BIG DATA CYBERSECURITY

LAB 5 NEW ENRICHMENT

ADDING A NEW ENRICHMENT TO APACHE METRON

Lab Description: In this assignment, we will add an enrichment to the Squid telemetry source created in Lab 3 [1]. WhoIs information will be added to the Squid dataset. This assignment repeats several steps of Lab 4.

Lab Files that are Needed: metron-cluster.ppk.

LAB EXERCISE/STEP 1

In the AWS cloud, create a new Metron virtual machine (VM) comprising a single-node Hadoop cluster as described in Lab 1. Establish an SSH connection with the VM as specified in Steps 1 through 4 of Lab 2. Note: it is recommended setting up a new VM for each lab to avoid misconfiguration because of the settings kept from other assignments.

LAB EXERCISE/STEP 2

A short sample of WhoIs data in comma-separated-values (CSV) format is given below. Copy the data into the buffer of your computer:

```
france24.com,FRANCE MEDIAS MONDE,FR,FRANCE MEDIAS MONDE,1133395200
linkedin.com,LinkedIn Corporation,US,LinkedIn Corporation,1036195200
csoonline.com,"CXO MEDIA, INC.",US,"CXO MEDIA, INC.",1016668800
wikipedia.org,"Wikimedia Foundation, Inc.",US,"Wikimedia Foundation, Inc.",979257600
ionos.com,PrivateName Services Inc.,CA,PrivateName Services Inc.,951782400
techtarget.com,"TechTarget, Inc.",US,"TechTarget, Inc.",937353600
godaddy.com,"Go Daddy Operating Company, LLC",US,"Go Daddy Operating Company, LLC",920332800
eenews.net,E&E NEWS,US,E&E NEWS,907027200
```

```
google.com,Google LLC,US,Google LLC,874281600
tripwire.com,"Domain Protection Services, Inc. ",US,"Domain Protection Services, Inc.",818121600
time.com,Time Inc.,US,Time Inc.,753580800
bloomberg.com,Bloomberg Finance L.P.,US,Bloomberg Finance L.P.,749260800
wired.com,Conde Nast Publications Inc.,US,Conde Nast Publications Inc.,722217600
norton.com,Symantec Corporation,US,Symantec Corporation,684979200
cisco.com,Cisco Technology Inc.,US,Cisco Technology Inc.,547948800
```

In the terminal window connected to the single-node Hadoop cluster, switch to the root user in order to run commands as the superuser:

```
sudo su
```

Navigate to the /root directory and create a file named whois_ref.csv with the commands below:

```
cd ~
vi whois_ref.csv
```

Press i to switch the vi editor to the Insert mode. Paste the WhoIs data from the buffer. If one or more empty lines appear in the end of the file, delete them.

Make a screenshot and store it in the submission document.

Press Escape.

```
:wq
```

LAB EXERCISE/STEP 3

The schema of this enrichment is

domain | owner | registeredCountry | registeredTimestamp.

Make sure you don't have an empty newline character as the last line of the CSV file, as that will result in a null pointer exception. The first thing we need to do is setup the enrichment source. In order to do this we first need to setup the extractor config:

Formatted JSON Data

```
{
  "config":{
    "columns":{
      "domain":0,
      "owner":1,
      "home_country":2,
      "registrar":3,
      "domain_created_timestamp":4
    },
    "indicator_column":"domain",
    "type":"whois",
    "separator":","
  },
  "extractor":"CSV"
}
```

Cut and paste this data into a file called extractor_config_temp.json on the virtual machine.

```
vi extractor_config_temp.json
```

Press i. Paste the data. Press Escape.

```
:wq
```

Copying and pasting may include some non-ascii invisible characters. To remove them run this command:

```
iconv -c -f utf-8 -t ascii extractor_config_temp.json -o
extractor_config.json
```

LAB EXERCISE/STEP 4

Another configuration must be done for Zookeeper enrichment.

```
{
  "zkQuorum":"metronserver.localdomain:2181",
  "sensorToFieldList":{
    "squid":{
      "type":"ENRICHMENT",
      "fieldToEnrichmentTypes":{
        "domain_without_subdomains":[
          "whois"
        ]
      }
    }
  }
}
```

Cut and paste this data into a file called enrichment_config_temp.json on the virtual machine.

```
vi enrichment_config_temp.json
```

Press i. Paste data. Press Escape.

```
:wq
```

Copying and pasting may include some non-ascii invisible characters. To remove them run this command:

```
iconv -c -f utf-8 -t ascii enrichment_config_temp.json -o
enrichment_config.json
```

Which means that the system will map the whois enrichment to the field URL. Then execute the following command:

```
${METRON_HOME}/bin/flatfile_loader.sh -n enrichment_config.json -i
whois_ref.csv -t enrichment -c t -e extractor_config.json
```

The last line of the output should be *INFO zookeeper.ClientCnxn: EventThread shut down*.

Make a screenshot and store it in the submission document.

LAB EXERCISE/STEP 5

Enrichment data will be loaded into Hbase and a Zookeeper mapping will be established. The data will be stored in the HBase table called enrichment.

To verify that the logs were properly ingested into HBase run the following command

```
echo "scan 'enrichment'" | hbase shell
```

Make a screenshot and store it in the submission document.

You should see the table bulk loaded with data from the CSV file. Now check if Zookeeper enrichment tag was properly populated:

```
${METRON_HOME}/bin/zk_load_configs.sh -m DUMP -z $ZOOKEEPER
```

This command displays configuration in the standard output. Among other entries, you should find one named squid.

LAB EXERCISE/STEP 6

Create a Kafka topic for the Squid data source

By design, in Metron architecture each new telemetry data source must be accompanied with a Kafka topic, which becomes a part of a Metron data stream. To create a new topic, execute the following command:

```
${HDP_HOME}/kafka-broker/bin/kafka-topics.sh --zookeeper $ZOOKEEPER
--create --topic squid --partitions 1 --replication-factor 1
```

To verify that the squid topic has been successfully created, list all existing Kafka topics:

```
${HDP_HOME}/kafka-broker/bin/kafka-topics.sh --zookeeper $ZOOKEEPER
--list
```

The output of this command should show squid among other topics.

Make a screenshot and store it in the submission document.

The following commands will send the Squid access.log to the corresponding topic we created earlier:

```
cat /var/log/squid/access.log | ${HDP_HOME}/kafka-broker/bin/kafka-
console-producer.sh --broker-list $BROKERLIST --topic squid
```

LAB EXERCISE/STEP 7

Ingest Squid logs into Kafka using the command below.

```
${HDP_HOME}/kafka-broker/bin/kafka-console-consumer.sh --zookeeper
$ZOOKEEPER --topic squid --from-beginning
```

After it displays approximately 20 lines of output, press the

```
Control C
```

key combination on the keyboard to terminate this command. Now we are ready to tackle the Metron parsing topology setup.

Commonly, log files contain semi-structured data—custom rules are required to extract information. Metron offers parsers for log files ingestion—software components transforming raw data into the JSON format [2]. In this assignment, we employ a Grok-based parser to extract data from Squid logs. Grok uses text and regular expression patterns to match textual data in log files or other data sources. Information can be extracted from the default Squid access log by applying the Grok code listed below [3-5].

```
SQUID_DELIMITED %{NUMBER:timestamp}[^0-9]*%{INT:elapsed}
%{IP:ip_src_addr} %{WORD:action}/%{NUMBER:code}
%{NUMBER:bytes} %{WORD:method} %{NOTSPACE:url}[^0-
9]*(%{IP:ip_dst_addr})?
```

By design, Metron requires parses to be stored in /apps/metron/patterns/ directory in HDFS. To place the Grok parser in that directory, we need to create a file in local Linux file system containing the Grok code above, save that file and then copy it to HDFS. To create a file in local file system and open it with the vi text editor:

```
vi /tmp/squid
```

Press the i button on the keyboard to switch vi to the Insert mode allowing to enter or paste new text. Now copy and paste (or type) the Squid code listed above into the new file. To save the results, press the Escape button and the following key combination:

```
:wq
```

LAB EXERCISE/STEP 8

We need to switch to the hdfs user, because this user has access privileges for the HDFS file system. To do so:

```
su - hdfs
```

Run this command to copy the file from the local file system /tmp/squid to the /apps/metron/patterns directory in HDFS:

```
hadoop fs -put -f /tmp/squid /apps/metron/patterns/
```

Make a screenshot and store it in the submission document.

And switch back to the root user:

```
exit
```

We need to define a parser configuration for the Metron Parsing Topology, which is responsible for processing a sensor input in its native format and converting it to the Metron JSON format. The parsing topology is formed by two components: (1) a Storm Kafka Spout for reading from a Kafka topic and sending data to a Storm topology and (2) Metron parser Kafka Bolt for parsing messages and sending them into a Kafka enrichment topic. Apache Zookeeper is responsible for preserving the configurations. Thus, the sensor configuration must be uploaded there after it has been created.

Create a Squid Grok parser configuration file:

```
vi ${METRON_HOME}/config/zookeeper/parsers/squid.json
```

Switch to the insert mode by pressing the i button. Check the file content. It must look exactly as listed below. If it is different, make corrections accordingly.

```
{
  "parserClassName": "org.apache.metron.parsers.GrokParser",
  "sensorTopic": "squid",
  "parserConfig": {
    "grokPath": "/patterns/squid",
    "patternLabel": "SQUID_DELIMITED",
    "timestampField": "timestamp"
  },
  "fieldTransformations" : [
     {
     "transformation" : "STELLAR"
    ,"output" : [ "full_hostname", "domain_without_subdomains" ]
    ,"config" : {
                    "full_hostname" : "URL_TO_HOST(url)"
                   ,"domain_without_subdomains" :
"DOMAIN_REMOVE_SUBDOMAINS(full_hostname)"
                   }
     } ]
}
```

To exit the Insert mode, press Escape. To save the file and exit, type:

```
:wq
```

And press Enter. In the parser configuration above, the fieldTransformations element uses the Stellar language in the parser configuration [6]. The Grok Parser is set up to extract complete URLs. For the purpose of this assignment, domain names without subdomains will be sufficient.

Stellar will be used for this field transformation. It supports multiple network-related string-processing functions and other operations. In the configuration above, the URL_TO_HOST function is applied to extract a hostname from a URL. For example, URL_TO_HOST(‘http://www.yahoo.com/foo’) would yield ‘www.yahoo.com’. Another function, DOMAIN_REMOVE_SUBDOMAINS deletes subdomains from a URL, e.g. DOMAIN_REMOVE_SUBDOMAINS(‘mail.yahoo.com’) yields ‘yahoo.com’. Thus, two new fields are added to each message, "full_hostname" and "domain_without_subdomains".

LAB EXERCISE/STEP 9

We need to setup index types and batch sizes by adding the lines below to the squid.json file. Edit the file for editing:

```
vi ${METRON_HOME}/config/zookeeper/indexing/squid.json
```

Switch to the insert mode by pressing i. Now, paste the lines into the file:

```
{
  "hdfs":{
    "index":"squid",
    "batchSize":5,
    "enabled":true
  },
  "elasticsearch":{
    "index":"squid",
    "batchSize":5,
    "enabled":true
  },
  "solr":{
    "index":"squid",
    "batchSize":5,
    "enabled":true
  }
}
```

Press the Escape button to exit the Insert mode and type

```
:wq
```

To save and exit the file.

LAB EXERCISE/STEP 10

In order to ensure the source IP and destination IP addresses are in the proper format, we will validate the messages. To do so, we will edit a global Metron configuration in the JSON format and push it into Zookeeper, which will perform the validation. Open the file for editing:

```
vi ${METRON_HOME}/config/zookeeper/global.json
```

The file already contains information. Remember to switch to the Insert mode. Then, locate the line containing *“parser.error.topic” : “indexing”,* make a new line right below it, and paste the following lines for IPv4 validation there:

```
"fieldValidations":[
  {
    "input":[
      "ip_src_addr",
      "ip_dst_addr"
    ],
    "validation":"IP",
    "config":{
      "type":"IPV4"
    }
  }
],
```

After pasting, content indentation may be lost. Correct formatting to keep the content readable. Then save and exit the file. Metron has a script to upload configurations to Zookeeper. To upload the changes we have just completed, run the command below. Notice that it uses the PUSH option.

```
${METRON_HOME}/bin/zk_load_configs.sh -i
${METRON_HOME}/config/zookeeper -m PUSH -z $ZOOKEEPER
```

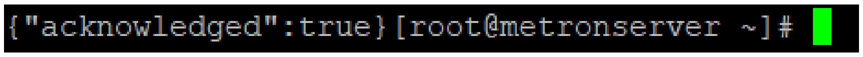
LAB EXERCISE/STEP 11

Create a new index in Elasticsearch for Squid data by executing the command below:

```
curl -XPUT 'http://metronserver.localdomain:9200/_template/squid_index'
-d '
{
  "template":"squid_index*",
  "mappings":{
    "squid_doc":{
      "dynamic_templates":[
        {
          "geo_location_point":{
            "match":"enrichments:geo:*:location_point",
            "match_mapping_type":"*",
            "mapping":{
              "type":"geo_point"
            }
          }
        },
        {
          "geo_country":{
            "match":"enrichments:geo:*:country",
            "match_mapping_type":"*",
            "mapping":{
              "type":"keyword"
            }
          }
        },
        {
          "geo_city":{
            "match":"enrichments:geo:*:city",
            "match_mapping_type":"*",
            "mapping":{
              "type":"keyword"
            }
          }
        },
        {
          "geo_location_id":{
            "match":"enrichments:geo:*:locID",
            "match_mapping_type":"*",
            "mapping":{
              "type":"keyword"
            }
          }
        },
        {
          "geo_dma_code":{
            "match":"enrichments:geo:*:dmaCode",
            "match_mapping_type":"*",
            "mapping":{
              "type":"keyword"
            }
          }
        },
        {
          "geo_postal_code":{
            "match":"enrichments:geo:*:postalCode",
            "match_mapping_type":"*",
            "mapping":{
              "type":"keyword"
            }
          }
        },
        {
          "geo_latitude":{
            "match":"enrichments:geo:*:latitude",
            "match_mapping_type":"*",
            "mapping":{
              "type":"float"
            }
          }
        },
        {
          "geo_longitude":{
            "match":"enrichments:geo:*:longitude",
            "match_mapping_type":"*",
            "mapping":{
              "type":"float"
            }
          }
        },
        {
          "timestamps":{
            "match":"*:ts",
            "match_mapping_type":"*",
            "mapping":{
              "type":"date",
              "format":"epoch_millis"
            }
          }
        },
        {
          "threat_triage_score":{
            "mapping":{
              "type":"float"
            },
            "match":"threat:triage:*score",
            "match_mapping_type":"*"
          }
        },
        {
          "threat_triage_reason":{
            "mapping":{
              "type":"text",
              "fielddata":"true"
            },
            "match":"threat:triage:rules:*:reason",
            "match_mapping_type":"*"
          }
        },
        {
          "threat_triage_name":{
            "mapping":{
              "type":"text",
              "fielddata":"true"
            },
            "match":"threat:triage:rules:*:name",
            "match_mapping_type":"*"
          }
        }
      ],
      "properties":{
        "timestamp":{
          "type":"date",
          "format":"epoch_millis"
        },
        "source:type":{
          "type":"keyword"
        },
        "ip_dst_addr":{
          "type":"ip"
        },
        "ip_dst_port":{
          "type":"integer"
        },
        "ip_src_addr":{
          "type":"ip"
        },
        "ip_src_port":{
          "type":"integer"
        },
        "alert":{
          "type":"nested"
        },
        "guid":{
          "type":"keyword"
        }
      }
    }
  }
}
'
```

LAB EXERCISE/STEP 12

The output of this command should like this:



```
{"acknowledged":true}[root@metronserver ~]#
```

Make a screenshot and store it in the submission document.

Start the new squid parser topology by executing the following line:

```
${METRON_HOME}/bin/start_parser_topology.sh -k $BROKERLIST -z
$ZOOKEEPER -s squid
```

Install the Squid Proxy Server.

Execute the following command to install Squid.

```
yum install squid -y
```

Start Squid by adjusting a general command to start a service in CentOS 7:

```
service squid start
```

LAB EXERCISE/STEP 13

Simulate user activity and populate Squid access log

At this point the log file should be empty since the Squid proxy server has just been installed on the system. We will simulate activity of proxy server users visiting various URLs with the use of *squidclient*, the Squid testing utility. To do so, run the following lines:

We are interested in access.log as that is the log that records the proxy usage. We see that initially the log is empty. Lets generate a few entries for the log.

```
squidclient "https://www.france24.com/en/tag/cyber-security/"

squidclient "https://www.linkedin.com/school/fairleigh-dickinson-university/about/"

squidclient "https://www.csoonline.com/article/3365239/congress-steers-clear-of-industrial-control-systems-cybersecurity.html"

squidclient "https://en.wikipedia.org/wiki/Computer_security"

squidclient "https://www.ionos.com/digitalguide/server/security/what-is-cybersecurity-the-current-dangers/"

squidclient "https://searchsecurity.techtarget.com/"

squidclient "https://www.godaddy.com/help/what-is-website-security-26451"

squidclient "https://www.eenews.net/energywire/2018/10/02/stories/1060100239"

squidclient "https://www.google.com/advanced_search"

squidclient "https://www.tripwire.com/misc/tripwire-cybersecurity-for-the-modern-enterprise/"

squidclient "http://time.com/tag/cybersecurity/"

squidclient "https://www.bloomberg.com/cybersecurity"

squidclient "https://www.wired.com/tag/cybersecurity/"

squidclient "https://us.norton.com/internetsecurity-how-to-cyber-security-best-practices-for-employees.html"
```

```
squidclient
"https://www.cisco.com/c/en/us/solutions/industries/government/defense-cybersecurity.html"
```

When the Squid proxy is used in a real-world setting in the forward mode, users' would change their web browser settings to specify IP address and port where Squid operates, in order to browse internet. Thus Squid would follow its configuration and collect URLs and other related information. We used the *squidclient* utility to simulate users navigating URLs as listed above. Squid access.log should now have several entries, which resembling the ones below.

```
1551302847.374    138 ::1 TCP_MISS/200 14345 GET
https://docs.hortonworks.com/HDPDocuments/HCP1/HCP-1.6.1/installation/content/introduction_to_metron.html -
HIER_DIRECT/13.32.250.64 text/html
1551302855.636    354 ::1 TCP_MISS/200 75016 GET
https://www.nsa.gov/What-We-Do/Cybersecurity/NCX/ -
HIER_DIRECT/23.218.61.50 text/html
1551302855.946    303 ::1 TCP_MISS/200 47716 GET https://niccs.us-cert.gov/formal-education/integrating-cybersecurity-classroom -
HIER_DIRECT/23.218.69.42 text/html
1551302857.174   1221 ::1 TCP_MISS/200 96902 GET
https://www.itpro.co.uk/hacking/30282/what-is-ethical-hacking-white-hat-hackers-explained - HIER_DIRECT/54.247.85.154 text/html
```

Squid format may be customized by including additional or removing existing features. By default, access log has the following attributes:

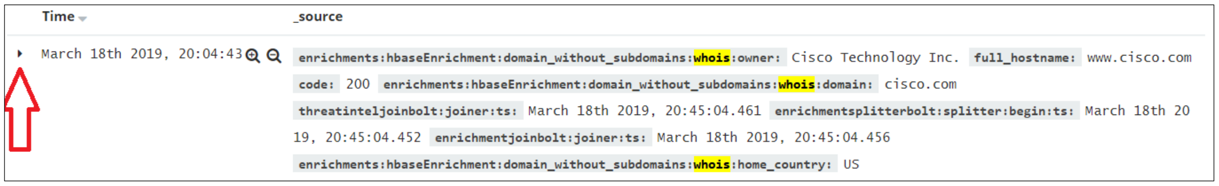
timestamp | *time elapsed* | *remotehost* | *code/status* | *bytes* | *method* | *URL rfc931* | *peerstatus/peerhost* | *type*

With the squid parser topology, generate data by running this command:

```
tail /var/log/squid/access.log | ${HDP_HOME}/kafka-broker/bin/kafka-console-producer.sh --broker-list $BROKERLIST --topic squid
```

LAB EXERCISE/STEP 14

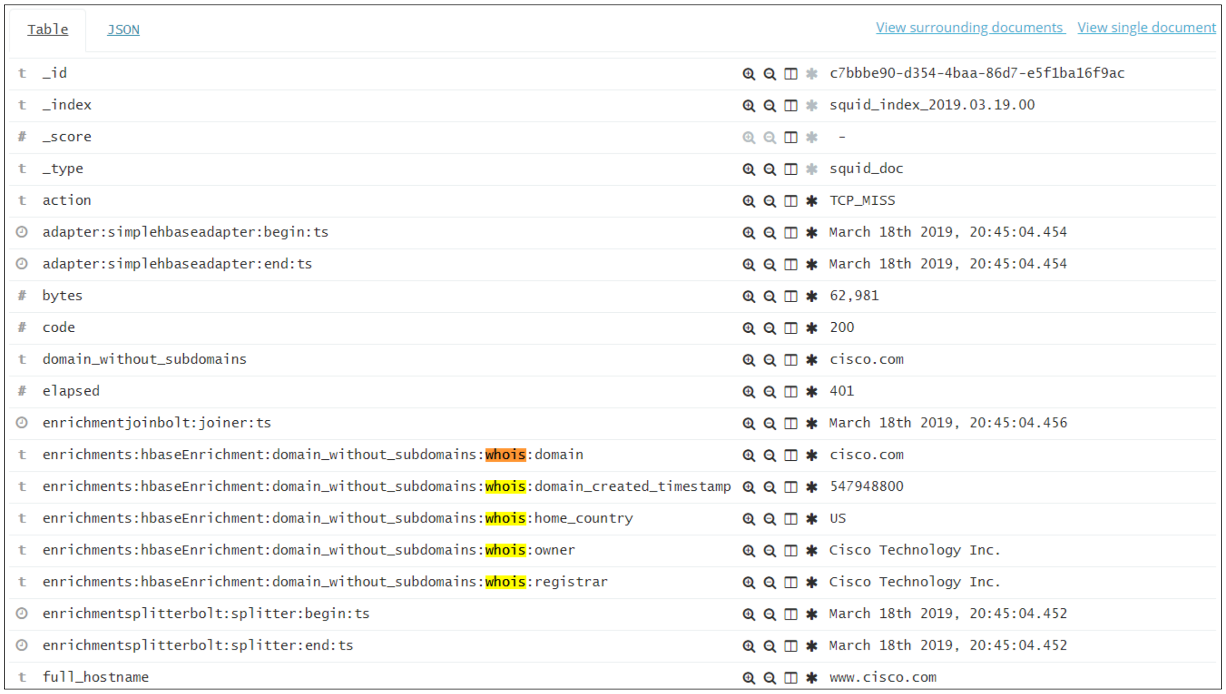
Observer the results in GUI.



Time	_source
March 18th 2019, 20:04:43	enrichments:hbaseEnrichment:domain_without_subdomains:whois:owner: Cisco Technology Inc. full_hostname: www.cisco.com code: 200 enrichments:hbaseEnrichment:domain_without_subdomains:whois:domain: cisco.com threatinteljoinbolt:joiner:ts: March 18th 2019, 20:45:04.461 enrichmentsplitterbolt:splitter:begin:ts: March 18th 2019, 20:45:04.452 enrichmentjoinbolt:joiner:ts: March 18th 2019, 20:45:04.456 enrichments:hbaseEnrichment:domain_without_subdomains:whois:home_country: US

Notice the fields added by enrichments on the screenshots: whois.owner, whois.domain_created_timestamp, whois.registrar, and whois.home_country.

Make a screenshot and store it in the submission document.



_id	c7bbbe90-d354-4baa-86d7-e5f1ba16f9ac
_index	squid_index_2019.03.19.00
_score	-
_type	squid_doc
action	TCP_MISS
adapter:simplehbaseadapter:begin:ts	March 18th 2019, 20:45:04.454
adapter:simplehbaseadapter:end:ts	March 18th 2019, 20:45:04.454
bytes	62,981
code	200
domain_without_subdomains	cisco.com
elapsed	401
enrichmentjoinbolt:joiner:ts	March 18th 2019, 20:45:04.456
enrichments:hbaseEnrichment:domain_without_subdomains:whois:domain	cisco.com
enrichments:hbaseEnrichment:domain_without_subdomains:whois:domain_created_timestamp	547948800
enrichments:hbaseEnrichment:domain_without_subdomains:whois:home_country	US
enrichments:hbaseEnrichment:domain_without_subdomains:whois:owner	Cisco Technology Inc.
enrichments:hbaseEnrichment:domain_without_subdomains:whois:registrar	Cisco Technology Inc.
enrichmentsplitterbolt:splitter:begin:ts	March 18th 2019, 20:45:04.452
enrichmentsplitterbolt:splitter:end:ts	March 18th 2019, 20:45:04.452
full_hostname	www.cisco.com

LAB EXERCISE/STEP 15

Be sure to terminate your AWS Metron VM to avoid budget depletion. Resources created in a cloud environment under your account have associated costs. AWS offers more than 60 products at a free tier with associated free tier usage limits [7].

What to submit

Submit a Word (or other text editor) document with embedded screenshots made as requested in the assignment and a brief description for each screenshot.

References

[1] The Apache Software Foundation, "Metron Tutorial - Fundamentals Part 2: Creating a New Enrichment," [Online]. Available: https://cwiki.apache.org/confluence/display/METRON/2016/04/28/Metron+Tutorial+-+Fundamentals+Part+2%3A+Creating+a+New+Enrichment. [Accessed Aug 8, 2019].

[2] The Apache Software Foundation, "Apache Metron Parsers," [Online]. Available: https://metron.apache.org/current-book/metron-platform/metron-parsers/index.html. [Accessed Aug 8, 2019].

[3] Grok, "Create a Grok Pattern," [Online]. Available: http://grok.nflabs.com/WhatIsPattern. [Accessed Aug 8, 2019].

[4] GitHub, "Logstash plugins. Logstash patterns core," [Online]. Available: https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns. [Accessed Aug 8, 2019].

[5] Elastic, "Grok filter plugin," [Online]. Available: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html. [Accessed Aug 8, 2019].

[6] The Apache Software Foundation, "Apache Metron Stellar Language," [Online]. Available: https://metron.apache.org/current-book/metron-stellar/stellar-common/index.html. [Accessed Aug 8, 2019].

[7] AWS, "AWS Free Tier," [Online]. Available: https://aws.amazon.com/free/. [Accessed Aug 8, 2019].