Working with the School of Professional Studies Elasticsearch Cluster: Tutorials for Students in Predictive Analytics

Students in Predictive Analytics have access to numerous Linux systems at Northwestern University. The School of Professional Studies Elasticsearch Cluster provides full user access to analytics, document storage, and search software, as well as a path to School of Professional Studies PostgreSQL database server.

Courses in Predictive Analytics utilize R, Python, and SAS as analytics software. R and Python are available on the Elasticsearch Cluster, along with H_2O algorithms for machine learning. SAS is available on the Social Sciences Computing Cluster (SSCC).

R, an object-oriented, open-source language for programming with data, is available worldwide and runs on PC/Windows, Mac/OSX, and Linux/Unix computers. We often begin by using R on our personal laptop or desktop computers. For large or computer-intensive jobs, you can use R with or without H_2O on the Elasticsearch Cluster.

Python is an object-oriented, open-source language. It is a general-purpose programming language especially strong in data and text preparation, as well as a wide range of applications relevant to predictive analytics and data science. Like R, it runs on PC/Windows, Mac/OSX, and Linux/Unix computers. We often begin by using Python on our personal laptop or desktop computers. For large or computer-intensive jobs, you can use Python with or without H_2O on the Elasticsearch Cluster.

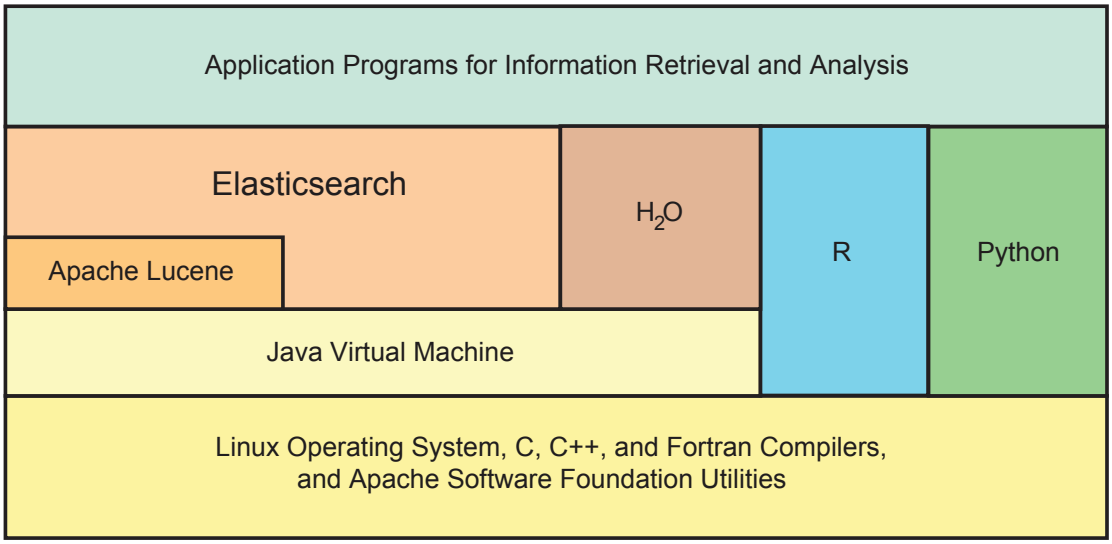
The Elasticsearch Cluster is a number of Linux computers in Evanston, Illinois. The initial configuration of each computer is as follows:

> Operating System: Red Had Enterprise Linux (RHEL 6.7)
> Memory Size: 16 GB
> CPU Count: 4
> Default Server Storage: / 10 GBs | /home 4 GBs | /usr 8 GBs | /var 12 GBs |
> Additional Hard Disk Storage: Data: 200GB

The Elasticsearch Cluster serves as a research and training facility for graduate students in Predictive Analytics. User accounts are not associated with individual courses or instructors. They are for your use only and remain available as long as you maintain a valid Northwestern NetID. Do not share your user account by giving your NetID password to others. Your account is tied to your Northwestern University network identity. Communication between you and SPS IT support for the Elasticsearch Cluster is through Northwestern University e-mail.

Hosea Lee: ubfhosea@northwestern.edu
Renata Oplatek: r-oplate@northwestern.edu

This diagram provides an overview of the Elasticsearch Cluster software stack used in Predictive Analytics courses:



User account holders have the ability to store files on their user accounts. Files can be uploaded and downloaded using software tools employing secure file transfer protocol (sftp). You may use sftp directly on a Mac OS X system or use a file transfer utility such as FileZilla on Mac or Windows.

Programming on the Elasticsearch Cluster means using Red Hat Enterprise Linux. Essential Linux commands include those for directory and file management, such as ls, mkdir, cd, and cp. Remember that Linux is case-sensitive. Course syllabi in Predictive Analytics provide Linux software references. Management of another computer facility, the Social Sciences Computing Cluster, provides a page with recommended Linux reference books at

http://www.it.northwestern.edu/research/sscc/booklist.html

To work on the system, you must first set up a virtual private network (VPN) connection. See this location for instructions:

http://www.it.northwestern.edu/oncampus/vpn/

After your VPN connection has been established, go to the command window on Windows or the Terminal application on Mac and log into the Real-Time Analytics Engine with the following command, substituting your NetID for **netid**:

```
ssh netid@129.105.88.91
```

If the connection is successful, you will be prompted for your password. Quit the **ssh** connection to the Elasticsearch Cluster by typing **logout**

Tutorial 1: Working with an Elasticsearch Index

As its name implies, the Elasticsearch Cluster has Elasticsearch at its core. Elasticsearch is a NoSQL database facility with strong search capabilities. Databases in Elasticsearch are called ***indices*** and indices are composed of documents of various ***types***.

NoSQL database systems, including Elasticsearch, are often described as being "schema-less." It is true that JSON documents may be read into (indexed by) Elasticsearch with no prior specification. Regardless, it is wise to define what is called a ***mapping*** for those types of documents that have a know format, identifying text fields, numeric fields, and dates, for example. An Elasticsearch mapping is analogous to a relational database schema.

The Enron E-mail Archive is available as an index on the Elasticsearch Cluster. The index name is **`enron`**, and there is one document type in this index called **`email`**. Field characteristics within this document type were defined by executing the following code on one of the nodes of the Elasticsearch Cluster. We provide a portion of this code here to show how the mapping was configured. There is no need to run this mapping code when working with the **`enron`** index:

```
curl -XPUT 'elasticsearch-cluster-node-name/enron' -d '{
  "settings":{
    "index.number_of_shards":5,
    "index.number_of_replicas":1,
    "index":{
      "analysis":{
        "analyzer":{
          "myanalyzer":{
            "type":"custom",
            "tokenizer":"uax_url_email"
          }
        }
      }
    }
  },
  "mappings":{
    "email":{
      "properties":{
        "body":{
          "type":"text"
        },
        "headers":{
          "type":"nested",
          "properties":{
            "Date":{
              "type":"date",
              "format":"EEE, dd MMM yyyy HH:mm:ss Z (z)"
            },
            "From":{
              "type":"text",
              "analyzer":"myanalyzer"
            },
```

```
            "Message-ID":{
              "type":"string",
              "include_in_all": false,
              "index": "no"
            },
            "Subject":{
              "type":"text"
            },
            "To":{
              "type":"text",
              "analyzer":"myanalyzer"
            },
            "X-From":{
              "type":"text"
            },
            "X-To":{
              "type":"text"
            },
            "X-bcc":{
              "type":"text"
            },
            "X-cc":{
              "type":"text"
            }
          }
        },
        "mailbox":{
          "type":"text"
        },
        "subFolder":{
          "type":"string",
          "include_in_all": false,
          "index": "no"
        }
      }
    }
  }
}'
```

A review of these mappings shows that most fields are indexed text fields, available for full text searches in the future. There is one date field for filtering searches across date ranges.

Especially important to the indexing process for the Enron E-mail Archive was the specification

```
      "tokenizer":"uax_url_email".
```

This instructs Elasticsearch to maintain e-mail and web URL addresses intact. This is important to the integrity of the data being indexed. We want to be able to search the index for specific e-mail address character strings.

Both Python and R offer client programs for working with Elasticsearch. These are convenience wrappers for accessing the Elasticsearch API. In this initial tutorial, we show how to access the Elasticsearch API directly using curl commands from the Linux bash shell. No additional software or programming is needed.

Assume that you have established your secure shell connection to a node on the Elasticsearch cluster. Elasticsearch is available on this node, accessible through the localhost port 9200. In the near future, you should be able to see the status of the **`enron`** index by typing

```
curl 'enron:spsdata@localhost:9200/enron/_stats?pretty
```

You should see that the enron index consists of 501,512 documents. The characters `?pretty` request pretty printing of the JSON response to this query.

If we wanted to see the top ten documents with matches to the exact word "silverpeak" in the body, we would type

```
curl 'enron:spsdata@localhost:9200/enron/email/_search?pretty' -d '{
  "query": {
    "match": {
      "body": "silverpeak"
    }
  }
}'
```

Alternatively we could search all fields of the email documents:

```
curl 'enron:spsdata@localhost:9200/enron/email/_search?pretty' -d '{
  "query": {
    "match": {
      "_all": "silverpeak"
    }
  }
}'
```

To show that there are fourteen documents that have "silverpeak" across just the `body` and `Subject` fields, we would type

```
curl 'enron:spsdata@localhost:9200/enron/email/_search?pretty' -d '{
  "query": {
    "multi_match": {
      "fields": ["body", "Subject"],
      "query": "silverpeak"
    }
  }
}'
```

Enron executives may have misspelled "silverpeak" in the `body` or `Subject` fields of their e-mails. So we will conduct a fuzzy search, which indicates that there were indeed three misspellings, yielding a total of seventeen documents in the search set:

```
curl 'enron:spsdata@localhost:9200/enron/email/_search?pretty' -d '{
  "query": {
    "multi_match": {
      "fields": ["body", "Subject"],
      "query": "silverpeak",
      "fuzziness": "AUTO"
    }
  }
}'
```

Elasticsearch offers a wide array of query and aggregation options for finding data within indices. These are summarized online at

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

Analysts familiar with Elasticsearch often use the word ***analysis*** to refer to the text preparation and annotation work that is done while indexing (while bringing data into the document database). Data analysis conducted on the document database or index or on filtered versions of the database is referred to as ***aggregation***.

Here are a few excellent references to consult regarding the Elasticsearch, the language of search, text analytics and information retrieval:

Buttcher, S., Clarke, C. L. A., and Cormack, Gordon V. (2010). *Information Retrieval: Implementing and Evaluating Search Engines.* Cambridge, Mass.: MIT Press. [ISBN-13: 978-0262026512]

Gheorghe, R., Hinman, M.L., and Russo, R. (2016). *Elasticsearch in Action.* Shetler Island, N.Y.: Manning. [ISBN-13: 978-1617291623]

Gormley, C. and Tong, Z. (2015). *Elasticsearch Search: The Definitive Guide.* Sebastopol, Calif.: O'Reilly. [ISBN-13: 978-1449358549]

Manning, C. D., Raghaven, P., and Schutze, H. (2008) *Introduction to Information Retrieval.* Cambridge, UK: Cambridge University Press. [ISBN-13: 978-0521865715] Available online at http://nlp.stanford.edu/IR-book/information-retrieval-book.html

Turnbull, D. and Berryman, J. (2016). *Relevant Search with Applications for Solr and Elasticsearch.* Shetler Island, N.Y.: Manning. [ISBN-13: 978-1617292774]

Tutorial 2: Accessing Elasticsearch from Python with Requests

It is possible to connect directly with an Elasticsearch index using Python and the requests package while working on the Elasticsearch Cluster. A sample Python program has been provided as jump-start code, where **xxx** is the current version number: **requests-enron-jump-start-vXXX.py**

```
# querying Elasticsearch with the requests package

# prepare for Python version 3x features and functions
from __future__ import division, print_function

import requests  # connection to web server or localhost
import json  # JSON-dictionary in/out
import pprint  # pretty printing of JSON

def search(index_location, headers, search_term):
    query = json.dumps({
        "query": {
            "match": {
                "body": search_term
            }
        }
    })

    response = requests.get(index_location, index_headers,
        data = query)
    results = json.loads(response.text)
    return results

if __name__ == '__main__':
# set location of index and x-pack security user and password
information
    index_location = 'http://enron:spsdata@localhost:9200/enron/email/_search'

    index_headers = '' # requests parameter headers not needed

# query/search term specified in call
# to the user-defined search function
    results = search(index_location, index_headers,
        search_term = "silverpeak")

# pretty-printing of JSON results to console
# comment out this pprint statement if search results are extensive
    pprint.pprint(results)

# save the results as JSON-formatted text file in current directory
    json_filename = 'search-results.json'
    with open(json_filename, 'w') as outfile:
        json.dump(results, outfile, encoding = 'utf-8')
```

On your personal computer locate yourself in the directory containing this program. Establish a VPN connection, go to a bash-enabled command window on Windows or the Terminal application on Mac and use the secure file transfer protocol to connect to the Elasticsearch Cluster, command, substituting your NetID for **netid**:

```
sftp netid@129.105.88.91
```

If the connection is successful, you will be prompted for your password.

Check the path to your current working directory, make a new directory for your Python work, place the Python program within that directory, check that the file has been successfully uploaded, and disconnect from the sftp connection:

```
pwd
mkdir python-work
put requests-enron-jump-start-vXXX.py
ls –la
quit
```

You are now ready to run the Python jump-start program on the Elasticsearch Cluster. Assuming that the VPN connection is still active, go to a bash-enabled command window on Windows or the Terminal application on Mac and log into the Elasticsearch Cluster with the following command, substituting your NetID for **netid**:

```
ssh netid@129.105.88.91
```

If the connection is successful, you will be prompted for your password. Later you will quit the **ssh** connection to the Elasticsearch Cluster by typing **logout**

Identify the path to your current working directory, switch to the python-work directory, and execute the Python program (where **XXX** is the version number of the Python program) and check for a new file called **search-results.json**:

```
pwd
cd python-work
python requests-enron-jump-start-vXXX.py
ls –la
```

The file **search-results.json** contains the search results of the query across email documents in the Elasticsearch enron index. You can examine these results directly on the Elasticsearch cluster host using more or less, or by loading these results in a host-based text editor, such as **vi**, **vim**, **emacs**, or **nano**. To examine the results in **nano**, which is an easy-to-use text editor on Linux, type

```
nano search-results.json
```

You may may want to use `nano` to edit the Python program as well. While in the **`python-work`** directory, type

```
nano requests-enron-jump-start-vXXX.py
```

where **`XXX`** is the version number of the Python program.

The text editor **`nano`** uses the console as its work area, responds to up-arrow and down-arrow keystrokes, and provides control-key command instructions at the bottom of the screen. The main instructions you will need are `control-O` for saving your edits to a file and `control-X` to exit the program.

After you have completed your Elasticsearch index work and have your search results output, you may well want to complete your work on your personal computer, in which case you will want to log out of your secure shell connection (`ssh`) and utilize secure file transfer protocol (`sftp`) to download your file to your personal computer. To download the file to your personal computer, use **`GET`** in place of **`PUT`** within `sftp`:

```
sftp netid@129.105.88.91
```

If the connection is successful, you will be prompted for your password.

Check the path to your current working directory, locate yourself within the **`python-work`** directory, and download the **`search-results.json`** file to your personal computer:

```
pwd
cd python-work
ls –la
get search-results.json
quit
```

You can now work with **`search-results.json`** on your personal computer, using the Python `json` package and other tools for text manipulation, exploratory data analysis, and modeling.

This second tutorial shows that standard Python packages may be used to access an Elasticsearch index, working on the Elasticsearch Cluster itself. Python programs have distinct advantages over `bash curl` commands, because we have the entire Python ecosystem at our disposal. Requests to Elasticsearch are JSON-formatted. Results are JSON-formatted, but by using the `json` package from the standard Python library, we are able to convert JSON objects into Python dictionaries and then work with alternative data structures. These capabilities are especially valuable when working with large collections of search results.