**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 H2O 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 H2O 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 H2O 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](mailto:ubfhosea@northwestern.edu)

Renata Oplatek: [r-oplate@northwestern.edu](mailto: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**](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/**](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:

curl -XPUT *'elasticsearch-cluster-node-name*/enron' -d '{

"settings":{

"index.number\_of\_shards":1,

"index.number\_of\_replicas":0,

"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"

}

}

}

}

}'

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. You can see the status of the enron index by typing

curl -XGET 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 'localhost:9200/enron/email/\_search?pretty' -d '{

"query": {

"match": {

"body": "silverpeak"

}

}

}'

Alternatively we could search all fields of the email documents:

curl '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 'localhost:9200/enron/email/\_count?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 conduce a fuzzy search, which indicates that there were indeed three misspellings, yielding a total of seventeen documents in the search set:

curl 'localhost:9200/enron/email/\_count?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

You may also want to consult one of these two primary references for Elasticsearch:

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]

**Tutorial 2: R Interactive and Batch Jobs**

For this first example, you can build a logistic regression model fit to data from the Sydney Transportation Study. The data are in the comma-delimited text file **sydney.csv** and the program is called **run\_sydney\_jump\_start.R**. Set up a directory **sydney\_work** and try running the R program.

You can run it as you normally would in interactive mode within the R console on your personal computer or within RStudio. Run the entire program interactively by typing

**source("run\_sydney\_jump\_start.R")**

The program produces two output files: **sydney\_results\_text.txt** and **sydney\_results\_plot.pdf**.

Having checked interactive execution on a personal computer, you can upload both the data and program to the Elasticsearch Cluster**.** Before beginning your connection to the Elasticsearch Cluster, locate yourself in the folder that contains the two files to be uploaded. Then connect to the Elasticsearch Cluster using the secure file transfer protocol from a utility on Windows or directly from the Terminal application on a Mac. Use your NetID in place of **netid**:

**sftp netid@129.105.88.91**

You will be asked for your password. After you enter your password, you should see the prompt **sftp>**

Set up a working directory under your home folder on the Elasticsearch Cluster and locate yourself in that directory:

**mkdir sydney\_work**

**cd sydney\_work**

You can check your location by typing **pwd** You transfer the data and program files and then quit **sftp**:

**put sydney.csv**

**put run\_sydney\_jump\_start.R**

**quit**

Now connect to the Elasticsearch Cluster using the secure shell, locate yourself in the working directory, set up a Linux process to record your work, and run the R program interactively:

**ssh netid@129.105.88.91**

**cd sydney\_work**

**script sydney\_work.log**

**R**

**source("run\_sydney\_jump\_start.R")**

**q()**

After entering the R console, you will see the **>** prompt for your commands until you quit the R session. Then you can check the working directory for the text output file and print the file to the command/Terminal window:

**ls -la**

**more sydney\_results\_text.txt**

When using the **more** process, you see a portion of the file and cursor through the file by hitting the return key to see additional lines. Or you can type **control c** to end the **more** process. The **more** process limits the amount of text displayed in the terminal window. A **cat** process, on the other hand, would show the entire file.

Remove the output files and check that they have been removed:

**rm sydney\_results\_text.txt**

**rm Sydney\_results\_plot.pdf**

**ls -la**

Now you are ready to run the R program again, this time in batch:

**R CMD BATCH run\_sydney\_jump\_start.R**

In batch mode, R creates a second file automatically. This is listing of what the R would have shown had the program been run interactively from the console. In this example, **run\_sydney\_jump\_start.Rout** is the name of the file.You can review its contents with the **more** process.

Many classroom assignments ask you to provide complete documentation for your work, including comma-delimited data files, R or Python programs as plain text files, output from those programs, and (when using remote systems such as the Elasticsearch Cluster) a log of the programming session.

When you are finished collecting information from the Linux session, you can type **exit** to exit the script logging application. The session log should be in file **sydney\_work.log** in the working directory on Linux because that is the name of the file you defined when beginning the **script** process. You can look at the log while connected to the Elasticsearch Cluster, again using a **more** process:

**more sydney\_work.log**

You quit the **ssh** connection to the Elasticsearch Cluster by typing **logout**

To obtain a record of your work on the Elasticsearch Cluster, you can again connect using a file transfer utility on Windows or directly with **sftp** on a Mac. Check the contents of the working directory and download the output and log files:

**sftp netid@129.105.88.91**

**cd sydney\_work**

**ls –la**

**get sydney\_results\_text.txt**

**get sydney\_results\_plot.pdf**

**get run\_sydney\_jump\_start.Rout**

**get sydney\_work.log**

**quit**

This first project does not require that any editing on the Elasticsearch Cluster itself. When editing is required on the Elasticsearch Cluster, you need to select a Linux-based editor or work within an integrated development environment (IDE). One option is the **vim** editor, with training information provided on Linux by **vimtutor**

Much program development and testing work can and should be conducted on your personal computer. Then after you have completed your coding and testing, you can transfer data and program files to the Elasticsearch Cluster for execution. If you are working on a Mac OS X system, commands in the Terminal application are similar to Linux commands.

Some assignments may call for an evaluation of processing times, which may be obtained within the R programming environment using the **system.time()** function in interactive mode. System time refers to the central processor system time need to run the process or job. For the Sydney example, you could type

**system.time(source("run\_sydney\_jump\_start.R"))**

For R batch jobs, the **Rout** listing shows processing times from **proc.time()**, including the system, user, and elapsed time. The precise definitions of these times vary from one computing environment to the next. To obtain processing times that are comparable across computing environments, it may be best to include code within the R program itself to obtain wall-clock or elapsed time. Use the system clock to bracket the code of interest, as shown here:

**start\_time <- Sys.time()**

**[R code to be timed]**

**end\_time <- Sys.time()**

**runtime <- end\_time - start\_time**

Alternatively, Linux commands may be used to obtain system times at the beginning and end of a process. For example, use **date %s** for clock time in seconds.

**Tutorial 3: R/H20 Interactive and Batch Jobs**

For this second example, you can fit a kmeans cluster analysis model fit to data for Major League Baseball players. Work from the zip archive **h2o\_work.zip**. The data are in the comma-delimited file **mlb\_batting\_cluster\_input.csv** and the program is called **run\_h2o\_jump\_start.R**. Set up a directory **h20\_work** on the Elasticsearch Cluster and upload the necessary files using **sftp** directly from a Mac OS X terminal window or using a secure file transfer protocol utility on Windows . Use your NetID in place of **netid**:

**sftp netid@129.105.88.91**

You will be asked for your password. While in **sftp**, Set up a working directory under your home folder and locate yourself in that directory:

**mkdir h2o\_work**

**cd h20\_work**

You can check your location by typing **pwd** and transfer the data and program files, and then quit **sftp**:

**put mlb\_batting\_cluster\_input.csv**

**put run\_h2o\_jump\_start.R**

**quit**

Now connect to the Elasticsearch Cluster using the secure shell, locate yourself in the working directory, set up a Linux process to record your work, and run the R program interactively:

**ssh netid@129.105.88.91**

**cd h2o\_work**

**script h2o\_work.log**

**R**

**source("run\_h2o\_jump\_start.R")**

**q()**

After entering the R console, you will see the **>** prompt for your commands until you quit the R session. Then you can check the working directory for the text output file and print the file to the command/Terminal window:

**ls -la**

**more h2o\_results\_text.txt**

When using the **more** process, cursor through the file by hitting the return key to see additional lines. Or you can type **control c** to end the **more** process. Or use a **cat** process to list the entire file.

Remove the output files and check that they have been removed:

**rm h2o\_results\_text.txt**

**rm h20\_results\_cluster.csv**

**ls -la**

Now you are ready to run the R program again, this time in batch:

**R CMD BATCH run\_h2o\_jump\_start.R**

In batch mode, R creates a second file automatically. This is a listing of what R would have shown had the program been run interactively from the console. In this example, **run\_h2o\_jump\_start.Rout** is the name of the file.You can review its contents with the **more** process.

Many classroom assignments ask you to provide complete documentation for your work, including comma-delimited data files, R or Python programs as plain text files, output from those programs, and (when using a remote system like the Elasticsearch Cluster) a log of the programming session.

When you are finished collecting information from the Linux session, you can type **exit** to exit the script logging application. The session log should be in file **h2o\_work.log** in the working directory on Linux because that is the name of the file you defined when beginning the **script** process. You can look at the log while connected to the Elasticsearch Cluster, again using a **more** process:

**more h2o\_work.log**

You quit the **ssh** connection to the Elasticsearch Cluster by typing **logout**

To obtain a record of your work, you can again connect using a file transfer utility on Windows or directly with **sftp** on a Mac. Check the contents of the working directory and download the output and log files:

**sftp netid@129.105.88.91**

**cd h2o\_work**

**ls –la**

**get h2o\_results\_text.txt**

**get h20\_results\_cluster.csv**

**get run\_h2o\_jump\_start.Rout**

**get h2o\_work.log**

**quit**