Aims:

This exercise aims to get you to:

1. Practice the use of ElasticSearch
2. Create an ElasticSearch index
3. Perform operations on the created index including queries using both query strings and ElasticSearch's DSL

Running ElasticSearch Server

ElasticSearch is a NoSQL indexing and search engine built on top of Apache Lucene. It allows you to create indexes over unstructured / semi-structured data / structured data. In order to run ElasticSearch server, first you need to open a terminal (command line tool) in VLab and run the following program:

```
$ 9313
```

This program will setup the environment needed to run ElasticSearch server. You should see an output like the following:

```
$ 9313
Welcome to COMP9313!
newclass starting new subshell for class COMP9313...
```

Next, run the command below to start ElasticSearch server:

```
$ elasticsearch
...
[2019-07-14T17:41:11,470][INFO        ][o.e.h.n.Netty4HttpServerTransport]        [8Uw8EI3]
publish_address {127.0.0.1:9200}, bound_addresses {127.0.0.1:9200}
[2019-07-14T17:41:11,470][INFO ][o.e.n.Node               ] [8Uw8EI3] started
[2019-07-14T17:41:11,479][INFO ][o.e.g.GatewayService     ] [8Uw8EI3] recovered [0] indices
into cluster_state
...
```

If everything goes fine, you should see an output similar to the one above. The server should be able to receive HTTP requests (using ElasticSearch's REST APIs) on http://localhost:9200. You can use the command below to check if ElasticSearch is up and running:

```
$ curl "localhost:9200/"

{
  "name" : "8Uw8EI3",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "1CEB4Ac1QvqBcdI_VzUXHQ",
  "version" : {
    "number" : "6.1.1",
    "build_hash" : "bd92e7f",
    "build_date" : "2017-12-17T20:23:25.338Z",
    "build_snapshot" : false,
    "lucene_version" : "7.1.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

If the server is up and running properly, you should see an output similar to the one shown above. Notice also that two new directories were created in your `HOME` directory, namely, `elasticsearch_data` and `elasticsearch_log`. The first will store all data of the indexes you create in ElasticSearch, while the second one will store logging data.

Create a new index to store vulnerability-related data

Let's now create a new index to store security vulnerability related data. Open a new terminal and run again the program 9313. Then, run the command below:

```
$ curl -X PUT "http://localhost:9200/vuln_demo?pretty"

{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "vuln_demo"
}
```

The command `curl` shown above is a program that allows to send HTTP request to a server. The command above sends a `PUT` HTTP request to ElasticSearch to create an index named `vuln_demo`. If everything goes fine, you should see the output shown above.

You can check the recently created index with the command below:

```
$ curl -X GET "http://localhost:9200/_cat/indices?v"

health status index     uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   vuln_demo b49GKK3XTMyV7FvSrfyJOQ   5   1          0            0      1.1kb          1.1kb
```

The output shows the created index along with additional information such as the number of replicas (`rep`), document counts (`docs.count`), store size (`store.size`), among other details.

Create a new mapping

Now, we will create a new mapping within the index just created. This mapping provides a "schema" for the vulnerability data we want to store in our index.

```
$ curl -X PUT "http://localhost:9200/vuln_demo/vulnerability/_mapping?pretty" -H "Content-
type: application/json" -d "{\"vulnerability\": {\"properties\": {\"id\": {\"type\":
\"text\"},\"description\": {\"type\": \"text\"}, \"cvss_score\": {\"type\": \"double\"}}}}"
{
  "acknowledged" : true
}
```

If the command executes properly, you should see the output `{"acknowledged":true}`.

Create, modify and delete documents

You can ***create*** a new document in the document with the command below:

```
$ curl -X PUT "http://localhost:9200/vuln_demo/vulnerability/CVE-1999-0001?pretty" -H
"Content-type: application/json" -d "{\"id\": \"CVE-1999-0001\",\"description\":\"ip_input.c
in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service
(crash or hang) via crafted packets.\", \"cvss_score\": 5.0}"
{
  "_index" : "vuln_demo",
  "_type" : "vulnerability",
  "_id" : "CVE-1999-0001",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

You can quickly check (retrieve from index) the recently created document with the command below:

```
$ curl -X GET "http://localhost:9200/vuln_demo/vulnerability/_search?pretty"
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "vuln_demo",
        "_type" : "vulnerability",
        "_id" : "_update",
        "_score" : 1.0,
        "_source" : {
          "cvss_score" : 6.5
        }
      },
      {
        "_index" : "vuln_demo",
        "_type" : "vulnerability",
        "_id" : "CVE-1999-0001",
        "_score" : 1.0,
        "_source" : {
          "cvss_score" : 5.0
        }
      }
    ]
  }
}
```

You can also ***update*** a document using the following command:

```
$  curl  -X  POST  "http://localhost:9200/vuln_demo/vulnerability/CVE-1999-0001?pretty"  -H
"Content-type: application/json" -d "{\"cvss_score\":6.5}"
{
  "_index" : "vuln_demo",
  "_type" : "vulnerability",
  "_id" : "CVE-1999-0001",
  "_version" : 4,
  "result" : "updated",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 4,
  "_primary_term" : 1
}
```

To ***delete*** a document, you can use the command below:

```
$ curl -X DELETE "http://localhost:9200/vuln_demo/vulnerability/CVE-1999-0001?pretty"
{
  "_index" : "vuln_demo",
  "_type" : "vulnerability",
  "_id" : "CVE-1999-0001",
  "_version" : 5,
  "result" : "deleted",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 5,
  "_primary_term" : 1
}
```

Querying data

Before providing examples on how to query data, let's create a couple of documents:

```
$  curl  -X  PUT  "http://localhost:9200/vuln_demo/vulnerability/CVE-1999-0001?pretty"  -H
"Content-type: application/json" -d "{\"id\": \"CVE-1999-0001\",\"description\":\"ip_input.c
in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service
(crash or hang) via crafted packets.\", \"cvss_score\": 5.0}"

{
  "_index" : "vuln_demo",
  "_type" : "vulnerability",
  "_id" : "CVE-1999-0001",
  "_version" : 3,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 10,
  "_primary_term" : 1
}
```

```
$  curl  -X  PUT  "http://localhost:9200/vuln_demo/vulnerability/CVE-1999-0002?pretty"  -H
"Content-type:  application/json"  -d  "{\"id\":  \"CVE-1999-0002\",\"description\":\"Buffer
overflow in NFS mountd gives root access to remote attackers, mostly in Linux systems.\",
\"cvss_score\": 10.0}"

{
  "_index" : "vuln_demo",
  "_type" : "vulnerability",
  "_id" : "CVE-1999-0002",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

Let’s now search for any document containing the string “BSD” anywhere in the document:

```
$ curl -X GET "http://localhost:9200/vuln_demo/vulnerability/_search?pretty&q=BSD"
{
  "took" : 10,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.2876821,
    "hits" : [
      {
        "_index" : "vuln_demo",
        "_type" : "vulnerability",
        "_id" : "CVE-1999-0001",
        "_score" : 0.2876821,
        "_source" : {
          "id" : "CVE-1999-0001",
          "description" : "ip_input.c in BSD-derived TCP/IP implementations allows remote
attackers to cause a denial of service (crash or hang) via crafted packets.",
          "cvss_score" : 5.0
        }
      }
    ]
  }
}
```

Next, let’s query our index by field and using Boolean operators (notice that use the string “%20” for representing spaces in the URL below):

```
$ curl -X GET
"http://localhost:9200/vuln_demo/vulnerability/_search?pretty&q=description:(TCP%20OR%20BSD
)"
{
  "took" : 7,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.5753642,
    "hits" : [
      {
        "_index" : "vuln_demo",
        "_type" : "vulnerability",
        "_id" : "CVE-1999-0001",
        "_score" : 0.5753642,
        "_source" : {
          "id" : "CVE-1999-0001",
          "description" : "ip_input.c in BSD-derived TCP/IP implementations allows remote
attackers to cause a denial of service (crash or hang) via crafted packets.",
          "cvss_score" : 5.0
        }
      }
    ]
  }
}
```

Yet, another query including a Boolean expression a little bit more complex that the previous one:

```
$ curl -X GET
"http://localhost:9200/vuln_demo/vulnerability/_search?pretty&q=(description:(Buffer%20AND%
20NFS)%20AND%20cvss_score:>4.0)"

{
  "took" : 7,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.5753641,
    "hits" : [
      {
        "_index" : "vuln_demo",
        "_type" : "vulnerability",
        "_id" : "CVE-1999-0002",
        "_score" : 1.5753641,
        "_source" : {
          "id" : "CVE-1999-0002",
          "description"  :  "Buffer  overflow  in  NFS  mountd  gives  root  access  to  remote
attackers, mostly in Linux systems.",
          "cvss_score" : 10.0
        }
      }
    ]
  }
}
```

Let's now search for vulnerabilities related to "TCP", but let's exclude documents containing "NFS".

```
$ curl -X GET
"http://localhost:9200/vuln_demo/vulnerability/_search?pretty&q=description:%20TCP%20-NFS"

{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.2876821,
    "hits" : [
      {
        "_index" : "vuln_demo",
        "_type" : "vulnerability",
        "_id" : "CVE-1999-0001",
        "_score" : 0.2876821,
        "_source" : {
          "id" : "CVE-1999-0001",
          "description"  :  "ip_input.c  in  BSD-derived  TCP/IP  implementations  allows  remote
attackers to cause a denial of service (crash or hang) via crafted packets.",
          "cvss_score" : 5.0
        }
      }
    ]
  }
}
```

Deleting an index

The command below will delete the index we created for this exercise.

```
$ curl -X DELETE "http://localhost:9200/vuln_demo?pretty"
{
  "acknowledged" : true
}
```

Stopping ElasticSearch server

You can stop the server with the combination of keys CTRL+C in the terminal where the server is running.