValue":"120s"},"serverUrl":{"source":"start","value":"https://kibana-cloud-apm.ap
m.us-east-1.aws.found.io/","commonName":"server_url"},"transactionSampleRate":{"source":"start","value":0.1,"commonName"
:"transaction_sample_rate"},"captureSpanStackTraces":{"source":"start","sourceValue":false},"secretToken":{"source":"sta
rt","value":"[REDACTED]","commonName":"secret_token"},"serviceName":{"source":"start","value":"kibana","commonName":"ser
vice_name"},"serviceVersion":{"source":"start","value":"8.15.0","commonName":"service_version"}},"activationMethod":"req
uire","message":"Elastic APM Node.js Agent v4.7.0"}
Native global console methods have been overridden in production environment.
[2024-08-16T18:56:01.577+05:30][INFO ][root] Kibana is starting
[2024-08-16T18:56:01.641+05:30][INFO ][node] Kibana process configured with roles: [background_tasks, ui]
[2024-08-16T18:59:18.630+05:30][INFO ][plugins-service] The following plugins are disabled: "cloudChat,cloudExperiments,
cloudFullStory,profilingDataAccess,profiling,securitySolutionServerless,serverless,serverlessObservability,serverlessSea
rch".
[2024-08-16T18:59:18.815+05:30][INFO ][http.server.Preboot] http server running at http://localhost:5601
[2024-08-16T18:59:19.259+05:30][INFO ][plugins-system.preboot] Setting up [1] plugins: [interactiveSetup]
[2024-08-16T18:59:19.677+05:30][INFO ][preboot] "interactiveSetup" plugin is holding setup: Validating Elasticsearch con
```

<u>**VISUALIZATION IN KIBANA**</u>

## 1. Steps to create Data View

- Navigate to **Stack Management -> Data Views** tab inside Kibana.
- Click the Create Data View and add the index named **logstash-file** (containing logs from the Employee Microservice) and **customer-logs** (containing logs from the Customer microservice) to create Data views.

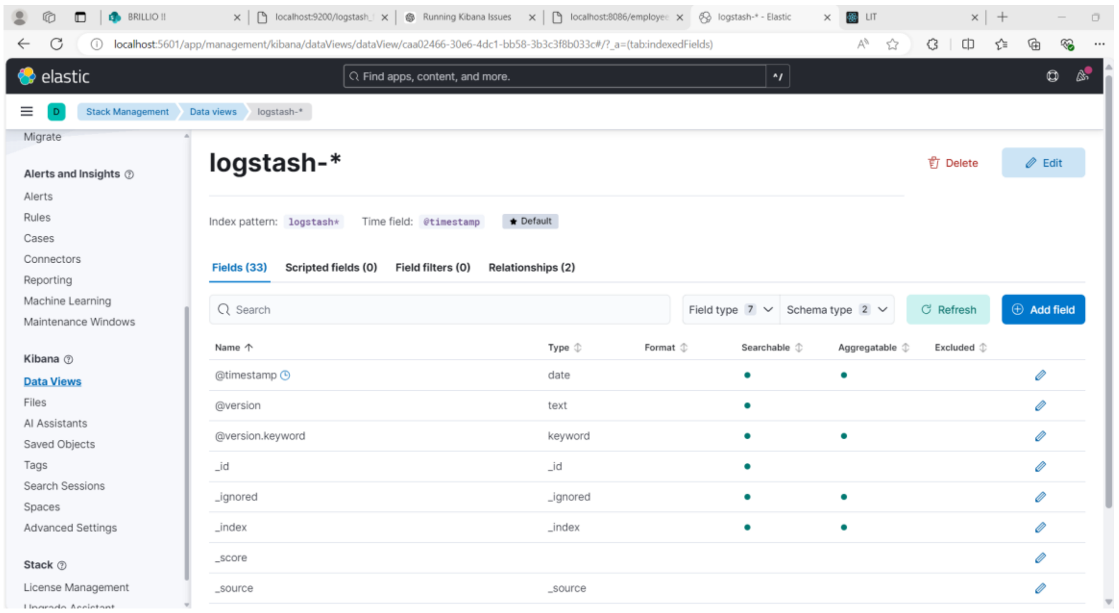


Data Views

Create and manage the data views that help you retrieve your data from Elasticsearch.

| Name | Spaces | Actions |
|---|---|---|
| logstash-* Default | D | 🗑 |
| .alerts-security.alerts-default,apm-*-transaction*,auditbeat-*,endgame-*,filebeat-*,logs-*,packetbeat-*,traces-apm*,winlogbeat-*,-*elastic-cloud-logs-* Security Data View | D | 🗑 |
| application_log | D | 🗑 |
| customer-logs | D | 🗑 |

- Click on the any of the data view eg: logstash_file which will show you the mapping(schema) of the respective index.



localhost:5601/app/management/kibana/dataViews/dataView/caa02466-30e6-4dc1-bb58-3b3c3f8b033c#/?_a=(tab:indexedFields)

Stack Management > Data views > logstash-*

# logstash-*

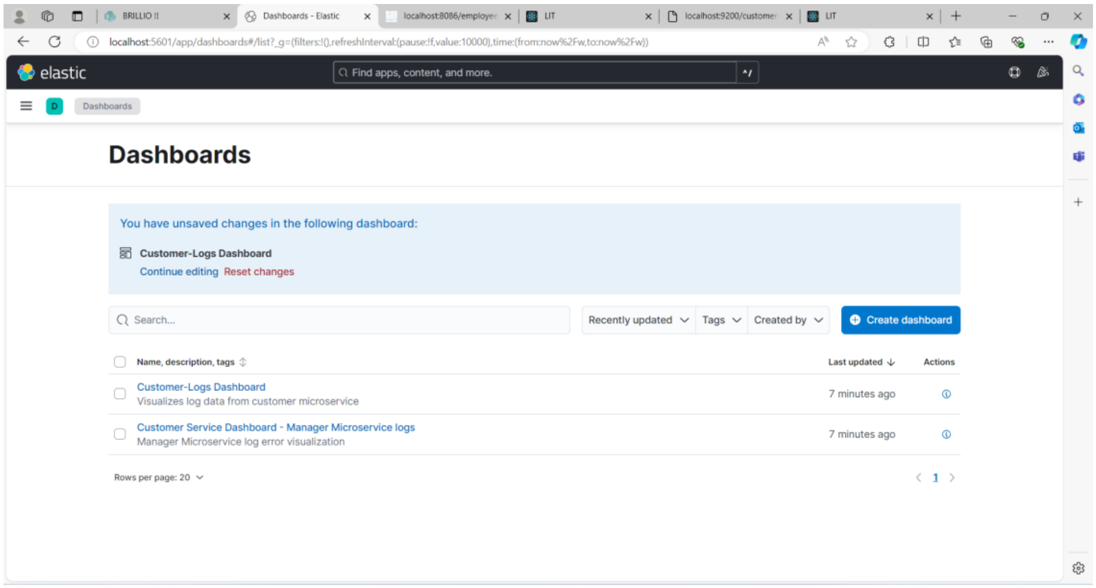
Index pattern: `logstash*` Time field: `@timestamp` ★ Default

Fields (33) | Scripted fields (0) | Field filters (0) | Relationships (2)

| Name | Type | Format | Searchable | Aggregatable | Excluded |
|---|---|---|---|---|---|
| @timestamp | date | | ● | ● | |
| @version | text | | ● | | |
| @version.keyword | keyword | | ● | ● | |
| _id | _id | | ● | | |
| _ignored | _ignored | | ● | ● | |
| _index | _index | | ● | ● | |
| _score | | | | | |
| _source | _source | | | | |

## 2. Steps to create Dashboard to visualize the log data

- Click on the **Hamburger** and Navigate to **Dashboard** under Discover tab.
- Click on the **Create Dashboards.**



localhost:5601/app/dashboards#/list?_g=(filters:!(),refreshInterval:(pause:!f,value:10000),time:(from:now%2Fw,to:now%2Fw))

# Dashboards

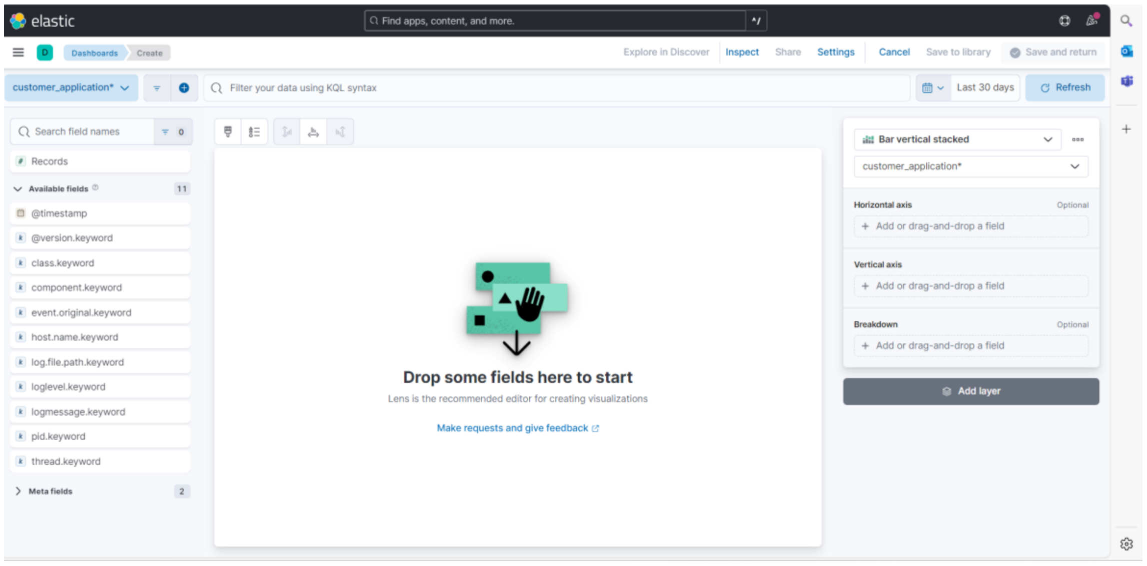
You have unsaved changes in the following dashboard:

**Customer-Logs Dashboard**
Continue editing Reset changes

| Name, description, tags | Last updated | Actions |
|---|---|---|
| Customer-Logs Dashboard<br>Visualizes log data from customer microservice | 7 minutes ago | ⓘ |
| Customer Service Dashboard - Manager Microservice logs<br>Manager Microservice log error visualization | 7 minutes ago | ⓘ |

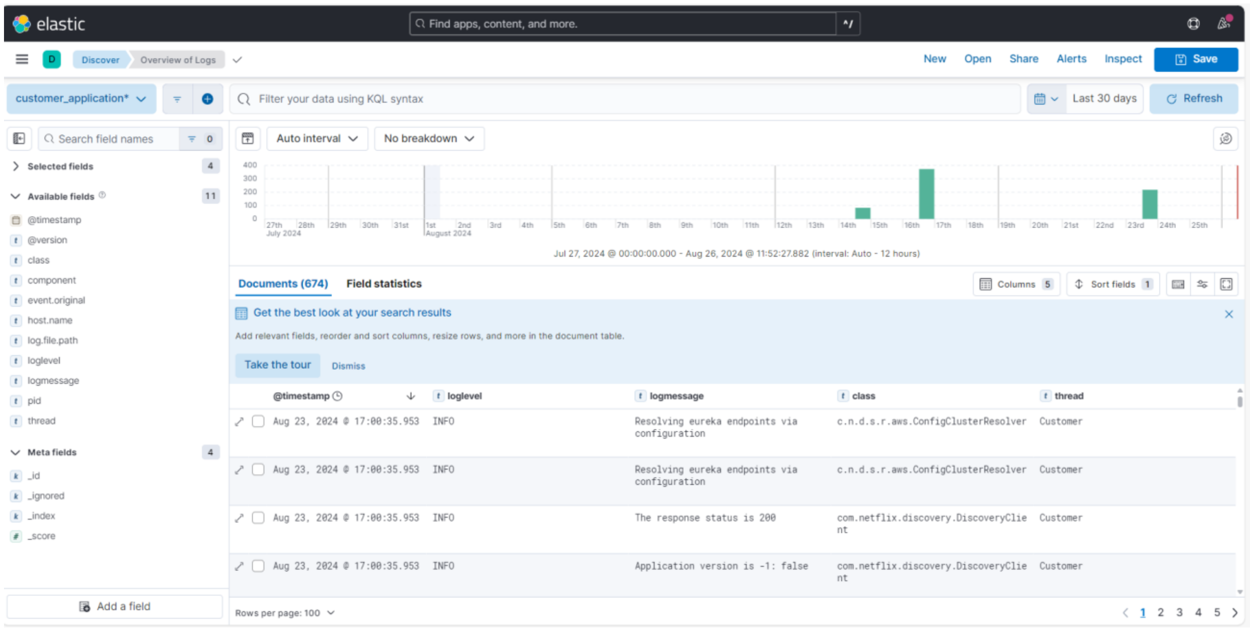
Rows per page: 20

- Click create visualization and it will navigate you the page which will look as shown in the picture.
- Here in **the left tab,** it will **display all the fields** from the selected **index** and in the **right tab** you can select **the type of visualization** as per your need to visualize the data**.**



## 3. Overview of the Logs

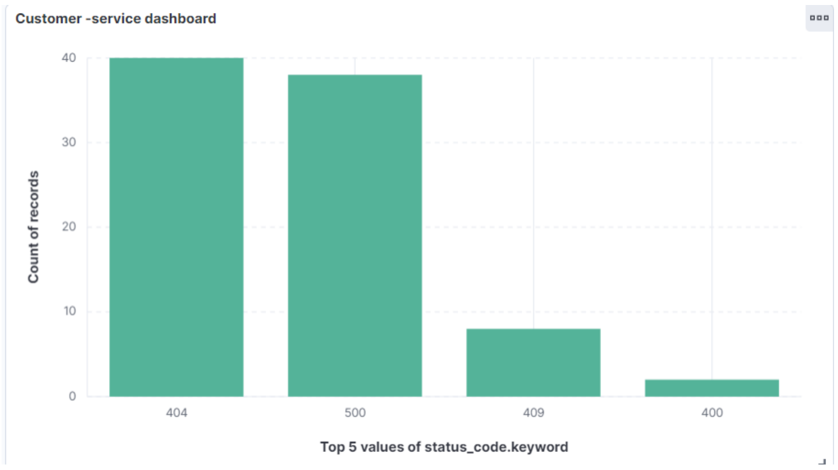
- Navigate to **Discover** in the **Analytics tab**
- The overview of the Logs for the selected index will be displayed as shown in the figure below



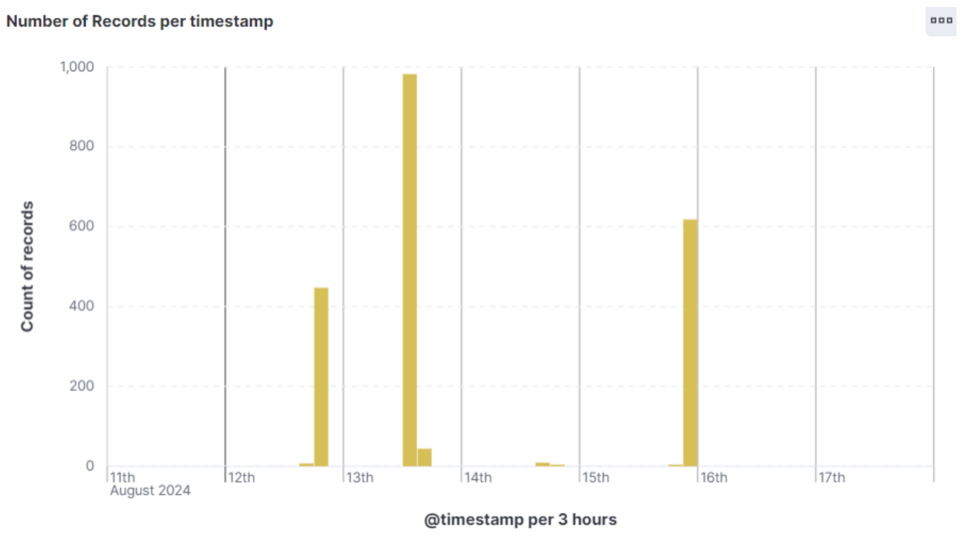
| @timestamp | loglevel | logmessage | class | thread |
|---|---|---|---|---|
| Aug 23, 2024 @ 17:00:35.953 | INFO | Resolving eureka endpoints via configuration | c.n.d.s.r.aws.ConfigClusterResolver | Customer |
| Aug 23, 2024 @ 17:00:35.953 | INFO | Resolving eureka endpoints via configuration | c.n.d.s.r.aws.ConfigClusterResolver | Customer |
| Aug 23, 2024 @ 17:00:35.953 | INFO | The response status is 200 | com.netflix.discovery.DiscoveryClient | Customer |
| Aug 23, 2024 @ 17:00:35.953 | INFO | Application version is -1: false | com.netflix.discovery.DiscoveryClient | Customer |

## Insights of Employee Microservice

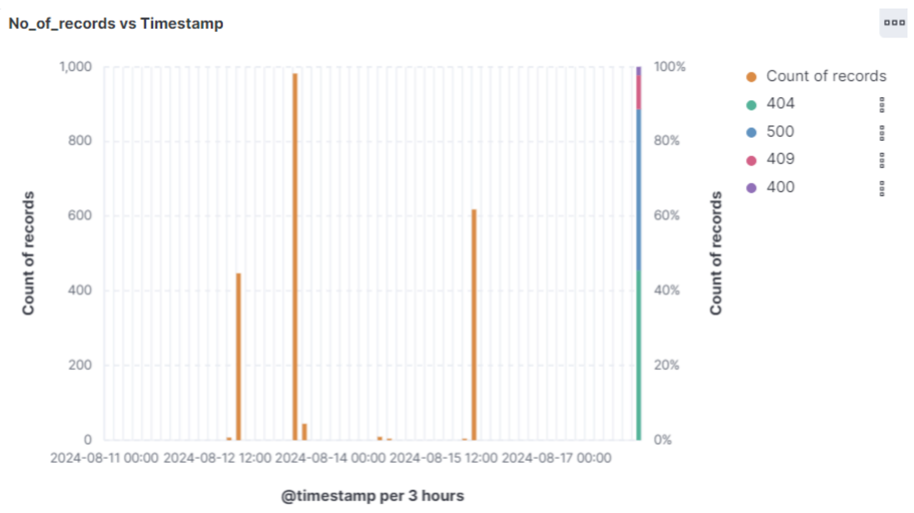
- Here I have created a **Vertical bar chart** which will display the Total counts of records with respect to the status code.
- Select the **Bar Vertical Stacked** and give:
  - ➢ Horizonatal axis: Top 5 values of status_code
  - ➢ Vertical axis: pid – No. of records in respective status_code



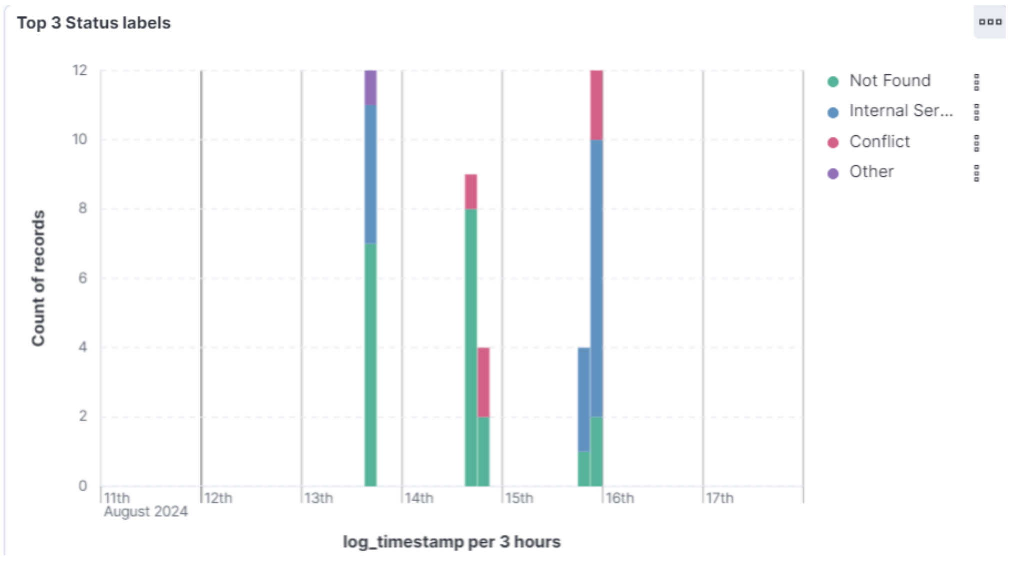
- Here I have created a **Vertical bar chart** which will display the Total counts of records with respect to the timestamp.
- Select the **Bar Vertical Stacked** and give:
  - ➢ Horizonatal axis: Timestamp (1 week)
  - ➢ Vertical axis: pid – No. of records in respective status_code



- Here I have created a **Vertical bar chart** which will display the Total counts of records with respect to the timestamp.
- Select the **Bar Vertical Stacked** and give:
    - ➢ Horizonatal axis: Timestamp
    - ➢ Vertical axis: No. of records in respective status_code along with the percentage of records in each status code(Bar vertical Percentage)

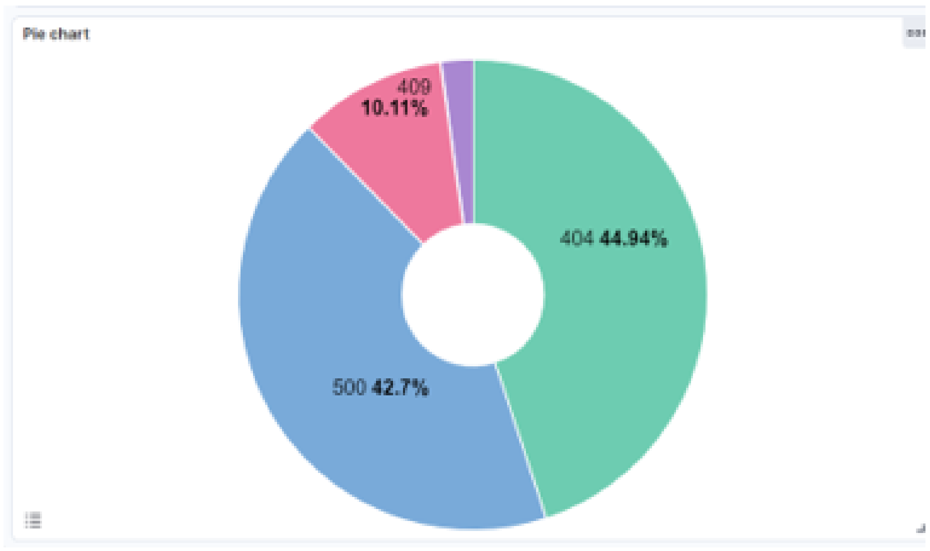


- Here I have created a **Vertical bar chart** which will display the Total counts of records with respect to the timestamp.
- Select the **Bar Vertical Stacked** and give:
    - ➢ Horizonatal axis: Timestamp
    - ➢ Vertical axis: pid – No. of records in given timeline categorized by the status_code

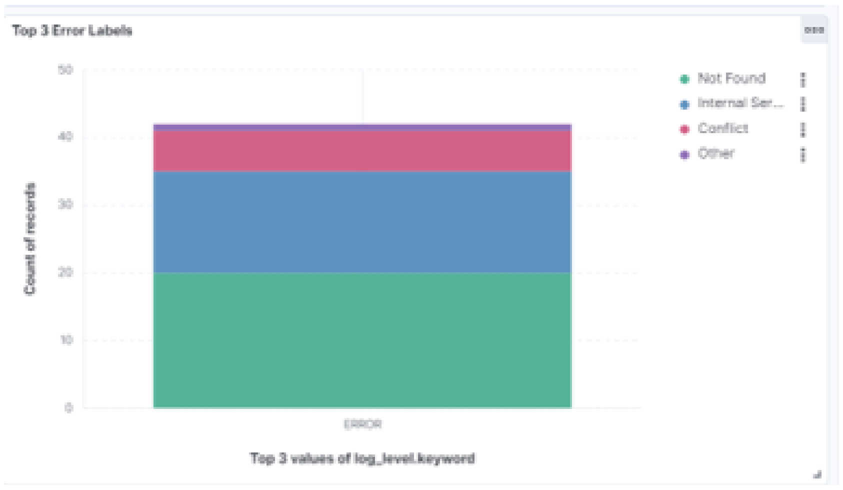


- Here I have created a **Donut chart** which will display the **percentage of logs** with respect to the **log level**.

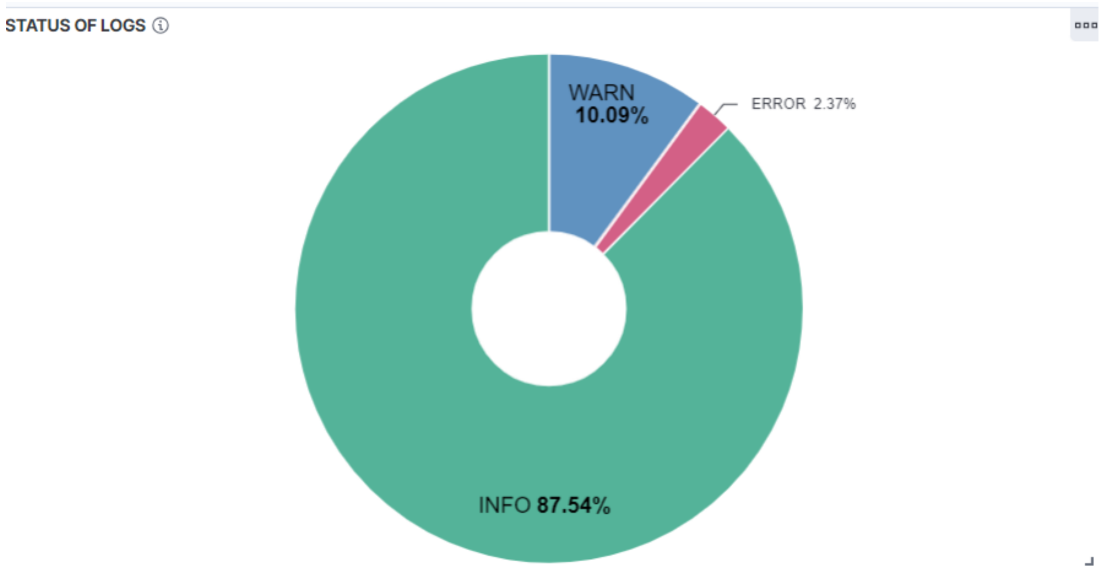
- Select the **Donut chart** and give:
    - Metrics: Status code



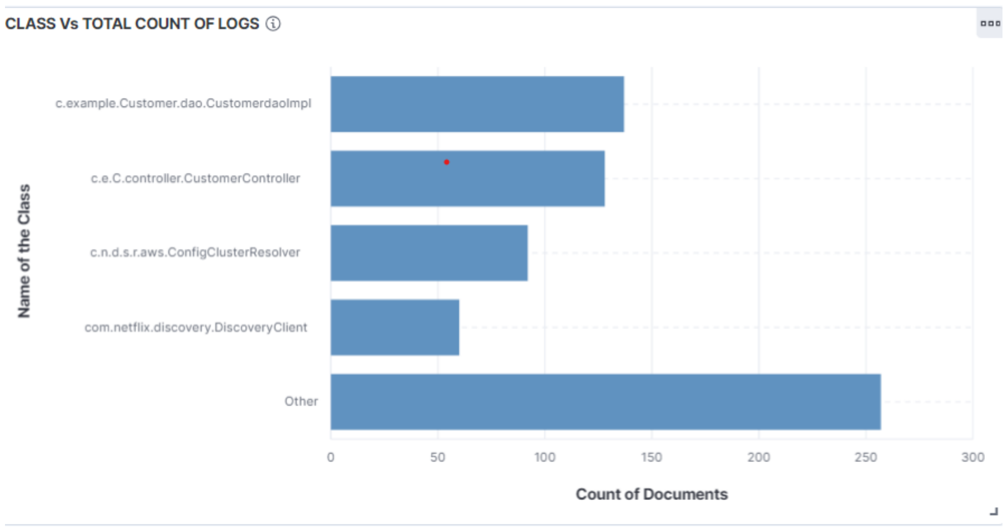
- Here I have created a **Donut chart** which will display the **percentage of logs** with respect to the **log level**.
- Select the **Donut chart** and give:
    - Metrics: Status label



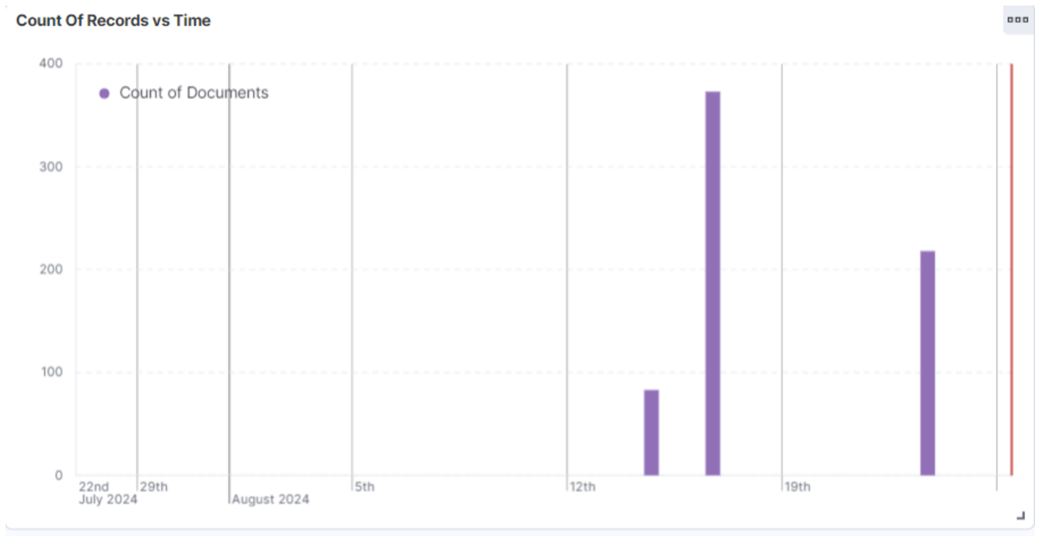
- Here I have created a **Donut chart** which will display the **percentage of logs** with respect to the **log level**.
- Select the **Donut chart** and give:
  - ➢ Metrics: pid – unique id for each document in the index



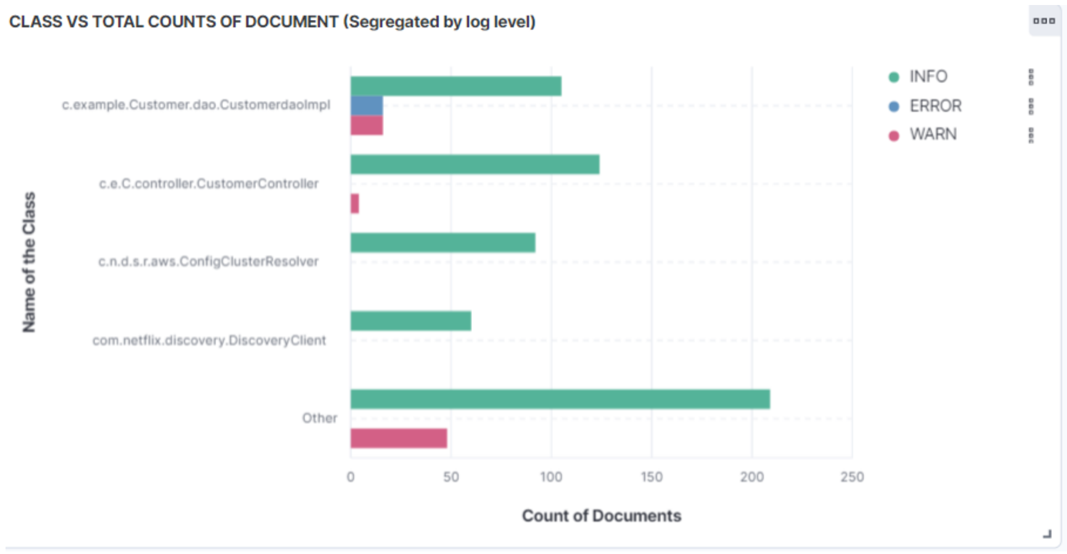
- Here I have created a **Horizontal bar chart** which will display the **Total number of logs** with respect to their **class**.(Only top 5 classess are displayed)
- Select the **Bar Horizontal** and give:
  - ➢ Vertical Axis: Class
  - ➢ Horizontal Axis: pid – unique id for each document in the index.



- Here I have created a **Vertical bar chart** which will display the **Total number of logs** with respect to timestamp.
- Select the **Bar Vertical** and give:
  - ➢ Vertical Axis: pid – unique id for each document in the index.
  - ➢ Horizontal Axis: timestamp



- Here I have created a **Horizontal bar chart** which will display the **Total number of logs** with respect to timestamp.
- Select the **Bar Horizontal** and give:
  - ➢ Vertical Axis: Class
  - ➢ Horizontal Axis: pid – unique id for each document in the index.
  - ➢ Breakdown: Log level


CLASS VS TOTAL COUNTS OF DOCUMENT (Segregated by log level)
Name of the Class
c.example.Customer.dao.Customerdaolmpl
c.e.C.controller.CustomerController
c.n.d.s.r.aws.ConfigClusterResolver
com.netflix.discovery.DiscoveryClient
Other
INFO
ERROR
WARN
Count of Documents