**App Installation Guide**

Stephen Blystone

Document Version 1.0

Last Updated: Apr 6, 2018

Table of Contents

[**Revision History** 3](#_Toc510952735)

[**Big Data Pipeline Architecture** 3](#_Toc510952736)

[**VM Parameters Used** 4](#_Toc510952737)

[**Pre-Requisites** 4](#_Toc510952738)

[**Installing and Configuring Apache Kafka** 4](#_Toc510952739)

[**Installing and Configuring Apache Spark** 5](#_Toc510952740)

[**Installing and Configuring Elasticsearch** 5](#_Toc510952741)

[**Installing and Configuring StreamSets** 5](#_Toc510952742)

[**Building Pipeline in StreamSets** 6](#_Toc510952743)

[**Testing Current Pipeline** 7](#_Toc510952744)

[**Stopping Current Pipeline** 9](#_Toc510952745)

[**Spark Streaming Python Resources** 10](#_Toc510952746)

[**ElasticSearch Python Resources** 10](#_Toc510952747)

# **Revision History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0 | 4/6/2018 | Initial Version |

# **Big Data Pipeline Architecture**

Below is a basic Big Data architecture for real-time processing from the Microsoft website. We are taking inspiration from there.

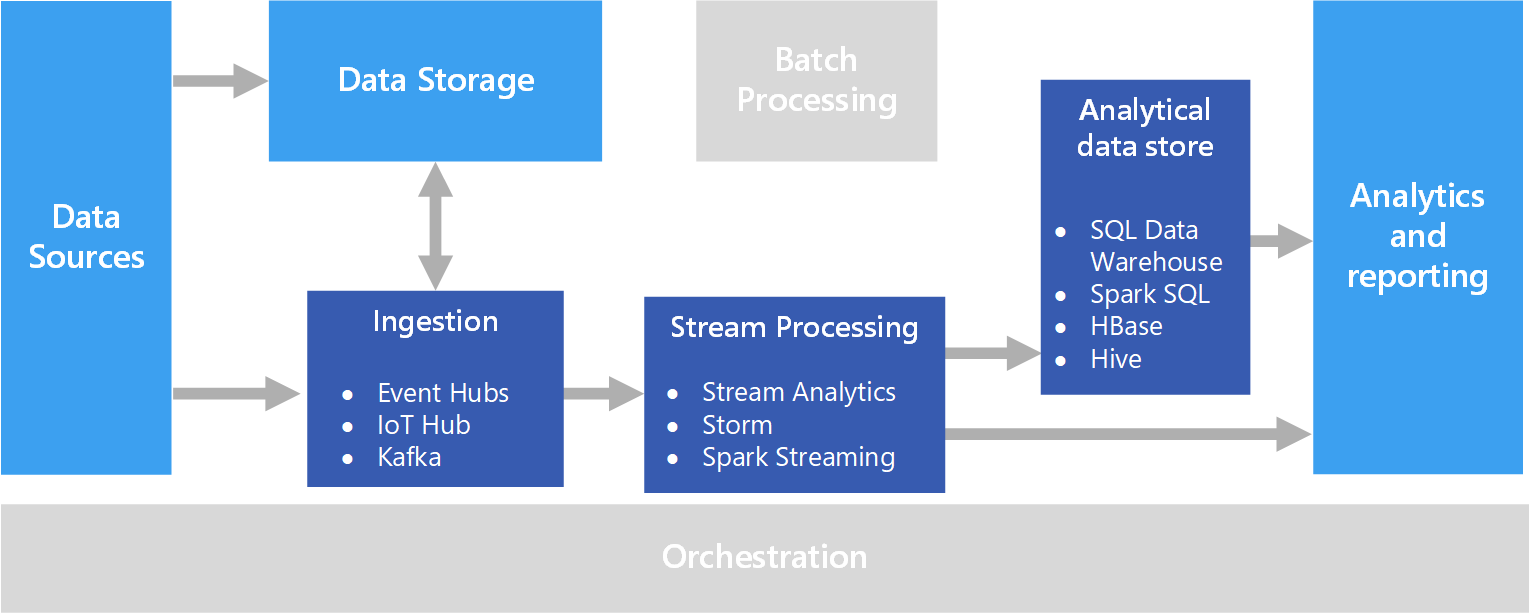


Figure 1 - https://docs.microsoft.com/en-us/azure/architecture/data-guide/scenarios/real-time-processing

We will have several components in our architecture:

1. Data Sources
   1. These are the Open vSwitch nodes sending NetFlow data.
2. Real-time Message Ingestion
   1. This component essentially performs Stream Buffering, providing the “Stream Processing” block enough time to process the stream data without being overrun with data.
   2. We will use Apache Kafka.
3. Stream Processing
   1. This component filters, aggregates, and prepares the data for analysis.
   2. We will use Apache Spark Streaming.
4. Analytical Data Store
   1. This is where the data is stored after being processed.
   2. We will use Elasticsearch.
5. Analytics and Reporting
   1. We will not use this part.
6. Machine Learning
   1. Not pictured in the above diagram.
   2. We will use the data in the Analytical data store to perform Machine Learning.
7. Analytics and Reporting
   1. We will use Kibana.

# **VM Parameters Used**

1. Memory: 6144MB (You may not need this much memory, I used a large value since all of the applications in this document are running in a single VM).
2. Hard Drive: 20GB

# **Pre-Requisites**

1. Install curl

sudo apt-get install curl

1. Install Java 8 onto Application VM
   1. Java8 commands:

sudo apt-get install software-properties-common -y && \

sudo add-apt-repository ppa:webupd8team/java -y && \

sudo apt-get update && \

echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 select true" | sudo debconf-set-selections && \

sudo apt-get install oracle-java8-installer oracle-java8-set-default -y

1. Add JAVA\_HOME to your .profile file and to PATH

vi ~/.profile

JAVA\_HOME=/usr/lib/jvm/java-8-oracle

PATH=$PATH:$JAVA\_HOME

1. Log out and back in to enable the changes.

# **Installing and Configuring Apache Kafka**

1. <https://kafka.apache.org/quickstart>
2. Download Kafka 1.0.0 (this is the version available in StreamSets)

cd ~

wget <http://apache.claz.org/kafka/1.0.0/kafka_2.11-1.0.0.tgz>

tar -xvzf kafka\_2.11-1.0.0.tgz

1. Vi into the Kafka Server configuration file and update that file.

vi ~/kafka\_2.11-1.0.0/config/server.properties

listeners=PLAINTEXT://streamsetApp:9092

advertised.listeners=PLAINTEXT://streamsetApp:9092

**NOTE:** streamsetApp is my VM hostname. You can find your hostname by running “hostname” in a terminal.

# **Installing and Configuring Apache Spark**

1. Download Apache Spark 2.3.0 to your VM. To choose your version you can go to <https://spark.apache.org/downloads.html>. Below is the mirror it gave me:

cd ~

wget <http://mirrors.ocf.berkeley.edu/apache/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz>

tar -xzvf spark-2.3.0-bin-hadoop2.7.tgz

1. Go to the “conf” directory

cd ~/spark-2.3.0-bin-hadoop2.7/conf

1. Copy the spark-env template

cp spark-env.sh.template spark-env.sh

1. Add “MASTER=local[2]” to the end of the spark-env.sh file.

vi spark-env.sh

G # Go to end of file.

o # Add new line below current cursor position and enter insert mode.

MASTER=local[2]

Esc # switch out of insert mode.

:x # save and exit.

1. Place sparkKafka.py into ~/spark-2.3.0-bin-hadoop2.7/python/

# **Installing and Configuring Elasticsearch**

1. <https://www.elastic.co/downloads/elasticsearch>
2. Download the file:

cd ~

wget <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.2.3.tar.gz>

tar -xvzf elasticsearch-6.2.3.tar.gz

1. <http://elasticsearch-py.readthedocs.io/en/master/>

# **Install necessary Python Libraries**

1. Install the required Python libraries.

sudo apt-get install python-setuptools

sudo easy\_install pip

pip install elasticsearch

pip install --user numpy

pip install --user pyspark

pip install --user requests

pip install --user sklearn

pip install --user pandas

# **Installing and Configuring Kibana**

1. <https://www.elastic.co/downloads/kibana>
2. Download the file:

cd ~

wget <https://artifacts.elastic.co/downloads/kibana/kibana-6.2.3-linux-x86_64.tar.gz>

tar -xvzf kibana-6.2.3-linux-x86\_64.tar.gz

1. Launch Kibana

cd ~/kibana-6.2.3-linux-x86\_64

bin/kibana

1. Go to <http://localhost:5601/app/kibana>
2. Go to “Dev Tools”
3. Enter the Index\_ES.txt information into the Console
4. Highlight what you entered in the Console and click  to run the commands.

# **Installing and Configuring Logstash**

1. <https://www.elastic.co/downloads/logstash>
2. Download the file:

cd ~

wget <https://artifacts.elastic.co/downloads/logstash/logstash-6.2.4.tar.gz>

tar -xvzf logstash-6.2.4.tar.gz

1. Edit the Logstash Configuration

cd ~/logstash-6.2.4

vi config/logstash.yml

modules:

- name: netflow

var.input.udp.port: 9997

1. Run the Netflow module

cd ~/logstash-6.2.4

bin/logstash --modules netflow --setup

NOTE: “--setup" creates a netflow-\* index pattern in Elasticsearch and imports Kibana dashboards and visualizations. Omit this option on subsequent runs to avoid overwriting existing dashboards.

# **Installing and Configuring StreamSets**

1. <https://streamsets.com/blog/visualizing-netflow-data-streamsets-data-collector-kudu-impala-d3/>
2. To simplify the flow between the different products, we can use Streamsets. <https://streamsets.com/opensource/>
3. Download the file:

cd ~

wget <https://archives.streamsets.com/datacollector/3.1.2.0/tarball/streamsets-datacollector-core-3.1.2.0.tgz>

tar -xvzf streamsets-datacollector-core-3.1.2.0.tgz

1. Increase the limit of open file descriptors.
   1. sudo vi /etc/security/limits.conf
   2. At the end of the file (above where it says # End of file) put the following entry:

\* soft nofile 32768

\* hard nofile 32768

1. Log out and back in for this to take effect.
2. You can verify this worked after logging back in by running “ulimit -n” and you should see 32768

# **Building Pipeline in StreamSets**

1. <https://streamsets.com/blog/visualizing-netflow-data-streamsets-data-collector-kudu-impala-d3/>
2. <https://kafka.apache.org/quickstart>
3. Run the Data Collector (from the directory you downloaded it in):

~/streamsets-datacollector-3.1.2.0/bin/streamsets dc

1. Go to http://<system-ip>:18630/
   1. Default username and password are “admin” and “admin”
2. Go to the packages icon  in the top right corner and install the following packages. Check the box next to the name of each package and then click  to install:
3. Apache Kafka 1.0.0
4. Create a new Pipeline for UDP to Kafka.
5. Name it “UDP Kafka Producer” with description “Source data from UDP to Kafka”
6. Run it on the “Data Collector”
7. Select Origin to be “UDP Source – Basic”
   * 1. In the Configuration Panel at the bottom, select the UDP tab and change the Data Format to NetFlow.
8. “Select Destination to connect” to Kafka Producer.
   * 1. In the Configuration Panel at the bottom, select the Kafka tab and change the “Topic” to “NETFLOW”
     2. In the Configuration Panel at the bottom, select the “Data Format” tab and change “Data Format” to “JSON Record”
9. Click the  icon. This will take you to “Error Records”. Select “Discard (Library: Basic)” from the drop-down menu.

# **Testing Current Pipeline**

1. Start ONOS (if not already started) in **ONOS VM**.

onos-buck run onos-local

1. Open a new terminal and launch StreamSets Web GUI (if not already running) in **APP VM**:

~/streamsets-datacollector-3.1.2.0/bin/streamsets dc

Go to <http://streamsetApp:18630/>

**NOTE:** streamsetApp is my VM hostname. You can find your hostname by running “hostname” in a terminal.

Default username/password is “admin” and “admin”

1. Open a new terminal and launch Spark Master and Slave servers:

cd ~/spark-2.3.0-bin-hadoop2.7/

sbin/start-master.sh

sbin/start-slave.sh spark://streamsetApp:7077

**NOTE:** streamsetApp is my VM hostname. You can find your hostname by running “hostname” in a terminal.

1. Open a new terminal and launch zookeeper server:

cd ~/kafka\_2.11-1.0.0/

bin/zookeeper-server-start.sh config/zookeeper.properties

1. Open a new terminal and launch Kafka server:

cd ~/kafka\_2.11-1.0.0/

bin/kafka-server-start.sh config/server.properties

1. Open a new terminal and create a Kafka Topic for NETFLOW:

cd ~/kafka\_2.11-1.0.0/

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

1. List Kafka Topics to verify NETFLOW is there

cd ~/kafka\_2.11-1.0.0/

bin/kafka-topics.sh --list --zookeeper localhost:2181

You should see NETFLOW in the output.

1. Run the StreamSets Pipeline. Open StreamSets and open your pipeline. Click 
2. Launch a Kafka Consumer Console to view the NETFLOW messages being received by Kafka.

cd ~/kafka\_2.11-1.0.0/

bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic NETFLOW

1. Open a new terminal and launch Elasticsearch

cd ~/elasticsearch-6.2.3/

bin/elasticsearch

1. Open a new terminal and launch Kibana

cd ~/kibana-6.2.3-linux-x86\_64/

bin/kibana

1. Open a new terminal and launch Logstash

cd ~/logstash-6.2.4

bin/logstash --modules netflow

1. Open a new terminal and launch your Python code for Spark Streaming

cd ~/spark-2.3.0-bin-hadoop2.7/

FOR CAPTURING NORMAL TRAFFIC TO STORE IN ELASTICSEARCH:

bin/spark-submit --packages org.apache.spark:spark-streaming-kafka-0-8\_2.11:2.3.0 python/sparkKafka.py | grep -v INFO | grep -v WARN

FOR READING STORED DATA IN ELASTICSEARCH AND CHECKING FOR ANOMALIES:

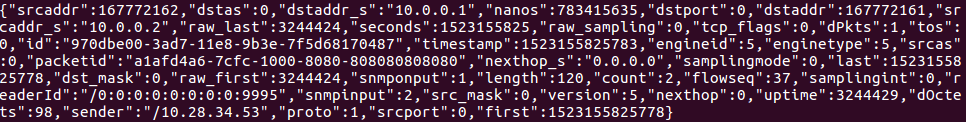
bin/spark-submit --packages org.apache.spark:spark-streaming-kafka-0-8\_2.11:2.3.0 python/sparkMachineLearning.py | grep -v INFO | grep -v WARN

NOTE: These commands will install any missing packages and necessary JAR files for spark and kafka.

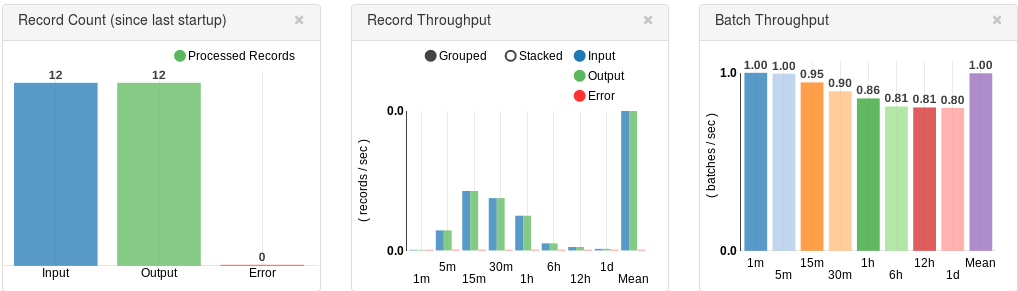
1. Start Mininet (if not already started) in **Mininet VM**.

sudo python setup\_topo.py

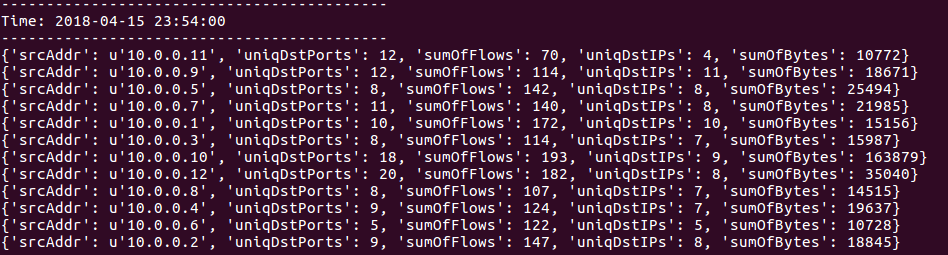
1. You should now see Netflow messages in the kafka-console-consumer window. Example:



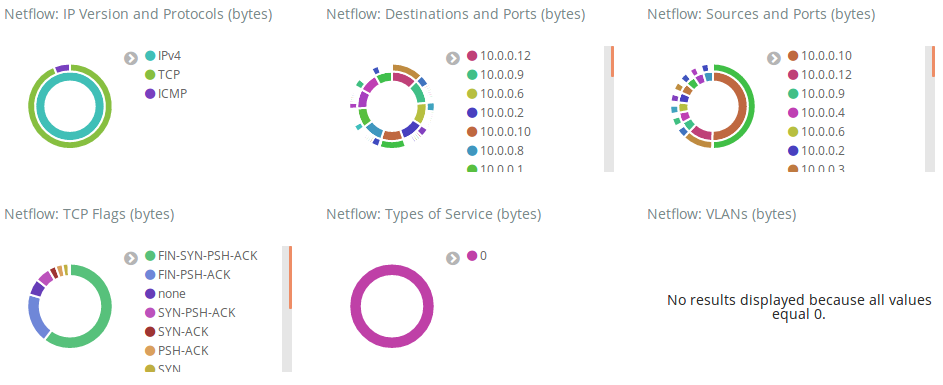
1. In StreamSets web GUI you should also see data showing up in the Summary window. Example:



1. You should see Output in your Spark Streaming terminal window.   
   NOTE: Due to buffering, it may take a few minutes to display the messages on the screen.  
     
   Example:



1. In Kibana Web Gui (<http://localhost:5601/app/kibana>) go to Dashboard.
   1. Click “Netflow: Overview” to see a visualization of the Netflow traffic being received from Mininet.



1. In **Mininet VM**, perform an ICMP Ping Flood:
   1. Type “xterm h11” at the Mininet prompt.
   2. Type “h11 ping h12” to begin a continuous ping.
   3. In the XTerm window that opens, run the following command:

sudo hping3 -V -c 100000 -d 9000 -S -w 64 --flood 10.0.0.2

* 1. This command sends a flood of 100000 pings to h12.

1. In the **App VM** within around 60 seconds (since Spark is configured to capture in 60-second windows), you should see a message in the Spark Streaming Terminal that indicates a REST API call was sent to ONOS to block all traffic from h11.
2. If you check your continuous ping in the **Mininet VM**, you should see that it has stopped receiving responses. You can also run the “pingall” command at the Mininet prompt to see that h11 cannot send packets anywhere, and no other host is able to ping h11.
3. In **Mininet VM**, perform a UDP port scan:
   1. Type “xterm h21” at the Mininet prompt.
   2. Type “h21 ping h22” to begin a continuous ping.
   3. In the XTerm window that opens, run the following command:

python scapyPortScan.py --ip 10.0.0.5

* 1. This Python script will run a port scan on h22 from port 1 to port 65535.

1. In the **App VM** within around 60 seconds (since Spark is configured to capture in 60-second windows), you should see a message in the Spark Streaming Terminal that indicates a REST API call was sent to ONOS to block all traffic from h21.
2. If you check your continuous ping in the **Mininet VM**, you should see that it has stopped receiving responses. You can also run the “pingall” command at the Mininet prompt to see that h11 and h21 cannot send packets anywhere, and no other host is able to ping h11 or h21.

# **Stopping Current Pipeline**

1. **NOTE: It is important you run these in order. Otherwise some items will refuse to shut down.**
2. Ctrl-c to stop the Spark Streaming Python code.
3. Ctrl-c to stop Logstash
4. Ctrl-c to stop Elasticsearch
5. Ctrl-c to stop Kibana
6. Stop the Spark Master and Slave servers

cd ~/spark-2.3.0-bin-hadoop2.7/

sbin/stop-slave.sh

sbin/stop-master.sh

1. Ctrl-c to stop the kafka-console-consumer.
2. Stop the Kafka Server

cd ~/kafka\_2.11-1.0.0/

bin/kafka-server-stop.sh

1. Stop the Zookeeper Server

cd ~/kafka\_2.11-1.0.0/

bin/zookeeper-server-stop.sh

1. Stop the StreamSets Pipeline by clicking  and answering “Yes” to confirm stopping.

# **Spark Streaming Python Resources**

<https://www.rittmanmead.com/blog/2017/01/getting-started-with-spark-streaming-with-python-and-kafka/>

<https://www.rittmanmead.com/blog/2017/01/data-processing-and-enrichment-in-spark-streaming-with-python-and-kafka/>

# **ElasticSearch Python Resources**

<https://www.elastic.co/products/hadoop>

<https://www.elastic.co/guide/en/elasticsearch/hadoop/master/spark.html>