

## FLUME

ApacheFlume es un sistema distribuido, confiable y preparado para recopilar, agregar y mover de forma eficiente grandes cantidades de datos de diversas fuentes a un almacén de datos centralizado.

Posee una arquitectura sencilla y flexible basada en flujos de datos en streaming.

Es robusto y tolerante a fallos con múltiples mecanismos de configuración.

Utiliza la transaccionalidad en la comunicación entre sus componentes.

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:

- a. Un agente: a1
- b. Que escucha por un puerto: 44444
- c. Un channel que almacena los datos en memoria
- d. Un sink que muestra datos por consola

```
[cloudera@quickstart ~]$ flume
bash: flume: command not found
[cloudera@quickstart ~]$ cd /etc/flume-ng/conf
[cloudera@quickstart conf]$ ls
flume.conf                flume-env.ps1.template  log4j.properties
flume-conf.properties.template  flume-env.sh.template
[cloudera@quickstart conf]$ cat flume-conf.properties.template
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
```

2. La máquina virtual con la que estamos trabajando no tiene Telnet instalado, por lo que lo instalaremos. Para ello ejecutar los siguientes pasos: a. `yum install telnet telnet-server -y` b. `sudo chmod 777 /etc/xinetd.d/telnet`

```
command line error: no such option: -y
[cloudera@quickstart ~]$ yum install telnet telnet-server -y sudo chmod 777 /etc
/xinetd.d/telnet
Loaded plugins: fastestmirror, security
You need to be root to perform this command.
[cloudera@quickstart ~]$
```

3. Editamos el archivo anterior y actualizamos la variable `disable=no`  
a. `vi /etc/xinetd.d/telnet` b. `disable=no`

4. comprobamos que está correcto

a. `cat /etc/xinetd.d/telnet`

```
[cloudera@quickstart ~]$ cat /etc/xinetd.d/telnet
# default: on
# description: The telnet server serves telnet sessions; it uses \
#      unencrypted username/password pairs for authentication.
service telnet
{
    disable = no
    flags          = REUSE
    socket_type    = stream
    wait          = no
    user           = root
    server         = /usr/sbin/in.telnetd
    log_on_failure += USERID
}
[cloudera@quickstart ~]$
```

## 5. Reiniciamos el servicio

### a. sudo service xinetd start

```
[cloudera@quickstart ~]$ sudo service xinetd start
Starting xinetd:
[cloudera@quickstart ~]$ █
```

## 6. Hacemos que el servicio se arranque automáticamente

### a. sudo chkconfig telnet on

### b. sudo chkconfig xinetd on

```
[cloudera@quickstart ~]$ sudo chkconfig telnet on
[cloudera@quickstart ~]$ sudo chkconfig xinetd on
[cloudera@quickstart ~]$ █
```

## 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.

```
[cloudera@quickstart conf]$ sudo vim flume-example.conf
```

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”.

```
[cloudera@quickstart conf]$ sudo vim flume-example.conf
```

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

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

```
[cloudera@quickstart ~]$ flume-ng agent --conf conf --conf-file /home/cloudera/example.conf  
--name a1 -Dflume.root.logger=INFO,console  
Info: Including Hadoop libraries found via (/usr/bin/hadoop) for HDFS access  
Info: Including HBASE libraries found via (/usr/bin/hbase) for HBASE access  
Info: Including Hive libraries found via () for Hive access  
+ exec /usr/java/jdk1.7.0_67-cloudera/bin/java -Xmx20m -Dflume.root.logger=INFO,console -cp  
'conf:/usr/lib/flume-ng/lib/*:/etc/hadoop/conf:/usr/lib/hadoop/lib/*:/usr/lib/hadoop/./*/:u  
sr/lib/hadoop-hdfs/./:/usr/lib/hadoop-hdfs/lib/*:/usr/lib/hadoop-hdfs/./*/:/usr/lib/hadoop-y  
arn/lib/*:/usr/lib/hadoop-yarn/./*/:/usr/lib/hadoop-mapreduce/lib/*:/usr/lib/hadoop-mapreduc  
e/./*/:/usr/lib/hbase/bin/./conf:/usr/java/jdk1.7.0_67-cloudera/lib/tools.jar:/usr/lib/hbas  
e/bin/./:/usr/lib/hbase/bin/./lib/activation-1.1.jar:/usr/lib/hbase/bin/./lib/apacheds-i18  
n-2.0.0-M15.jar:/usr/lib/hbase/bin/./lib/apacheds-kerberos-codec-2.0.0-M15.jar:/usr/lib/hba  
se/bin/./lib/api-asn1-api-1.0.0-M20.jar:/usr/lib/hbase/bin/./lib/api-util-1.0.0-M20.jar:/u  
sr/lib/hbase/bin/./lib/asm-3.2.jar:/usr/lib/hbase/bin/./lib/avro.jar:/usr/lib/hbase/bin/./  
lib/aws-java-sdk-bundle-1.11.134.jar:/usr/lib/hbase/bin/./lib/commons-beanutils-1.9.2.jar:  
/usr/lib/hbase/bin/./lib/commons-beanutils-core-1.8.0.jar:/usr/lib/hbase/bin/./lib/commons  
-cli-1.2.jar:/usr/lib/hbase/bin/./lib/commons-codec-1.9.jar:/usr/lib/hbase/bin/./lib/commo  
ns-collections-3.2.2.jar:/usr/lib/hbase/bin/./lib/commons-compress-1.4.1.jar:/usr/lib/hbase
```

10. Abrimos otro shell donde ejecutamos a. telnet localhost 44444

```
[cloudera@quickstart ~]$ telnet localhost 44444  
Trying 127.0.0.1...  
telnet: connect to address 127.0.0.1: Connection refused  
[cloudera@quickstart ~]$
```

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

a. sudo mkdir -p /var/log/apache/flumeSpool

b. sudo chmod 777 /var/log/apache/flumeSpool

```
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ sudo mkdir -p /var/log/apache/flumeSpool
[cloudera@quickstart ~]$ sudo chmod 777 /var/log/apache/flumeSpool
[cloudera@quickstart ~]$ sudo mkdir -p /mnt/flume/checkpoint
[cloudera@quickstart ~]$ sudo mkdir -p /mnt/flume/data
[cloudera@quickstart ~]$ sudo chmod 777 /mnt/flume/checkpoint
[cloudera@quickstart ~]$ sudo chmod 777 /mnt/flume/data
[cloudera@quickstart ~]$
```

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 a.

- a. `sudo mkdir -p /mnt/flume/checkpoint`
- b. `sudo mkdir -p /mnt/flume/data`
- c. `sudo chmod 777 /mnt/flume/checkpoint`
- d. `sudo chmod 777 /mnt/flume/data`

```
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ sudo mkdir -p /var/log/apache/flumeSpool
[cloudera@quickstart ~]$ sudo chmod 777 /var/log/apache/flumeSpool
[cloudera@quickstart ~]$ sudo mkdir -p /mnt/flume/checkpoint
[cloudera@quickstart ~]$ sudo mkdir -p /mnt/flume/data
[cloudera@quickstart ~]$ sudo chmod 777 /mnt/flume/checkpoint
[cloudera@quickstart ~]$ sudo chmod 777 /mnt/flume/data
[cloudera@quickstart ~]$
```

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

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

```
ta [cloudera@quickstart ~]$ sudo vim example2.conf
ta [cloudera@quickstart ~]$ ls
ta cloudera-manager enterprise-deployment.json Music Public
ta cm_api.py example2.conf nada Templates
ta Desktop example.conf nada01 Videos
ta Documents express-deployment.json nada2 workspace
ta Downloads hive parcels
ta eclipse kerberos Pictures
ta ejercicios lib pig_1649319503309.log
[cloudera@quickstart ~]$ flume-ng agent --conf conf --conf-file /home/cloudera/ex
ample2.conf --name a1 -Dflume.root.logger=DEBUG,console -Dorg.apache.flume.log.pr
intconfig=true -Dorg.apache.flume.log.rawdata=true
Info: Including Hadoop libraries found via (/usr/bin/hadoop) for HDFS access
Info: Including HBASE libraries found via (/usr/bin/hbase) for HBASE access
Info: Including Hive libraries found via () for Hive access
+ exec /usr/java/jdk1.7.0_67-cloudera/bin/java -Xmx20m -Dflume.root.logger=DEBUG,
console -Dorg.apache.flume.log.printconfig=true -Dorg.apache.flume.log.rawdata=tr
ue -cp 'conf:/usr/lib/flume-ng/lib/*:/etc/hadoop/conf:/usr/lib/hadoop/lib/*:/usr/
```

4. Arrancamos flume en un shell a. 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

```
ta [cloudera@quickstart ~]$ sudo vim example2.conf
ta [cloudera@quickstart ~]$ ls
ta cloudera-manager enterprise-deployment.json Music Public
ta cm_api.py example2.conf nada Templates
ta Desktop example.conf nada01 Videos
ta Documents express-deployment.json nada2 workspace
ta Downloads hive parcels
ta eclipse kerberos Pictures
ta ejercicios lib pig_1649319503309.log
[cloudera@quickstart ~]$ flume-ng agent --conf conf --conf-file /home/cloudera/ex
ample2.conf --name a1 -Dflume.root.logger=DEBUG,console -Dorg.apache.flume.log.pr
intconfig=true -Dorg.apache.flume.log.rawdata=true
Info: Including Hadoop libraries found via (/usr/bin/hadoop) for HDFS access
Info: Including HBASE libraries found via (/usr/bin/hbase) for HBASE access
Info: Including Hive libraries found via () for Hive access
+ exec /usr/java/jdk1.7.0_67-cloudera/bin/java -Xmx20m -Dflume.root.logger=DEBUG,
console -Dorg.apache.flume.log.printconfig=true -Dorg.apache.flume.log.rawdata=tr
ue -cp 'conf:/usr/lib/flume-ng/lib/*:/etc/hadoop/conf:/usr/lib/hadoop/lib/*:/usr/
```

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.

```
[cloudera@quickstart ~]$ cd /var/log/apache/flumeSpool
[cloudera@quickstart flumeSpool]$ sudo vim prueba
[cloudera@quickstart flumeSpool]$
```

6. Prestar atención al Shell donde tenemos flume corriendo y ver cómo se envían y muestran los ficheros por consola.

```
22/04/10 15:45:16 INFO avro.ReliableSpoolingFileEventReader: Last read took us just up to a file boundary. Rolling to the next file, if there is one.
22/04/10 15:45:16 INFO avro.ReliableSpoolingFileEventReader: Preparing to move file /var/log/apache/flumeSpool/prueba to /var/log/apache/flumeSpool/prueba.COMPLETED
22/04/10 15:45:19 INFO sink.LoggerSink: Event: { headers:{file=/var/log/apache/flumeSpool/prueba} body: 68 6F 6C 61 hola }
```

### 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

a. `hadoop fs -mkdir /flume` b. `hadoop fs -mkdir /flume/events`

```
[cloudera@quickstart ~]$ hadoop fs -ls /flume
Found 1 items
drwxr-xr-x - cloudera supergroup          0 2022-04-10 15:50 /flume/events
[cloudera@quickstart ~]$
```

2. Creamos un nuevo fichero de configuración, example

```
[cloudera@quickstart ~]$ sudo vim example3.conf
[cloudera@quickstart ~]$ sudo vim example3.conf
[cloudera@quickstart ~]$ ls
cloudera-manager enterprise-deployment.json lib pig_1649319503309.log
cm_api.py example2.conf Music Public
Desktop example3.conf nada Templates
Documents example.conf nada01 Videos
Downloads express-deployment.json nada2 workspace
eclipse hive parcels
ejercicios kerberos Pictures
[cloudera@quickstart ~]$ 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
Info: Including Hadoop libraries found via (/usr/bin/hadoop) for HDFS access
Info: Including HBASE libraries found via (/usr/bin/hbase) for HBASE access
Info: Including Hive libraries found via () for Hive access
+ exec /usr/java/jdk1.7.0_67-cloudera/bin/java -Xmx20m -Dflume.root.logger=DEBUG
```

3.conf, igual que el del ejemplo anterior, pero sustituyendo la descripción del sink por el tipo HDFS y el path que acabamos de crear en el paso anterior

a. `a1.sources.r1.type = hdfs`

b. `a1.sources.r1.bind = /flume/events` 3. Corremos el agente flume a.  
`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

```
[cloudera@quickstart ~]$ sudo vim example3.conf
[cloudera@quickstart ~]$ ls
cloudera-manager  enterprise-deployment.json  lib          pig_1649319503309.log
cm_api.py         example2.conf              Music        Public
Desktop          example3.conf              nada         Templates
Documents        example.conf               nada01       Videos
Downloads        express-deployment.json    nada2        workspace
eclipse          hive                       parcels
ejercicios       kerberos                  Pictures
[cloudera@quickstart ~]$ 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
Info: Including Hadoop libraries found via (/usr/bin/hadoop) for HDFS access
Info: Including HBASE libraries found via (/usr/bin/hbase) for HBASE access
Info: Including Hive libraries found via () for Hive access
+ exec /usr/java/jdk1.7.0_67-cloudera/bin/java -Xmx20m -Dflume.root.logger=DEBUG
```

4. Nos posicionamos en el directorio pool 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. Deteneos un momento en observar el resultado y abrid uno de los ficheros importados para ver su contenido.

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



```

File Edit View Search Terminal Help
me the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = spooldir
a1.sources.r1.spoolDir = /var/log/apache/flumeSpool
a1.sources.r1.fileHeader = true

# Describe the sink
a1.sinks.k1.type = 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
a1.sinks.k1.hdfs.writeFormat = Text
a1.sinks.k1.hdfs.fileType = DataStream

# 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

```

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.

```

[cloudera@quickstart flumeSpool]$ hadoop fs -cat /flume/events/*
SEQ[org.apache.hadoop.io.LongWritable"org.apache.hadoop.io.BytesWritable
hola manuel

```

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 laS propiedades que hace que se muestren los datos en formato Texto.

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