DAY 1

ELK Pre-requisite:
Java
Python (Because ELK has a lot of rest services, you can write your own apis and in industry python is a way to go)

Software Requirements:
Windows 10/11 or Mac
16 GB RAM
ELK
Anaconda Navigator
Postman software
VS Code
Rancher Desktop
Docker Desktop

Monolithic Architecture:
Client
Server

Microservices Architecture:
We dont use any central servers.
Dont write everything under one umbrella.
Actuator framework for Microsrvices

ELK is not only used for logging, metrics but is also used for NLP (AI) for faster results etc.
ELK is HA architecture.
ELK stack

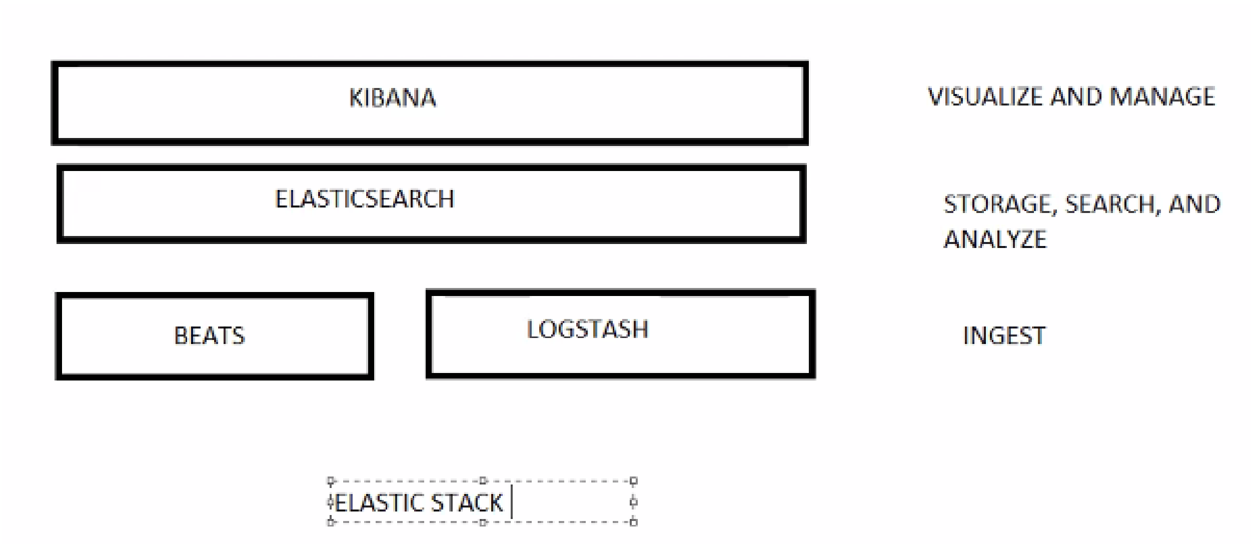
ELK: uses logstash
EFK: Uses Fluentd

Users can take any data from any source and perform searching, analyzing and creating visualizations.

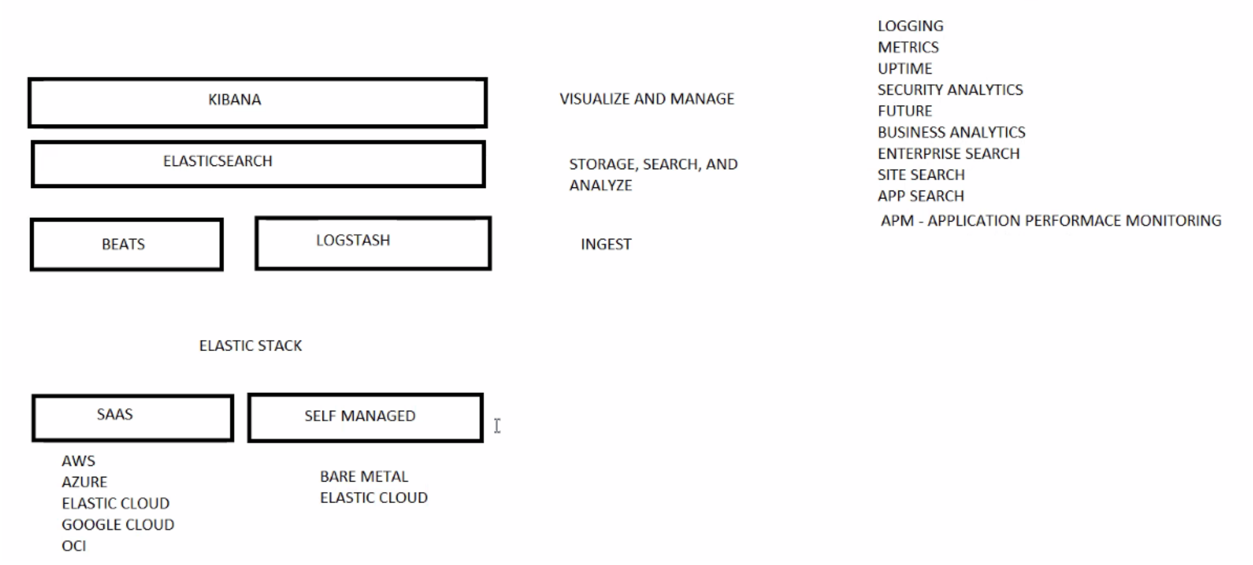
REST will help in understanding
Came in 2012 and from 2012 onwards people started using ELK

Components Of Elastic Stack:
1. ElasticSearch: Store the data (SQL, NO SQL, JSON, CSV). Most important and basic component of ELK. It is open source and distributed and supports
restful and JSON based.
2. Logstash: Can write data pipeline used to read the data from various sources and write data into different target data sources.
3. Kibana: GUI used to visualize the data (Analytics, dashboards)
4. Beats: Lightweight data shipper on every target machine and sends data to elastic search directly or via Logstash.

Apache Lucene: High performance text searching engine written in java.

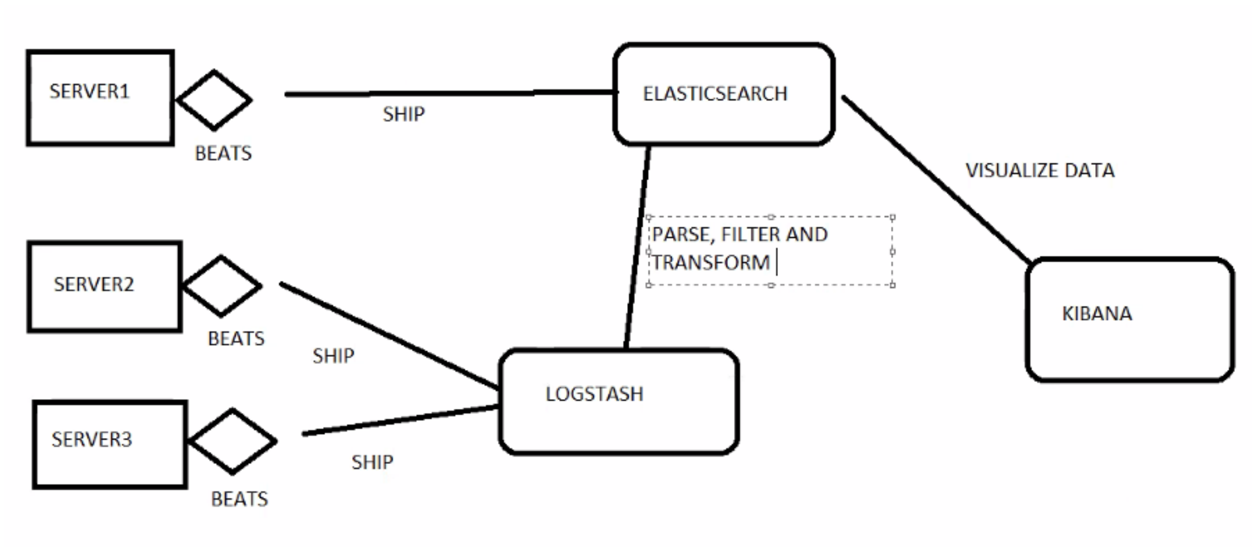


Services that ELK offers:
Logging, Metrics, Uptime, Security Analytics, Future, Business Analytics, Enterprise Search, Site search, App search, APM



Two ways to send data:

1. Server sending directly to Elastic Search and display on Kibana
2. Server sends data to elastic search via Logstash and visualize data on kibana.



We can write our own API for our data pipeline flow, no need for LogStash, just send it directly to Elastic Search. Logstash parses, filters and transforms the data and sends it but if you already have something that can parse, filter or transform the data then Logstash can be skipped.

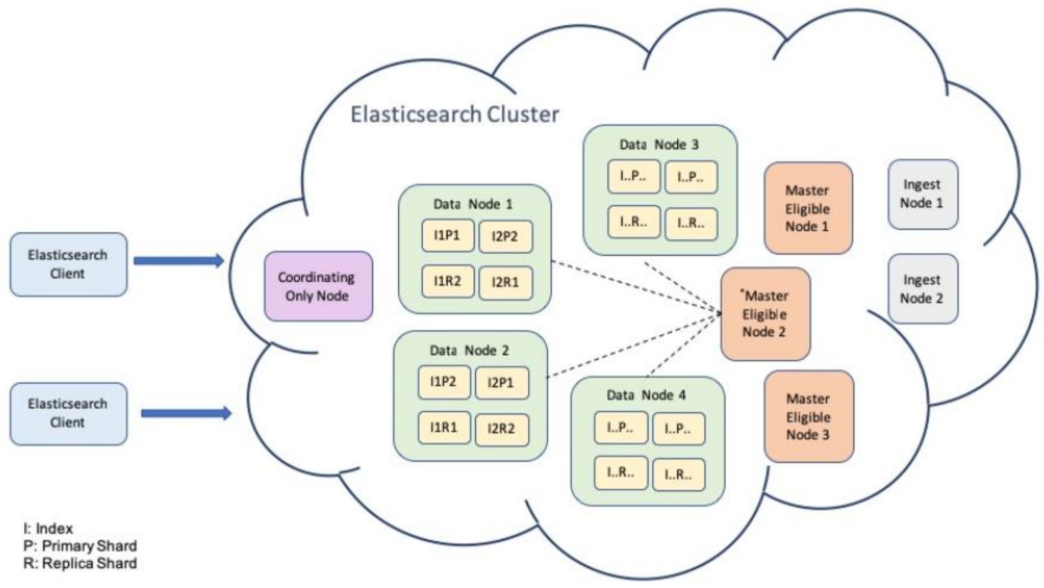
ElasticSearch:
Full text search engine. NO SQL DB. Can be used as analytics engine. Schema less and works in real time. Written in java rest APIs. 12 factor app design. Structured and unstructured data can be used
Kibana:
It's a GUI used to do multiple activities. It discovers your data by exploring. It can analyze the data and implement the metrics. It can visualize the data. It can apply ML on the data (forecasting, anomalies, etc). It can do APM. It can Manage roles and users. Elastic search expressions. It can analyze time series data using the timeline.

LogStash:
It's a data pipeline. Push the data into Elasticsearch. You can take any type of data using Logstash. It parses, filters and transforms the data.

Beats:
It's a lightweight software agent. It can talk to logstash or Elasticsearch directly. There are different beats such as for network, windows, etc we have different beats agent (file beat, winlog beat packet beat).

X-pack:
It is a security extension. Elastic stack extension to provide security. It provides secured Alerting, secured monitoring, secured reporting, and secured ML. It is not free.

An Elasticsearch cluster is a group of one or more Elasticsearch nodes that are connected together. Let's first outline how it is laid out, as shown in the following diagram:



Data node: Contains the data that contains indexed documents. Performs CRUD operation.
Ingest Node: One of the ways to process the document in your pipeline mode before indexing the document.
Disable it using node.ingest=false in elasticsearch.yaml file

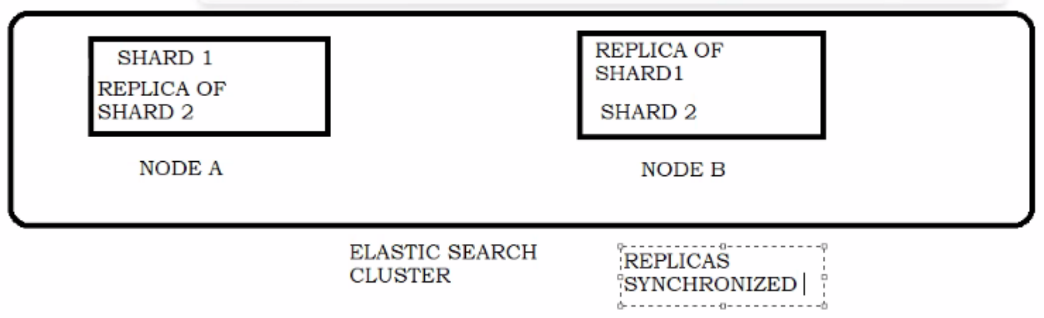
Coordinating Only Node: Disable data node ingest node. At such time they can only route the request and distribute work using a bulk indexing mechanism.

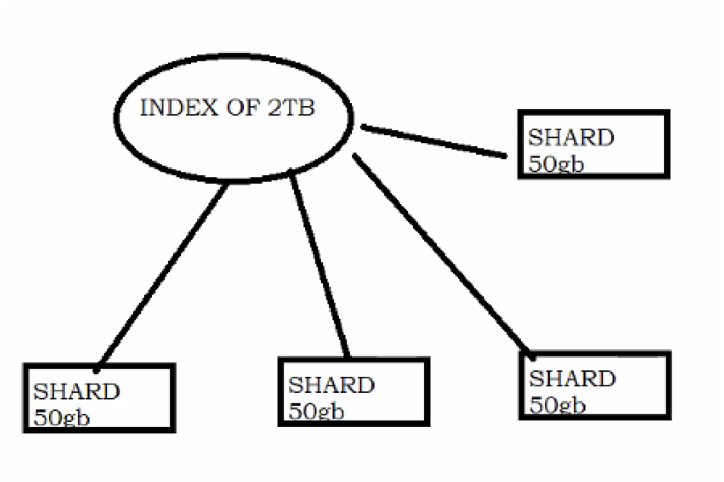
Disabling the different components will tell elastic search what role is it playing.

Fields: Smallest individual unit of data.
Multifields: Can be indexed in more than one way to produce more results.

Shards: Single Lucene index
Index size is a major cause of elastic search collapse. Indexing can begin to fail. Therefore, shards are heavily used.
Split the indexes horizontally into pieces. Helps in distributed environment and will have multiple nodes.

Replica: Copies of indexes. Replicas are not placed on the same nodes as shards.





Sharding: Improves performance, Provides high availability and fault tolerance.

Installation:

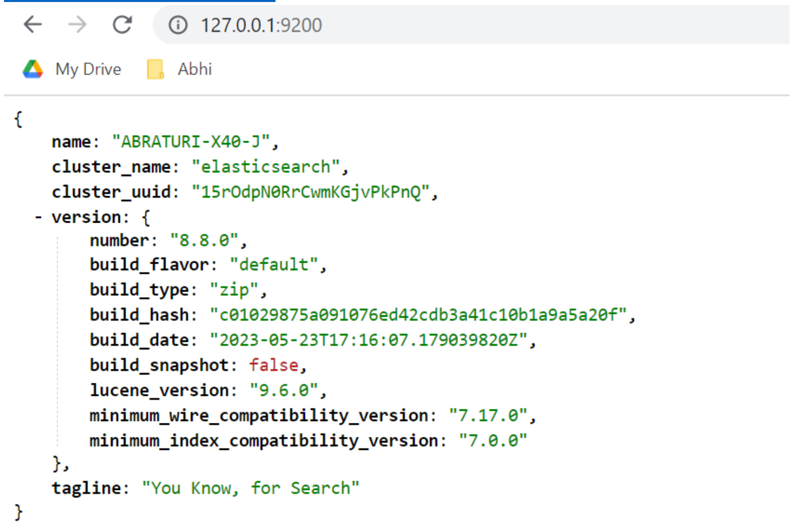
Download the zip packages for all three components from here:
https://www.elastic.co/downloads/elasticsearch
https://www.elastic.co/downloads/logstash
https://www.elastic.co/downloads/kibana

Now create a folder ELK and put all three zip files under this and extract all three.

Install jdk if not already done.
Start the elastic search using below:
C:\Users\abraturi\Desktop\ELK\elasticsearch-8.8.0\bin\elasticearch.bat

Edit the below file to disable the security feature: xpack.security.enabled: false
C:\Users\abraturi\Desktop\ELK\elasticsearch-8.8.0\config\elasticsearch.yml

Now restart the Elasticsearch
Hit the URL in your browser to see the information (if prompts for credentials appear just cancel it):
http://127.0.0.1:9200/



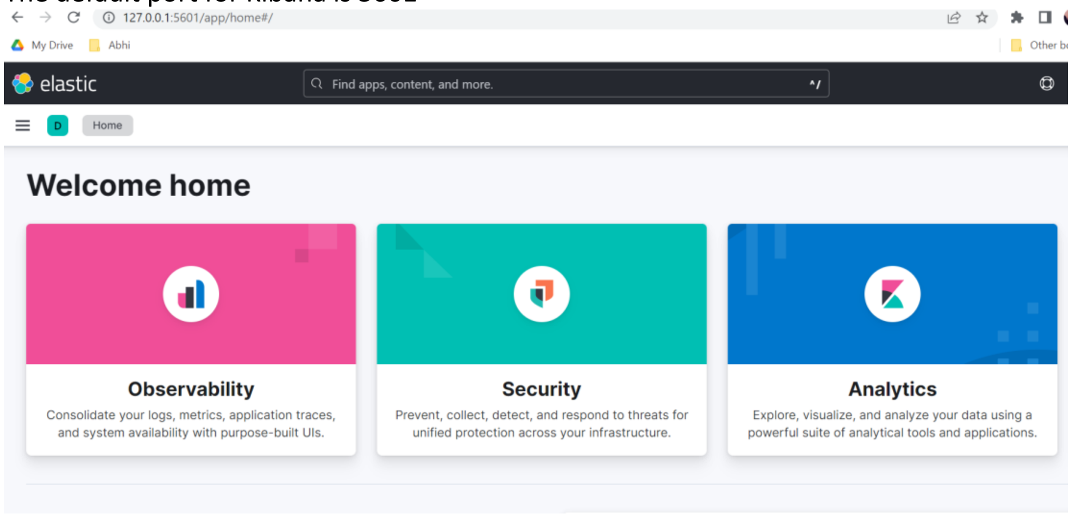
```
{
    name: "ABRATURI-X40-J",
    cluster_name: "elasticsearch",
    cluster_uuid: "15rOdpN0RrCwmKGjvPkPnQ",
  - version: {
        number: "8.8.0",
        build_flavor: "default",
        build_type: "zip",
        build_hash: "c01029875a091076ed42cdb3a41c10b1a9a5a20f",
        build_date: "2023-05-23T17:16:07.179039820Z",
        build_snapshot: false,
        lucene_version: "9.6.0",
        minimum_wire_compatibility_version: "7.17.0",
        minimum_index_compatibility_version: "7.0.0"
    },
    tagline: "You Know, for Search"
}
```

You can also use Postman to send the request and you will see the json results similar to what we saw in the browser.

Now start Kibana:

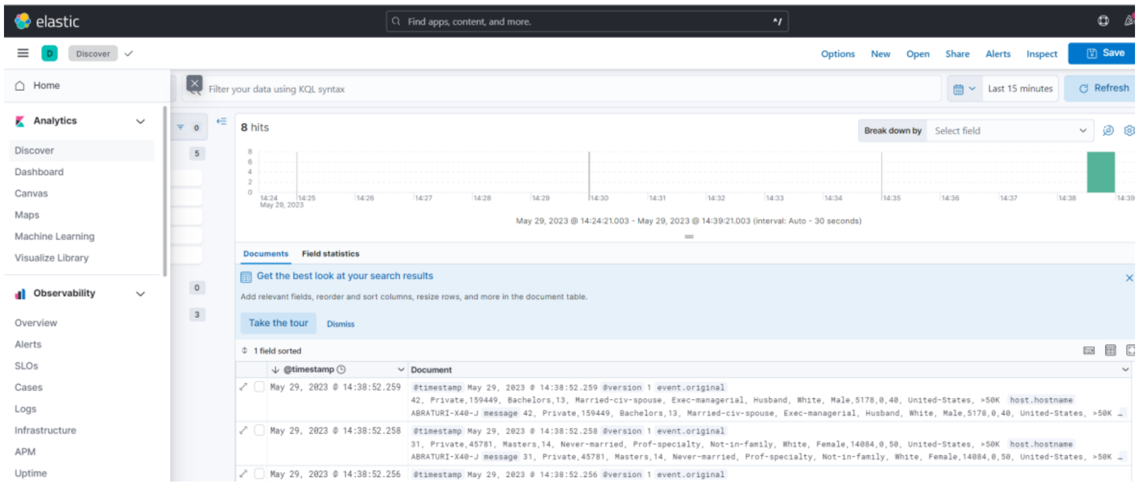
C:\Users\abraturi\Desktop\ELK\kibana-8.8.0\bin\kibana.bat

The default port for Kibana is 5601


127.0.0.1:5601/app/home#/
My Drive
Abhi
elastic
Find apps, content, and more.
Home
Welcome home
Observability
Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.
Security
Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.
Analytics
Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

Now create a file logstash.conf under path: C:\Users\abraturi\Desktop\ELK\logstash-8.8.0\logstash.conf

Logstash.conf:
```
input {
  stdin {}
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "indexforlogstash"
  }
}
```
Now go to directory: C:\Users\abraturi\Desktop\ELK\logstash-8.8.0\bin
Once done execute the below command: .\logstash.bat -f .\logstash.conf
Now in this same terminal or cmd provide the data by just copying and pasting.
Check the data in Kibana:
Go to discover:

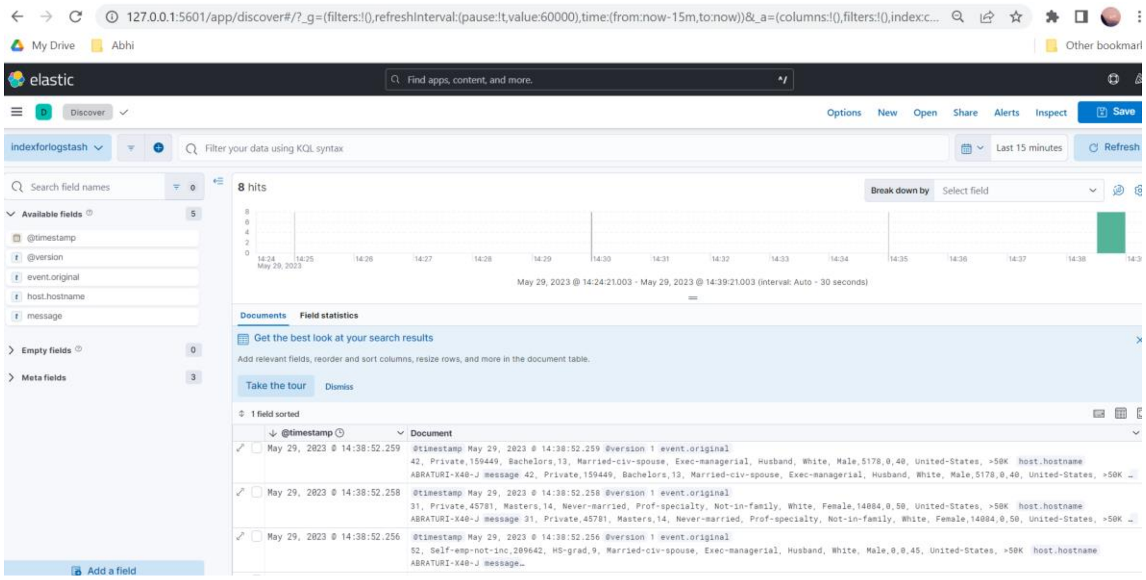


May 29, 2023 @ 14:24:21.003 - May 29, 2023 @ 14:39:21.003 (interval: Auto - 30 seconds)

```
@timestamp                  Document
May 29, 2023 @ 14:38:52.259 @timestamp May 29, 2023 @ 14:38:52.259 @version 1 event.original
                            42, Private,159449, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White, Male,5178,0,40, United-States, >50K host.hostname
                            ABRATURI-X40-J message 42, Private,159449, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White, Male,5178,0,40, United-States, >50K …
May 29, 2023 @ 14:38:52.258 @timestamp May 29, 2023 @ 14:38:52.258 @version 1 event.original
                            31, Private,45781, Masters,14, Never-married, Prof-specialty, Not-in-family, White, Female,14084,0,50, United-States, >50K host.hostname
                            ABRATURI-X40-J message 31, Private,45781, Masters,14, Never-married, Prof-specialty, Not-in-family, White, Female,14084,0,50, United-States, >50K …
May 29, 2023 @ 14:38:52.256 @timestamp May 29, 2023 @ 14:38:52.256 @version 1 event.original
```

For the first time provide the search name and indexname (“indexforlogstash” in this case) and click on create search.

You will start seeing the data:


127.0.0.1:5601/app/discover#/?_g=(filters:!(),refreshInterval:(pause:!t,value:60000),time:(from:now-15m,to:now))&_a=(columns:!(),filters:!(),index:c...
elastic
Find apps, content, and more.
Discover
Options
New
Open
Share
Alerts
Inspect
Save
indexforlogstash
Filter your data using KQL syntax
Last 15 minutes
Refresh
Search field names
Available fields
@timestamp
@version
event.original
host.hostname
message
Empty fields
Meta fields
Add a field
8 hits
Break down by
Select field
May 29, 2023 @ 14:24:21.003 - May 29, 2023 @ 14:39:21.003 (interval: Auto - 30 seconds)
Documents
Field statistics
Get the best look at your search results
Add relevant fields, reorder and sort columns, resize rows, and more in the document table.
Take the tour
Dismiss
1 field sorted
@timestamp
Document

```
May 29, 2023 @ 14:38:52.259   @timestamp May 29, 2023 @ 14:38:52.259 @version 1 event.original
                              42, Private,159449, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White, Male,5178,0,40, United-States, >50K host.hostname
                              ABRATURI-X40-J message 42, Private,159449, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White, Male,5178,0,40, United-States, >50K …
May 29, 2023 @ 14:38:52.258   @timestamp May 29, 2023 @ 14:38:52.258 @version 1 event.original
                              31, Private,45781, Masters,14, Never-married, Prof-specialty, Not-in-family, White, Female,14084,0,50, United-States, >50K host.hostname
                              ABRATURI-X40-J message 31, Private,45781, Masters,14, Never-married, Prof-specialty, Not-in-family, White, Female,14084,0,50, United-States, >50K …
May 29, 2023 @ 14:38:52.256   @timestamp May 29, 2023 @ 14:38:52.256 @version 1 event.original
                              52, Self-emp-not-inc,209642, HS-grad,9, Married-civ-spouse, Exec-managerial, Husband, White, Male,0,0,45, United-States, >50K host.hostname
                              ABRATURI-X40-J message…
```

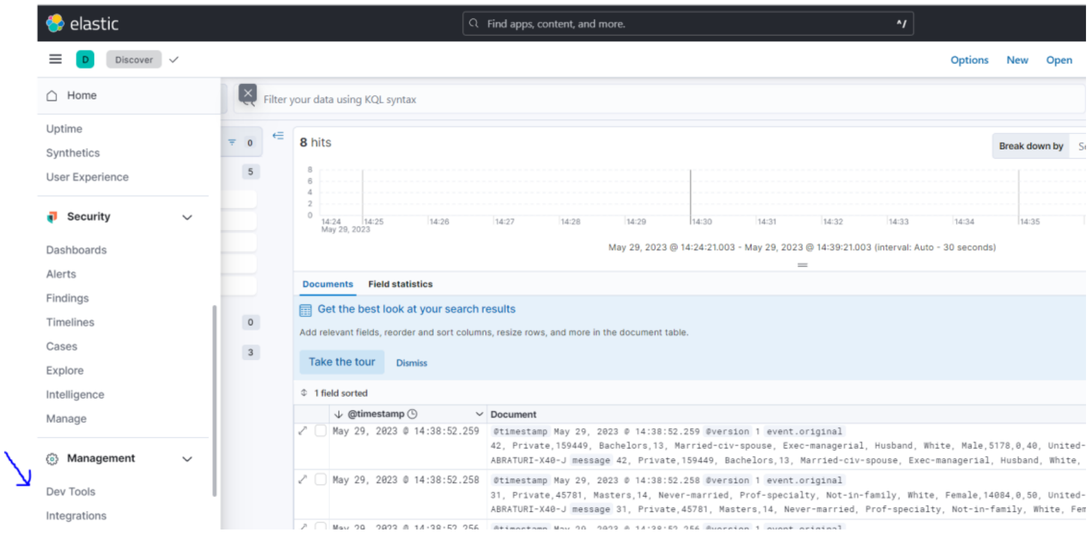
If you want to change the data path and log path then update elasticsearch.yaml with the below and restart elasticsearch server:

```
#admin defined configuration changes

path:
  data : "C:\\Users\\abraturi\\Desktop\\ELK\\elasticsearch-8.8.0\\data"
  logs : "C:\\Users\\abraturi\\Desktop\\ELK\\elasticsearch-8.8.0\\logs"
```

Start the elasticsearch:

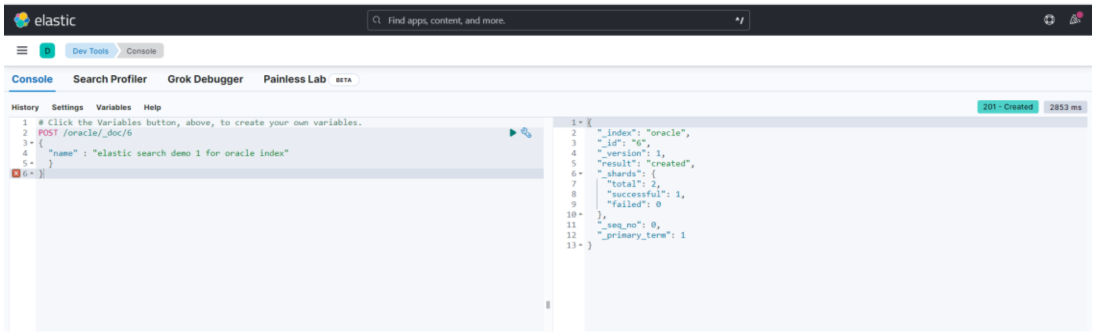
```
C:\Users\\abraturi\Desktop\ELK\elasticsearch-8.8.0\bin\elasticsearch
```

You can send the data using Postman or from devtools in kibana.


elastic
Find apps, content, and more.
Discover
Options
New
Open
Home
Uptime
Synthetics
User Experience
Security
Dashboards
Alerts
Findings
Timelines
Cases
Explore
Intelligence
Manage
Management
Dev Tools
Integrations
Filter your data using KQL syntax
8 hits
Break down by
May 29, 2023 @ 14:24:21.003 - May 29, 2023 @ 14:39:21.003 (interval: Auto - 30 seconds)
Documents
Field statistics
Get the best look at your search results
Add relevant fields, reorder and sort columns, resize rows, and more in the document table.
Take the tour
Dismiss
1 field sorted
@timestamp
Document

```
May 29, 2023 @ 14:38:52.259   @timestamp May 29, 2023 @ 14:38:52.259 @version 1 event.original
                              42, Private,159449, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White, Male,5178,0,40, United-
                              ABRATURI-X40-J message 42, Private,159449, Bachelors,13, Married-civ-spouse, Exec-managerial, Husband, White,
May 29, 2023 @ 14:38:52.258   @timestamp May 29, 2023 @ 14:38:52.258 @version 1 event.original
                              31, Private,45781, Masters,14, Never-married, Prof-specialty, Not-in-family, White, Female,14084,0,50, United-
                              ABRATURI-X40-J message 31, Private,45781, Masters,14, Never-married, Prof-specialty, Not-in-family, White, Fem
```

Now, post the query and run

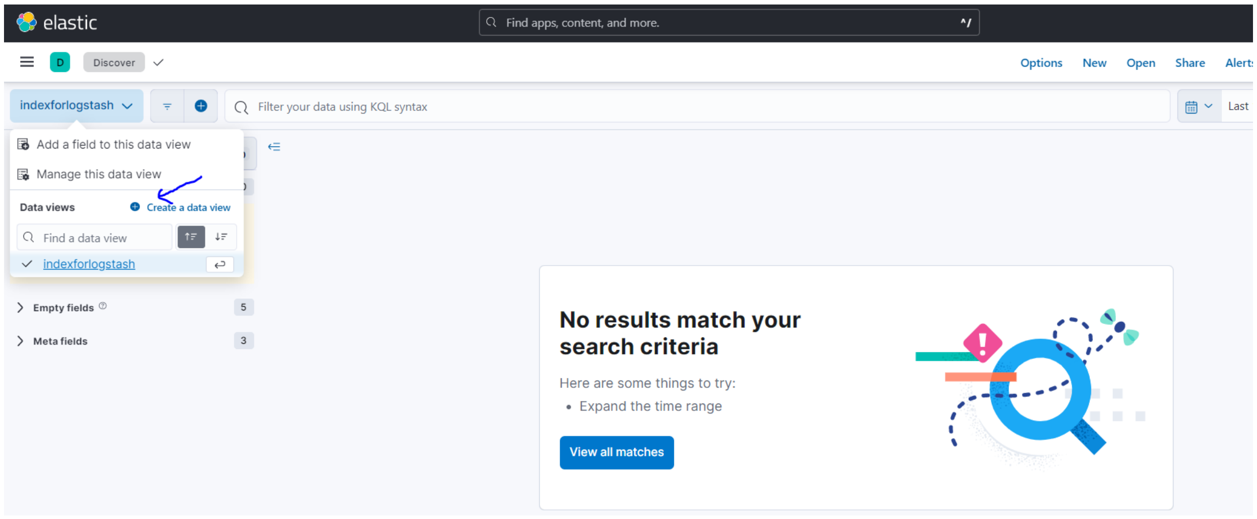


```
# Click the Variables button, above, to create your own variables.
POST /oracle/_doc/6
{
  "name" : "elastic search demo 1 for oracle index"
  }
}
```

```
{
  "_index": "oracle",
  "_id": "6",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 0,
  "_primary_term": 1
}
```

201 - Created 2853 ms

Now to check the logs sent via the POST method, create another data view that will see the logs in the index "oracle"

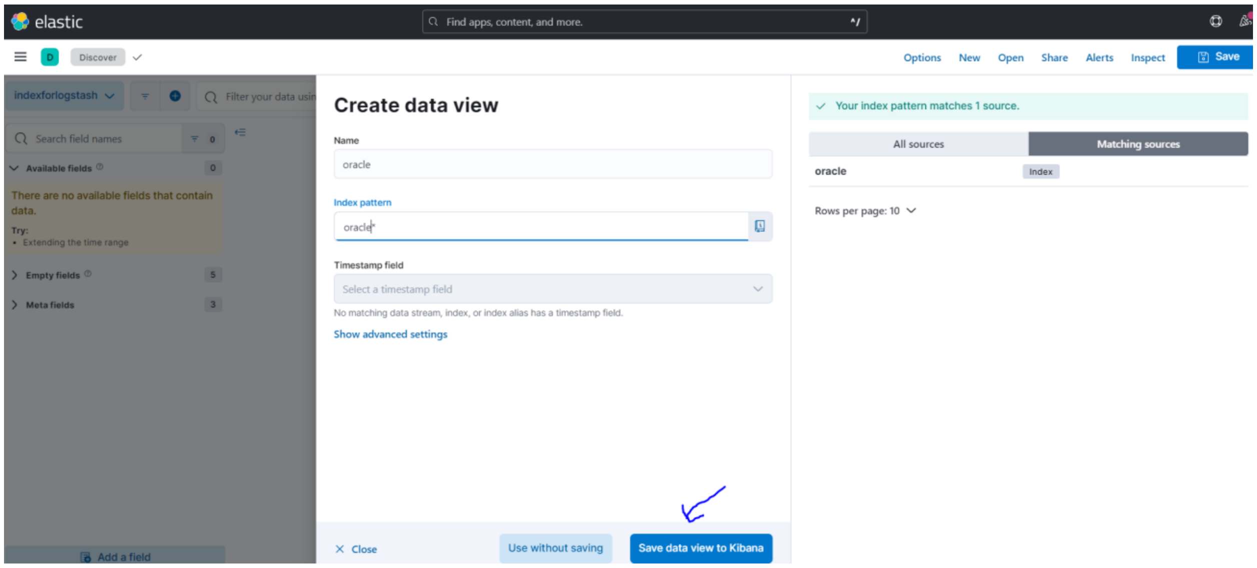


No results match your search criteria

Here are some things to try:

- Expand the time range

Fill the details and click on "Save data view"



Create data view

Name: oracle

Index pattern: oracle*

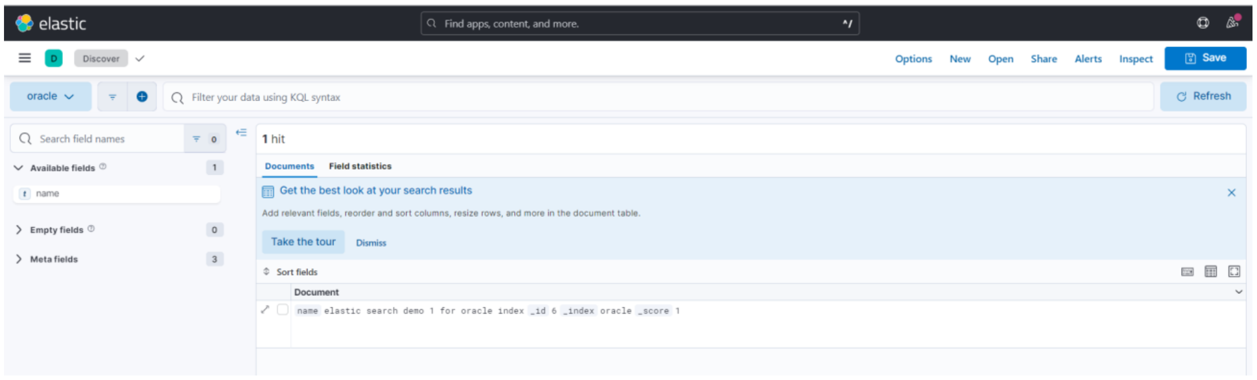
Timestamp field: Select a timestamp field

No matching data stream, index, or index alias has a timestamp field.

Your index pattern matches 1 source.

oracle — Index

You will see the data:



```
name elastic search demo 1 for oracle index _id 6 _index oracle _score 1
```

Performing Operations using curl command:

Create index:

```
curl --location --request PUT “http://localhost:9200/cf_etf”
```

Status of index:

```
curl --location --head http://localhost:9200/cf_etf
```

Get details of an index:

```
curl --location --request GET http://localhost:9200/cf_etf
```

Get details of an index:

```
curl --location --request GET http://localhost:9200/indexforlogstash
```

Delete an index:

```
curl --location --request DELETE  http://localhost:9200/cf_etf
```

Close the index and it will not show in kibana (GUI)

```
curl --location --request POST http://localhost:9200/cf_etf/_close
```

Open the index and it show in kibana (GUI)

```
curl --location --request POST http://localhost:9200/customer/_open
```

You can also change the content by REST calls:

```
curl --location --request PUT "http://localhost:9200/customer/_settings" --header "Content-Type:
application/json" --data "{ \"index\" : { \"number_of_replicas\" : 5 }}"
```

```
curl --location --request GET http://localhost:9200/customer/_settings
```

Creating index template using Postman (REST calls):

## Create customer index template

~~~
```
http://localhost:9200/_template/customer_template
PUT
Header - content-type - application/json

BODY

{
  "index_patterns" : ["customer*"],
  "settings" : {
    "number_of_replicas" : 3,
    "codec" : "best_compression"
  }
~~~

}
```

get customer index template
```
http://localhost:9200/_template/customer_template
GET
Header - content-type - application/json
```

check if exists customer index template
```
http://localhost:9200/_template/customer_template
HEAD
Header - content-type - application/json
```

check if exists customer index template
```
http://localhost:9200/_template/customer_template
DELETE
Header - content-type - application/json
```

DAY 2

Alias points to the main object itself. It is used to do pointing capability. You cannot do reading and writing simultaneously in elasticsearch (cannot do on a single object).

Creating an index:
```
http://localhost:9200/oracle1

PUT
Header - content-type - application/json
```

Creating alias for index "oracle":
```
http://localhost:9200/oracle/_alias/oracle__alias

PUT
Header - content-type - application/json
```

Change alias from one index to another index with zero downtime:
```
http://localhost:9200/_aliases

POST
Header - content-type - application/json
```
```

```
```
BODY
{
  "actions" : [
    {"remove" : {"index" : "cf_etf", "alias" : "cf_etf_production"}},
    {"add" : {"index" : "cf_etf_new", "alias" : "cf_etf_production"}}
  ]
}
```

Get the alias details for all indexes:
```
http://localhost:9200/oracle/_alias
GET
```

Get the alias details of all indexes that has alias "oracle__alias":
```
http://localhost:9200/*/_alias/oracle__alias
GET
```
Creating a view:
```
http://localhost:9200/cf_view
PUT
```

```
BODY
{
    "mappings" : {
        "properties" : {
            "symbol" : {"type" : "keyword"},
            "category" : {"type" : "keyword"}
            }
        }
}

```

Creating alias with filter:
```
http://localhost:9200/_aliases

POST
```
```
BODY
{
    "actions" : [
        {"add" : {"index" : "cf_view", "alias" : "cf_view_international", "filter" : {"term" :
 {"category" : "International"}}}}

    ]
}
```

Get the search details from an index:
```
http://localhost:9200/cf_view_international/_search
GET
```

Get the full statistics details for an index:
```
http://localhost:9200/cf_view/_stats
GET
```

Get the segment details for an index:
```
http://localhost:9200/cf_view/_segments
GET
```

Get the shard details for an index:
```
http://localhost:9200/cf_view/_shard_stores
GET
```

Changing the settings of index:
```
http://localhost:9200/cf_etf/_settings
PUT
```
```
BODY
{
    "settings" : {
        "index.blocks.write" : true

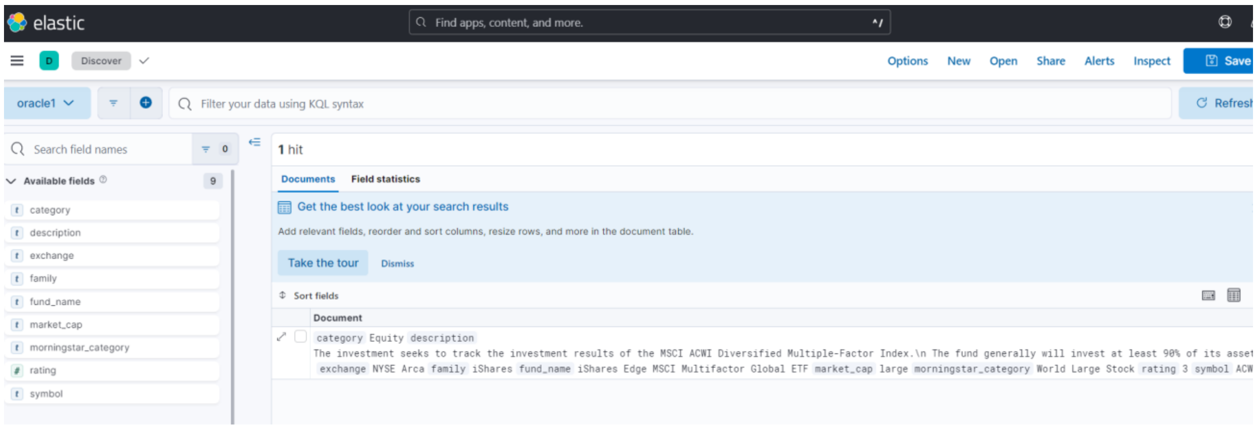
    }
}

```

Create the shards of an index:
```
http://localhost:9200/cf_etf/_split/cf_etf_split
POST
```
{
    "settings" : {
        "index.number_of_shards" : 4

    }
}
```

**Creating the document and sending the data (Using Postman):**

```
http://localhost:9200/cf_etf/_doc/1
PUT
```

```
{
 "symbol": "ACWF",
 "fund_name": "iShares Edge MSCI Multifactor Global ETF",
 "rating": 3,
 "morningstar_category": "World Large Stock",
 "category": "Equity",
 "family": "iShares",
 "market_cap": "large",
 "description": "The investment seeks to track the investment results of the MSCI ACWI Diversified Multiple-Factor Index.\\n The fund generally will invest at least 90% of its assets in the component securities of the underlying index and in investments that have economic characteristics that are substantially identical to the component securities of the underlying index. The underlying index is designed to contain equity securities from the MSCI ACWI Index (the \\\"parent index\\\") that have high exposure to four investment style factors: value, quality, momentum and low size, while maintaining a level of risk similar to that of the parent index.",
 "exchange": "NYSE Arca"
}
```



1 hit

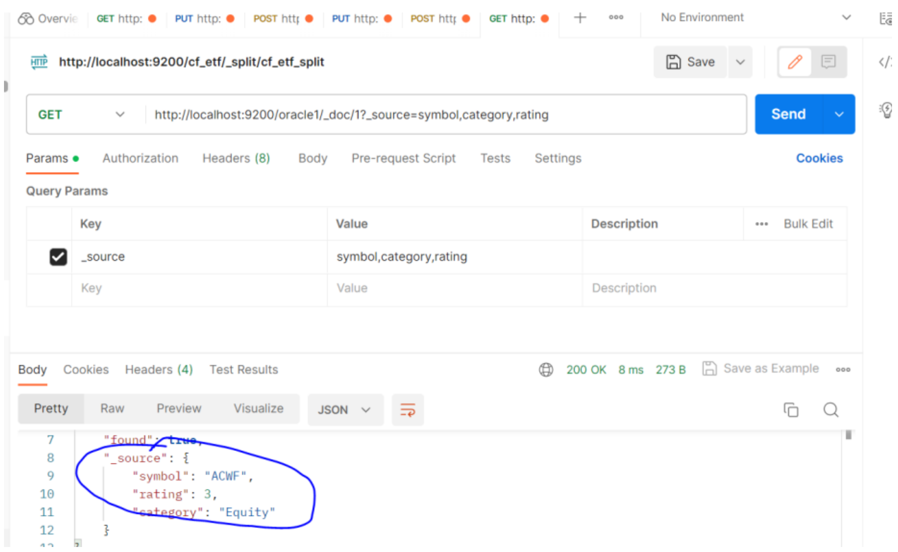
Get the best look at your search results

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

```
Document
category Equity description
The investment seeks to track the investment results of the MSCI ACWI Diversified Multiple-Factor Index.\n The fund generally will invest at least 90% of its asset
exchange NYSE Arca family iShares fund_name iShares Edge MSCI Multifactor Global ETF market_cap large morningstar_category World Large Stock rating 3 symbol ACW
```

Getting the selected fields only from the data using the partial search:

```
http://localhost:9200/oracle1/_doc/1?_source=symbol,category,rating
GET
```



http://localhost:9200/cf_etf/_split/cf_etf_split

GET http://localhost:9200/oracle1/_doc/1?_source=symbol,category,rating

Query Params

	Key	Value	Description
✓	_source	symbol,category,rating	
	Key	Value	Description

```
    "found": true,
    "_source": {
        "symbol": "ACWF",
        "rating": 3,
        "category": "Equity"
    }
```

Update the selected fields only in the data:

```
http://localhost:9200/oracle1/_update/1
POST
```

```
BODY
{
    "doc" : {
        "market_cap" : "mid"
    }
}

```

Multiple Documents:

Get multiple docs:

```
http://localhost:9200/_mget
POST
```

```
BODY
{
    "docs" : [
        { "_index" : "cf_view", "_id" : "1", "_source" : ["symbol","category"]}
}

```

Get docs by id:
```
http://localhost:9200/_mget
POST
```
```
BODY
{
    "docs" : [
        { "_index" : "cf_view", "_id" : "1", "_source" : ["symbol","category"]},
        { "_index" : "cf_etf", "_id" : "1", "_source" : ["symbol","category"]}
    ]
}

```

Perform bulk actions:
```
http://localhost:9200/_bulk
POST
```
```
BODY
{"delete": {"_index":"cf_etf", "_id":"1"}}
{"create": {"_index":"cf_etf", "_id":"1"}}
{"symbol": "ACWF", "fund_name": "iShares Edge MSCI Multifactor Global ETF", "rating": 3,
"morningstar_category": "World Large Stock", "category": "Equity", "family": "iShares",
"market_cap": "large", "description": "The investment seeks to track the investment results of
the MSCI ACWI Diversified Multiple-Factor Index.\\n The fund generally will invest at least
90% of its assets in the component securities of the underlying index and in investments that
have economic characteristics that are substantially identical to the component securities of
the underlying index. The underlying index is designed to contain equity securities from the
MSCI ACWI Index (the \\\"parent index\\\") that have high exposure to four investment style
factors: value, quality, momentum and low size, while maintaining a level of risk similar to
that of the parent index.","exchange": "NYSE Arca"}
```

Update by Query:
```
http://localhost:9200/ct_etf/_update_by_query
POST
```
```
BODY
{"script": "if(ctx._source['rating']==3) {ctx._source['rating']==4}"}
```

Reindex:
```
http://localhost:9200/_reindex
POST
```
```
BODY
{
       "source": {"index": "cf_view"},
       "dest": {"index": "cf_etf"}
}
```

Mapping:

Dynamic mapping: Based on type or values maps will be created
Custom Mapping: User explicitly maps the content

Create default mapping index:
```
http://localhost:9200/default_mappins_index/
PUT
```
```
BODY
{
    "mappings" : {
        "dynamic_date_formats" : ["yyyy/MM/dd"]
    }
}

**Creating doc with default mapping index:**
```
http://localhost:9200/default_mappins_index/_doc/1
PUT
```
```
BODY
{
 "simple_string_value": "a string",
 "string_w_date_value": "2019/02/07",
 "string_w_floating_point_value": "1.11",
 "string_w_integer_value" : "1",
 "integer_value": 1,
 "floating_piont_value": 1.1,
 "json_object" : {"level_1" : {"level2_1":"a string", "level_2_2":1,
"level_2_3":1.11}},
 "array_of_integer": [1, 2],
 "array_of_float": [1.11, 2.22],
 "array_of_simple_string": ["ab", "cd"],
 "array_of_integer_string": ["1", "2"],
 "array_of_float_string": ["1.11", "2.22"],
 "boolean_value" : true,
 "null_value": null
}

Search the default mapping index:
```
http://localhost:9200/default_mappins_index/_search
GET
```
Create the custom mapping index with different structure:

```
http://localhost:9200/custom_mappins_index/
PUT
```
```
```

BODY
```
{
    "mappings" : {
        "date_detection" : false,
        "numeric_detection" : true
    }
}
```

Get the custom mapping index details:
```
http://localhost:9200/custom_mappins_index/_mappings
PUT
```

Perform the join in Dev tools in Kibana:

```
PUT test

PUT test/_mapping
{
  "properties" : {
    "id" : {"type" : "keyword"},
    "date" : {"type" : "date"},
    "customer_id" : {"type" : "keyword"},
    "sent" : {"type" : "boolean"},
    "item" : {"type" : "object",
      "properties" : {
        "name" : {"type" : "keyword"},
        "quantity" : {"type" : "long"},
        "cost" : {"type": "alias", "path":"item.price"},
        "price" : {"type" : "double"},
        "vat" : {"type" : "double"}
      }
    }
  }
}


PUT test/_doc/1?refresh
{
  "id": "1",
  "date": "2018-11-16T20:07:45Z",
  "customer_id": "100",
  "sent": true,
  "item": [
    {
      "name": "tshirt",
      "quantity": 10,
      "price": 4.3,
      "vat": 8.5
    }
  ]
}

GET test/_mapping

GET test/_search
{
  "query": {
    "term" : {
      "item.cost" : 4.3
    }
  }
}
```

Q. Why there is a need to create one alias for multiple indexes? Tell one scenario or a case study on this?
Uber has a marketplace data real time data analysis with elasticsearch. Uber marketplace app is a supply and demand. Interaction between them is creating trips and fare.

DAY 3

Analyzer:
Mainly used for doing analysis. Helps you to do text analysis (unstructured data). It can do data mining, some amount of linguistic search. Performs full text searching where it can search relevant results rather than exact search. Tokenization, normalization (lemmatization, stemming) is used.
Analyzer does not have NLP model but has a similar logic behind the analyzer. But yu can use NLP model as an analyzer as well (custom setup).
Elasticsearch used standard analyzer.
Different analyzers are there e.g. whitespace analyzer, etc.

Creating Analyzer:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "<body> You all working on Elastic search </body>",
    "tokenizer" : "standard"
}
```

Creating lowercase Analyzer:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "<body> You'll working on Elastic search </body>",
    "tokenizer" : "standard",
    "filter" : ["lowercase"]
}
```

Creating Analyzer with filter:

```
http://localhost:9200/_analyze
POST
```

HEADER
Content-type application/json

```
BODY
{
    "text" : "<body> You'll working on Elastic search </body>",
    "char_filter" : ["html_strip", {"type" : "mapping", "mappings" : ["You'll => You will"]}],
    "tokenizer" : "standard",
    "filter" : ["lowercase"]
}
```

Using Analyzer to replace:

```
http://localhost:9200/_analyze
POST
```

HEADER
Content-type application/json

```
BODY
{
    "text" : "<body> You'll working on Elastic search 8.8</body>",
    "char_filter" : ["html_strip", {"type" : "mapping", "mappings" : ["You'll => You will"]},
    {"type" : "pattern_replace", "pattern" : "(\\d+).(\\d+)", "replacement" : "v$1"}],
    "tokenizer" : "standard",
    "filter" : ["lowercase"]
}
```

Using Analyzer for URL:

```
http://localhost:9200/_analyze
POST
```

HEADER
Content-type application/json

```
BODY
{
    "text" : "POST https://api.iextrding.com/1.0/stock/acwf/company /usr/local",
    "tokenizer" : "standard"
}
```

Using whitespace Analyzer for URL:

```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "POST https://api.iextrding.com/1.0/stock/acwf/company /usr/local",
    "tokenizer" : "whitespace"
}

**Using uax_url_email Analyzer for URL (URL is retained):**
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "POST https://api.iextrding.com/1.0/stock/acwf/company /usr/local",
    "tokenizer" : "uax_url_email"
}

Ngram tokenizer in Analyzer:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "ElasticSearch 8.8",
    "tokenizer" : {"type" : "ngram", "min-
gram" : 2, "max_gram" : 2, "token_chars" : ["punctuation", "digit"] }
}

**Simple_pattern tokenizer in Analyzer:**
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY

~~~
{
    "text" : "ElasticSearch 8.8",
    "tokenizer" : {"type" : "simple_pattern",  "pattern" : "[a-zA-Z]*" }
}
~~~

```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY

~~~
{
    "text" : "<body> You'll working on Elastic search 8.8</body>",
    "tokenizer" : "standard",
    "filter" : ["fingerprint"]
}
~~~

**Analyzer with stemming filter:**
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
~~~
{
    "text" : "love loves loved",
    "tokenizer" : "standard",
    "filter" : [{"type": "stemmer", "name" : "english"}]
}
~~~

Analyzer with stemming filter returning only unique stemming word:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
~~~
{
    "text" : "love loves loved",
    "tokenizer" : "standard",
    "filter" : [{"type": "stemmer", "name" : "english"}, "unique"]
}
~~~
```

Analyzer with filter and script:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "Elasticsearch 8.8 is the search and analytics engine that powers the Elastic Engine",
    "tokenizer" : "standard",
    "filter" : [{"type": "condition", "script" : {"source" : "token.getType()=='<ALPHANUM>'"},
 "filter" : ["reverse"]}]
}
```

Analyzer with delimiter:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "Elasticsearch 8.8 is the search and analytics engine that powers the Elastic Engine",
    "tokenizer" : "standard",
    "filter" : [{"type": "word_delimiter", "generate_word_parts" : true }]
}
```

Analyzer with catenate:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "Elasticsearch 8.8 is the search and analytics engine that powers the Elastic Engine",
    "tokenizer" : "standard",
    "filter" : [{"type": "word_delimiter", "catenate_all" : true }]
}
```

Analyzer with split_on_case_change:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "Elasticsearch 8.8 is the search and analytics engine that powers the Elastic Eng
ine",
    "tokenizer" : "standard",
    "filter" : [{"type": "word_delimiter", "split_on_case_change" : true }]
}
```

Standard Analyzer:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "In ElasticSearch 8.8",
    "analyzer" : "standard"
}
```

Whitespace Analyzer:
```
http://localhost:9200/_analyze
POST
```
HEADER
Content-type application/json
```
BODY
{
    "text" : "In ElasticSearch 8.8",
    "analyzer" : "whitespace"
}
```

Other Analyzers: fingerprint, keyword, English, pattern, simple

NOTE: Analyzer is a wrapper on top of entities whereas tokenizer is used with filters. We can create custom analyzer under the index.
Custom Analyzer:

First create file “stopwords” under the config in elasticsearch.
```
http://localhost:9200/cf_etf
PUT
```
HEADER
Content-type application/json
```
BODY
{
        "mappings": {
                "dynamic": false,
                "properties": {
                        "symbol": {"type": "keyword"},
                        "fund_name": {"type": "text", "fields": {"raw": {"type":"keyword"}},
"analyzer":"description_analyzer", "search_analyzer":"description_analyzer"},
                        "rating": {"type": "byte"},
                        "morningstar_category": {"type": "keyword"},
                        "category": {"type": "keyword"},
                        "family": {"type": "keyword"},
                        "market_cap": {"type": "keyword"},
                        "description": {"type": "text", "fields": {"raw": {"type":"keyword"}},
"analyzer":"description_analyzer", "search_analyzer":"description_analyzer"},
                        "exchange": {"type": "keyword"}
                }
        },
        "settings": {
                "analysis": {
                        "analyzer" : {
                                "description_analyzer": {
                                        "type": "custom",
                                        "tokenizer": "description_tokenizer",
                                        "filter": ["lowercase", "description_pattern_replace_filter",
"description_stemmer_filter", "description_stop_filter", "description_length_filter",
"unique"]
                                }
                        },
                        "tokenizer": {
                                "description_tokenizer" : {"type":"char_group",
"tokenize_on_chars": ["whitespace", "digit", "symbol", "\\n", ",", ":", "!", "?", ";", ",",
"_", "{", "[", "}", "]", "(", ")", "\\", "\/","\""]}
                        },
                        "filter": {
                                "description_pattern_replace_filter": {"type":"pattern_replace",
"pattern": "(\\w{2,})\\.", "replacement":"$1"},
                                "description_stemmer_filter" : {"type":"stemmer",
"name":"light_english"},
                                "description_stop_filter": {"type":"stop",
"stopwords_path":"stopwords"},
                                "description_length_filter": {"type":"length", "min":2}
                        }
                }
        }
```

}

Now analyze the text using the above custom analyzer that was created:

```
http://localhost:9200/cf_etf/_analyze
GET
```

HEADER
Content-type application/json

```
BODY

{

       "text":"The investment seeks to track the price and yield performance, before fees and
expenses, of the Bloomberg Barclays U.S. Aggregate Enhanced Yield Index (the \\\"index\\\").\\
n The index is designed to broadly capture the U.S. investment grade, fixed income securities
market while seeking to enhance yield within desired risk parameters and constraints. The fund
 is non-diversified.",

       "analyzer": "description_analyzer"

}
```

Now using analyzer with search:

git clone https://github.com/vishymails/ELK_RAW.git

curl -XPUT localhost:9200/cf_etf -H "Content-Type:application/json" --data-binary
@define_custom_analyzer.json

curl -XPOST localhost:9200/cf_etf/_bulk?pretty -H "Content-Type:application/json" --data-binary
@cf_etf_list_bulk.json

Now check the results using REST calls in Postman:

```
http://localhost:9200/cf_etf/_search?from=2&size=2&sort=symbol:asc&_source=false
POST
```

HEADER
Content-type application/json

```
-----------------------------------------------------------------------------------------------
```

```
http://localhost:9200/cf_etf/_search?q=fund_name:ishares edge gobal&from=2&size=2&sort=_score:asc
POST
```

HEADER
Content-type application/json
```

```
http://localhost:9200/cf_etf/_search?q=fund_name:ishares%20edge%20global&from=2&default_operator=AND&analyzer=keyword
POST
```
HEADER
Content-type application/json
```

-----------------------------------------------------------------------------------------------------

```
http://localhost:9200/cf_etf/_search?q=fund_name:ishares%20edge%20global&from=2&default_operator=AND&explain=true
POST
```
HEADER
Content-type application/json
```

You can also search and filter:
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "query_string" : {
            "query" : "family:iShares",
            "default_operator" : "AND"
        }
    }
}
------------------------------------------------------------------------------------------
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
```

```
{
    "query" : {
        "query_string" : {
            "query" : "family:iShares",
            "default_operator" : "AND"
        }
    },
    "sort" : {
        "rating" :"desc",
        "symbol" : "asc"

    }
}
```
-------------------------------------------------------------------------------------------

```
http://localhost:9200/cf_etf/_search
POST
```

HEADER
Content-type application/json

```
BODY
{
    "query" : {
        "query_string" : {
            "query" : "fund_name:ishares edge global",
            "default_operator" : "AND"
        }
    },
    "sort" : {"rating" :"desc","morningstar_category" : "desc"

    }
}
```

Searching using the query string:

```
http://localhost:9200/cf_etf/_search
POST
```

HEADER
Content-type application/json

```
BODY
{
    "query" : {
        "query_string" : {
            "query" : "description:emerging non-diversified high-yield gobal division",
            "analyzer" : "description_analyzer"
        }
    }

}
```

Searching using the query string (return events which has all keywords in the search):

```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "query_string" : {
            "query" : "description:emerging non-diversified high-yield gobal division",
            "analyzer" : "description_analyzer",
            "default_operator" : "AND"
        }
    }


}

**Searching using the query and rescore:**

```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "query_string" : {
            "query" : "description:emerging non-diversified high-yield global dividend",
            "analyzer" : "description_analyzer"
        }
    },
    "rescore" : {
        "window_size" : 500,
        "query" : {
            "rescore_query" : {
                "query_string" : {
                    "query" : "description:global",
                    "analyzer" : "description_analyzer"

                }
            },
            "query_weight" : 0.1,
            "rescore_query_weight" : 0.9
        }
    }
}

Searching using the query and highlighting:

```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json

```
BODY
~~~
{
    "query" : {
        "bool" : {
            "should" : [
                {
                    "query_string" : {
                        "_name" : "Asia",
                        "query" : "description:asia",
                        "analyzer" : "description_analyzer"
                }},
                {
                    "query_string" : {
                        "_name" : "usa",
                        "query" : "description:usa",
                        "analyzer" : "description_analyzer"
                }},
                {
                    "query_string" : {
                        "_name" : "europe",
                        "query" : "description:europe",
                        "analyzer" : "description_analyzer"
                }}

            ]
        }

    },
    "highlight" : {
        "fields" : {
            "description" : {}
            }
        }
    }
~~~

**Multiple queries simultaneous execution:**
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
~~~
{
    "query" : {
        "bool" : {
            "should" : [
                {"query_string" : {
                    "_name" : "asia",
                    "query" : "description:asia",
~~~

```
                    "analyzer" : "description_analyzer"}},
                {"query_string" : {
                    "_name" : "europe",
                    "query" : "description:europe",
                    "analyzer" : "description_analyzer"}}

            ]
        }

    }

}
```

NOTE: Rescore can do 3 things filtering, ordering,

Search using OR condition:

~~~~
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "match" : {
            "fund_name" : {
                "query" : "ishares global"
            }
        }
    }
}
~~~~

Search using Match with AND condition:

~~~~
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "match" : {
            "fund_name" : {
                "query" : "ishares global",
                "operator" : "and"
            }
        }
    }
}
~~~~

Search using Match with AND condition and fuziness:
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "match" : {
            "fund_name" : {
                "query" : "ishares global",
                "operator" : "and",
                "fuzziness" : 2

            }
        }
    }
}
```

DAY 4

Match Queries:

Search using must and script:
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "bool" : {
            "must" : {"match_phrase_prefix" : {"fund_name" : "ishares edge"}},
            "filter" : {
                "script" : {
                    "script" : {
                        "lang" : "painless",
                        "source" : "if (!doc['rating'].empty && doc['rating'].value >= 4) retu
rn true;else false;"
                    }
                }
            }
        }
        }
    }
```

Search using must_not and script:
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "bool" : {
            "must" : {"match_phrase_prefix" : {"fund_name" : "ishares edge"}},
            "must_not" : {"match" : {"fund_name" : "global" }
                }
            }
        }

}

**Search using boosting:**
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "boosting":{
            "positive" : {"match_phrase_prefix" : {"fund_name" : "ishares edge"}},
            "negative" : {"term" : {"category" : "Equity"}},
            "negative_boost" : 0.1
        }
            }
}

Search using functions:
```
http://localhost:9200/cf_etf/_search
POST
```
HEADER
Content-type application/json
```
BODY
{
    "query" : {
        "function_score" : {
            "query" : {"match_phrase" : {"fund_name" : "ishares edge"}},
            "functions" : [
                {"filter" : {"wildcard" : {"morningstar_category" : "*Stock"}}, "weight" : 2},
                {"filter" : {"wildcard" : {"morningstar_category" : "*Growth"}}, "weight" : 1.
5},
                {"filter" : {"wildcard" : {"morningstar_category" : "*Blend"}}, "weight" : 0.5
}

            ]
        }
    }
}
```

Rank Evaluation:

```
http://localhost:9200/cf_etf/_rank_eval
POST
```

HEADER
Content-type application/json

```
BODY
{
    "requests" : [
        {
            "id" : "Asia",
            "request" : {"query" : {"match" : {"description" : "asia"}}},
            "ratings" :[]
        },
        {
            "id" : "Global",
            "request" : {"query" : {"match" : {"description" : "global"}}},
            "ratings" :[]
        }
    ],
    "metric" : {
        "precision" : {
            "k" : 3,
            "relevant_rating_threshold" :1
        }
    }
}
```

Suggesters:
Provides auto completion sort of behavior. Faster search mechanism. e.g. songs suggestion based on your search.
Global Suggester:
Term Suggester:
Field base Suggester:
Completion Suggester:
Context Suggester:

Geo Context: Based on latitude and longitude you can search

Suggester:
```
http://localhost:9200/cf_etf/_rank_eval
POST
```
HEADER
Content-type application/json
```
BODY
{
    "suggest" : {
        "text" : "asian stock",
        "suggester_1" : {
            "term" : {
                "field" : "fund_name"
        }
    },
        "suggester_2" : {
            "term" : {
                "field" : "description"
            }



    }}

}
```

Suggester:
```
http://localhost:9200/cf_etf/_rank_eval
POST
```
HEADER
Content-type application/json
```
BODY
{
    "suggest" : {
        "text" : "asian stock",
        "suggester_1" : {
            "term" : {
                "field" : "fund_name"
        }
    },
        "suggester_2" : {
            "term" : {
                "field" : "description"
            }
    }}

}
```

```
http://localhost:9200/cf_etf/_rank_eval
POST
```
HEADER
Content-type application/json
```
BODY
{
    "suggest" : {
        "text"  : "divide weigh index",
        "suggester" : {
            "phrase" : {
                "field" : "description"
            }
        }
    }

}
```

Create index:

```
curl -XDELETE "http://localhost:9200/mybooks"

curl -XPUT "http://localhost:9200/mybooks" -H 'Content-Type: application/json' -d'
{
 "mappings": {
   "properties": {
    "join_field": {
     "type": "join",
     "relations": {
      "order": "item"
     }
    },
    "position": {
     "type": "integer",
     "store": true
    },
    "uuid": {
     "store": true,
     "type": "keyword"
    },
    "date": {
     "type": "date"
    },
    "quantity": {
     "type": "integer"
```

```
        },
        "price": {
          "type": "double"
        },
        "description": {
          "term_vector": "with_positions_offsets",
          "store": true,
          "type": "text"
        },
        "title": {
          "term_vector": "with_positions_offsets",
          "store": true,
          "type": "text",
          "fielddata": true,
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    }
}'

curl -XPOST "http://localhost:9200/_bulk?refresh" -H 'Content-Type: application/json' -d'
{"index":{"_index":"mybooks", "_id":"1"}}
{"uuid":"11111","position":1,"title":"Joe Tester","description":"Joe Testere nice guy","date":"2015-10-
22","price":4.3,"quantity":50}
{"index":{"_index":"mybooks", "_id":"2"}}
{"uuid":"22222","position":2,"title":"Bill Baloney","description":"Bill Testere nice guy","date":"2016-06-
12","price":5,"quantity":34}
{"index":{"_index":"mybooks", "_id":"3"}}
{"uuid":"33333","position":3,"title":"Bill Klingon","description":"Bill is not\n                nice guy","date":"2017-09-
21","price":6,"quantity":33}
'

curl -XDELETE "http://localhost:9200/mybooks-join"

curl -XPUT "http://localhost:9200/mybooks-join" -H 'Content-Type: application/json' -d'
{
  "mappings": {
      "properties": {
        "join": {
          "type": "join",
          "relations": {
            "book": "author"
          }
        },
        "position": {
          "type": "integer",
          "store": true
        },
        "uuid": {
          "store": true,
          "type": "keyword"
        },
        "date": {
          "type": "date"
        },
        "quantity": {
          "type": "integer"
        },
        "price": {
          "type": "double"
        },
        "rating": {
          "type": "double"
        },
        "description": {
          "term_vector": "with_positions_offsets",
          "store": true,
          "type": "text"
        },
        "title": {
          "term_vector": "with_positions_offsets",
          "store": true,
          "type": "text",
          "fielddata": true,
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "name": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "surname": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "versions": {
          "type": "nested",
          "properties": {
            "color": {
              "type": "keyword"
            },
            "size": {
              "type": "integer"
            }
          }
        }
      }
    }
}'


curl -XPOST "http://localhost:9200/_bulk?refresh" -H 'Content-Type: application/json' -d'
{"index":{"_index":"mybooks-join", "_id":"1"}}
{"uuid":"11111","position":1,"title":"Joe Tester","description":"Joe Testere nice guy","date":"2015-10-
22","price":4.3,"quantity":50,"join": {"name": "book"}, "versions":[{"color":"yellow", "size":5},{"color":"blue",
"size":15}]}
{"index":{"_index":"mybooks-join", "_id":"a1", "routing":"1"}}
{"name":"Peter","surname":"Doyle","rating":4.5,"join": {"name": "author", "parent":"1"}}
{"index":{"_index":"mybooks-join", "_id":"a12", "routing":"1"}}
{"name":"Mark","surname":"Twain","rating":4.2,"join": {"name": "author", "parent":"1"}}
{"index":{"_index":"mybooks-join", "_id":"2"}}
{"uuid":"22222","position":2,"title":"Bill Baloney","description":"Bill Testere nice guy","date":"2016-06-
12","price":5,"quantity":34,"join": {"name": "book"}, "versions":[{"color":"red", "size":2},{"color":"blue",
"size":10}]}
{"index":{"_index":"mybooks-join", "_id":"a2", "routing":"2"}}
{"name":"Agatha","surname":"Princeton","rating":2.1,"join": {"name": "author", "parent":"2"}}
{"index":{"_index":"mybooks-join", "_id":"3"}}
{"uuid":"33333","position":3,"title":"Bill Klingon","description":"Bill is not\n        nice guy","date":"2017-09-
21","price":6,"quantity":33,"join": {"name": "book"}, "versions":[{"color":"red", "size":2}]}
{"index":{"_index":"mybooks-join", "_id":"a3", "routing":"3"}}
{"name":"Martin","surname":"Twisted","rating":3.2,"join": {"name": "author", "parent":"3"}}
'

curl -XPOST "http://localhost:9200/mybooks-join/_refresh"

curl -XDELETE 'http://localhost:9200/mygeo-index'
curl -XPUT "http://localhost:9200/mygeo-index" -H 'Content-Type: application/json' -d'
{
  "mappings": {
    "properties": {
      "pin": {
        "properties": {
          "location": {
            "type": "geo_point"
          }
        }
      }
    }
  }
}'
```

```
curl -XPUT 'http://localhost:9200/mygeo-index/_doc/1' -H 'Content-Type: application/json' -d '{"pin": {"location":
{"lat": 40.12, "lon": -71.34}}}'
curl -XPUT 'http://localhost:9200/mygeo-index/_doc/2' -H 'Content-Type: application/json' -d '{"pin": {"location":
{"lat": 40.12, "lon": 71.34}}}'
curl -XPOST 'http://localhost:9200/mygeo-index/_refresh'
```

Execute the below in Dev tools in kibana (you can also perform using Postman)
#prefix

```
POST /mybooks/_search
{
  "query": {
    "prefix": {
      "uuid": "222"
    }
  }
}
```

#wildcard

```
POST /mybooks/_search
{
  "query": {
    "wildcard": {
      "uuid": "22?2*"
    }
  }
}
```

#regex

```
POST /mybooks/_search
{
  "query": {
    "regexp": {
      "description": {
        "value": "j.*",
        "flags": "INTERSECTION|COMPLEMENT|EMPTY"
      }
    }
  }
}
```

#span queries

```
POST /mybooks/_search
{
  "query": {
    "span_first": {
      "match": {
        "span_term": {
          "description": "joe"
        }
      },
      "end": 5
    }
  }
}
```

```
POST /mybooks/_search
{
  "query": {
    "span_or": {
      "clauses": [
        {
          "span_term": {
            "description": "nice"
          }
        },
        {
          "span_term": {
            "description": "cool"
          }
        },
        {
          "span_term": {
            "description": "wonderful"
          }
        }
      ]
    }
  }
}
```

```
POST /mybooks/_search
{
  "query": {
    "span_multi": {
      "match": {
        "prefix": {
          "description": {
            "value": "jo"
          }
        }
      }
    }
  }
}
```

```
POST /mybooks/_search
{
  "query": {
    "span_near": {
      "clauses": [
        {
          "span_term": {
            "description": "nice"
          }
        },
        {
          "span_term": {
            "description": "joe"
          }
        },
        {
          "span_term": {
            "description": "guy"
          }
        }
      ],
      "slop": 3,
      "in_order": false
    }
  }
}
```

```
POST /mybooks/_search
{
  "query": {
    "span_not": {
      "include": {
        "span_term": {
          "description": "nice"
        }
      },
      "exclude": {
        "span_near": {
          "clauses": [
            {
              "span_term": {
                "description": "not"
              }
            },
            {
              "span_term": {
                "description": "nice"
              }
            }
          ],
          "slop": 1,
          "in_order": true
        }
      }
    }
  }
}
```

```
POST /mybooks/_search
{
 "query": {
  "span_containing": {
   "little": {
    "span_term": {
     "description": "nice"
    }
   },
   "big": {
    "span_near": {
     "clauses": [
      {
       "span_term": {
        "description": "not"
       }
      },
      {
       "span_term": {
        "description": "guy"
       }
      }
     ],
     "slop": 5,
     "in_order": true
    }
   }
  }
 }
}
```

```
POST /mybooks/_search
{
 "query": {
  "span_within": {
   "little": {
    "span_term": {
     "description": "nice"
    }
   },
   "big": {
        "span_near": {
          "clauses": [
            {
              "span_term": {
                "description": "not"
              }
            },
            {
              "span_term": {
                "description": "guy"
              }
            }
          ],
          "slop": 5,
          "in_order": true
        }
      }
    }
  }
}
```

#match query

```
POST /mybooks/_search
{
  "query": {
    "match": {
      "description": {
        "query": "nice guy",
        "operator": "and"
      }
    }
  }
}
```

```
POST /mybooks/_search
{
  "query": {
    "match_phrase": {
      "description": "nice guy"
    }
  }
}
```

```
POST /mybooks/_search
{
  "query": {
    "match_phrase_prefix": {
      "description": "nice gu"
    }
  }
}

POST /mybooks/_search
{
  "query": {
    "multi_match": {
      "fields": [
        "description",
        "name"
      ],
      "query": "Bill",
      "operator": "and"
    }
  }
}
```

#query string query

```
POST /mybooks/_search
{
  "query": {
    "query_string": {
      "query": """"nice guy" -description:not price:{ * TO 5 } """,
      "fields": [
        "description^5"
      ],
      "default_operator": "and"
    }
  }
}
```

#simple query string query

```
POST /mybooks/_search
{
  "query": {
    "simple_query_string": {
      "query": """"nice guy" -not""",
      "fields": [
        "description^5",
        "_all"
      ],
      "default_operator": "and"
    }
  }
}
```

# range

```
POST /mybooks/_search
{
  "query": {
    "range": {
      "position": {
        "from": 3,
        "to": 4,
        "include_lower": true,
        "include_upper": false
      }
    }
  }
}
```

#common text

```
POST /mybooks/_search
{
  "query": {
    "common": {
      "description": {
        "query": "nice guy",
        "cutoff_frequency": 0.001
      }
    }
  }
}
```

#ids

```
POST /mybooks/_search
{
  "query": {
    "ids": {
      "type": "test-type",
      "values": [
        "1",
        "2",
        "3"
      ]
    }
  }
}
```

Ingestion:
Will occur before indexing. e.g. enriching the data before indexing it.
Each Processors will be sequentially executed. Elasticsearch adds transforms this to index.
Whenever you create ingest pipeline (using script or using Kibana) both of them stores in a cluster.
Enabled by default in 8 and after versions. Pipelines are executed in ingest node. Pipeline has to be attached to the index (you can do at id level or doc level).

In Kibana -> stack management -> ingest pipeline

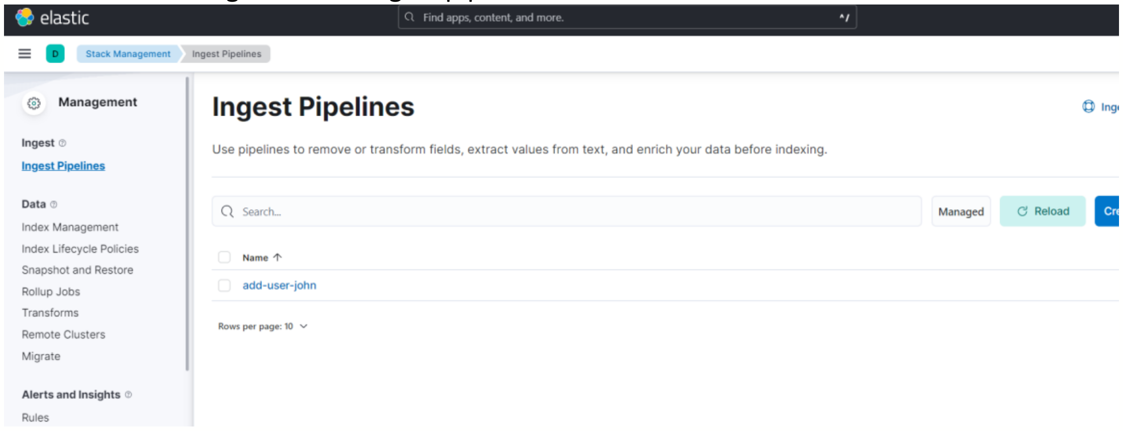
Every pipeline will have processors.

In Dev tools as well we can create piepeline.

Creating pipeline:
Got o Dev tools -> and execute below:

```
PUT /_ingest/pipeline/add-user-john
{
 "description": "Add user john field",
 "processors": [
   {
    "set" : {
      "field" : "user",
      "value": "john"
    }
   }
 ],
 "version" : 2
}
```

Check the pipeline created:
Go to “Stack management” -> ingest pipelines



Ingest Pipelines

Use pipelines to remove or transform fields, extract values from text, and enrich your data before indexing.

add-user-john

```
PUT /my_index/_doc/my_id?pipeline=add-user-john
{}


GET /my_index/_doc/my_id


GET /_ingest/pipeline/add-user-john

GET /_ingest/pipeline/*


GET /_ingest/pipeline/add-*

DELETE /_ingest/pipeline/add-user-john
```

#Simulate an ingest pipeline

```
POST /_ingest/pipeline/_simulate
{
  "pipeline": {
    "description": "Add user john field",
    "processors": [
      {
        "set": {
          "field": "user",
          "value": "john"
        }
      },
      {
        "set": {
          "field": "job",
          "value": 10
        }
      }
    ],
    "version": 1
  },
  "docs": [
    {
      "_index": "index",
      "_type": "type",
      "_id": "1",
      "_source": {
        "name": "docs1"
      }
    },
    {
      "_index": "index",
      "_type": "type",
      "_id": "2",
      "_source": {
        "name": "docs2"
      }
    }
  ]
}
```

Multiple processors can be created. You can have different step by step processes. We can do the simulation. We can perform different types of activities. Simulation is for testing purpose without creating any object to test the pipeline.

DAY 5

Activities performing with multiple processors:

```
POST /_ingest/pipeline/_simulate
{
  "pipeline": {
    "description": "Testing some build-processors",
    "processors": [
      {
        "dot_expander": {
          "field": "extfield.innerfield"
        }
      },
      {
        "remove": {
          "field": "unwanted"
        }
      },
      {
        "trim": {
          "field": "message"
        }
      },
      {
        "set": {
          "field": "tokens",
          "value": "{{message}}"
        }
      },
      {
        "split": {
          "field": "tokens",
          "separator": "\\s+"
        }
      },
      {
        "sort": {
          "field": "tokens",
          "order": "desc"
        }
      },
      {
        "convert": {
          "field": "mynumbertext",
          "target_field": "mynumber",
          "type": "integer"
        }
      }
    ]
  },
  "docs": [
    {
      "_index": "index",
      "_type": "type",
      "_id": "1",
      "_source": {
        "extfield.innerfield": "booo",
        "unwanted": 32243,
        "message": "155.2.124.3 GET /index.html 15442 0.038",
        "mynumbertext": "3123"
      }
    }
  ]
}
```

Grok filter:

```
POST /_ingest/pipeline/_simulate
{
  "pipeline": {
    "description": "Testing grok pattern",
    "processors": [
      {
        "grok": {
          "field": "message",
          "patterns": [
            "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}"
          ]
        }
      }
    ]
  },
  "docs": [
    {
      "_index": "index",
      "_type": "type",
      "_id": "1",
      "_source": {
        "message": "155.2.124.3 GET /index.html 15442 0.038"
      }
    }
  ]
}
```

```
POST /_ingest/pipeline/_simulate
{
  "pipeline": {
    "description": "custom grok pattern",
    "processors": [
      {
        "grok": {
          "field": "message",
          "patterns": [
            "my favorite color is %{COLOR:color}"
          ],
          "pattern_definitions": {
            "COLOR": "RED|GREEN|BLUE"
          }
        }
      }
    ]
  },
  "docs": [
    {
      "_source": {
        "message": "my favorite color is RED"
      }
    },
    {
      "_source": {
        "message": "happy fail!!"
      }
    }
  ]
}
```

Encode in pipeline:

#Using the ingest attachment plugin

```
PUT /_ingest/pipeline/attachment
{
  "description": "Extract data from an attachment via Tika",
  "processors": [
    {
      "attachment": {
        "field": "data"
      }
    }
  ],
  "version": 1
}


PUT /my_index/_doc/my_id1?pipeline=attachment
{
  "data" :
"SWQsIE1vZGVsLCBZZWFyLCBTY3JlZW5TaXplLCBSQU0sIEhERCwgVywgRCwgSCwgV2VpZ2h0CjEsTWFjQm9vayBQc
m8sMjAxNSwiMTVcIiIsMTZHQiw1MTJHQiBTU0QsMTMuNzUsOS40OCwwLjYxLDQuMDIKMixNYWNCb29rLDIwMTY
sIjEyXCIiLDhHQiwyNTZHQiBTU0QsMTEuMDQsNy43NCwwLjUyLDIuMDMKMyxNYWNCb29rIEFpciwyMDE2LCIxMy4
zXCIiLDhHQiwxMjhHQiBTU0QsMTIuOCw4Ljk0LDAuNjgsMi45Ngo0LGlNYWMsMjAxNywiMjdcIiIsNjRHQiwxVEIgU1N
ELDI1LjYsOC4wLDIwLjMsMjAuOA=="
}


GET /my_index/_doc/my_id1
```

Geo IP concept in pipeline:

#Using the ingest GeoIP plugin

```
PUT /_ingest/pipeline/geoip
{
  "description": "Extract geopoint from an IP",
  "processors": [
    {
      "geoip": {
        "field": "ip"
      }
    }
  ],
  "version": 1
}
```

```
PUT /my_index/_doc/my_ip?pipeline=geoip
{
  "ip" : "46.51.224.0"
}
```

```
GET /my_index/_doc/my_ip
```

LogStash

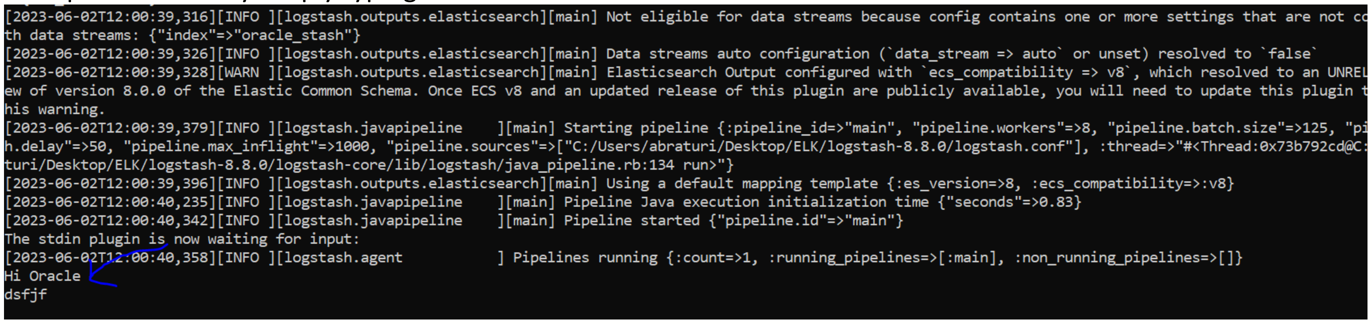
Is an independent product. You have different categories of activities that can be performed. Logs stash configuration using logstash.conf.

Edit the logstash.conf with the below:

```
input {
   stdin {}

}

output {
   elasticsearch {
      hosts => ["localhost:9200"]
      index => "oracle_stash"
   }
}
```

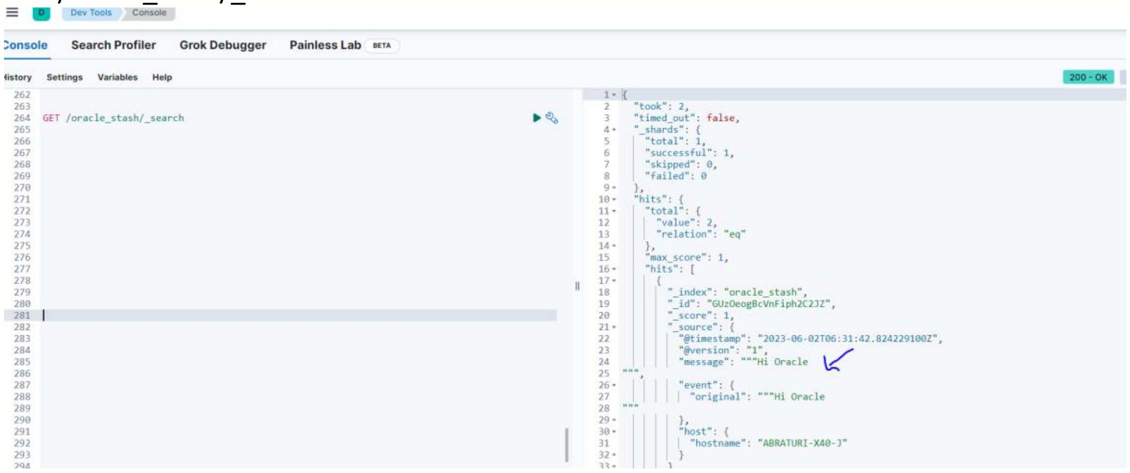
Now go to directory: C:\Users\abraturi\Desktop\ELK\logstash-8.8.0\bin
Once done execute the below command: .\logstash.bat -f .\logstash.conf

Now provide data by simply typing and hit enter:



```
[2023-06-02T12:00:39,316][INFO ][logstash.outputs.elasticsearch][main] Not eligible for data streams because config contains one or more settings that are not co
th data streams: {"index"=>"oracle_stash"}
[2023-06-02T12:00:39,326][INFO ][logstash.outputs.elasticsearch][main] Data streams auto configuration (`data_stream => auto` or unset) resolved to `false`
[2023-06-02T12:00:39,328][WARN ][logstash.outputs.elasticsearch][main] Elasticsearch Output configured with `ecs_compatibility => v8`, which resolved to an UNREL
ew of version 8.0.0 of the Elastic Common Schema. Once ECS v8 and an updated release of this plugin are publicly available, you will need to update this plugin t
his warning.
[2023-06-02T12:00:39,379][INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline.workers"=>8, "pipeline.batch.size"=>125, "pi
h.delay"=>50, "pipeline.max_inflight"=>1000, "pipeline.sources"=>["C:/Users/abraturi/Desktop/ELK/logstash-8.8.0/logstash.conf"], :thread=>"#<Thread:0x73b792cd@C:
turi/Desktop/ELK/logstash-8.8.0/logstash-core/lib/logstash/java_pipeline.rb:134 run>"}
[2023-06-02T12:00:39,396][INFO ][logstash.outputs.elasticsearch][main] Using a default mapping template {:es_version=>8, :ecs_compatibility=>:v8}
[2023-06-02T12:00:40,235][INFO ][logstash.javapipeline    ][main] Pipeline Java execution initialization time {"seconds"=>0.83}
[2023-06-02T12:00:40,342][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
The stdin plugin is now waiting for input:
[2023-06-02T12:00:40,358][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}
Hi Oracle
dsfjf
```

Check the data in kibana using Dev tools:
GET /oracle_stash/_search



```
GET /oracle_stash/_search
```

```
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 2,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "oracle_stash",
        "_id": "GUzOeogBcVnFiph2C2JZ",
        "_score": 1,
        "_source": {
          "@timestamp": "2023-06-02T06:31:42.824229100Z",
          "@version": "1",
          "message": """Hi Oracle
""",
          "event": {
            "original": """Hi Oracle
"""
          },
          "host": {
            "hostname": "ABRATURI-X40-J"
          }
        }
```

Filebeat configuration:
Logstash and beat has to be on same machine.
Install file beat and put the filebeat-example.yml file there

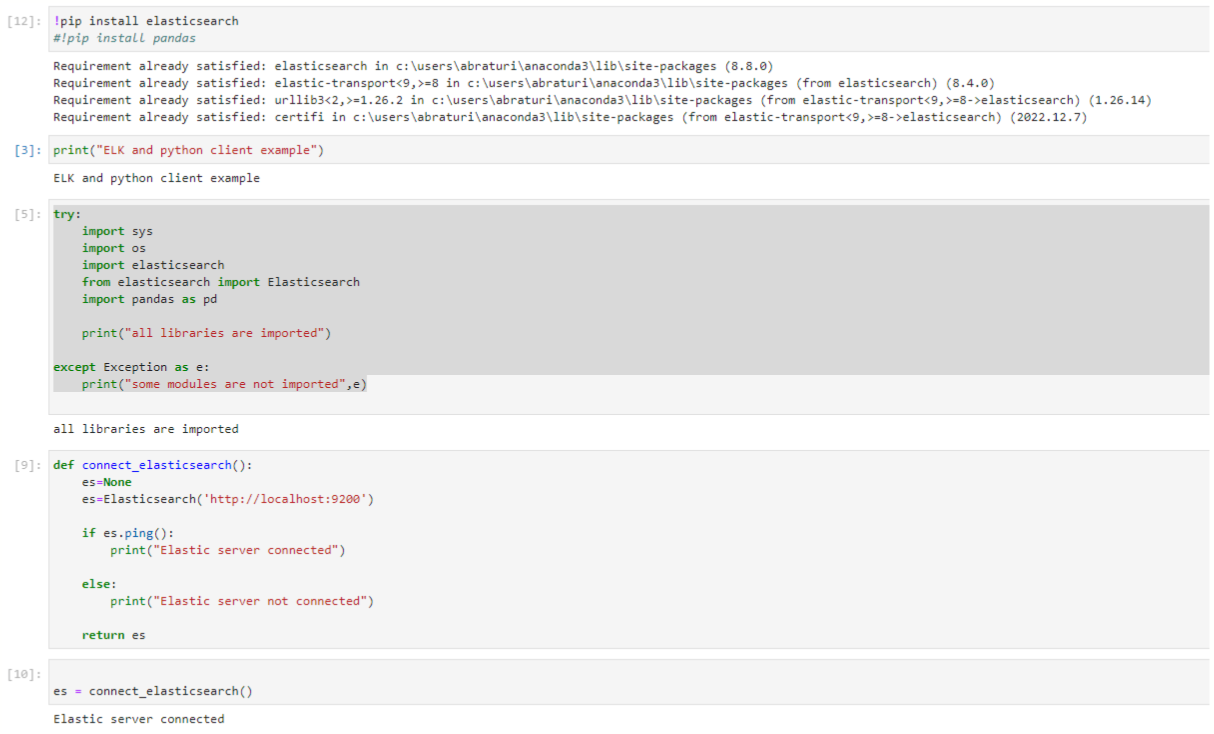
Now under logstash put the filebeat-demo.conf file.

Also put the logstash-dataset under the same path
https://github.com/vishymails/ELK_RAW/tree/main

Logstash will collect the data and send to elasticsearch amd elasticsearch will store in nosql database in the form of indexes.

Using python to connect to Elasticsearch:
Install Anaconda Navigator
Open Jupyter lab from the Anaconda Navigator
Cretae ipynb file with below code

```
!pip install elasticsearch
!pip install pandas
print("ELK and python client example")
try:
    import sys
    import os
    import elasticsearch
    from elasticsearch import Elasticsearch
    import pandas as pd

    print("all libraries are imported")

except Exception as e:
    print("some modules are not imported",e)


def connect_elasticsearch():
    es=None
    es=Elasticsearch('http://localhost:9200')

    if es.ping():
        print("Elastic server connected")

    else:
        print("Elastic server not connected")

    return es


es = connect_elasticsearch()
```



```
[12]: !pip install elasticsearch
      #!pip install pandas

      Requirement already satisfied: elasticsearch in c:\users\abraturi\anaconda3\lib\site-packages (8.8.0)
      Requirement already satisfied: elastic-transport<9,>=8 in c:\users\abraturi\anaconda3\lib\site-packages (from elasticsearch) (8.4.0)
      Requirement already satisfied: urllib3<2,>=1.26.2 in c:\users\abraturi\anaconda3\lib\site-packages (from elastic-transport<9,>=8->elasticsearch) (1.26.14)
      Requirement already satisfied: certifi in c:\users\abraturi\anaconda3\lib\site-packages (from elastic-transport<9,>=8->elasticsearch) (2022.12.7)

 [3]: print("ELK and python client example")

      ELK and python client example

 [5]: try:
          import sys
          import os
          import elasticsearch
          from elasticsearch import Elasticsearch
          import pandas as pd

          print("all libraries are imported")

      except Exception as e:
          print("some modules are not imported",e)

      all libraries are imported

 [9]: def connect_elasticsearch():
          es=None
          es=Elasticsearch('http://localhost:9200')

          if es.ping():
              print("Elastic server connected")

          else:
              print("Elastic server not connected")

          return es

[10]:
      es = connect_elasticsearch()

      Elastic server connected
```

```
es.indices.create(index="python-index", ignore=400)
es.indices.create(index="python-index1", ignore=400)

res=es.indices.get_alias(index="*")
for Name in res:
    print(Name)

from datetime import datetime
doc={
    'author' : 'ABhishek Raturi',
    'text'  : 'Elastic Search Example',
    'customer' : 'Oracle India',
    'session_date' : datetime.now()
}
resp=es.index(index="python-index",id=2,document=doc)
```

```
print(resp['result'])
resp = es.get(index='python-index', id=1)
print(resp['_source'])


from datetime import datetime
doc={
    'author' : 'ABhishek Raturi',
    'text'  : 'Elastic Search Example2',
    'customer' : 'Oracle India',
    'session_date' : datetime.now()
}
resp=es.index(index="python-index",id=2,document=doc)
print(resp['result'])


es.delete(index='python-index', id=1)
```