

# Day 4 - Lab

## Flume

---

### Task 1: Preparation

- At the command prompt, enter the following to install **Flume**.

```
yum install -y flume
```

bash

### Task 2: Configuration

Then we create a configuration file **example.conf** using

```
vi example.conf
```

bash

### Example.conf

bash

```
# example.conf: A single-node Flume configuration

# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# Describe the sink
a1.sinks.k1.type = logger

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

This configuration defines a single agent named a1. a1 has a source that listens for data on port 44444, a channel that buffers event data in memory, and a sink that logs event data to the console.

Enter the following as a *single line*.

bash

```
flume-ng agent --conf conf --conf-file example.conf
--name a1 -Dflume.root.logger=INFO,console
```

## Task 5: Testing

To test our configuration, we will use `nc` command.

bash

```
nc 127.0.0.1 44444
```

type in anything, e.g. `hello world` , and you shall see the result `OK` .

We can try to type in something longer. e.g. `The quick brown fox jump over the lazy dog` .

You will see that now our text has been truncated to only the first 16 characters (default limitation of a console logger).

---

## Kafka

---

First, login as `root` and switch the working directory to kafka, by typing:

```
cd /usr/hdp/2.2.4.2-2/kafka/bin
```

Start our Kafka broker by typing: `bash kafka start`

## Setup Topic

Let's imagine we want to monitor events log of a transport company. We can create an event log for all the trucks. Let's start **kafka** by using:

```
bash
kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1
--topic truckevent
```

## Setup Truck Events Producer

We will setup a *TruckEvents* folder, and download the code using following commands:

```
bash
mkdir /opt/TruckEvents

cd /opt/TruckEvents

wget https://github.com/cherhan/hadooptraining/raw/master/Day%204/trucks.zip

unzip trucks.zip
```

You shall a long list of messages about inflating, that means the file has been unzipped correctly.

Next, we will need to setup *Maven* to compile our Java code that has just been downloaded.

---

```
bash
wget http://www.carfab.com/apachsoftware/maven/maven-3/3.2.5/binaries/apache-maven-3.2.5-bin.tar.gz

tar xvf apache-maven-3.2.5-bin.tar.gz

mv apache-maven-3.2.5 /usr/local/

export PATH=/usr/local/apache-maven-3.2.5/bin:$PATH

mvn -version
```

Now we have downloaded Maven, we can proceed to compile our code.

```
bash
cd /opt/TruckEvents/Tutorials-master

mvn clean package
```

If everything works fine, we should be able to start our Kafka producer

```
bash
java -cp target/Tutorial-1.0-SNAPSHOT.jar com.hortonworks.tutorials.tutorial1.TruckEvents
Producer sandbox.hortonworks.com:6667 sandbox.hortonworks.com:2181
```

We should see the following screen as the messages are emitting.

```
15/08/19 23:38:35 INFO tutorial1.TruckEventsProducer: Sending Message #: route17k: 4, msg:2015-08-19 23:38:35.503|2|12|Normal|-74.025588|41.501001
15/08/19 23:38:36 INFO tutorial1.TruckEventsProducer: Sending Message #: route208: 4, msg:2015-08-19 23:38:36.509|3|13|Normal|-74.189190000000053|41.3389069999999914
15/08/19 23:38:37 INFO tutorial1.TruckEventsProducer: Sending Message #: route27: 4, msg:2015-08-19 23:38:37.513|4|14|Unsafe tail distance|-73.990962000000081|40.6643
679999999968
15/08/19 23:38:38 INFO tutorial1.TruckEventsProducer: Sending Message #: route17: 5, msg:2015-08-19 23:38:38.518|1|11|Normal|-79.741729000000078|42.130435000000034
15/08/19 23:38:39 INFO tutorial1.TruckEventsProducer: Sending Message #: route17k: 5, msg:2015-08-19 23:38:39.522|2|12|Normal|-74.026581|41.501068
15/08/19 23:38:40 INFO tutorial1.TruckEventsProducer: Sending Message #: route208: 5, msg:2015-08-19 23:38:40.527|3|13|Normal|-74.189262999999997|41.339009999999753
15/08/19 23:38:41 INFO tutorial1.TruckEventsProducer: Sending Message #: route27: 5, msg:2015-08-19 23:38:41.531|4|14|Normal|-73.990215000000035|40.663669999999911
15/08/19 23:38:42 INFO tutorial1.TruckEventsProducer: Sending Message #: route17: 6, msg:2015-08-19 23:38:42.535|1|11|Normal|-79.741653000000042|42.130458000000009
15/08/19 23:38:43 INFO tutorial1.TruckEventsProducer: Sending Message #: route17k: 6, msg:2015-08-19 23:38:43.539|2|12|Normal|-74.030852|41.501377
15/08/19 23:38:44 INFO tutorial1.TruckEventsProducer: Sending Message #: route208: 6, msg:2015-08-19 23:38:44.544|3|13|Normal|-74.189845999999989|41.339833999999961
15/08/19 23:38:45 INFO tutorial1.TruckEventsProducer: Sending Message #: route27: 6, msg:2015-08-19 23:38:45.547|4|14|Normal|-73.988505999999916|40.662078999999949
15/08/19 23:38:46 INFO tutorial1.TruckEventsProducer: Sending Message #: route17: 7, msg:2015-08-19 23:38:46.551|1|11|Normal|-79.738769000000048|42.131307999999999
15/08/19 23:38:47 INFO tutorial1.TruckEventsProducer: Sending Message #: route17k: 7, msg:2015-08-19 23:38:47.556|2|12|Normal|-74.031654|41.501435
15/08/19 23:38:48 INFO tutorial1.TruckEventsProducer: Sending Message #: route208: 7, msg:2015-08-19 23:38:48.561|3|13|Normal|-74.190127999999959|41.342054999999917
15/08/19 23:38:49 INFO tutorial1.TruckEventsProducer: Sending Message #: route27: 7, msg:2015-08-19 23:38:49.565|4|14|Normal|-73.986490999999887|40.660000000000082
15/08/19 23:38:50 INFO tutorial1.TruckEventsProducer: Sending Message #: route17: 8, msg:2015-08-19 23:38:50.569|1|11|Normal|-79.738090000000057|42.131597999999994
15/08/19 23:38:51 INFO tutorial1.TruckEventsProducer: Sending Message #: route17k: 8, msg:2015-08-19 23:38:51.572|2|12|Normal|-74.022702|41.501516
```

This shows that our producer is running properly.

## Test our consumer

Start another putty session, and run the following:

```
bash
cd /usr/hdp/2.2.4.2-2/kafka/bin/

sh kafka-console-consumer.sh -zookeeper localhost:2181 -topic truckevent
```

Alternatively, we can add `-from-beginning` option, to read the whole log from the start.

```
sh kafka-console-consumer.sh -zookeeper localhost:2181 -topic truckevent -from-beginning
```

## Storm

Make sure we are login to `root` and `cd ~/` to our main directory.

- Login to Ambari and make sure your Storm is up and running.
- Download the storm starter using the following command

```
wget http://public-repo-1.hortonworks.com/HDP-LABS/Projects/Storm/0.9.0.1/storm-starter-0.0.1-storm-0.9.0.1.jar
```

Three components are included in this starter kit:

1. Sentence Generator Spout
2. Sentence Split Bolt
3. WordCount Bolt

We can then start our storm process using the following commands:

```
jar storm-starter-0.0.1-storm-0.9.0.1.jar storm.starter.WordCountTopology WordCount -c storm.starter.WordCountTopology WordCount -c nimbus.host=sandbox.hortonworks.com
```

Once successfully started, we can login to Storm UI from <http://127.0.0.1:8744/> and visualize the results.

## Spark

Connect to Sandbox as 'root'

- Download some sample data using

```
wget http://en.wikipedia.org/wiki/Hortonworks
```

Upload the data to sandbox

```
hadoop fs -put ~/Hortonworks /user/guest/Hortonworks
```

Launch **pyspark** program from the prompt, using:

```
pyspark
```

This is actually a python prompt, loaded with Spark tools, so now we can load the data using

```
myLines = sc.textFile('hdfs://sandbox.hortonworks.com/user/guest/Hortonworks')
```

python

and we filter it, and take non-empty lines.

```
myLines_filtered = myLines.filter( lambda x: len(x) > 0 )
```

python

Finally, we can run some commands on Spark, using:

```
myLines_filtered.count()
```

python