Práctica sesión 7 - INGESTA DE DATOS

Resumen

En este documento se trata de explicar mediante ejercicios prácticos el funcionamiento de SQOOP, FLUME y NIFI, con los que el alumno podrá experimentar el funcionamiento básico de la ingesta de datos en el entorno de Hadoop con diferentes herramientas.

Contenido

[Contenido 1](#_Toc72172598)

[1. SQOOP 2](#_Toc72172599)

[1.1. Ejemplo 1 – Importar tablas con Sqoop 2](#_Toc72172600)

[1.2. Ejemplo 2 – Compresión y formato Avro 4](#_Toc72172601)

[2. FLUME 5](#_Toc72172602)

[2.1 Ejemplos operaciones básicas con Flume 5](#_Toc72172603)

[3. NIFI 8](#_Toc72172604)

[3.1. Instalación 8](#_Toc72172605)

[3.2. Mover datos 10](#_Toc72172606)

## SQOOP

### Ejemplo 1 – Importar tablas con Sqoop

Con SQOOP podemos importar/exportar desde una BBDD a nuestro HDFS y viceversa. En esta práctica vamos a trabajar con mysql que ya tenemos instalada en nuestra máquina virtual. Cuando se lanza SQOOP convierte las marcas de tiempo de nuestra base de datos origen y las convierte a la hora del sistema servidor por lo que tenemos que especificar en nuestra base de datos la zona horaria.

La forma de realizarlo es muy simple, simplemente editamos el fichero mysqld.cnf que se encuentra en /etc/mysql/mysql.conf.d/



Y añadimos la siguiente propiedad:

|  |
| --- |
|  |

Ahora solo nos queda reiniciar el servicio de mysql:

|  |
| --- |
| sudo service mysql restart |

OJO: Esto en la sesión práctica nos dio error, si al cambiar la propiedad del fichero da error al reiniciar mysql, hay que pasar el serverTimezone por parámetro en la cadena JDBC

jdbc:mysql://localhost/ejemplo?serverTimezone=Europe/Madrid

Una vez realizado el paso anterior, vamos a crear un par de tablas en mysql, para ello abrimos la Shell de mysql:

|  |
| --- |
| sudo mysql |

Y creamos una base de datos que se va a llamar ejemplo

|  |
| --- |
| mysql> create database ejemplo;  mysql> use ejemplo; |

Creamos un par de tablas:

|  |
| --- |
| mysql> CREATE TABLE profesores(  id MEDIUMINT NOT NULL AUTO\_INCREMENT,  nombre CHAR(30) NOT NULL,  edad INTEGER(30),  materia CHAR(30),  PRIMARY KEY (id) );  mysql> CREATE TABLE profesores2(  id MEDIUMINT NOT NULL AUTO\_INCREMENT,  nombre CHAR(30) NOT NULL,  edad INTEGER(30),  materia CHAR(30),  PRIMARY KEY (id) ); |

Hacemos una inserción en la primera tabla:

|  |
| --- |
| INSERT INTO profesores (nombre, edad, materia) VALUES  ("Carlos", 24, "Matematicas"),  ("Pedro", 32, "Lenguaje"),  ("Juan", 35, "Tecnologia"),  ("Jose", 48, "Matematicas"),  ("Paula", 24, "Informatica"),  ("Susana", 32, "Informatica"),  ("Lorena", 54, "Informatica"); |

Arrancamos hdfs y yar:

|  |
| --- |
| start-dfs.sh  start-yarn.sh |

Con el comando SQOOP listamos todas las tablas de ejemplo (NO COPIAR Y PEGAR)

|  |
| --- |
| sqoop list-tables --connect jdbc:mysql://localhost/ejemplo?serverTimezone=Europe/Madrid --username=hadoop --password=hadoop |

Importamos los datos de nuestra BBDD a HDFS mediante el comando IMPORT:

|  |
| --- |
| sqoop import --connect jdbc:mysql://localhost/ejemplo?serverTimezone=Europe/Madrid  --username=hadoop --password=hadoop  --table=profesores --driver=com.mysql.jdbc.Driver  --target-dir=/user/hadoop/sqoop/profesores\_hdfs  --fields-terminated-by=’,’ --lines-terminated-by ‘\n’ |

A través del navegador web de ficheros de hdfs localhost:9870 podremos comprobar en el directorio /user/hadoop/sqoop que ha creado el directorio que hemos especificado junto con los archivos:

|  |
| --- |
| **EJERCICIO [Adjuntar captura]** |

Ahora vamos a hacer el paso contrario, desde hdfs vamos a exportar los ficheros a nuestra otra tabla que habíamos creado en mysql para ello lanzamos la siguiente orden:

|  |
| --- |
| sqoop export --connect jdbc:mysql://localhost/ejemplo?serverTimezone=Europe/Madrid  --username=hadoop  --password=hadoop  --table=profesores2  --export-dir=/user/hadoop/sqoop/profesores\_hdfs  --columns=”id,nombre,edad,materia” |

Con la opción --columns=”columna1,columna2” restringimos el número de columnas.

Con la opción --where=”campo>valor” restringimos las condiciones de importación

|  |
| --- |
| sqoop import --connect jdbc:mysql://localhost/ejemplo?serverTimezone=Europe/Madrid  --username=hadoop --password=hadoop  --table=profesores  --driver=com.mysql.jdbc.Driver  --target-dir=/user/hadoop/sqoop/where\_hdfs  --columns=”nombre,edad”  --where=“edad>30” |

### Ejemplo 2 – Compresión y formato Avro

Avro es un formato de almacenamiento basado en filas para Hadoop que es usado ampliamente como plataforma de serialización.

Snappy es una biblioteca de compresión y descompresión de datos rápida que es utilizado con frecuencia en proyectos Big Data.

Para que funcione la serialización con Avro hay que copiar el fichero .jar que viene en el directorio de Sqoop para Avro con el siguiente comando:

|  |
| --- |
| cp $SQOOP\_HOME/lib/avro-1.8.1.jar $HADOOP\_HOME/share/hadoop/common/lib/  rm $HADOOP\_HOME/share/hadoop/common/lib/avro-1.7.7.jar |

Ahora ya podemos lanzar el siguiente comando:

|  |
| --- |
| sqoop import --connect jdbc:mysql://localhost/ejemplo?serverTimezone=Europe/Madrid  --username=hadoop --password=hadoop  --table=profesores  --driver=com.mysql.jdbc.Driver  --target-dir=/comprimido\_hdfs  --compress  --compression-codec org.apache.hadoop.io.compress.SnappyCodec  --as-avrodatafile |

Probamos a hacer la ingesta con compresión BZip2 y formato secuencial:

|  |
| --- |
| sqoop import --connect jdbc:mysql://localhost/ejemplo?serverTimezone=Europe/Madrid  --username=hadoop --password=hadoop  --table=profesores  --driver=com.mysql.jdbc.Driver  --target-dir=/BZip\_hdfs  --compress  --compression-codec org.apache.hadoop.io.compress.BZip2Codec  --as-sequencefile |

## FLUME

### 2.1 Ejemplos operaciones básicas con Flume

Vamos a ver con un par de ejemplos cómo podemos a través de flume, generar, recopilar y agregar datos desde un origen a nuestro sistema HDFS.

* Nos movemos hasta el directorio $FLUME\_HOME y creamos un fichero llamado seq\_gen.conf en el directorio /conf con el siguiente contenido:

|  |
| --- |
| #Nombramos a los componentes del agente  SeqGenAgent.sources = SeqSource SeqGenAgent.channels = MemChannel  SeqGenAgent.sinks = HDFS    # Describimos el tipo de origen  SeqGenAgent.sources.SeqSource.type = seq    # Describimos el destino  SeqGenAgent.sinks.HDFS.type = hdfs  SeqGenAgent.sinks.HDFS.hdfs.path = hdfs://hadoop-VirtualBox:9000/user/hadoop/seqgen\_data/  SeqGenAgent.sinks.HDFS.hdfs.filePrefix = log SeqGenAgent.sinks.HDFS.hdfs.rollInterval = 0 SeqGenAgent.sinks.HDFS.hdfs.rollCount = 10000 SeqGenAgent.sinks.HDFS.hdfs.fileType = DataStream    # Describimos la configuración del canal  SeqGenAgent.channels.MemChannel.type = memory SeqGenAgent.channels.MemChannel.capacity = 1000 SeqGenAgent.channels.MemChannel.transactionCapacity = 100    # Unimos el origen y el destino a través del canal  SeqGenAgent.sources.SeqSource.channels = MemChannel SeqGenAgent.sinks.HDFS.channel = MemChannel |

Ejecutamos el siguiente comando:

|  |
| --- |
| ./bin/flume-ng agent --conf ./conf/ --conf-file ./conf/seq\_gen.conf --name SeqGenAgent |

Ahora vamos a crear otro ejemplo de generación de información. En el mismo directorio $FLUME\_HOME\conf, creamos un nuevo fichero con el nombre *netcat.conf y copiamos el siguiente código:*

|  |
| --- |
| #Nombramos a los componentes del agente  NetcatAgent.sources = Netcat  NetcatAgent.channels = MemChannel  NetcatAgent.sinks = HDFS  # Describimos el tipo de origen  NetcatAgent.sources.Netcat.type = netcat  NetcatAgent.sources.Netcat.bind = localhost  NetcatAgent.sources.Netcat.port = 44444  NetcatAgent.sources.Netcat.channels = MemChannel  # Describimos el destino  NetcatAgent.sinks.HDFS.type=hdfs  NetcatAgent.sinks.HDFS.hdfs.path=hdfs://hadoop-VirtualBox:9000/user/hadoop/net\_data/  NetcatAgent.sinks.HDFS.hdfs.writeFormat=Text  NetcatAgent.sinks.HDFS.hdfs.fileType=DataStream  NetcatAgent.sinks.HDFS.channel=MemChannel  # Unimos el origen y el destino a través del canal  NetcatAgent.channels.MemChannel.type = memory  NetcatAgent.channels.MemChannel.capacity = 1000  NetcatAgent.channels.MemChannel.transactionCapacity = 100 |

Lanzamos al agente:

|  |
| --- |
| ./bin/flume-ng agent --conf ./conf/ --conf-file ./conf/netcat.conf --name NetcatAgent -Dflume.root.logger=INFO,console |

En una nueva pestaña introducimos el siguiente comando y escribimos

|  |
| --- |
|  |

Nos vamos al navegador web de HDFS y comprobamos que se ha creado el fichero:

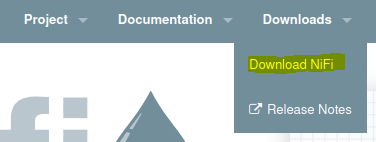
|  |
| --- |
| **[Ejercicio] Adjuntar captura de pantalla** |

## NIFI

### Instalación

Para instalar Nifi accedemos al siguiente enlace: [https://nifi.apache.org](https://nifi.apache.org/)

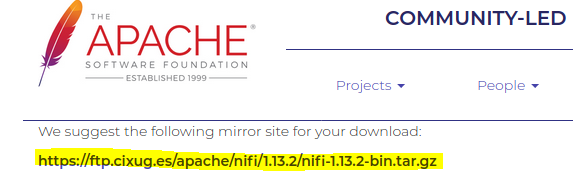
Una vez dentro seleccionamos la opción de descarga:



Y seleccionamos el binario de la versión 1.13.2



A continuación seleccionamos el mirror que nos sugiere para realizar la descarga:



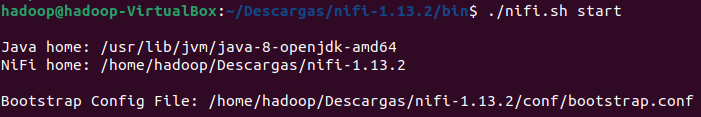
Una vez se ha realizado la descarga descomprimimos el archivo:

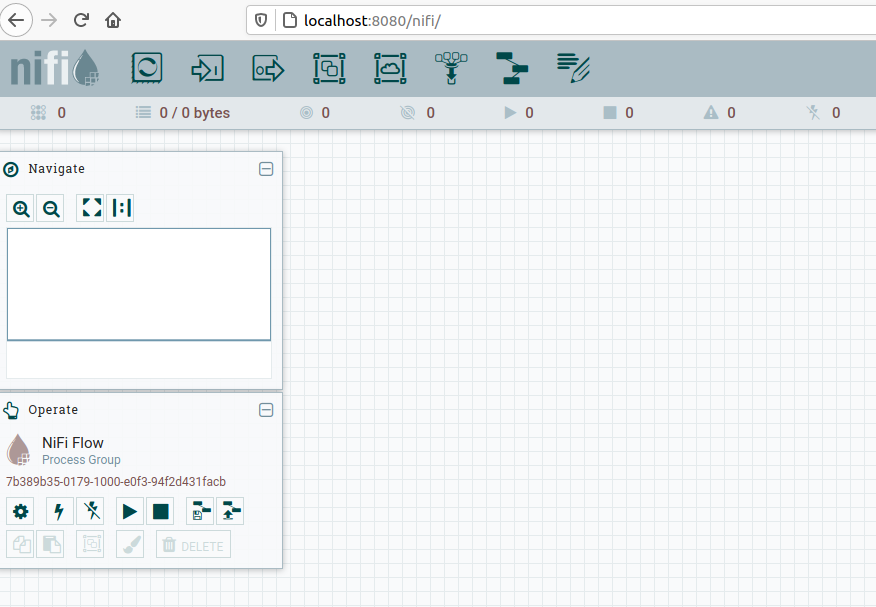
|  |
| --- |
| tar xvf nifi-1.13.2-bin.tar.gz |

Para ejecutar nifi, tendremos que ejecutar el script nifi.sh que se encuentra en el directorio bin:

|  |
| --- |
| ./nifi.sh start |

Si vemos el siguiente mensaje es que nifi ha arrancado correctamente:

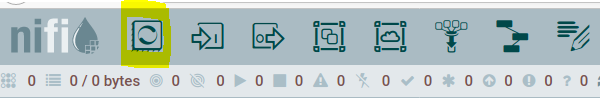


Para acceder al entorno de trabajo introducimos en el navegador: [http://localhost:8080/nifi](http://localhost:8080/nifi)

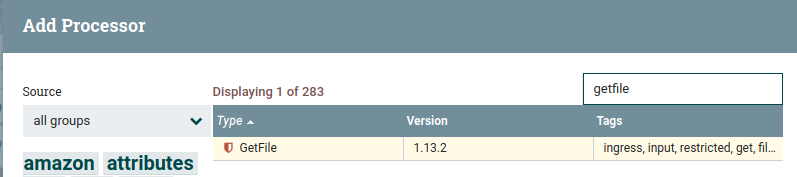
### Mover datos

Vamos a hacer un pequeño ejercicio con Nifi para familiarizarnos con el entorno desarrollando un flujo sencillo que mueva un fichero de un directorio a otro.

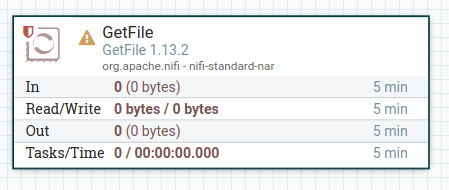
1. Seleccionamos un procesador y lo colocamos en nuestra área de trabajo



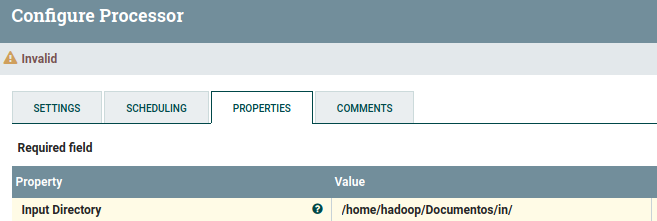
1. Indicamos el tipo de procesador getfile



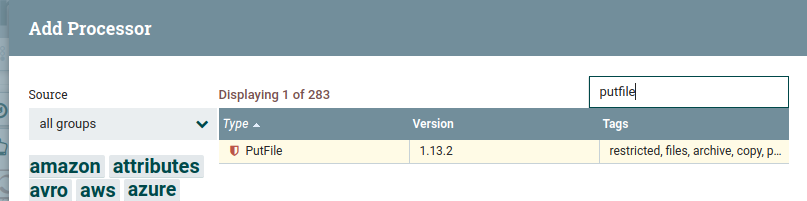
1. Damos doble **click** sobre el elemento gráfico que representa nuestro procesador.



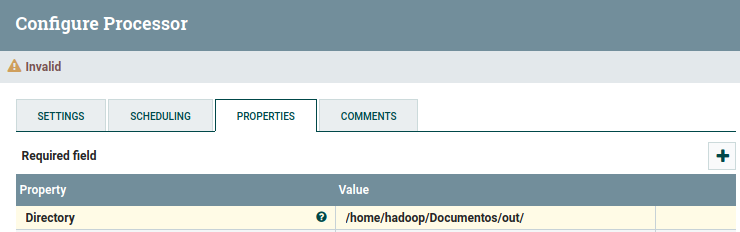
1. En **properties** indicamos el directorio de entrada de donde tendrá que recoger el fichero, en este caso:



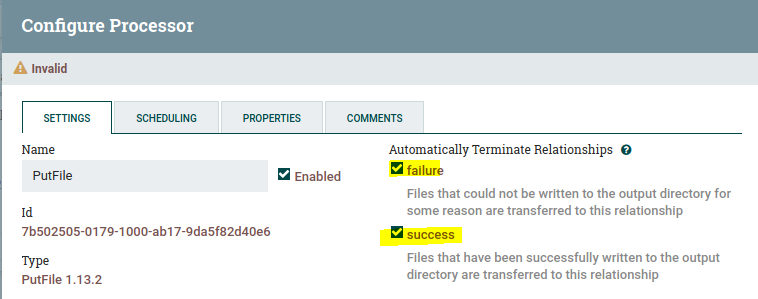
1. Añadimos un nuevo procesador de tipo PutFile



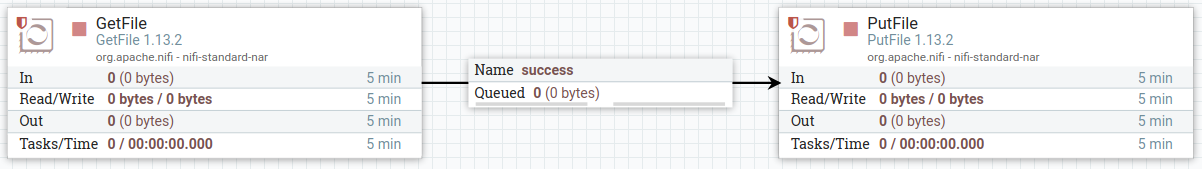
1. En **properties** indicamos el directorio de salida



1. Para este ejemplo sencillo “autoterminaremos” las 2 relaciones:



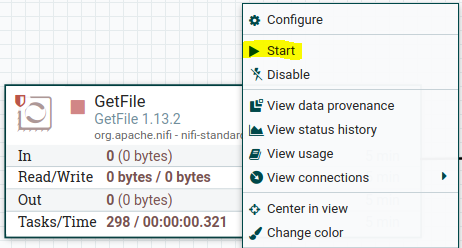
1. Ya solo nos queda unir los dos procesadores



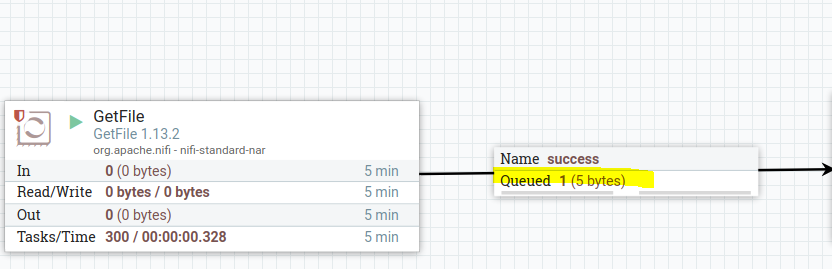
1. Antes de arrancarlo, creamos un pequeño fichero en el directorio que hemos puesto como entrada



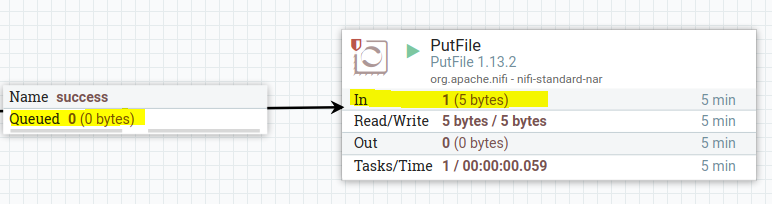
1. Con el segundo botón sobre el procesados le damos a **START**



1. Y vemos que tenemos fichero en la cola



1. Damos a START al proceso PUTFILE y vemos como el fichero pasa de la cola a este proceso marcando la entrada:



1. Comprobamos los directorios /in y /out para comprobar que ha movido los ficheros.