# Day 4 - Lab

## Flume

### Task 1: Preparation

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

```bash
yum install -y flume
```

### Task 2: Configuration

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

```bash
vi example.conf
```

#### 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/apachesoftware/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.

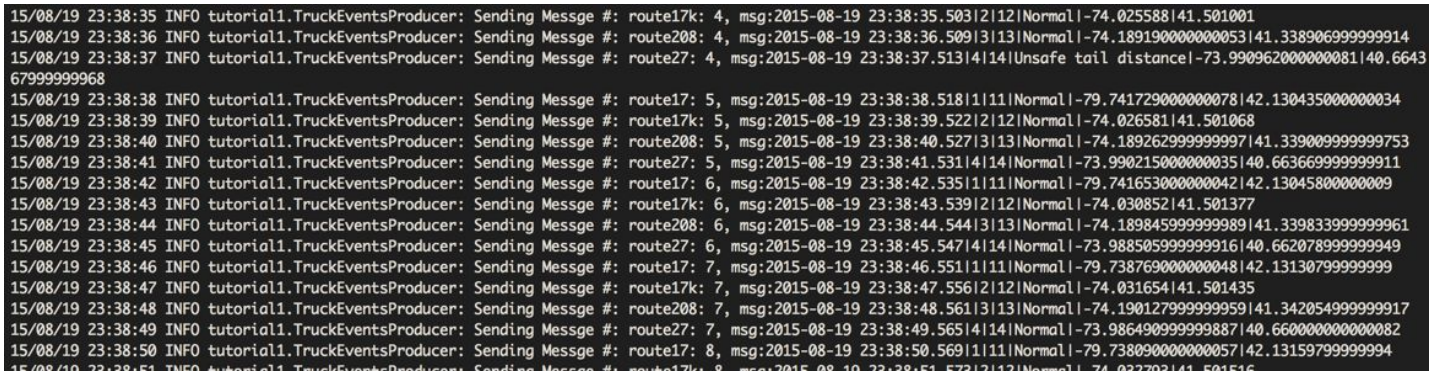
```bash
cd /opt/TruckEvents/Tutorials-master

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.TruckEventsProducer sandbox.hortonworks.com:6667 sandbox.hortonworks.com:2181
```

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




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.

```bash
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

```bash
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:

```bash
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

```bash
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

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

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

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

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

```python
myLines_filtered.count()
```