**Ejercicios Flume**

**1. Configuración básica de Flume**

1. La configuración de un agente Flume está almacenada en un fichero. A continuación, se detalla el contenido que ha de tener dicho fichero. Para este ejercicio vamos a definir:
   ```
   Un agente: a1
   Que escucha por un puerto: 44444
   Un channel que almacena los datos en memoria
   Un sink que muestra datos por consola
   ```
2. La máquina virtual con la que estamos trabajando no tiene Telnet instalado, por lo que lo instalaremos. Para ello ejecutar los siguientes pasos:
   ```
   yum install telnet telnet-server -y
   sudo chmod 777 /etc/xinetd.d/telnet
   ```
3. Editamos el archivo anterior y actualizamos la variable disable=no
   ```
   vi /etc/xinetd.d/telnet
   disable=no
   ```
4. comprobamos que está correcto
   ```
   cat /etc/xinetd.d/telnet
   ```
5. Reiniciamos el servicio
   ```
   sudo service xinetd start
   ```
6. Hacemos que el servicio se arranque automáticamente
   ```
   sudo chkconfig telnet on
   sudo chkconfig xinetd on
   ```
7. En el directorio /etc/flume-ng están las carpetas que contienen las plantillas que hay que configurar para realizar una importación de datos con flume.
8. Para este ejercicio vamos a hacer la prueba de escribir por consola lo que escribamos a través de Telnet en un Shell. Para ello creamos un fichero llamado “example.conf” y lo guardamos en “/home/cloudera”. El contenido de este fichero es

```
# 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
```

9. Abrimos un Shell. En la primera de ellas ejecutamos el siguiente comando, que arranca el agente flume.

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

10. Abrimos otro shell donde ejecutamos

```
telnet localhost 44444
```

11. Ahora probamos a escribir algo en este Segundo shell, donde hemos ejecutado el telnet, y vemos cómo se envía al primer shell

**2. Importar datos de un spool-dir**

La idea de este ejercicio es ir dejando ficheros en un directorio (spool-dir) y ver como flume los va consumiendo.

1. Creamos el directorio spool y le damos permisos

```
sudo mkdir -p /var/log/apache/flumeSpool
sudo chmod 777 /var/log/apache/flumeSpool
```

2. Tendríamos que crear también los directorios checkpoint y datadir. Si no lo hacemos, flume lo crea por nosotros. Para poder utilizarlo le damos permisos a dicho directorio, ya que sabemos dónde se va a montar. A continuación, les damos permisos

```
sudo mkdir -p /mnt/flume/checkpoint
sudo mkdir -p /mnt/flume/data
sudo chmod 777 /mnt/flume/checkpoint
sudo chmod 777 /mnt/flume/data
```

3. Creamos un fichero de configuración en la misma ruta que en el ejemplo anterior, y modificamos la configuración del source, cambiándola por esta

```
a1.sources.r1.type = spooldir
a1.sources.r1.spoolDir = /var/log/apache/flumeSpool
a1.sources.r1.fileHeader = true
```

4. Arrancamos flume en un shell

```
flume-ng agent --conf conf --conf-file
/home/cloudera/example2.conf --name a1 -
Dflume.root.logger=DEBUG,console -
Dorg.apache.flume.log.printconfig=true -
Dorg.apache.flume.log.rawdata=true
```

5. Para comprobar que funciona, abrimos una nueva Shell, nos posicionamos en la ruta donde hemos definido el spool-dir y creamos un fichero con el editor vi (recomendable) o con el explorador de archivos de Linux.
6. Prestar atención al Shell donde tenemos flume corriendo y ver cómo se envían y muestran los ficheros por consola.

**3. Importar datos desde un spool-dir a HDFS**

1. Creamos el directorio en HDFS donde vamos a dejar los datos importados desde el spool-dir a través del channel de flume

```
hadoop fs -mkdir /flume
hadoop fs -mkdir /flume/events
```

2. Creamos un nuevo fichero de configuración, example3.conf, igual que el del ejemplo anterior, pero sustituyendo la descripción del sink por lo siguiente

```
a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path=/flume/events
```

3. Corremos el agente flume

```
flume-ng agent --conf conf --conf-file
/home/cloudera/example3.conf --name a1 -
Dflume.root.logger=DEBUG,console -
Dorg.apache.flume.log.printconfig=true -
Dorg.apache.flume.log.rawdata=true
```

4. Nos posicionamos en el directorio spool y creamos un fichero con algo escrito. Después accedemos a la carpeta HDFS donde se supone que debe estar y vemos si está. Tarda un poco.
5. Prestad atención al nombre de la carpeta
6. Para mejorar un poco la info que nos devuelve Flume, añadimos la siguiente configuración de hdfs.

```
a1.sinks.k1.hdfs.path = /flume/events/%y-%m-
%d/%H%M/%S
a1.sinks.k1.hdfs.filePrefix = events-
a1.sinks.k1.hdfs.round = true
a1.sinks.k1.hdfs.roundValue = 10
a1.sinks.k1.hdfs.roundUnit = minute
```

7. Observad cómo cambia la estructura de carpetas donde se almacenan los datos en el sink. Creamos un nuevo fichero en el spool y vamos a HDFS para ver cómo se ha importado.
   (hadoop fs -cat /flume/events/*
   hadoop fs -rm /flume/events/*)
8. Si habéis abierto uno de los ficheros de datos importados, os habréis dado cuenta de que el contenido del fichero que enviáis tiene caracteres extraños. Esto es porque por defecto flume escribe datos serializados (….BytesWritable). Si recordáis del primer día de clase, una de las propiedades de Hadoop es que serializa los datos con los que trabaja (interfaz Writable). Existe una forma de solucionar esto, que es lo que tenéis que hacer en este punto. Acceded a la web de flume y buscad la propiedad que hace que se muestren los datos en formato Texto:

```
a1.sinks.k1.hdfs.writeFormat = Text
a1.sinks.k1.hdfs.fileType = DataStream
```