

Lab 4 Elasticsearch and Kibana (I)

In this lab, you will learn how to use Elasticsearch and Kibana, including [JSON-based Query DSL \(Domain Specific Language\)](#), Analyzer API, Mapping and Coercion.



elasticsearch kibana

Task 1 Getting started with Elasticsearch and Kibana

Please download Elasticsearch and Kibana. The version we are going to use in this paper is 7.7.0. Elasticsearch and Kibana take around 500 MB and 1GB, respectively. The Elasticsearch 7.7.0 requires a [JDK 11 or newer version](#), but [JDK 8](#) turns out to be fine.

The download links are as follows:

- Elasticsearch: <https://www.elastic.co/downloads/past-releases/elasticsearch-7-7-0>
- Kibana: <https://www.elastic.co/downloads/past-releases/kibana-7-7-0>

For Windows system, when you download and extract both zip files, launch elastsearch.bat located in the following folder:

elasticsearch-7.7.0-windows-x86_64 > elasticsearch-7.7.0 > bin	
Name	Date modified
elasticsearch	5/12/2020 1:58 AM
elasticsearch	5/12/2020 1:58 AM
elasticsearch-certgen	5/12/2020 2:03 AM
elasticsearch-certgen	5/12/2020 2:03 AM

The screenshot below shows Elasticsearch runs at the port 9200.




```
C:\WINDOWS\system32\cmd.exe
g template [.monitoring-logstash] for index patterns [.monitoring-logstash-7-*]
[2020-08-26T10:25:56,211][INFO ][o.e.c.m.MetaDataIndexTemplateService] [LEO-WORKSTATION] addin
g template [.monitoring-es] for index patterns [.monitoring-es-7-*]
[2020-08-26T10:25:56,283][INFO ][o.e.c.m.MetaDataIndexTemplateService] [LEO-WORKSTATION] addin
g template [.monitoring-beats] for index patterns [.monitoring-beats-7-*]
[2020-08-26T10:25:56,348][INFO ][o.e.c.m.MetaDataIndexTemplateService] [LEO-WORKSTATION] addin
g template [.monitoring-alerts-7] for index patterns [.monitoring-alerts-7]
[2020-08-26T10:25:56,430][INFO ][o.e.c.m.MetaDataIndexTemplateService] [LEO-WORKSTATION] addin
g template [.monitoring-kibana] for index patterns [.monitoring-kibana-7-*]
[2020-08-26T10:25:56,495][INFO ][o.e.x.i.a.TransportPutLifecycleAction] [LEO-WORKSTATION] addi
ng index lifecycle policy [watch-history-ilm-policy]
[2020-08-26T10:25:56,561][INFO ][o.e.x.i.a.TransportPutLifecycleAction] [LEO-WORKSTATION] addi
ng index lifecycle policy [ml-size-based-ilm-policy]
[2020-08-26T10:25:56,599][INFO ][o.e.h.AbstractHttpServerTransport] [LEO-WORKSTATION] publish_
address {127.0.0.1:9200}, bound addresses {127.0.0.1:9200}, {:::1}:9200}
[2020-08-26T10:25:56,600][INFO ][o.e.n.Node] [LEO-WORKSTATION] started
[2020-08-26T10:25:56,625][INFO ][o.e.x.i.a.TransportPutLifecycleAction] [LEO-WORKSTATION] addi
ng index lifecycle policy [ilm-history-ilm-policy]
[2020-08-26T10:25:56,673][INFO ][o.e.x.i.a.TransportPutLifecycleAction] [LEO-WORKSTATION] addi
ng index lifecycle policy [slm-history-ilm-policy]
[2020-08-26T10:25:56,844][INFO ][o.e.l.LicenseService] [LEO-WORKSTATION] license [b072aa6
f-3c1c-44ff-9089-0a3c40710179] mode [basic] - valid
[2020-08-26T10:25:56,846][INFO ][o.e.x.s.s.SecurityStatusChangeListener] [LEO-WORKSTATION] Act
ive license is now [BASIC]; Security is disabled
```

Open another command prompt and use [curl](#) to check if the Elasticsearch is running properly.

```
Command Prompt
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\weihu>curl localhost:9200
{
  "name" : "LEO-WORKSTATION",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "4F5wslWJTq0-xUswMf_FNA",
  "version" : {
    "number" : "7.7.0",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "81ale9eda8e6183f5237786246f6dced26a10eaf",
    "build_date" : "2020-05-12T02:01:37.602180Z",
    "build_snapshot" : false,
    "lucene_version" : "8.5.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

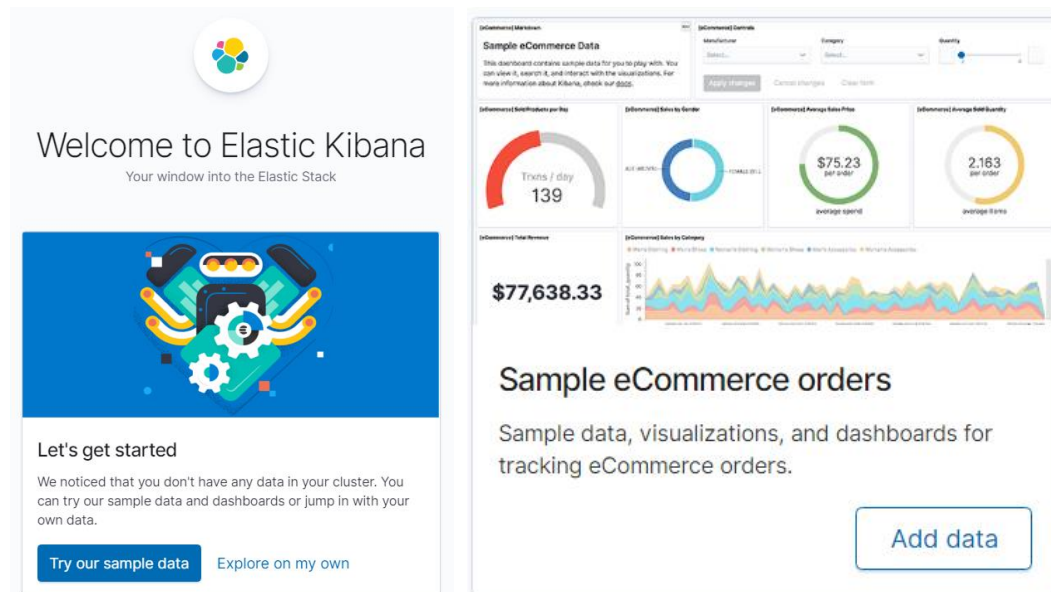
Launch Kibana by executing the kibana.bat in the folder below.

DATA (D:) > Big Data > kibana-7.7.0-windows-x86_64 > bin	
Name	Date modified
 kibana	5/12/2020 2:53 AM
 kibana-keystore	5/12/2020 2:53 AM
 kibana-plugin	5/12/2020 2:53 AM

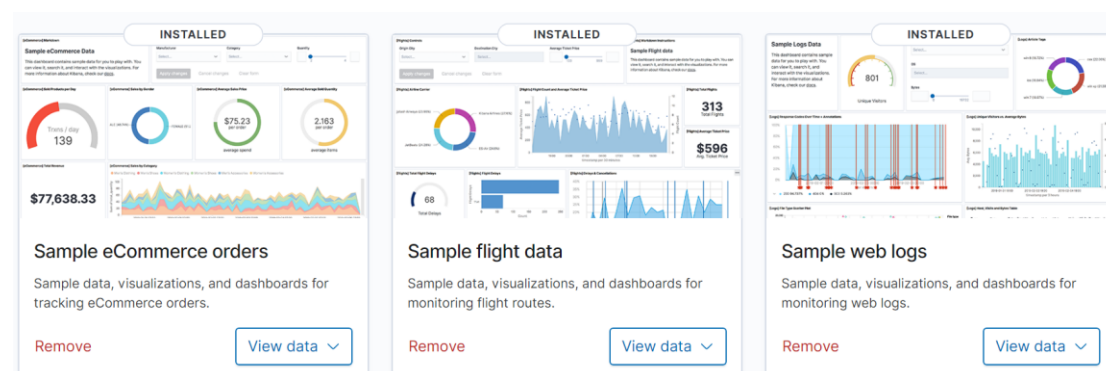
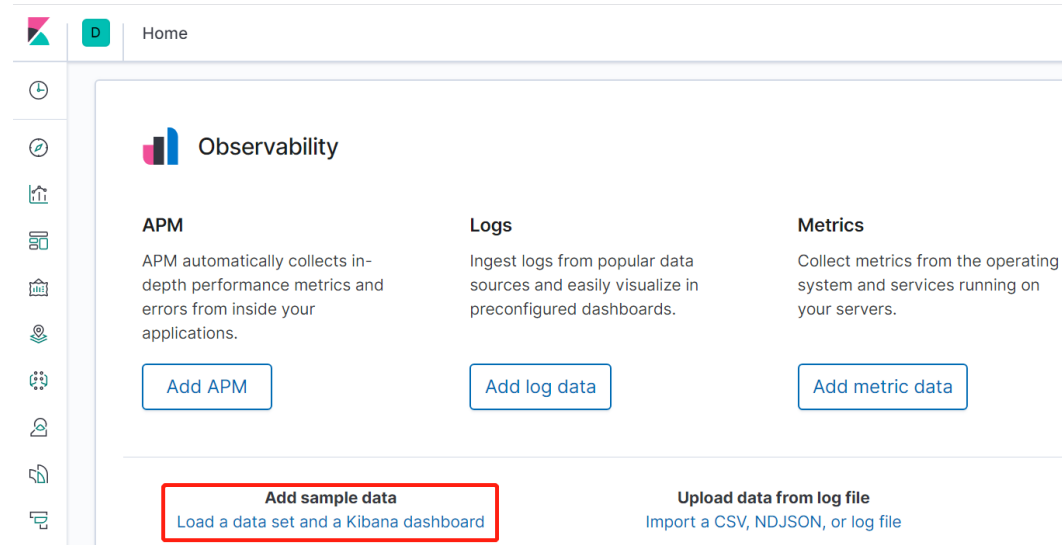
Kibana runs at the port 5601. After that, open your browser and access **localhost:5601**

```
C:\WINDOWS\system32\cmd.exe
log [22:39:07.360] [info][status][plugin:markdown_vis@7.7.0] Status changed from uninitialized
to green - Ready
log [22:39:07.362] [info][status][plugin:table_vis@7.7.0] Status changed from uninitialized to
green - Ready
log [22:39:07.365] [info][status][plugin:tagcloud@7.7.0] Status changed from uninitialized to
green - Ready
log [22:39:07.367] [info][status][plugin:timelion_vis@7.7.0] Status changed from uninitialized
to green - Ready
log [22:39:07.370] [info][status][plugin:vis_type_vega@7.7.0] Status changed from uninitializ
d to green - Ready
log [22:39:08.144] [warning][reporting] Generating a random key for xpack.reporting.encryption
Key. To prevent pending reports from failing on restart, please set xpack.reporting.encryptionKey
in kibana.yml
log [22:39:08.149] [info][status][plugin:reporting@7.7.0] Status changed from uninitialized to
green - Ready
log [22:39:08.185] [info][listening] Server running at http://localhost:5601
log [22:39:08.571] [info][server][Kibana][http] http server running at http://localhost:5601
```

When you see the figure below in your browser, choose “try out sample data”. Add **Sample eCommerce orders**. It optional for you to add the other two sample datasets since they won’t be required in this course.



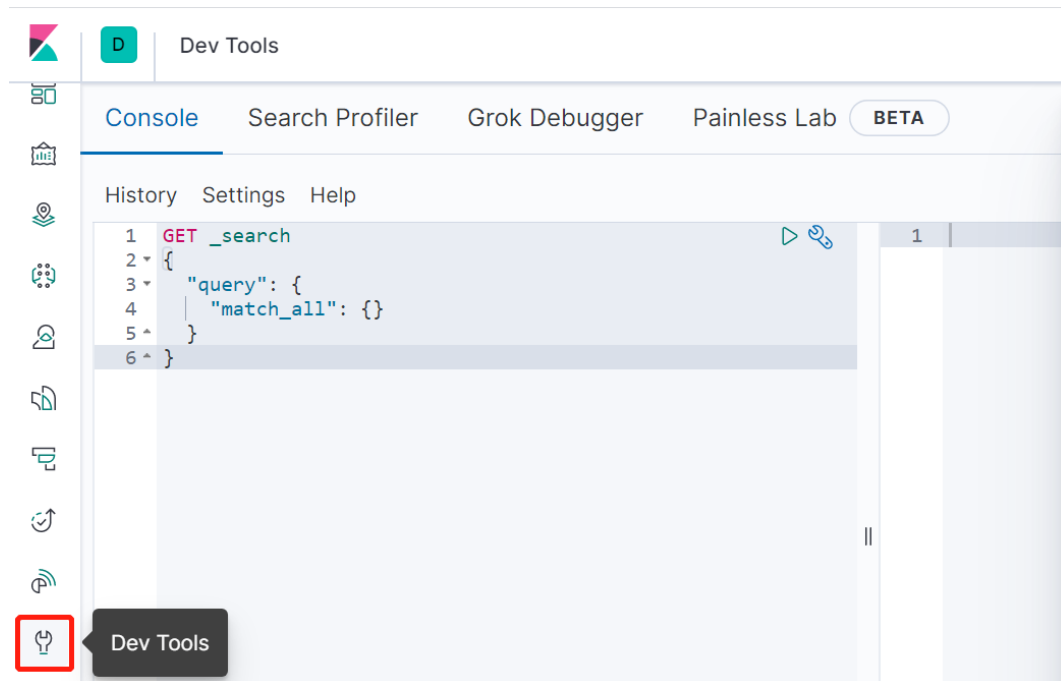
You can also add sample data after getting in the Home page of Kibana.



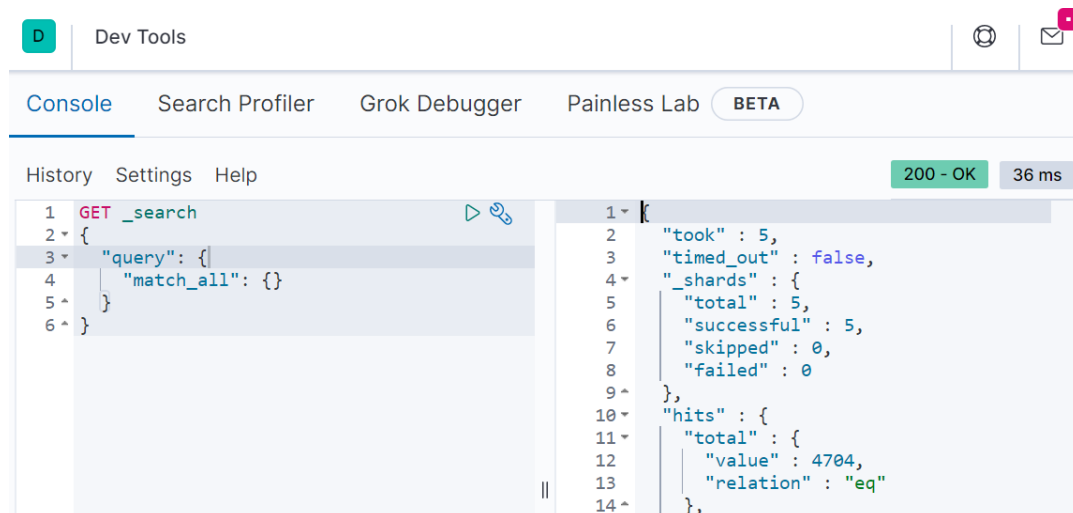
Task 2 Domain Specific Language (DSL)

Elasticsearch is part of the ELK Stack and is built on [Lucene](#), the search library from Apache, and exposes Lucene's query syntax.

Select **Dev Tools** and you will see the figure below.



Execute the code block using ctrl+enter. The results show on the right-hand side.



Getting started – Query all the records

```
GET _search
{
  "query": {
    "match_all": {}
  }
}
```

Follow the instructions below, execute scripts and see the outputs. More importantly, learn how to use DSL.

#see the cluster status

```
GET _cat/health?v
GET _cat/nodes?v
GET _cat/indices?v
```

#create index

```
PUT /sales
```

#create an order

```
PUT /sales/_doc/123
{
  "orderid": "123",
  "orderAmount" : "500"
}
```

search all documents from an index

```
GET sales/_search
```

get a document from the index

```
GET sales/_doc/123
```

delete a document

```
DELETE /sales/_doc/123
```

create an order with an auto-generated ID

```
POST /sales/_doc
{
  "orderid": "123",
  "orderAmount" : "500"
}
```

delete index

```
DELETE sales
```

bulk ingestion

```
POST _bulk
{"index": {"_index": "bulk_test", "_id": "1" }}
{"col1": "val1" }
{"index": {"_index": "bulk_test", "_id": "2" }}
{"col1": "val2" }
{"index": {"_index": "bulk_test", "_id": "3" }}
{"col1": "val3" }
```

check the results of bulk ingestion

```
GET _cat/indices
```

```
GET bulk_test/_search
```

Update and check the results

```
# update
POST bulk_test/_doc/2/
{
  "name": "Leo",
  "skills": "Data, AI and Programming"
}

# check the results
GET bulk_test/_search

# delete index
DELETE bulk_test
```

As we have already imported some sample data into kibana_sample_data_ecommerce index, let's show all the imported sample data.

```
GET kibana_sample_data_ecommerce/_search
```

Task 3 Using Analyzer API

An analyzer in Elasticsearch is used to explain how the text will be indexed and searched. Only text fields support the analyzer mapping parameter.

Analyzer can be tested by using analyzer API.

analyze the text by using standard analyzer. This is also the default analyzer.

```
POST _analyze
{
  "text": "I found COMP810 very interesting! This is one of my favourite courses ^_^",
  "analyzer": "standard"
}
```

analyze the text by using keyword analyzer. Please compare the difference.

```
POST _analyze
{
  "text": "I found COMP810 very interesting ^_^",
  "analyzer": "keyword"
}
```

analyze a text "array". There is no "array" type, but it uses a concatenation.

```
POST /_analyze
```

```
{
  "text": ["smartphone","desktops","computers"],
  "analyzer": "standard"
}
```

Task 4 Dynamic and Explicit Mapping

In Elasticsearch, the automatic detection and addition of new fields is called **dynamic mapping**. Elasticsearch generates field mappings automatically. When a document is indexed, the index, type, and fields will be generated automatically.

Elasticsearch also allows manually defining mapping, which is called **explicit mapping**.

Previously, we have created a sales index and index a few records. Now, let us use dynamic mapping.

create an order and dynamic mapping, please note that we give **"orderid": 123**, where 123 is not surrounded with double quotation marks.

```
PUT /sales/_doc/123
{
  "orderid": 123,
  "orderAmount" : "500"
}
```

Check the dynamic mapping

```
GET sales/_mapping
```

As can be seen from the results that the orderAmount is created as text field and orderid is long.

```
{
  "sales" : {
    "mappings" : {
      "properties" : {
        "orderAmount" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "orderid" : {
          "type" : "long"
        }
      }
    }
  }
}
```

Create an explicit mapping by specifying the fields and type manually

```
PUT /reviews
{
  "mappings": {
    "properties": {
      "rating": {"type": "float"},
      "content": {"type": "text"},
      "product_id": {"type": "integer"},
      "author": {
        "properties": {
          "first_name": {"type": "text"},
          "last_name": {"type": "text"},
          "email": {"type": "keyword"}
        }
      }
    }
  }
}
```

Check the mapping of reviews index

```
GET /reviews/_mapping
```

Create an explicit mapping by using dot notation

```
PUT /reviews_dot_notation
{
  "mappings": {
    "properties": {
      "rating": {"type": "float"},
      "content": {"type": "text"},
      "product_id": {"type": "integer"},
      "author.first_name": {"type": "text"},
      "author.last_name": {"type": "text"},
      "author.email": {"type": "keyword"}
    }
  }
}
```

Check the mapping of reviews_dot_notation index

```
GET /reviews_dot_notation/_mapping
```


Task 5 Understand Coercion

Coercion attempts to clean up dirty values to fit the data type of a field. For example: Strings will be coerced to numbers, floating points will be truncated for integer values.

Recall that we have re-created sales index by using dynamic mapping, having the mapping presented as follows:

```
{
  "sales" : {
    "mappings" : {
      "properties" : {
        "orderAmount" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "orderid" : {
          "type" : "long"
        }
      }
    }
  }
}
```

Index a document

```
POST /sales/_doc/
{
  "orderid": 777,
  "orderAmount" : "1500"
}
```

Index a document with coercion

```
POST /sales/_doc/
{
  "orderid": "1234",
  "orderAmount" : 1500
}
```

Index a document with coercion

```
POST /sales/_doc/
{
  "orderid": "1234",
  "orderAmount" : true
}
```

See what happens

```
POST /sales/_doc/
{
  "orderid": "abcd",
  "orderAmount": "I input a string as the order amount"
}
```

Coercion fails. “abcd” cannot be coerced to long, but true can be coerced to text.

```
{
  "error" : {
    "root_cause" : [
      {
        "type" : "mapper_parsing_exception",
        "reason" : "failed to parse field [orderid] of type [long] in document with id 'DuxlF3wB8sPiQ0JlRXBa'. Preview of field's value: 'abcd'"
      }
    ],
    "type" : "mapper_parsing_exception",
    "reason" : "failed to parse field [orderid] of type [long] in document with id 'DuxlF3wB8sPiQ0JlRXBa'. Preview of field's value: 'abcd'",
    "caused_by" : {
      "type" : "illegal_argument_exception",
      "reason" : "For input string: \"abcd\""
    }
  },
  "status" : 400
}
```

Reference and Resources

- Elasticsearch Reference [7.x] » REST APIs » Document APIs, <https://www.elastic.co/guide/en/elasticsearch/reference/7.x/docs.html>
- Query DSL (Domain Specific Language), <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>
- Elasticsearch Queries: A Thorough Guide, Jurgens du Toit, Aug 16, 2020, <https://logz.io/blog/elasticsearch-queries/>
- Build your own dashboard, <https://www.elastic.co/guide/en/kibana/7.9/tutorial-build-dashboard.html>
- A Basic Guide to Elasticsearch Aggregations, Daniel Berman, Aug 29, 2019, <https://logz.io/blog/elasticsearch-aggregations/>
- Elasticsearch Java API, <https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/index.html>
- Explore Kibana using sample data, <https://www.elastic.co/guide/en/kibana/current/tutorial-sample-data.html>
- Text similarity search with vector fields, Julie Tibshirani, Aug 28, 2019, <https://www.elastic.co/blog/text-similarity-search-with-vectors-in-elasticsearch>