Steps to install spark:

1. wget [**http://mirrors.estointernet.in/apache/spark/spark-2.4.3/spark-2.4.3-bin-hadoop2.7.tgz**](http://mirrors.estointernet.in/apache/spark/spark-2.4.3/spark-2.4.3-bin-hadoop2.7.tgz)
2. sudo tar zxvf spark-2.4.3-bin-hadoop2.7.tgz -C /opt
3. sudo chown -R ubuntu:ubuntu /opt/spark-2.4.3-bin-hadoop2.7
4. sudo ln -fs spark-2.4.3-bin-hadoop2.7 /opt/spark
5. open .bashrc file and append the following:

export SPARK\_HOME=/opt/spark

PATH=$PATH:$SPARK\_HOME/bin

export PATH

6. source .bashrc

7. sudo apt-get update

8. sudo apt-get install default-jre

9. sudo apt-get install default-jdk(sudo apt install openjdk-8-jdk)

10. sudo nano /etc/environment

11. JAVA\_HOME=" /usr/lib/jvm/java-8-openjdk-amd64/jre/”

12. source /etc/environment

13. echo $JAVA\_HOME

14. wget <https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-x86_64.sh>

Wget <https://repo.anaconda.com/archive/Anaconda3-2019.03-Linux-x86_64.sh>

|  |
| --- |
|  |
|  |
|  |

15. bash Anaconda3-5.0.1-Linux-x86\_64.sh -b -p ~/anaconda

16. bash Anaconda3-5.0.1-Linux-x86\_64.sh –u(update)

17. jupyter notebook --generate-config

18. sudo apt install jupyter-core

19. sudo apt install jupyter-notebook

20. mkdir certs

21. cd certs

22. sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem

23. cd ~/.jupyter/

24. vi jupyter\_notebook\_config.py

If error /opt/spark/bin/spark-class: line 71: /usr/lib/jvm/java-8-openjdk-amd64/jre/bin//bin/java: No such file or directory

Update /bin/ vim load-spark-env.sh by

**JAVA\_HOME=" /usr/lib/jvm/java-8-openjdk-amd64/jre/”**

**/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java**

**Hostname -f**

sudo apt-get install scala

scala –version

(optional)wget <http://www.scala-lang.org/files/archive/scala-2.11.8.deb>

sudo dpkg -i scala-2.11.8.deb

sudo apt install imagemagick-6.q16

sudo apt install graphicsmagick-imagemagick-compat

sudo apt install imagemagick-6.q16hdri

**Uninstalling jupyter (use sudo according to your requirement)**

**pip3 uninstall jupyter**

**pip3 uninstall jupyter\_core**

**pip3 uninstall jupyter-client**

**pip3 uninstall jupyter-console**

**pip3 uninstall notebook**

**pip3 uninstall qtconsole**

**pip3 uninstall nbconvert**

**pip3 uninstall nbformat**

issue with jupyter notebook opening, check with permissions of directory where you are trying to open notebook:

**sudo chown –R user:user file/folder/directory name**

for jupyter notebook server connection:

**pip install --upgrade tornado==5.1.1**

**scala plugins error:**

**(update) Conflicting cross-version suffixes in: org.scala-lang.modules:scala-xml, org.scala-lang.modules:scala-parser-combinators**

**Soln: comment sbtPlugins :=True from build.sbt**

**Anomaly Detection commands:**

1. **in spark\_train folder write this command**

sbt assembly

sbt package

spark-submit --class AnomalyDetection \

target/scala-2.11/anomalydetection\_2.11-1.0.jar

1. **in streaming\_validation folder type this command**

sbt assembly

sbt package

spark-submit --class AnomalyDetectionTest \

--jars target/scala-2.11/AnomalyDetectionTest-assembly-1.0.jar \

target/scala-2.11/anomalydetectiontest\_2.11-1.0.jar

**Cassandra installation:**

1. sudo apt update
2. sudo apt upgrade –y(for rebbot instance)
3. Add the Apache repository of Cassandra to /etc/apt/sources.list.d/cassandra.sources.list.  
   $ sudo echo deb <http://www.apache.org/dist/cassandra/debian> 311x main |sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
4. Add the Apache Cassandra repository keys:  
   $ sudo curl <https://www.apache.org/dist/cassandra/KEYS> | sudo apt-key add –
5. Update the repositories:  
   $ sudo apt update
6. Install Cassandra   
   $ sudo apt install Cassandra
7. Stop Cassandra Service  
   $ sudo service cassandra stop

Now in order to create a Cluster of these 3 nodes or add a new node to existing cluster, follow above 1–7 steps and then below steps:

Step 1: Goto Cassandra conf Directory.  
$ cd /etc/cassandra

Step 2: Take backup of main configuration file before you make any change in it.  
$ sudo cp cassandra.yaml cassandra.yaml.bak

Step 3: Open cassandra.yaml in your favorite editor and edit below parameters as mentioned below:  
cluster\_name: ‘My Cluster’authenticator: PasswordAuthenticator (**optional**)seeds: “node\_private\_ip\_address”listen\_address:<node\_private\_ip\_address>rpc\_address: 0.0.0.0broadcast\_rpc\_address:<node\_private\_ip\_address>endpoint\_snitch: Ec2Snitch

Step 4: Save the cassandra.yaml file.

Step 5: Clear the default data from the Cassandra system table in order to import the new values set in the cassandra.yaml config file:  
$ sudo rm -rf /var/lib/cassandra/data/system/\*

Step 6: Start Cassandra Service on that node.  
$ sudo service cassandra start

Step 7: Wait for 10 second and check cluster status.  
$ sudo nodetool status

Write cqlsh command for Cassandra query.

Cassandra.apache.org/doc/cql3

**Testing trace on on Cassandra:**

**Cqlsh>Tracing on;**

**Now tracing is enabled**

**Cqlsh>CONSISTENCY QUORUM.**

**Consistency level set to quorum.**

 CREATE KEYSPACE WebLogDatabase WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication\_factor' : 1};

$ USE WebLogDatabase; CREATE TABLE WebLog (Time text PRIMARY KEY, IP text, URL text, Status varint, UserAgent text);

USE WebLogDatabase; INSERT INTO WebLog (Time, IP, URL, Status, UserAgent) VALUES ('29/Nov/2015:03:50:05', '156.23.45.89', '/news', 520, 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0)');

$ USE WebLogDatabase; SELECT \* FROM WebLog;

Error of cqlsh:

"/home/ubuntu/anaconda3/bin/cqlsh", line 121 except ImportError, e:

**Soln: except ImportError as e:**

This means that you need to either downgrade to Python 2.x or get a version of Cassandra that is compatible with Python 3.x.

Comparison between Cassandra, mongodb, hdfs

Scalability, Data Lake, highly availability, scalability (telemetry)

Zabbics agent in each vms host data automatically (os purpose)(matrics)

Kibana application matrics(json data log)(lstat)monitoring.

How to decompress gz files in linus:

**Gunzip filename.gz**

* File "/opt/spark-2.4.3-bin-hadoop2.7/bin/batch\_model\_builder.py", line 10, in <module>from pyspark.mllib.regression import LabeledPoint
* File "/opt/spark-2.4.3-bin-hadoop2.7/python/lib/pyspark.zip/pyspark/mllib/\_\_init\_\_.py", line 28, in <module>ImportError: No module named numpy

Slon:Append spark-env.sh

export PYSPARK\_PYTHON=/home/ubuntu/Anaconda/bin/python

export PYSPARK\_DRIVER\_PYTHON=home/ubuntu/Anaconda/bin/python

export HADOOP\_CONF\_DIR= /home/ubuntu/hadoop-2.8.0/etc/hadoop

Steps to download the **kafka:**

Step1 : Wget [**http://mirrors.estointernet.in/apache/kafka/2.2.0/kafka\_2.12-2.2.0.tgz**](http://mirrors.estointernet.in/apache/kafka/2.2.0/kafka_2.12-2.2.0.tgz)

Step2: tar -xzf kafka\_2.12-2.2.0.tgz

Step3: cd kafka\_2.12-2.2.0

**Run single-node zookeeper instance (one-terminal)**

Step4: **bin sudo ./zookeeper-server-start.sh ../config/zookeeper.properties**

netstat -nlp|grep 2181

sudo lsof -i :2181

kill -9 zookeeperProcessId

**Start the kafka server (different-terminal)**

Step5: **sudo ./kafka-server-start.sh ../config/server.properties**

To test process is running proper or not

ps -ef | grep "zookeeper.properties"

**Create a topic**

Step6: sudo ./kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test

./kafka-topics.sh --create --zookeeper *zookeeper-server*:2181 --partitions 1 --replication-factor 1 --topic wordcounttopic

**./kafka-topics.sh --create --zookeeper localhost:2181 --partitions 1 --replication-factor 1 --topic wordcounttopic**

Step7: sudo ./kafka-topics.sh --list --bootstrap-server localhost:9092

**Send some messages**

Step8: sudo ./kafka-console-producer.sh --broker-list localhost:9092 --topic test

**Start a consumer**

Step9: sudo ./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

**\***sudo ./kafka-topics.sh --describe --topic test --zookeeper localhost:2181

\*sudo ./kafka-console-producer --broker-list 0.0.0.0:9092 --topic MyFirstTopic1

\*sudo ./kafka-console-producer.sh --broker-list 0.0.0.0:9092 --topic test

\*sudo ./kafka-console-consumer.sh --bootstrap-server 0.0.0.0:9092 --topic test --from-beginning

Spark and Cassandra integration:

YOUR\_SPARK\_HOME/bin/spark-submit **\**

--class "SimpleApp" **\**

--master local[4] **\**

target/scala-2.12/simple-project\_2.12-1.0.jar

**jar files**:

steps:

* git clone <https://github.com/datastax/spark-cassandra-connector.git>
* cd spark-cassandra-connector
* sbt/sbt assembly
* spark-shell –-jars /home/ubuntu/spark-cassandra-connector/spark-cassandra-connector-java/target/scala-2.10/spark-cassandra-connector-java-assembly-1.4.0-SNAPSHOT.jar
* /home/ubuntu/target/scala-2.12/weblogprocessing\_2.12-1.0.jar
* **./spark-submit --class "WebLogProcessing" --master local[2] /home/ubuntu/target/scala-2.12/weblogprocessing\_2.12-1.0.jar 0.0.0.0:2181 testLogs**

Packages:

* spark-shell --packages "com.datastax.spark:spark-cassandra-connector\_2.11:2.0.2"
* spark-shell --packages org.apache.spark:spark-streaming-kafka-0-10\_2.11:2.1.0-SNAPSHOT
* --jar /opt/spark/bin/spark-cassandra-connector/spark-cassandra-connector/target/full/scala-2.11/spark-cassandra-connector-assembly-2.4.1-20-g23c09965.jar

Create build.sbt config file & add below in the file

name := "WebLogProcessing"

version := "1.0"

scalaVersion := "2.12.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.4.3"

name := "WebLogProcessing"

version :="1.0"

scalaVersion :="2.12.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.4.3"

libraryDependencies ++= Seq(

"org.apache.spark" %% "spark-streaming" % sparkVersion,

//"org.apache.spark" %% "spark-streaming-kafka" % sparkVersion

"org.apache.spark" % "spark-streaming-kafka-0-10\_2.11" % sparkVersion

//libraryDependencies += "org.apache.spark" %% "spark-streaming-kafka" % "1.6.3"

//libraryDependencies += "com.datastax.spark" %% "spark-cassandra-connector" % "2.0.2"

resolvers += "Spark Packages Repo" at "https://dl.bintray.com/spark-packages/maven"

libraryDependencies += "datastax" % "spark-cassandra-connector" % "2.4.1-s\_2.11"

libraryDependencies += "org.apache.spark" %% "spark-streaming-kafka-0-10" % "2.0.1"

sudo ./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic testLogs

libraryDependencies ++= Seq("org.apache.spark" % "spark-core\_2.10" % "1.2.1",

"org.apache.spark" % "spark-mllib\_2.10" % "1.2.1")

**KAFKA AND SPARK Integration commands**

spark-submit --master local[2] --conf "spark.dynamicAllocation.enabled=false" --jars

*SPARK\_HOME*/jars/spark-streaming\_2.11-2.4.3.jar kafka\_wordcount.py *zookeeper 0.0.0.0:*2181 wordcounttopic

https://repo1.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8-assembly\_2.11/2.4.3/spark-streaming-kafka-0-8-assembly\_2.11-2.4.3.jar

spark-submit --master local[2] --conf "spark.dynamicAllocation.enabled=false" --jars

*SPARK\_HOME*/examples/jars/spark-examples\_2.11-2.4.3.jar kafka\_wordcount.py *zookeeper 0.0.0.0:*2181 wordcounttopic

sudo ./spark-submit --master yarn --deploy-mode client --conf "spark.dynamicAllocation.enabled=false" --jars /opt/spark/examples/jars/spark-examples\_\*.jar kafka\_wordcount.py nightly511:2181 wordcounttopic

**sudo ./spark-submit --master local[2] --packages org.apache.spark:spark-streaming-kafka\_2.10:1.6.2 kafka\_wordcount.py localhost:2181 wordcounttopic**

**kafka-console-producer --broker-list *kafka\_broker*:9092 --topic wordcounttopic**

./spark-submit --packages org.apache.spark:spark-streaming-kafka-0-10\_2.11:2.2.0 kafka\_wordcount.py 0.0.0.0:2181 wordcounttopic

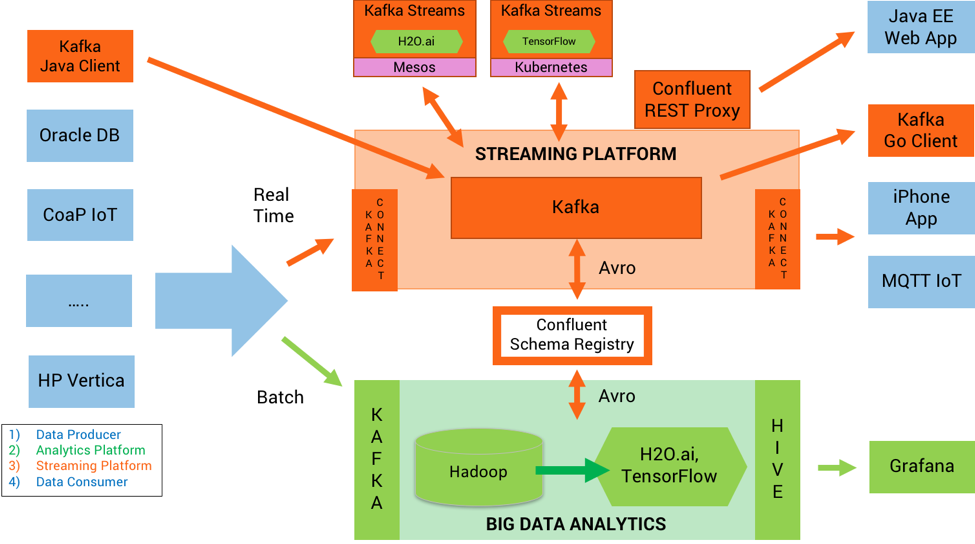
if you use something else than --master local, modify accordingly):

"PYSPARK\_SUBMIT\_ARGS": "--master local --packages org.apache.spark:spark-streaming-kafka-0-8:2.3.0 pyspark-shell"

2nd way is to put the following entry in your spark-defaults.conf file:

spark.jars.packages org.apache.spark:spark-streaming-kafka-0-10\_2.11:2.2.0

<https://repo1.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8_2.10/2.2.3/spark-streaming-kafka-0-8_2.10-2.2.3.pom>



Error Path:/admin/preferred\_replica\_election Error:KeeperErrorCode = NoNode for /admin/preferred\_replica\_election (org.apache.zookeeper.server.PrepRequestProcessor)

These log entries in ZK are normal. When kafka is started for the first time, it needs to create some paths in ZK. The initial creation of a path could fail because some parent paths are missing. Kafka will detect that and automatically create the missing parent paths

'ulimit –u’<larger num>

Ulimit –a

Jps –lvm

Cat /proc/sys/kernel/pid\_max

Cat /proc/sys/kernel/threads-max

ps -u ubuntu -L | wc –l

.ps -elf | wc -l:number of process running

2.ps -elfT | wc -l :all threads

3.ps -p JBOSSPID -lfT | wc -l:threads are spawned by jobid

4. jstack -l JBOSSPID | grep tid | wc -l

5./usr/bin/jps -J-Djava.io.tmpdir=/var/tmp/jboss -lv | grep -i jboss | cut -d ' ' -f 1:determine any JBOSS PID for any jboss version

ps -eLo pid,cmd,nlwp | wc –l: count threads

ps huH

/opt/spark/bin/hadoop-2.8.0/etc/hadoop$

Update core-site.xml

1. <configuration>
2. <property>
3. <name>fs.default.name</name>
4. <value>hdfs://localhost:9000</value>
5. </property>
6. </configuration>

# Real-Time-Data-Processing-using-Kafka-Spark-Streaming-and-Cassandra

This project aims to mine the logs with error status codes from Kafka real-time streaming data and save them into Cassandra database.

Step 1. Set up Cassandra

1.1 Download DataStax Distribution of Apache Cassandra from the link below, and then install it in Windows

<http://www.planetcassandra.org/cassandra/>

1.2. Start Cassandra server:

$ cd C:\cassandra\apache-cassandra\bin

$ cassandra

1.3. Open Datastax DevCenter, and execute the folowing cql in orders:

$ CREATE KEYSPACE WebLogDatabase WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication\_factor' : 1};

$ USE WebLogDatabase; CREATE TABLE WebLog (Time text PRIMARY KEY, IP text, URL text, Status varint, UserAgent text);

$ USE WebLogDatabase; INSERT INTO WebLog (Time, IP, URL, Status, UserAgent) VALUES ('29/Nov/2015:03:50:05', '156.23.45.89', '/news', 520, 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0)');

$ USE WebLogDatabase; SELECT \* FROM WebLog;

Step 2. Set up Kafka:

2.1. Install Kafka in Windows by following the instruction in the link below:

<https://dzone.com/articles/running-apache-kafka-on-windows-os>

2.2. Start Zookeeper server

$ cd C:\zookeeper\zookeeper-3.3.6\bin

$ zkserver

2.3. Start Kafka broker:

$ cd C:\kafka\kafka\_2.11-0.10.0.1

$ .\bin\windows\kafka-server-start.bat .\config\server.properties

2.4. Create a topic "testLogs"

$ cd C:\kafka\kafka\_2.11-0.10.0.1\bin\windows

$ kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic testLogs

Step 3. Run the spark program

Step 4. Create a producer to send the web log to Kafka broker

$ cd C:\kafka\kafka\_2.11-0.10.0.1\bin\windows

$ kafka-console-producer.bat --broker-list localhost:9092 --topic testLogs < web\_log\_data.txt

**!q for exiting from vim if it doesn’t have write permission**

E: Sub-process /usr/bin/dpkg returned an error code (1)

1. sudo dpkg --configure -a

2. sudo apt-get install -f

3. sudo apt remove

4. ls -l /var/lib/dpkg/info | grep -i python-\*

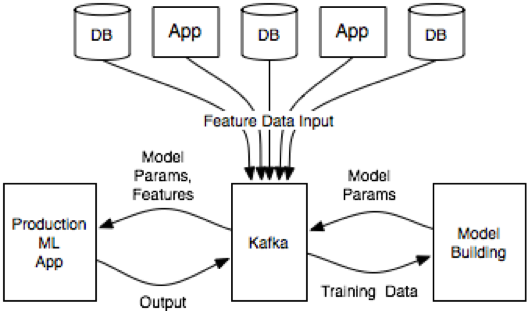
5. sudo mv /var/lib/dpkg/info/python-\*.\* /tmp

6. sudo apt upgrade

\*sudo apt-get remove --purge packagename

dpkg-deb -c /var/cache/apt/archives/libc6\_2.13-37\_i386.deb | awk {'print $6'} | cut -f2- -d. | sed 's|^/$|/.|' | sed 's|/$||' > /var/lib/dpkg/info/libc6:i386.list

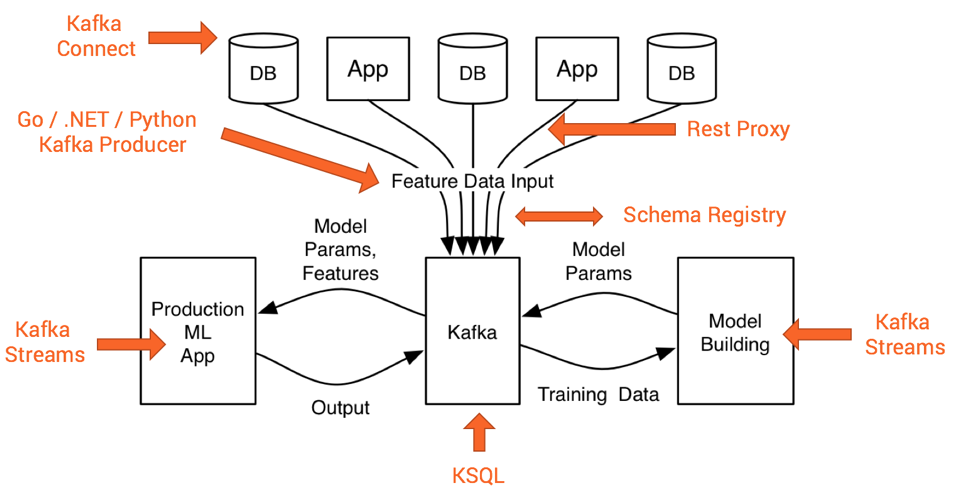
|  |  |
| --- | --- |
| **Strategy** | **Technologies** |
| Scalable Infrastructure /Elastic | Spark, Cassandra, Kafka |
| Partition for Scale, Network Topology Aware | Cassandra, Spark, Kafka, Akka cluster |
| Replicate For Resiliency | Spark, Cassandra , Akka cluster all hash the node ring |
| Share nothing, Masterless | Cassandra, Akka cluster both Dynamo Style |
| Fault Tolerance and no single point of failure | Spark, Cassandra, Kafka |
| Replay From Any Point of Failure | Spark, Cassandra, Kafka, Akka+ Akka persistence |
| Failure Detection | Cassandra, Spark, Akka, Kafka |
| Consensus and Gossip | Cassandra and Akka clutser |
| Parallelism | Spark, Cassandra, Kafka, Akka |
| Asynchronous Data Processing | Kafka, AKka, Spark |
| Fast, Low Latency ,Data Locality | Cassandra, Spark, Kafka |
| Location Transparency | Akka, Spark, Cassandra, Kafka |



* The essence of this architecture is that it uses Kafka as an intermediary between the various data sources from which feature data is collected, the model building environment where the model is fit, and the production application that serves predictions.
* Feature data is pulled into Kafka from the various apps and databases that host it.

Kafka becomes the central nervous system in the ML architecture to feed, build, apply and monitor analytic models. This establishes huge benefits:

* Data pipelines are simplified
* Building analytic modules is decoupled from servicing them
* Usage of real time or batch as needed
* Analytic models can be deployed in a performant, scalable and mission-critical environment
* In addition to leveraging Kafka as a scalable, distributed messaging broker, you can also add optional components of the Kafka ecosystem like Kafka Connect, Kafka Streams, Confluent REST Proxy, Confluent Schema Registry or KSQL instead of relying on the Kafka Producer and Consumer APIs:



Stream data from kafka to Cassandra

Stream data from kafka to spark and write to Cassandra

Stream from Cassandra to spark

Read data from Spark streaming source and write to C\*

Read data from Cassandra to spark

Batch analytics:

Analysis after data has accumulated decreases the weight of the data by the time processed.

Stream analytics:

Analytics as data arrives, The data won’t be stale neither analytics.

**Commands to run Anomaly detection**:

1. spark-submit -- master yarn -- deploy-mode client -- driver-memory 2g -- num-executors 2 -- executor-cores 2 -- executor-memory 2g -- packages com.databricks:spark-csv\_2.10:1.4.0 batch\_model\_builder.py -- input\_data\_path ${ENRICHED\_DATA\_PATH} -- data\_format json -- model\_path ${MODEL\_PATH}.tmp

**\*Need to start yarn client ./start-yarn.sh**

1. spark-submit --master yarn-client --packages com.databricks:spark- csv\_2.10:1.4.0 batch\_model\_builder.py
2. spark-submit -- master yarn -- deploy-mode client -- driver-memory 2g -- num-executors 2 -- executor-cores 1 -- packages org. apache.spark:spark-streaming-kafka\_2.10:1.6.0 spark\_streaming\_processor.py -- kafka\_zookeeper\_hosts ${KAFKA\_HOST}:2181 -- kafka\_broker\_list ${KAFKA\_HOST}:6667 -- kafka\_message\_topic ${KAFKA\_MESSAGE\_TOPIC} -- kafka\_alert\_topic ${KAFKA\_ALERT\_TOPIC} -- kafka\_enriched\_data\_topics ${KAFKA\_ENRICHED\_DATA\_TOPIC} -- streaming\_batch\_duration\_sec 15 ${MODEL\_ARG}
3. spark-submit --master yarn-client --packages org.apache.spark:spark-streaming-kafka\_2.10:1.6.0 spark\_streaming\_processor.py
4. ./streaming\_data\_generator.py -- kdd\_file /opt/spark/bin/kddcup.data -- kafka\_zookeeper\_hosts ${KAFKA\_HOST}:2181 –kafka\_broker\_list ${KAFKA\_HOST}:6667 -- kafka\_message\_topic ${KAFKA\_MESSAGE\_TOPIC} – throttle\_message\_per\_sec 100

Steps to import json data into Cassandra:

There are four ways to do it.

* **sstable2json** - it produces quite big json files which are hard to parse - because tool puts data in one row and uses complicated schema (ex. 300Mb Data file = ~2Gb json), it takes a lot of time to dump and Cassandra likes to change source file names according its internal mechanism
* **COPY** - causes timeouts on quite fast EC2 instances for big number of records
* **CAPTURE** - like above, causes timeouts
* **reads with pagination** - I used timeuuid for it, but it returns about 1,5k records per second