

ASSIGNMENT 3

Threat Intelligence - Static and Dynamic Detection by using
Data Engineering

GROUP 7

Deema Sami Barham Al Dogom - 200548717

Rimi Mondal - 200543066

Navid Saleh Sadik - 200544271

Bhavesht Natwarlal Waghela - 200532173

Georgian College
Data Collection and Curation BDAT 1008 01

Contents

Description of the Project	2
Comments on the Codes used in the project	2
LogSimulator.scala	2
KinesisSparkStreamingApp.scala	2
Illustrating Our Work: A Detailed Explanation with Screenshots	3
1. Setting Up Logstash on Local Machine	3
Tools used in the step	3
1.1. Configuring IAM User with S3 and Kinesis Access	4
Tools used in the step	4
2. Uploading MIP File to S3	5
Tools used in the step	5
3. Storing Malicious IP in Database.....	6
Tools used in the step	6
4. Configuring Kinesis Data Stream.....	8
Tools used in the step	8
5. Installing Maven.....	9
Tools used in the step	9
6. Creating JAR File with Maven	10
Tools used in the step	10
7. Setting Up Spark Services on Hadoop.....	11
Tools used in the step	11
8. Running Spark Commands on Hortonworks.....	12
Tools used in the step	12
Achievement	13
Project Participation Table.....	14

Description of the Project

The Threat Intelligence project focuses on implementing static and dynamic detection methods using data engineering techniques. The solution aims to collect random logs from scala code, process them using Spark, and store the identified malicious logs in a database for future querying.

The project involves several steps, including setting up Logstash on a local system, configuring an IAM user with access to S3 and Kinesis, uploading a malicious IP (MIP) file to S3, saving the malicious IP addresses to the database, utilizing a Kinesis Data Stream, installing and creating a JAR file using Maven, running Spark services on Hadoop, and executing Spark commands on Hortonworks.

Comments on the Codes used in the project

LogSimulator.scala

In particular, 'LogSimulator.scala' serves as a vital utility that generates simulated Apache logs. These logs mimic real-world log entries and provide a representative sample of data for analysis. By generating these logs, the file emulates log events that would typically be encountered in a production environment. Generating such logs is essential for testing and validating the Threat Intelligence solution's capabilities in handling diverse log data.

Once the Apache logs are generated, 'LogSimulator.scala' facilitates the transmission of these logs to the Kinesis Data Stream. By utilizing Kinesis, the code efficiently and reliably streams the generated logs, ensuring that they are available for subsequent processing by the Threat Intelligence system.

KinesisSparkStreamingApp.scala

On the other hand, 'KinesisSparkStreamingApp.scala' is responsible for retrieving the log data from the Kinesis Data Stream and processing it accordingly. This file employs Spark Streaming, a component of the Apache Spark framework, to enable real-time data ingestion and analysis.

Once the log data is retrieved from the Kinesis Data Stream, 'KinesisSparkStreamingApp.scala' performs the necessary computations, analyses, and evaluations to identify any malicious logs. This process ensures that only logs that are classified as malicious IP addresses are persisted, while non-malicious logs are excluded.

In summary, 'LogSimulator.scala' and 'KinesisSparkStreamingApp.scala' are instrumental in generating representative Apache logs, transmitting them to the Kinesis Data Stream, and subsequently processing and storing the identified malicious logs in the database. By leveraging data engineering techniques and combining static and dynamic detection methods, the Threat Intelligence project enables efficient identification and storage of malicious logs, enhancing the overall security posture and enabling future analysis and investigation.

Illustrating Our Work: A Detailed Explanation with Screenshots

These step-by-step actions provided a comprehensive explanation of how we implemented the Threat Intelligence - Static and Dynamic Detection project. We began by setting up Logstash and ensuring the necessary infrastructure was in place. Then, we processed the logs using Spark's distributed computing abilities, and finally stored the identified malicious IP addresses in a database for further analysis and querying.

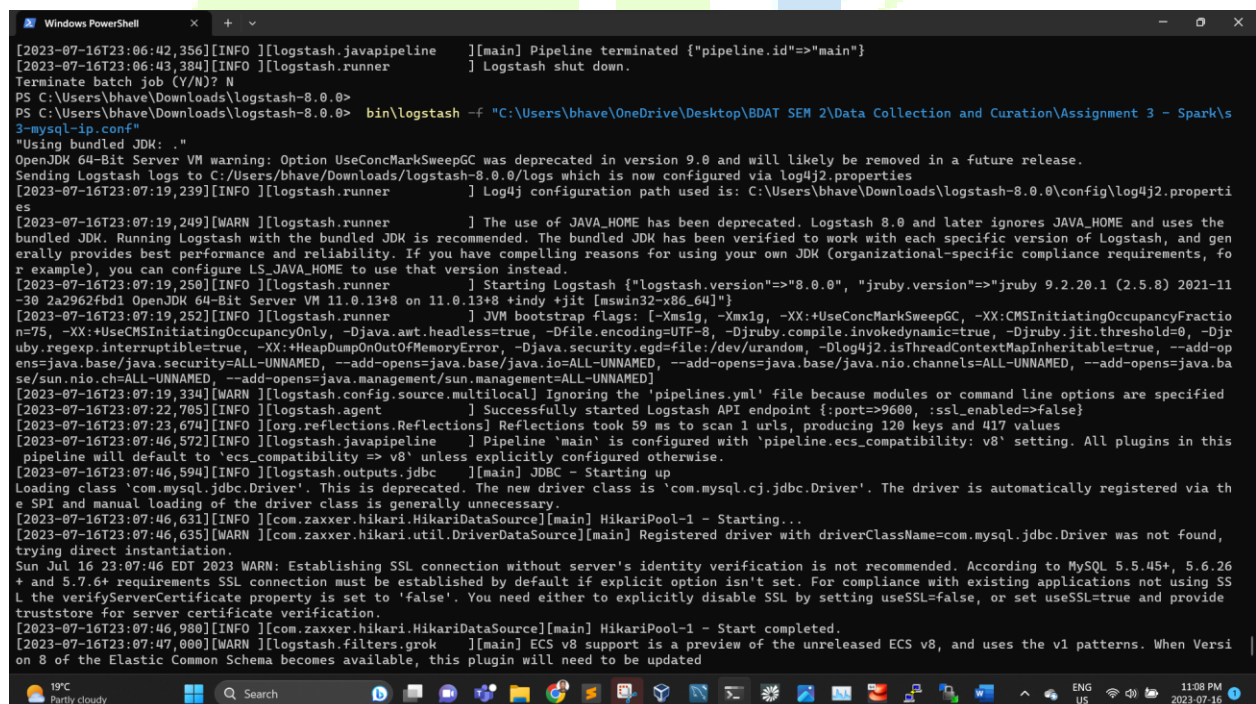
Step-by-Step Explanation for Threat Intelligence - Static and Dynamic Detection Project:

1. Setting Up Logstash on Local Machine

We started by setting up Logstash on our local machine. Logstash is an open-source data processing pipeline that allows us to collect, transform, and send logs or data from various sources. In our case, we configured Logstash to receive Malicious IP addresses from an S3 bucket. By leveraging Logstash's capabilities, we established a connection between S3 and Logstash, enabling the ingestion of Malicious IP addresses.

Tools used in the step

- **Logstash:** An open-source data processing pipeline used to collect, transform, and send logs or data from various sources.
- **S3 (Simple Storage Service):** A scalable object storage service provided by AWS for storing the Malicious IP (MIP) file.



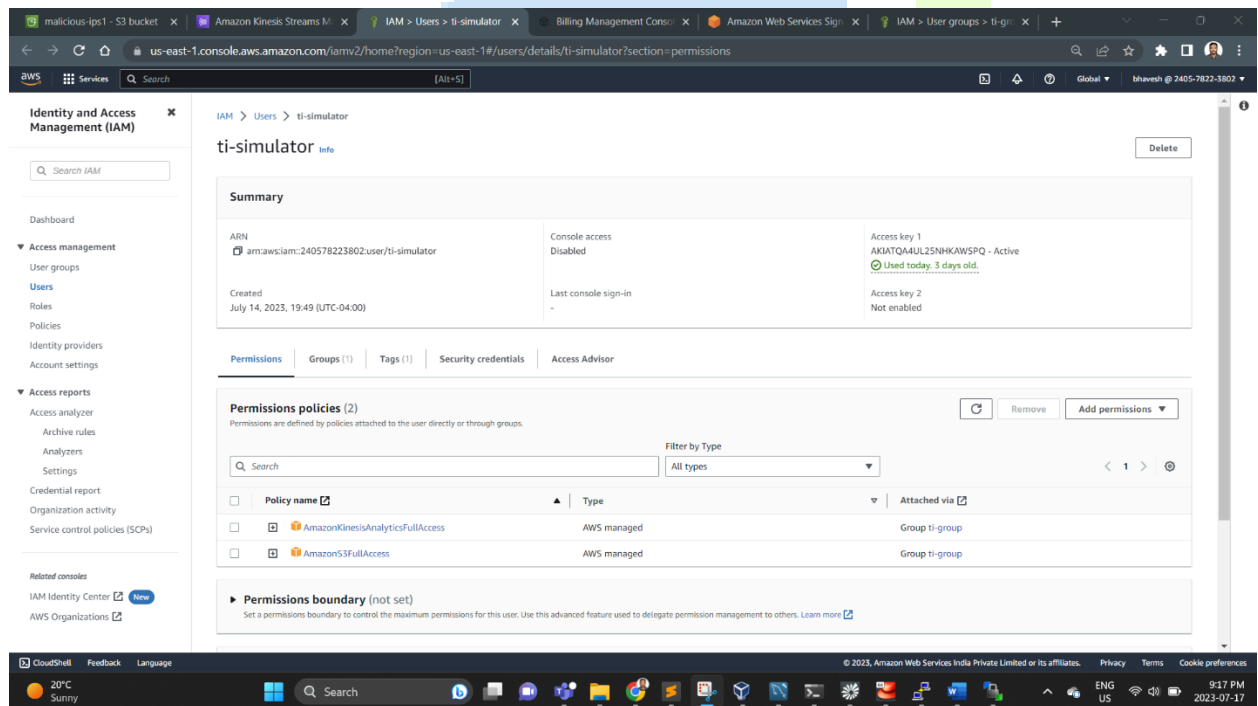
```
[2023-07-16T23:06:42,356][INFO ][logstash.javapipeline ][main] Pipeline terminated {"pipeline.id"=>"main"}
[2023-07-16T23:06:43,384][INFO ][logstash.runner ][main] Logstash shut down.
Terminate batch job (Y/N)? N
PS C:\Users\bhave\Downloads\logstash-8.0.0>
PS C:\Users\bhave\Downloads\logstash-8.0.0> bin\logstash -f "C:\Users\bhave\OneDrive\Desktop\BDAT SEM 2\Data Collection and Curation\Assignment 3 - Spark\3-mysql-ip.conf"
"Using bundled JDK: ."
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
Sending Logstash logs to C:\Users\bhave\Downloads\logstash-8.0.0\logs which is now configured via log4j2.properties
[2023-07-16T23:07:19,239][INFO ][logstash.runner ][main] Log4j configuration path used is: C:\Users\bhave\Downloads\logstash-8.0.0\config\log4j2.properties
[2023-07-16T23:07:19,249][WARN ][logstash.runner ][main] The use of JAVA_HOME has been deprecated. Logstash 8.0 and later ignores JAVA_HOME and uses the bundled JDK. Running Logstash with the bundled JDK is recommended. The bundled JDK has been verified to work with each specific version of Logstash, and generally provides best performance and reliability. If you have compelling reasons for using your own JDK (organizational-specific compliance requirements, for example), you can configure LS_JAVA_HOME to use that version instead.
[2023-07-16T23:07:19,258][INFO ][logstash.runner ][main] Starting Logstash {"logstash.version"=>"8.0.0", "jruby.version"=>"jruby 9.2.20.1 (2.5.8) 2021-11-30 2a2962fbd1 OpenJDK 64-Bit Server VM 11.0.13+8 on 11.0.13+8 +indy +jit [mswin32-x86_64]}"
[2023-07-16T23:07:19,252][INFO ][logstash.runner ][main] JVM bootstrap flags: [-Xms1g, -Xmx1g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatingOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djruby.compile.invokedynamic=true, -Djruby.jit.threshold=0, -Djruby.regexp.interruptible=true, -XX:+HeapDumpOnOutOfMemoryError, -Djava.security.egd=file:/dev/urandom, -Dlog4j2.isThreadContextMapInheritable=true, --add-opens=java.base/java.security=ALL-UNNAMED, --add-opens=java.base/java.io=ALL-UNNAMED, --add-opens=java.base/java.nio.channels=ALL-UNNAMED, --add-opens=java.base/sun.nio.ch=ALL-UNNAMED, --add-opens=java.management/sun.management=ALL-UNNAMED]
[2023-07-16T23:07:19,334][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' file because modules or command line options are specified
[2023-07-16T23:07:22,705][INFO ][logstash.agent ][main] Successfully started Logstash API endpoint {"port":9600, "ssl.enabled":false}
[2023-07-16T23:07:22,674][INFO ][org.reflections.Reflections] Reflections took 59 ms to scan 1 urls, producing 120 keys and 417 values
[2023-07-16T23:07:46,572][INFO ][logstash.javapipeline ][main] Pipeline 'main' is configured with 'pipeline.ecs_compatibility: v8' setting. All plugins in this pipeline will default to 'ecs_compatibility => v8' unless explicitly configured otherwise.
[2023-07-16T23:07:46,594][INFO ][logstash.outputs.jdbc ][main] JDBC - Starting up
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
[2023-07-16T23:07:46,631][INFO ][com.zaxxer.hikari.HikariDataSource][main] HikariPool-1 - Starting...
[2023-07-16T23:07:46,635][WARN ][com.zaxxer.hikari.util.DriverDataSource][main] Registered driver with driverClassName=com.mysql.jdbc.Driver was not found, trying direct instantiation.
Sun Jul 16 23:07:46 EDT 2023 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
[2023-07-16T23:07:46,980][INFO ][com.zaxxer.hikari.HikariDataSource][main] HikariPool-1 - Start completed.
[2023-07-16T23:07:47,000][WARN ][main] ECS v8 support is a preview of the unreleased ECS v8, and uses the v1 patterns. When Version 8 of the Elastic Common Schema becomes available, this plugin will need to be updated
```

1.1. Configuring IAM User with S3 and Kinesis Access

To ensure smooth data handling, we created an IAM (Identity and Access Management) user with appropriate permissions for accessing S3 (Simple Storage Service) and Kinesis. This user was granted the necessary privileges to interact with S3 for data storage and Kinesis for data streaming.

Tools used in the step

- **IAM (Identity and Access Management):** A service provided by AWS to manage user access and permissions.
- **S3 (Simple Storage Service):** A scalable object storage service provided by AWS for storing the Malicious IP (MIP) file.
- **Kinesis:** A real-time data streaming service provided by AWS for efficient and reliable data streaming.



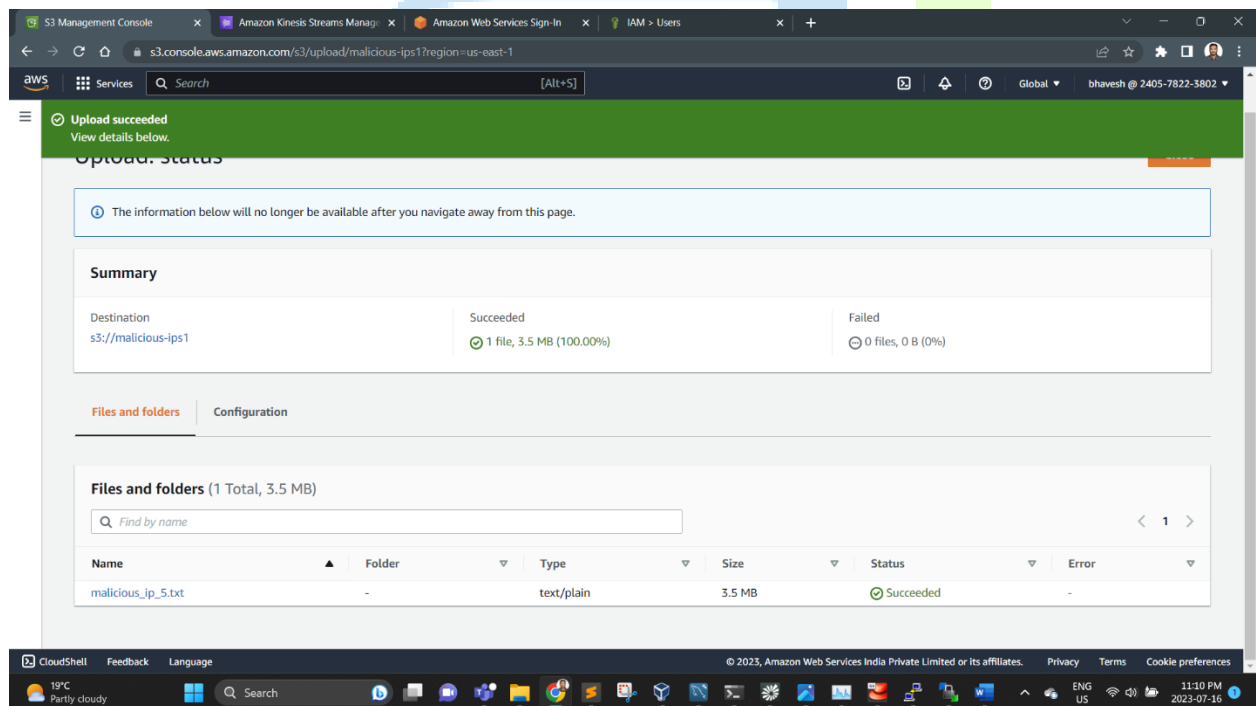
2. Uploading MIP File to S3

We obtained the Malicious IP (MIP) file from the GitHub link provided by the professor in the video. The link, <https://raw.githubusercontent.com/stamparm/ipsum/master/levels/1.txt>, contained a list of known malicious IP addresses. We created a text file and compiled all the IP addresses into it.

Once we had the MIP file ready, we proceeded to upload it to an S3 bucket. By storing the MIP file in an S3 bucket, we ensured easy accessibility and retrieval of the file for subsequent steps. This allowed Logstash to access the file during its data processing pipeline and facilitated the comparison and identification of malicious logs.

Tools used in the step

- **S3 (Simple Storage Service):** The AWS service used to store the Malicious IP (MIP) file containing a list of known malicious IP addresses.

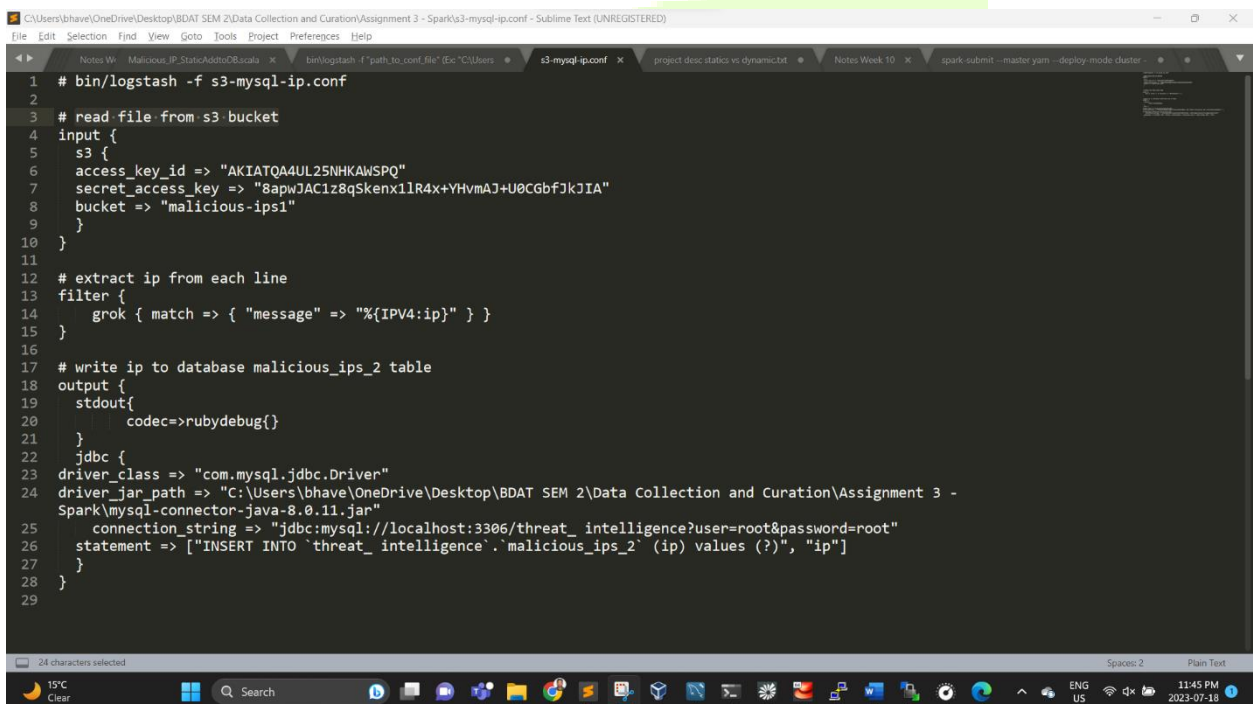


3. Storing Malicious IP in Database

To store the identified malicious IP addresses, we set up a database system, such as MySQL Workbench. This allowed us to create a table or collection specifically designed to store the malicious IP addresses. Logstash was configured to read the data and then write it to MySQL. By doing so, we ensured that we could query the database later for any relevant information.

Tools used in the step

- **Database System (MySQL Workbench):** A database management system used to create a table or collection specifically designed to store the malicious IP addresses.



```
1 # bin/logstash -f s3-mysql-ip.conf
2
3 # read file from s3 bucket
4 input {
5   s3 {
6     access_key_id => "AKIATQA4UL2SNHKAWSPQ"
7     secret_access_key => "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"
8     bucket => "malicious-ips1"
9   }
10 }
11
12 # extract ip from each line
13 filter {
14   grok { match => { "message" => "%{IPV4:ip}" } }
15 }
16
17 # write ip to database malicious_ips_2 table
18 output {
19   stdout {
20     codec => rubydebug {}
21   }
22   jdbc {
23     driver_class => "com.mysql.jdbc.Driver"
24     driver_jar_path => "C:\Users\bhave\OneDrive\Desktop\BDAT SEM 2\Data Collection and Curation\Assignment 3 - Spark\mysql-connector-java-8.0.11.jar"
25     connection_string => "jdbc:mysql://localhost:3306/threat_intelligence?user=root&password=root"
26     statement => ["INSERT INTO 'threat_intelligence'.malicious_ips_2 (ip) values (?)", "ip"]
27   }
28 }
29
```

```
Windows PowerShell

"message" => "206.62.50.70\r\n",
"event" => {
  "original" => "206.62.50.70\r\n"
},
"ip" => "206.62.50.70"
}

"@version" => "1",
"@timestamp" => 2023-07-17T03:14:29.990313900Z,
"message" => "103.233.155.93\r\n",
"event" => {
  "original" => "103.233.155.93\r\n"
},
"ip" => "103.233.155.93"
}

"@version" => "1",
"@timestamp" => 2023-07-17T03:14:29.990313900Z,
"message" => "109.117.131.101\r\n",
"event" => {
  "original" => "109.117.131.101\r\n"
},
"ip" => "109.117.131.101"
}

"@version" => "1",
"@timestamp" => 2023-07-17T03:14:29.990313900Z,
"message" => "91.243.167.196\r\n",
"event" => {
  "original" => "91.243.167.196\r\n"
},
"ip" => "91.243.167.196"
}

"@version" => "1",
"@timestamp" => 2023-07-17T03:14:29.990313900Z,
"message" => "153.205.157.249",
"event" => {
  "original" => "153.205.157.249"
},
}
```

MySQL Workbench

MyConnection x

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

- ada
- sakila
- sys
- threat_intelligence
 - Tables
 - malicious_event_details
 - Columns
 - id
 - raw_event
 - malicious_ips
 - Indexes
 - Foreign Keys
 - Triggers
 - malicious_ips_2
 - Columns
 - id
 - ip
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information

Table: malicious_ips_2

Columns:

- id int(11) PK
- ip mediumText

Object Info Session

malicious_ips_2 malicious_event_details malicious_event_details malicious_ips_2 malicious_event_details malicious_event_details malicious_ips

1 SELECT * FROM 'threat_intelligence'.malicious_ips_2;

Result Grid

#	id	ip
1	124.227.31.0	
2	121.237.169.0	
3	124.235.138.0	
4	124.88.55.0	
5	120.85.110.5	
6	124.88.113.0	
7	120.85.111.0	
8	124.225.44.0	
9	121.237.168.0	
10	124.88.112.0	
11	61.177.172.160	
12	167.94.138.125	

Output

#	Time	Action	Message	Duration / Fetch
26	23:03:49	DROP TABLE 'threat_intelligence'.malicious_ips	0 row(s) affected	0.047 sec
27	23:04:04	SELECT * FROM 'threat_intelligence'.malicious_ips LIMIT 0.50000	Error Code: 1146. Table 'threat_intelligence.malicious_ips' doesn't exist	0.000 sec
28	23:15:34	SELECT * FROM 'threat_intelligence'.malicious_ips_2 LIMIT 0.50000	50000 row(s) returned	0.000 sec / 0.047 sec
29	23:41:30	SELECT * FROM 'threat_intelligence'.malicious_event_details LIMIT 0.50000	0 row(s) returned	0.016 sec / 0.000 sec
30	23:41:32	SELECT * FROM 'threat_intelligence'.malicious_event_details LIMIT 0.50000	0 row(s) returned	0.000 sec / 0.000 sec
31	23:49:11	SELECT * FROM 'threat_intelligence'.malicious_ips_2 LIMIT 0.50000	50000 row(s) returned	0.000 sec / 0.047 sec

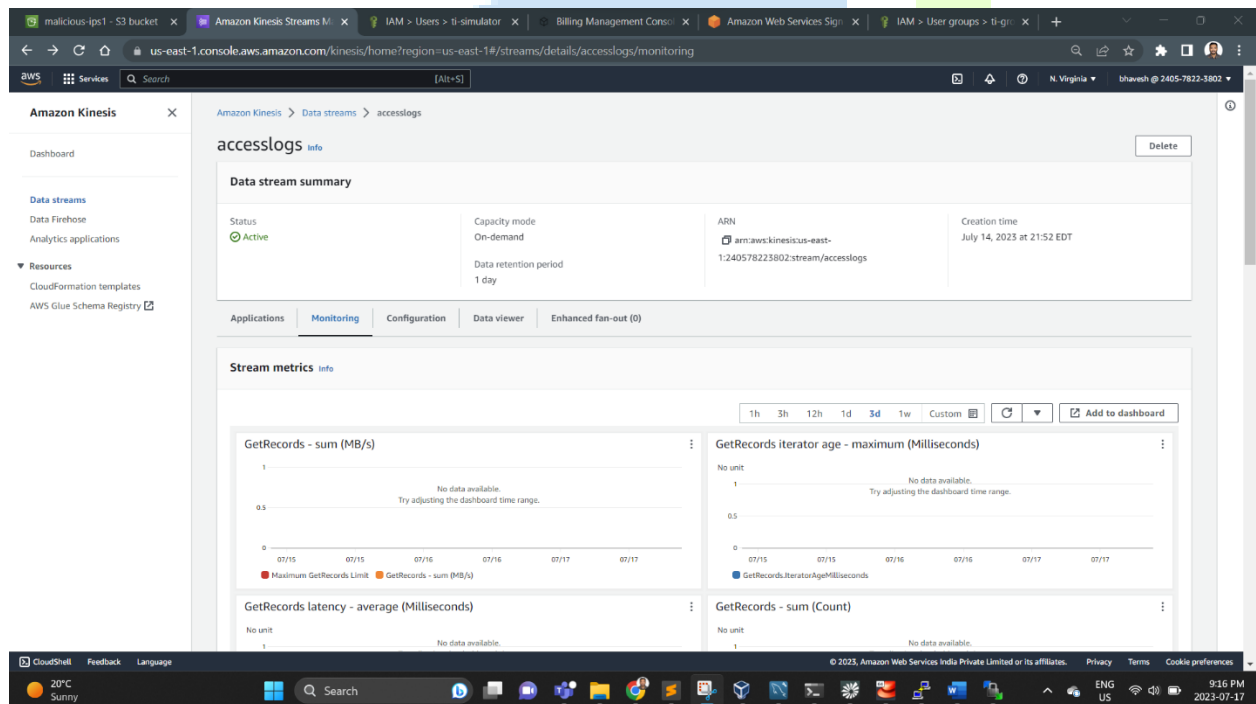
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

4. Configuring Kinesis Data Stream

To facilitate the flow of data, we designed a Scala code that generates Apache Logs with IP addresses. These logs are then stored in a Kinesis Data Stream for further processing. By configuring the Kinesis Data Stream to receive the logs, we established a smooth data flow from the log source to the subsequent processing steps.

Tools used in the step

- **Kinesis:** The AWS service used to create a data stream that acts as a conduit for processing incoming logs from Scala code.
- **Scala:** A programming language used to generate Apache Logs with IP addresses.
- **Kinesis Data Stream:** A service provided by AWS used to store the generated logs for further processing.

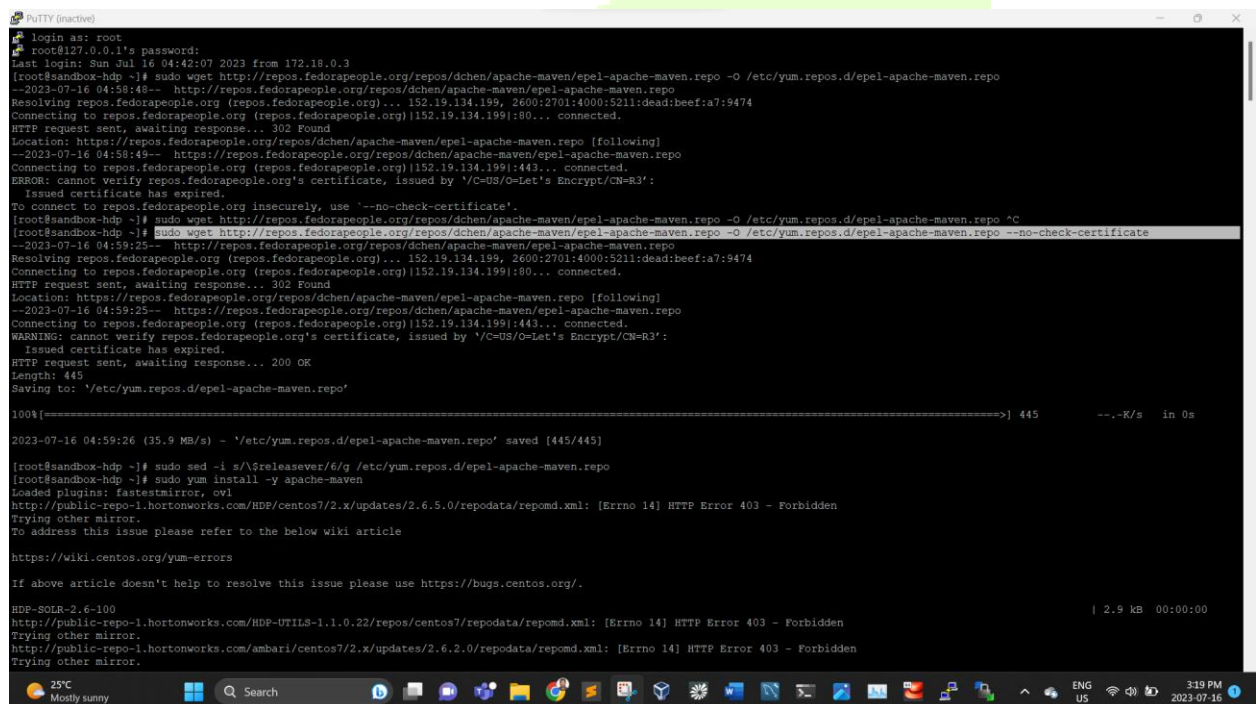


5. Installing Maven

We installed Apache Maven, a build automation tool, on our systems. Maven simplifies dependency management and project build processes, specifically for Java projects. This tool helped us manage the project's dependencies efficiently and ensured that all required libraries and packages were properly included.

Tools used in the step

- **Apache Maven:** A build automation tool used to manage project dependencies and simplify the project build process, particularly for Java projects.



```
login as: root
root@127.0.0.1's password:
Last login: Sun Jul 16 04:42:07 2023 from 172.18.0.3
[root@sandbox-hdp ~]# sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
--2023-07-16 04:58:48-- http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo
Resolving repos.fedorapeople.org (repos.fedorapeople.org)... 152.19.134.199, 2600:2701:4000:5211::dead:beef:a7:9474
Connecting to repos.fedorapeople.org (repos.fedorapeople.org)|152.19.134.199|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo [following]
--2023-07-16 04:58:49-- https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo
Connecting to repos.fedorapeople.org (repos.fedorapeople.org)|152.19.134.199|:443... connected.
ERROR: cannot verify repos.fedorapeople.org's certificate, issued by '/C=US/O=Let's Encrypt/CN=R3':
  Issued certificate has expired.
To connect to repos.fedorapeople.org insecurely, use '--no-check-certificate'.
[root@sandbox-hdp ~]# sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo --no-check-certificate
--2023-07-16 04:59:25-- http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo
Resolving repos.fedorapeople.org (repos.fedorapeople.org)... 152.19.134.199, 2600:2701:4000:5211::dead:beef:a7:9474
Connecting to repos.fedorapeople.org (repos.fedorapeople.org)|152.19.134.199|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo [following]
--2023-07-16 04:59:25-- https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo
Connecting to repos.fedorapeople.org (repos.fedorapeople.org)|152.19.134.199|:443... connected.
WARNING: cannot verify repos.fedorapeople.org's certificate, issued by '/C=US/O=Let's Encrypt/CN=R3':
  Issued certificate has expired.
HTTP request sent, awaiting response... 200 OK
Length: 445
Saving to: '/etc/yum.repos.d/epel-apache-maven.repo'

100%[=====] 445 --.-K/s in 0s

2023-07-16 04:59:26 (35.9 MB/s) - '/etc/yum.repos.d/epel-apache-maven.repo' saved [445/445]

[root@sandbox-hdp ~]# sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
[root@sandbox-hdp ~]# sudo yum install -y apache-maven
Loaded plugins: fastestmirror, ovl
http://public-repo-1.hortonworks.com/HDP/centos7/2.x/updates/2.6.5.0/reposdata/repomd.xml: [Errno 14] HTTP Error 403 - Forbidden
Trying other mirror.
To address this issue please refer to the below wiki article
https://wiki.centos.org/yum-errors
If above article doesn't help to resolve this issue please use https://bugs.centos.org/.

HDP-SOLR-2.6-100
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.22/repos/centos7/reposdata/repomd.xml: [Errno 14] HTTP Error 403 - Forbidden
Trying other mirror.
http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.6.2.0/reposdata/repomd.xml: [Errno 14] HTTP Error 403 - Forbidden
Trying other mirror.
```

6. Creating JAR File with Maven

Using Maven, we navigated to the project directory and executed the necessary commands to create a JAR (Java Archive) file. This JAR file contained the compiled code and all the required dependencies. By packaging the project into a JAR file, we ensured portability and ease of deployment in different environments.

Tools used in the step

- **Apache Maven:** Used to create a JAR (Java Archive) file containing the compiled code and required dependencies. The JAR file ensures portability and ease of deployment.

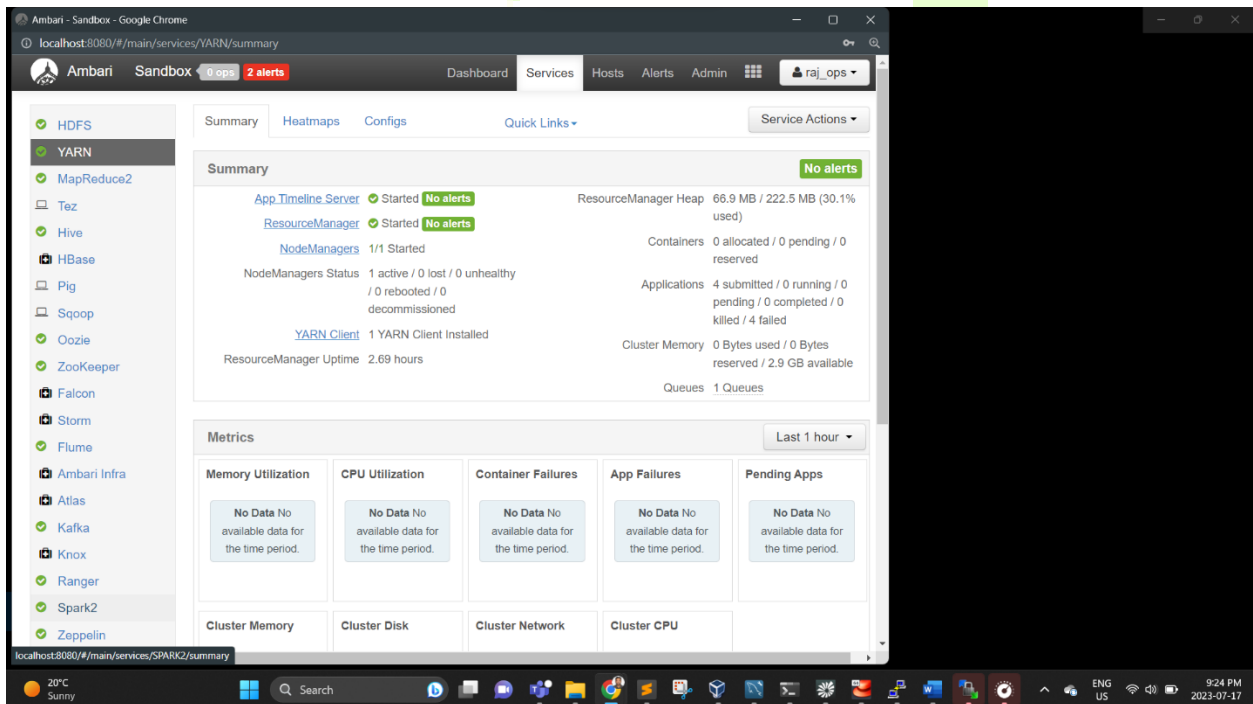
[illegible]

7. Setting Up Spark Services on Hadoop

As part of the project's infrastructure setup, we established a Hadoop cluster with Spark services enabled. Hadoop is a distributed processing framework, while Spark is a fast and general-purpose cluster computing system. By configuring Spark services on the Hadoop cluster, we leveraged the power of distributed computing for efficient data processing.

Tools used in the step

- **Hadoop:** A distributed processing framework used to process large datasets across a cluster of computers.
- **Apache Spark:** A fast and general-purpose cluster computing system that provides high-level APIs for distributed data processing.

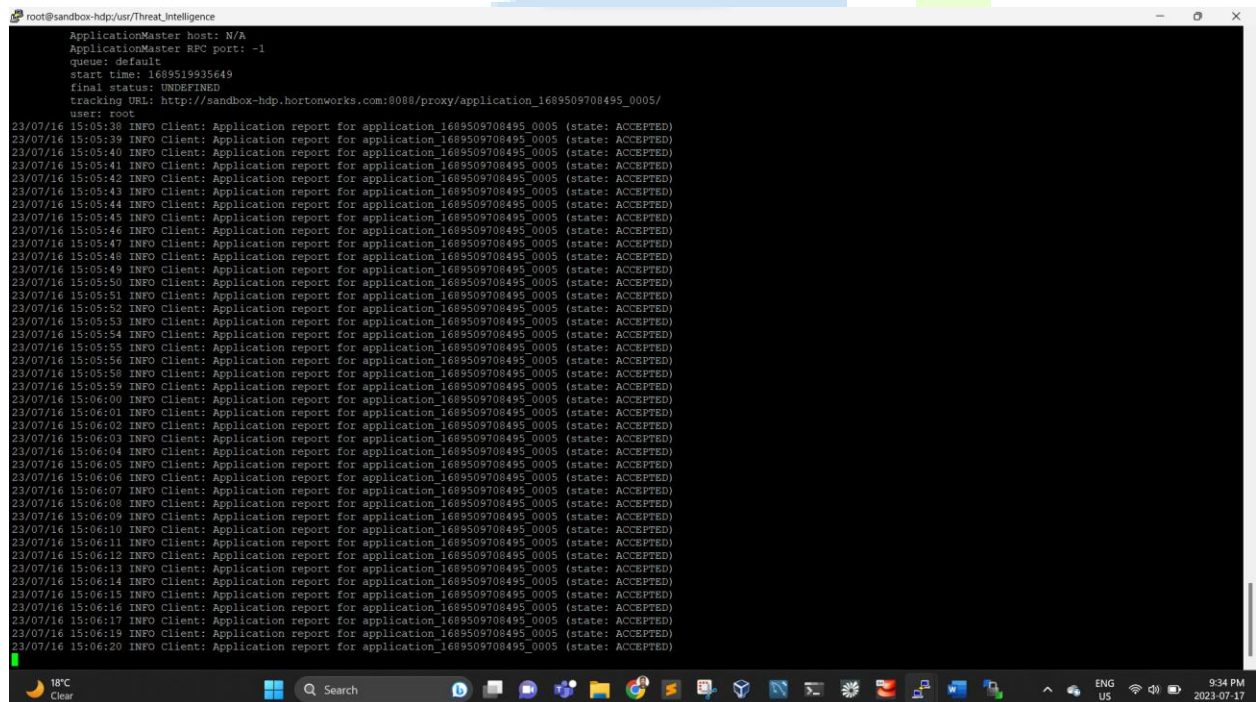


8. Running Spark Commands on Hortonworks

Finally, on the Hortonworks platform, we ran the Spark command to execute our threat intelligence solution. This command involved reading the logs from the Kinesis data stream, processing them using Spark's powerful capabilities, and identifying malicious logs through comparison with the MIP file. The identified malicious logs were then written to the database we set up earlier, excluding logs associated with non-malicious IP addresses.

Tools used in the step

- **Hortonworks:** A platform where the Spark command is executed to read logs from the Kinesis data stream and process them using Spark's capabilities.
- **Apache Spark:** Used to process the logs, compare them with the MIP file to identify malicious logs, and write the identified logs to the database.



The screenshot shows a terminal window titled "root@sandbox-hdp:~/Threat_Intelligence". The terminal displays the output of a Spark application. At the top, it shows configuration details for the ApplicationMaster, including host, RPC port, queue, start time, final status, and tracking URL. Below this, a series of log entries are shown, each starting with a timestamp (e.g., 23/07/16 15:05:38) and the text "INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)". The logs continue down to 23/07/16 15:06:20. The terminal window is overlaid on a Windows desktop background, with the taskbar visible at the bottom showing various application icons and the system clock indicating 9:34 PM on 2023-07-17.

```
root@sandbox-hdp:~/Threat_Intelligence
ApplicationMaster host: N/A
ApplicationMaster RPC port: -1
queue: default
start time: 1689519935649
final status: UNDEFINED
tracking URL: http://sandbox-hdp.hortonworks.com:8080/proxy/application_1689509708495_0005/
user: root
23/07/16 15:05:38 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:39 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:40 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:41 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:42 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:43 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:44 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:45 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:46 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:47 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:48 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:49 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:50 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:51 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:52 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:53 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:54 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:55 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:56 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:58 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:05:59 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:00 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:01 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:02 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:03 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:04 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:05 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:06 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:07 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:08 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:09 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:10 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:11 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:12 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:13 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:14 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:15 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:16 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:17 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:19 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
23/07/16 15:06:20 INFO Client: Application report for application_1689509708495_0005 (state: ACCEPTED)
```

Achievement

The Threat Intelligence - Static and Dynamic Detection project has provided valuable learnings in the field of data engineering. Through the implementation of static and dynamic detection methods, we have gained insights into effectively handling diverse log data and enhancing security measures. By utilizing data engineering techniques, we have successfully collected random logs from scala code, processed them using Spark, and stored identified malicious logs in a database for future analysis.

The key codes used in the project, 'LogSimulator.scala' and 'KinesisSparkStreamingApp.scala,' have played significant roles in achieving the project's objectives. 'LogSimulator.scala' has enabled the generation of representative Apache logs, emulating real-world log events encountered in production environments. This step has been crucial for testing and validating the capabilities of the Threat Intelligence solution.

On the other hand, 'KinesisSparkStreamingApp.scala' has been responsible for retrieving log data from the Kinesis Data Stream and processing it in real-time. By leveraging Spark Streaming, this code has facilitated efficient data ingestion and analysis, allowing for near real-time processing and response. Through the computations, analyses, and evaluations performed by 'KinesisSparkStreamingApp.scala,' malicious logs have been identified and stored in the database, ensuring comprehensive security measures.

The step-by-step explanation and accompanying screenshots have provided a detailed illustration of the project's implementation. From setting up Logstash and configuring IAM user access to S3 and Kinesis, to uploading the malicious IP file, storing malicious IP addresses in the database, and configuring the Kinesis Data Stream, each step has contributed to the successful execution of the project.

Additionally, the installation of Maven, creation of a JAR file, setup of Spark services on Hadoop, and execution of Spark commands on Hortonworks have demonstrated the utilization of powerful tools and frameworks for efficient data processing and analysis.

Overall, the Threat Intelligence - Static and Dynamic Detection project has not only showcased the application of data engineering techniques but has also provided valuable insights into handling diverse log data, identifying security threats, and enhancing overall security posture. The project's achievements have laid a solid foundation for future analysis, investigation, and improvement of threat intelligence systems.

Project Participation Table

Name of students who has participated in the assignment.
Student name: Deema Sami Barham Al Dogom - 200548717
Student name: Rimi Mondal - 200543066
Student name: Navid Saleh Sadik - 200544271
Student name: Bhavesht Natwarlal Waghela - 200532173
Student name:

Name of students who has not participated in the assignment.
Student name:
Student name:
Student name:
Student name:
Student name: