

Contents

Flume Helper Lab	2
Step 1. VM Networking.....	2
Step 2. Ensure the correct version of Java exists	3
Step 3: Install Flume	3
Step 4: Running Flume	6

Flume Helper Lab

The following is a helper lab for streaming data to HDFS using Flume. The lab assumes you have a recent Ubuntu LTS VM installed from prior class work.

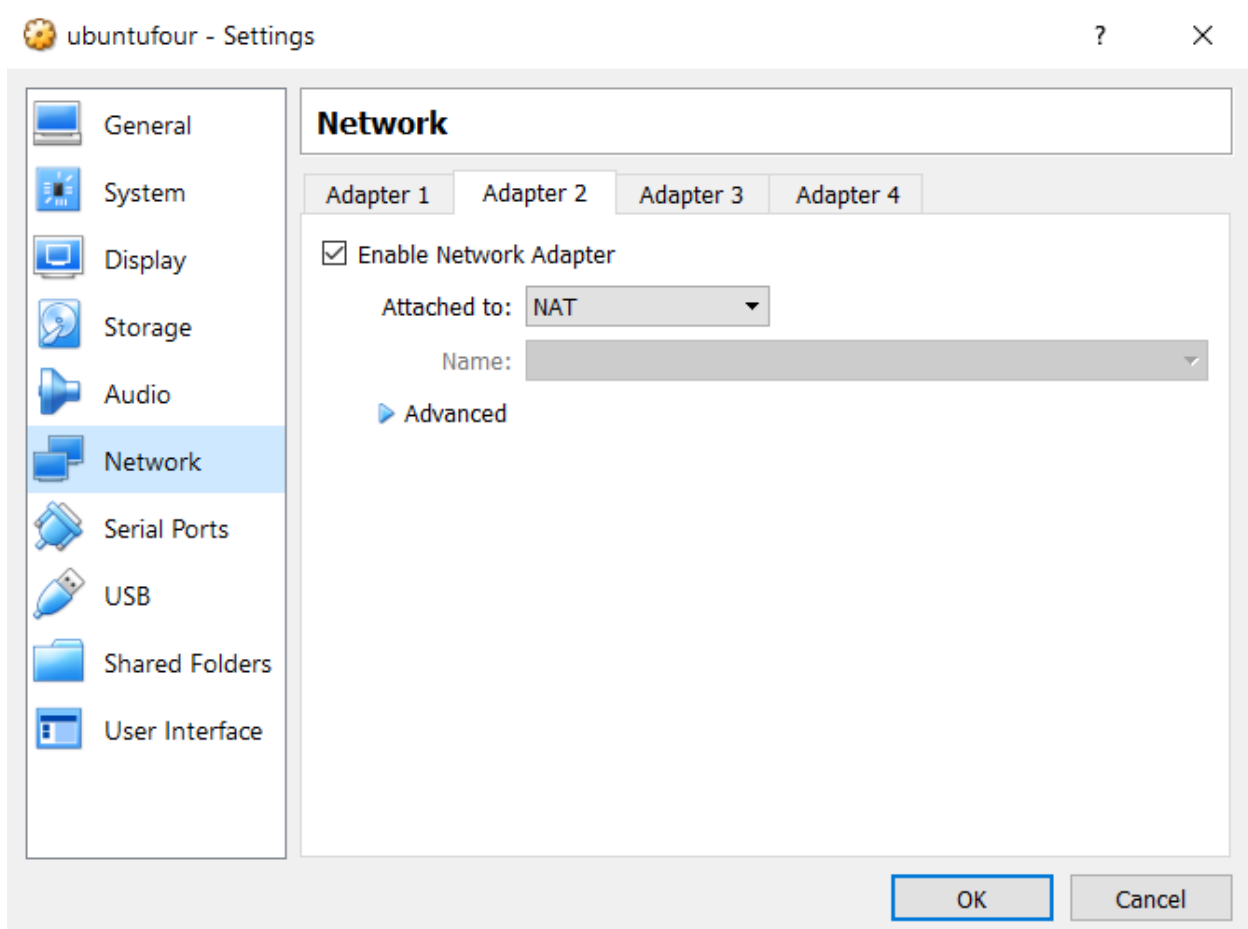
The class textbook provides the fundamentals for this lab:

<https://github.com/PacktPublishing/Big-Data-Architects-Handbook>

Step 1. VM Networking

The first step is to make sure you enable NAT or bridged mode on your Ubuntu VM. It is necessary for the VM to have access to the Internet. If you cannot ping www.google.com from your Ubuntu CLI, you need to change your NIC settings.

You may need to change your Adapter 1 to Bridged mode or NAT mode or you may also be able to have two NICs enabled with one in host-only mode and another in NAT or bridged mode. If two NICs do not work, simply change your Adapter 1 to NAT or bridged mode. See this example:



Step 2. Ensure the correct version of Java exists

Flume should work with newer versions of Java per documentation but should use the same version of Java that Hadoop/HDFS is running. Start by checking the Java version:

```
update-alternatives --config java
```

```
update-alternatives --config javac
```

Select Java 8 if Hadoop is using Java 8 and so forth.

Step 3: Install Flume

The official site for installing and downloading Flume is found her:

<https://flume.apache.org/download.html>

Download the Flume binary to your Hadoop home directory. A current download is:

```
wget https://www.apache.org/dyn/closer.lua/flume/1.11.0/apache-flume-1.11.0-bin.tar.gz
```

```
wget https://dlcdn.apache.org/flume/1.11.0/apache-flume-1.11.0-bin.tar.gz
```

Once you have the file, uncompress the compressed file in your \$HOME directory.

```
tar -xvzf apache-flume-1.11.0-bin.tar.gz
```

Rename the new folder to “flume.” Change the new folder to “flume”.

```
mv apache-flume-1.11.0-bin ./flume
```

```
cd $home
```

Next, from your home directory edit .bashrc:

```
nano .bashrc
```

Set the paths to your correct paths:

```
#Apache Flume
```

```
export FLUME_HOME=/home/bobsmith/flume
```

```
export PATH=$PATH:$FLUME_HOME/bin
```

Save the .bashrc file and source it:

```
source .bashrc
```

A common Flume error is due to an issue with the guava file being incompatible with Hadoop/HDFS. Thus, make a backup copy and remove it from the Flume library.

```
cd /home/bobsmith/flume/lib
```

ls

```
cp guava-11.0.2.jar ./guava-11.0.2.JAR.ORIGINAL
```

```
sudo rm guava-11.0.2.jar
```

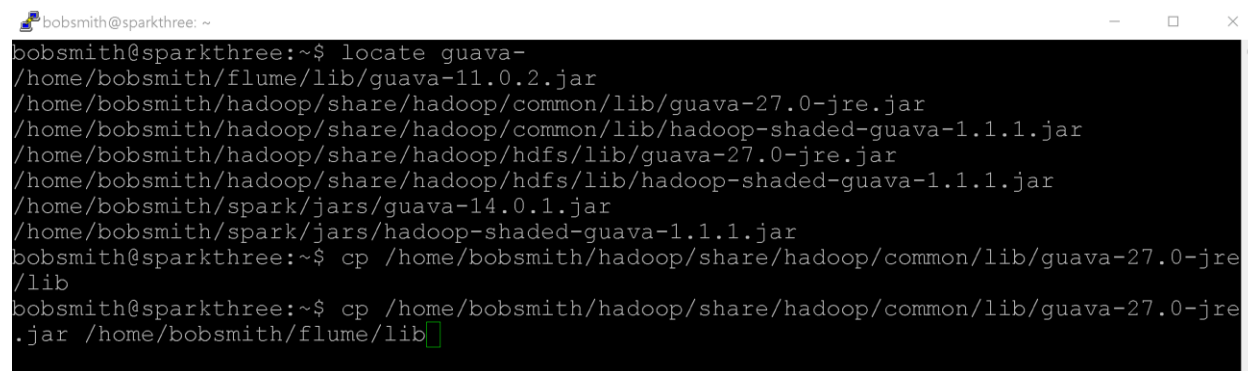
We are going to use the locate command to find the Hadoop guava jar. First, update your locate database to ensure you can find the right files:

```
sudo updatedb
```

Next, you need to locate the guava file that Hadoop uses. Thus, use the locate command to find this on your OS:

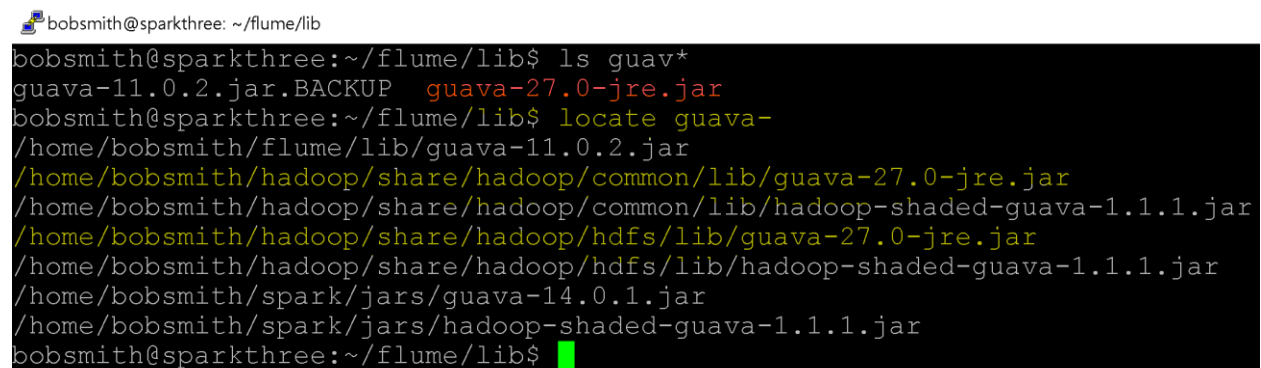
```
locate guava-
```

Copy the Hadoop guava jar file to the /flume/lib directory. Here is a screenshot as an example:



```
bobsmith@sparkthree: ~  
bobsmith@sparkthree:~$ locate guava-  
/home/bobsmith/flume/lib/guava-11.0.2.jar  
/home/bobsmith/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar  
/home/bobsmith/hadoop/share/hadoop/common/lib/hadoop-shaded-guava-1.1.1.jar  
/home/bobsmith/hadoop/share/hadoop/hdfs/lib/guava-27.0-jre.jar  
/home/bobsmith/hadoop/share/hadoop/hdfs/lib/hadoop-shaded-guava-1.1.1.jar  
/home/bobsmith/spark/jars/guava-14.0.1.jar  
/home/bobsmith/spark/jars/hadoop-shaded-guava-1.1.1.jar  
bobsmith@sparkthree:~$ cp /home/bobsmith/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar /lib  
bobsmith@sparkthree:~$ cp /home/bobsmith/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar /home/bobsmith/flume/lib
```

Make sure your Flume guava jar version is the same as Hadoop and HDFS, like this:



```
bobsmith@sparkthree: ~/flume/lib  
bobsmith@sparkthree:~/flume/lib$ ls guav*  
guava-11.0.2.jar.BACKUP  guava-27.0-jre.jar  
bobsmith@sparkthree:~/flume/lib$ locate guava-  
/home/bobsmith/flume/lib/guava-11.0.2.jar  
/home/bobsmith/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar  
/home/bobsmith/hadoop/share/hadoop/common/lib/hadoop-shaded-guava-1.1.1.jar  
/home/bobsmith/hadoop/share/hadoop/hdfs/lib/guava-27.0-jre.jar  
/home/bobsmith/hadoop/share/hadoop/hdfs/lib/hadoop-shaded-guava-1.1.1.jar  
/home/bobsmith/spark/jars/guava-14.0.1.jar  
/home/bobsmith/spark/jars/hadoop-shaded-guava-1.1.1.jar  
bobsmith@sparkthree:~/flume/lib$
```

Finally, make sure the default Java version associates with the JDK for Flume. At the time of this writing, Java 8 seems to work well with Flume. As an example:

```
bobsmith@sparkthree:~/flume/conf$ sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).
```

Selection	Path	Priority	Status
0	/usr/lib/jvm/java-11-openjdk-amd64/bin/java	1111	auto mode
1	/usr/lib/jvm/java-11-openjdk-amd64/bin/java	1111	manual mode
* 2	/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java	1081	manual mode

```
Press <enter> to keep the current choice[*], or type selection number: 2
bobsmith@sparkthree:~/flume/conf$
```

Step 4: Configuring Flume

To run Flume we have to define a source, channel, and sink as we have learned from our textbook. Go to the configuration directory:

```
cd $FLUME_HOME/conf
```

Next, make a new configuration file under the conf directory:

```
nano flume.conf
```

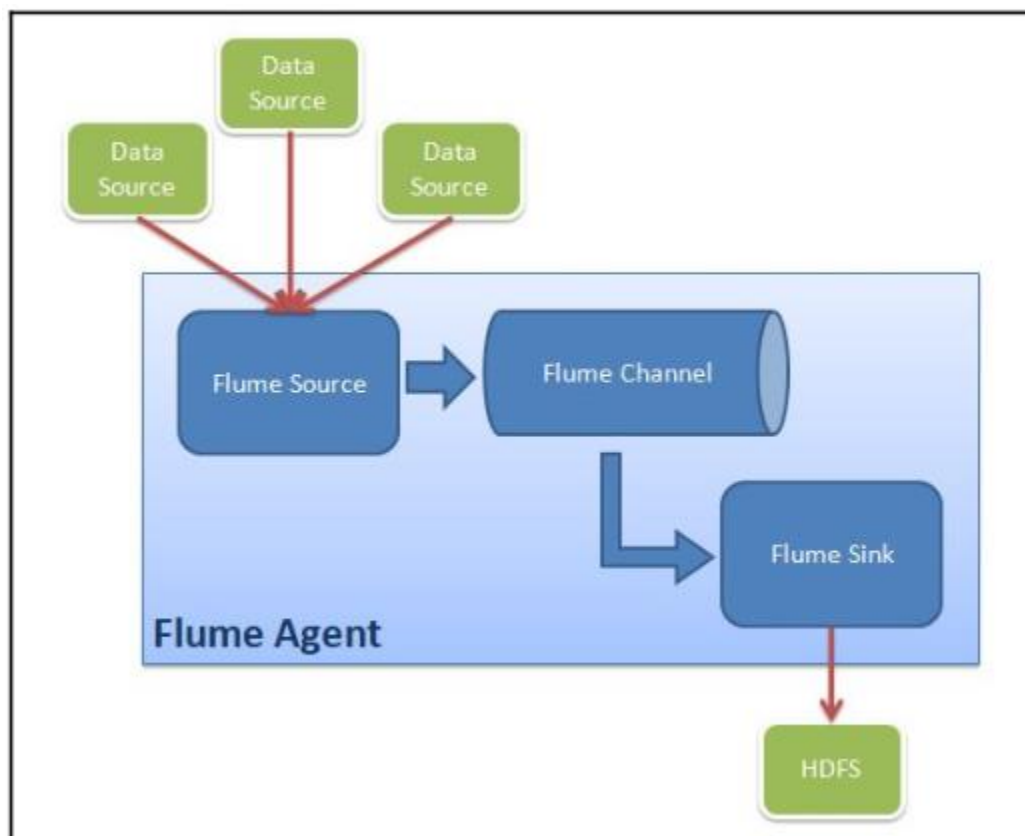
Go the website: <https://flume.apache.org/FlumeUserGuide.html>

To get the flume basic configurations. In our example, we are going to use a Netcat source to stream data. See the documentation here:

<https://netcat.sourceforge.net/>

Netcat can be used to simulate network traffic using the TCP/IP protocol. This allows use to create a source similar to a social networking account without having to use one of our actual social networking sites. Thus, think of it like one of your social networking sites.

Normally, we would stream data from one of these sites and connect it at the Flume source. The Flume source then sends data from your social networking site to the Flume channel, which buffers it prior to the HDFS write within the Flume Sink. Recall from the diagram in our Diaz textbook:



To see a working example of the flume configuration reference our textbook. The lab modifies some of the textbook configurations but here is the example of the textbook's flume configuration:

```
agent.sources=netcatSource
agent.sinks=hdfsWriter
agent.channels=memoryChannel
agent.sources.netcatSource.type=netcat
agent.sources.netcatSource.bind=127.0.0.1
agent.sources.netcatSource.port=22222
agent.sinks.hdfsWriter.type=hdfs
agent.sinks.hdfsWriter.hdfs.path=hdfs://localhost:9000/flumeData/
agent.sinks.hdfsWriter.rollInterval=60
agent.sinks.hdfsWriter.hdfs.writeFormat=Text
agent.sinks.hdfsWriter.hdfs.fileType=DataStream
agent.channels.memoryChannel.type=memory
agent.channels.memoryChannel.capacity=10000
agent.sources.netcatSource.channels=memoryChannel
agent.sinks.hdfsWriter.channel=memoryChannel
```

A few things you want to change are the `agent.sources.netcatSource.bind=127.0.0.1` and `agent.sinks.hdfsWriter.hdfs.path=hdfs://localhost:9000/flumeData/`

Change the IP addresses to your Hadoop environment. Netcat is a networking application that will listen for data streams and thus proper networking is a key aspect of flume being operational.

Here is my final configuration file after I made these changes:

```
agent.sources=netcatSource
agent.sinks=hdfsWriter
agent.channels=memoryChannel
agent.sources.netcatSource.type=netcat
agent.sources.netcatSource.bind=10.100.100.40
agent.sources.netcatSource.port=22222
agent.sinks.hdfsWriter.type=hdfs
agent.sinks.hdfsWriter.hdfs.path=hdfs://namenode:9000/flume
agent.sinks.hdfsWriter.rollInterval=60
agent.sinks.hdfsWriter.hdfs.writeFormat=Text
agent.sinks.hdfsWriter.hdfs.fileType=DataStream
agent.channels.memoryChannel.type=memory
agent.channels.memoryChannel.capacity=10000
agent.sources.netcatSource.channels=memoryChannel
agent.sinks.hdfsWriter.channel=memoryChannel
```

Note, in this flume configuration I am running multiple Hadoop/Spark nodes and thus hdfs://namenode:8020/flume exists in reality. The default Hadoop port may be 9000 or another port. Check your core-site.xml in Hadoop and use netstat to see what port is listening for HDFS. You may need use port 9000 for example:

```
hdfs://hadoopprimaryservername:9000/flume
```

Again, remember to change to the correct IP address as well!

***If this configuration is not running, you may need to change various elements such as the memoryChannel capacity so that it streams properly on your server specifically.**

Step 5: Running Flume with Hadoop HDFS

The goal of streaming services is to stream massive amounts of data into big data systems. Thus, we want to use Hadoop HDFS to stream data using Flume.

Start Hadoop HDFS and ensure HDFS is working properly before attempting to stream data with Flume.

Next, you want to make sure the HDFS path is set to an HDFS directory for Flume. For example:

```
hdfs dfs -mkdir -p /flume
hdfs dfs -chmod 777 /flume
```

To run flume, use your directory structure properly and modify this example:

```
/home/bobsmith/flume/bin/flume-ng agent -f /home/bobsmith/flume/conf/flume.conf -n agent
```

Leave Flume running in the CLI. Next, open a second terminal window and try to get the first 10 lines or so of a log file like the textbook example or you can use any unstructured data such as your eBook. I am using the Alice.txt eBook. To test it you may want to just try a few lines of the book. Thus, use the tail command to just get a few of the last lines of the book and use your proper IP address of your Flume server. Then, try to stream that data using Netcat. Netcat requires a hostname or IP address followed by a TCP/IP port to stream the data. We need to use a TCP/IP port not already in use. In the example, we are going to use port 22222. However, you may need to change the TCP/IP port to one that is not in use if it does not work.

Here is a complete example:

```
tail /home/bobsmith/alice.txt | nc 10.100.100.10 22222
```

Replace the IP address and directories with the correct replacements given your environment. We can stream any type of unstructured data. For example, Flume is often used for log files:

```
tail hadoop-bobsmith-namenode-namenode.log | nc 10.100.100.10 22222
```

In the window where Flume is still running, you should see it writing the file contents to HDFS. You can now navigate to the /flume directory in HDFS to see the files that Flume streamed.

Here is an example screenshot of streaming a log file using “cat /home/bobsmith/log.txt | nc 127.0.0.1 22222”

