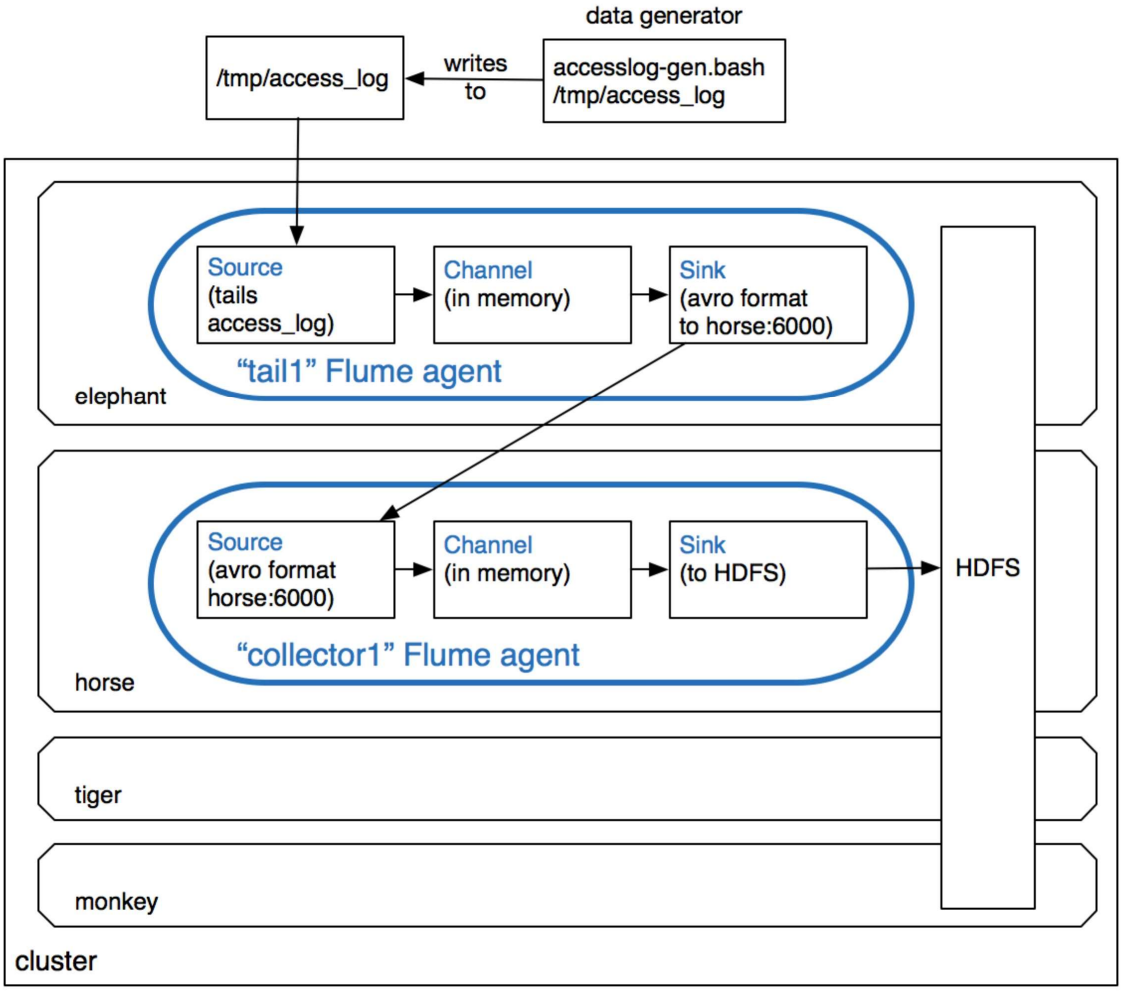
Hands-On Exercise: Using Flume to Put Data into HDFS

In this Hands-On Exercise you will use Flume to import dynamically generated data into HDFS. A very common use case for Flume is to collect access logs from a large number of Web servers on your network; we will simulate a simple version of this in the following exercise.

The diagram below shows the data flow that will occur once you complete the Exercise.



IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

Adding the Flume Service

1. Add the Flume service on `elephant` and `horse`.

   From the Cloudera Manager Home page, click the down arrow next to your cluster and choose "Add a Service".

   Select "Flume" and click "Continue". Note that HDFS is a dependency, however the HDFS service is already added. Click "Continue" again.

   Use the "Select hosts" button to add the Flume Agent on both `elephant` and `horse` then click OK and Continue. At the Congratulations screen click "Finish".

   Note: you may notice that there are now two new 'Hosts' configuration issues , this time on `elephant` and `horse` that - like the one that appeared on `monkey` earlier - are related to memory overcommit validation thresholds. In a true production cluster, you would want to address all configuration issues, however you can safely ignore these in the classroom environment.

2. Update the configuration for the Flume agent on `elephant`.

From the Cloudera Manager Home page, click the Flume link and then click on the Instances tab.

Select the "Agent" that resides on `elephant`, then click Configuration.

Delete the default contents of the two properties listed in the table below entirely and replace with the lines shown below.

Note: the Configuration File lines are also available in `~/training_materials/admin/scripts/flume-tail1.txt`

Tip: You can expand the Configuration File text area to make it easier to edit in by dragging it out the bottom right corner of the text box.

Property	**Value**
Agent Name	`tail1`

Configuration File	`tail1.sources = src1`
	`tail1.channels = ch1`
	`tail1.sinks = sink1`
	
	`tail1.sources.src1.type = exec`
	`tail1.sources.src1.command = tail -F /tmp/access_log`
	`tail1.sources.src1.channels = ch1`
	
	`tail1.channels.ch1.type = memory`
	`tail1.channels.ch1.capacity = 500`
	
	`tail1.sinks.sink1.type = avro`
	`tail1.sinks.sink1.hostname = horse`
	`tail1.sinks.sink1.port = 6000`
	`tail1.sinks.sink1.batch-size = 1`
	`tail1.sinks.sink1.channel = ch1`

Click Save Changes.

3. Update the configuration for the Flume agent on `horse`.

From the Cloudera Manager Flume page, click on the Instances tab.

Select the Agent that resides on `horse`, then click Configuration.

Delete the default contents of the two properties listed in the table below entirely and replace with the lines shown below.

Note: the Configuration File lines are also available in `~/training_materials/admin/scripts/flume-collector1.txt`

Property	**Value**
Agent Name	`collector1`

Configuration File	`collector1.sources = src1`
	`collector1.channels = ch1`
	`collector1.sinks = sink1`
	
	`collector1.sources.src1.type = avro`
	`collector1.sources.src1.bind = horse`
	`collector1.sources.src1.port = 6000`
	`collector1.sources.src1.channels = ch1`
	
	`collector1.channels.ch1.type = memory`
	`collector1.channels.ch1.capacity = 500`
	
	`collector1.sinks.sink1.type = hdfs`
	`collector1.sinks.sink1.hdfs.path =`
	`hdfs://elephant/user/flume/collector1`
	`collector1.sinks.sink1.hdfs.filePrefix = access_log`
	`collector1.sinks.sink1.channel = ch1`

Ensure that "collector1.sinks.sink1.hdfs.path = hdfs://elephant/user/flume/collector1" shown above is all on a single line.

Click "Save changes."

4. Create the `/user/flume/collector1` directory in HDFS to store the files.

   On **elephant**:

```
$ sudo -u hdfs hdfs dfs -mkdir -p \
/user/flume/collector1
$ sudo -u hdfs hdfs dfs -chown -R flume /user/flume
```

Starting the Data Generator

1 Open a new terminal window on elephant (or an ssh connection to elephant). In this terminal window, run the `accesslog-gen.bash` shell script, which simulates a Web server creating log files. This shell script also rotates the log files regularly.

   On **elephant**:

```
$ accesslog-gen.sh /tmp/access_log
```

   Note: The `accesslog-gen.bash` script is specific to the training environment and is *not* part of CDH.

2 Open a second new terminal window on `elephant` (or an ssh connection to `elephant`). Verify that the log file has been created. Notice that the log file is rotated periodically.

   On **elephant**:

```
$ ls -l /tmp/access*
-rw-rw-r-- 1 training training  498 Nov 15 15:12 /tmp/access_log
-rw-rw-r-- 1 training training  997 Nov 15 15:12 /tmp/access_log.0
-rw-rw-r-- 1 training training 1005 Nov 15 15:11 /tmp/access_log.1
```

Starting the Flume Collector Agent

Here you start the Flume agent that will insert the data into HDFS. This agent receives data from the source Flume agent.

1. Start the collector1 Flume Agent on `horse`.

   In Cloudera Manager, go to Flume's Instances tab. Select the Agent hosted on `horse`.

   From the "Actions for Selected" menu choose Start. In the "Start" window that appears, click "Start".

   In the "Command Details: Start" screen wait for confirmation that the agent started successfully. Click Close.

Starting the Flume Source Agent

Here you start the agent that reads the source log files and passes the data along to the collector agent you have already started.

1. Start the tail1 Flume Agent on `elephant`.

   From the Cloudera Manager Flume Instances tab, select the Agent hosted on `elephant`.

   From the "Actions for Selected" menu choose Start. In the "Start" window that appears, click "Start".

   In the "Command Details: Start" screen wait for confirmation that the agent started successfully. Click Close.

Viewing Data in HDFS

1. Confirm the data is being written into HDFS.

   In Cloudera Manager browse to the HDFS page for your cluster and click on File Browser.

   Drill down into /user/flume/collector1. You should see many access_log files.

2. View Metric Details.

   Return to the Flume page and click on the Metric Details tab. Here you can see details related to the Channels, Sinks, and Sources of your running Flume agents. If you are interested, an explanation of the metrics available is at http://bit.ly/flumemetrics.

Increase the File Size in HDFS (Optional)

These two steps are optional, but may be of interest to some students.

1 Edit the Collector1 agent configuration settings on `horse` by adding these three additional name value pairs:

```
collector1.sinks.sink1.hdfs.rollSize = 2048
collector1.sinks.sink1.hdfs.rollCount = 100
collector1.sinks.sink1.hdfs.rollInterval = 60
```

Click Save Changes.

2 From the Flume Status page, click on the  Stale Configuration - refresh needed icon and follow the prompts to refresh the cluster.

3 Execute `hdfs dfs -ls /user/flume/collector1` in a terminal window and note the file size of the more recent content posted by Flume to HDFS.

Viewing the Logs

1. Check the log files to see messages.

   In Cloudera Manager choose Diagnostics > Logs.

   Click Select Sources and configure as follows:
   - Uncheck all sources except Flume
   - Set the Minimum Log Level to INFO
   - Leave the timeframe of your search set to 30 minutes

   Click Search.

Browse through the logged actions from both Flume agents.

Cleaning Up

1. Stop the log generator by hitting Ctrl-C in the first terminal window.

2. Stop both Flume agents from the Flume Instances page in Cloudera Manager.

3. Remove the generated access log files from the /tmp directory to clear up space on your virtual machine.

   On **`elephant`**:

```
$ rm -rf /tmp/access_log*
```

This is the end of the Exercise.

Hands-On Exercise: Importing Data with Sqoop

For this exercise you will import data from a relational database using Sqoop. The data you load here will be used in a subsequent exercise.

Consider the MySQL database `movielens`, derived from the MovieLens project from University of Minnesota. (See note at the end of this exercise.) The database consists of several related tables, but we will import only two of these: `movie`, which contains about 3,900 movies; and `movierating`, which has about 1,000,000 ratings of those movies.

IMPORTANT: This exercise builds on the previous one. If you were unable to complete the previous exercise or think you may have made a mistake, run the following command and follow the prompts to prepare for this exercise before continuing:

```
$ ~/training_materials/admin/scripts/reset_cluster.sh
```

Perform all steps in this exercise on `elephant`.

Reviewing the Database Tables

First, review the database tables to be loaded into Hadoop.

1. Log on to MySQL.

   On **`elephant`**:

   ```
   $ mysql --user=training --password=training movielens
   ```

2. Review the structure and contents of the `movie` table.

   On **`elephant`**:

```
mysql> DESCRIBE movie;
. . .
mysql> SELECT * FROM movie LIMIT 5;
```

3. Note the column names for the table.

---

4. Review the structure and contents of the `movierating` table.

   On **`elephant`**:

```
mysql> DESCRIBE movierating;
. . .
mysql> SELECT * FROM movierating LIMIT 5;
```

5. Note these column names.

---

6. Exit MySQL.

   On **`elephant`**:

```
mysql> quit;
```

Adding the Sqoop 1 Client

1. Add the Sqoop 1 Client gateway on `elephant`.

   From the Home page in Cloudera Manager, click the down arrow icon next to Cluster 1 and choose "Add a Service".

   Select `Sqoop 1 Client` and click Continue.

   At the "Custom Role Assignments" page, click on the Select hosts box and choose to add the Gateway on `elephant`. Click Continue.

The “Progress” page appears. Once the client configuration has been deployed successfully, click Continue. At the “Congratulations” screen click Finish.

2. Using `sudo`, create a symlink to the MySQL JDBC driver.

   On **`elephant`**:

```
$ sudo ln -s /usr/share/java/mysql-connector-java.jar \
/opt/cloudera/parcels/CDH-5.3.2-1.cdh5.3.2.p0.10\
/lib/sqoop/lib/
```

   Now run the command below to confirm the symlink was properly created.

   On **`elephant`**:

```
$ readlink -f /opt/cloudera/parcels/CDH-5.3.2-\
1.cdh5.3.2.p0.10/lib/sqoop/lib/mysql-connector-java.jar
```

   If the symlink was properly defined, the command should return the `/usr/share/java/mysql-connector-java.jar` path.

Importing with Sqoop

You invoke Sqoop on the command line to perform several commands. With it you can connect to your database server to list the databases (schemas) to which you have access, and list the tables available for loading. For database access, you provide a connect string to identify the server, and your username and password.

1. Show the commands available in Sqoop.

   On **`elephant`**:

```
$ sqoop help
```

   You can safely ignore the warning that Accumulo does not exist since this course does not use Accumulo.

2. List the databases (schemas) in your database server.

On **elephant**:

```
$ sqoop list-databases \
--connect jdbc:mysql://localhost \
--username training --password training
```

(Note: Instead of entering `--password training` on your command line, you may prefer to enter `-P`, and let Sqoop prompt you for the password, which is then not visible when you type it.)

3. List the tables in the `movielens` database.

On **elephant**:

```
$ sqoop list-tables \
  --connect jdbc:mysql://localhost/movielens \
  --username training --password training
```

4. Import the `movie` table into Hadoop.

On **elephant**:

```
$ sqoop import \
--connect jdbc:mysql://localhost/movielens \
--table movie --fields-terminated-by '\t' \
--username training --password training
```

The `--fields-terminated-by '\t'` option separates the fields in the HDFS file with the tab character, which is sometimes useful if users will be working with Hive and Pig.

Warnings that packages such as hbase, hive-hcatalog, and accumulo are not installed are expected. It is not a problem that these packages are not installed on your system.

Notice how the INFO messages that appear show that a MapReduce job consisting of four map tasks was completed.

5. Verify that the command has worked.

   On **`elephant`**:

```
$ hdfs dfs -ls movie
$ hdfs dfs -tail movie/part-m-00000
```

6. Import the `movierating` table into Hadoop using the command in step 4 as an example.

   Verify that the `movierating` table was imported using the command in step 5 as an example or by using the Cloudera Manager HDFS page's File Browser.

7. Optionally observe the results in Cloudera Manager's YARN Applications page.

   Navigate to the YARN Applications page.

   Notice the last two YARN applications that ran (movie.jar and movierating.jar).

   Explore the job details for either or both of these jobs.

This is the end of the Exercise

Note:

This exercise uses the MovieLens data set, or subsets thereof. This data is freely available for academic purposes, and is used and distributed by Cloudera with the express permission of the UMN GroupLens Research Group. If you would like to use this data for your own research purposes, you are free to do so, as long as you cite the GroupLens Research Group in any resulting publications. If you would like to use this data for commercial purposes, you must obtain explicit permission. You may find the full dataset, as well as detailed license terms, at http://www.grouplens.org/node/73