大数据技术之Flume

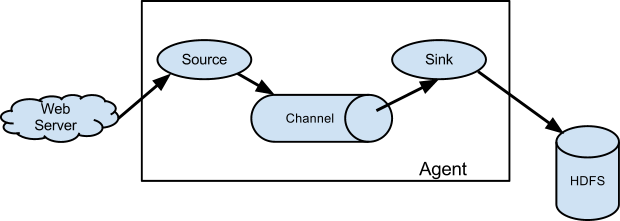
# **一、Flume简介**

1) Flume提供一个分布式的，可靠的，对大数据量的日志进行高效收集、聚集、移动的服务，Flume只能在Unix环境下运行。

2) Flume基于流式架构，容错性强，也很灵活简单。

3) Flume、Kafka用来实时进行数据收集，Spark、Flink用来实时处理数据，impala用来实时查询。

# **二、Flume角色**



## **2.1、Source**

用于采集数据，Source是产生数据流的地方，同时Source会将产生的数据流传输到Channel，这个有点类似于Java IO部分的Channel。

## **2.2、Channel**

用于桥接Sources和Sinks，类似于一个队列。

## **2.3、Sink**

从Channel收集数据，将数据写到目标源(可以是下一个Source，也可以是HDFS或者HBase)。

## **2.4、Event**

传输单元，Flume数据传输的基本单元，以事件的形式将数据从源头送至目的地。

# **三、Flume传输过程**

source监控某个文件或数据流，数据源产生新的数据，拿到该数据后，将数据封装在一个Event中，并put到channel后commit提交，channel队列先进先出，sink去channel队列中拉取数据，然后写入到HDFS中。

# **四、Flume部署及使用**

## **4.1、文件配置**

查询JAVA\_HOME: echo $JAVA\_HOME

|  |
| --- |
| 显示/opt/module/jdk1.8.0\_144 /opt/module/jdk1.8.0\_144 |

安装Flume

|  |
| --- |
| [itstar@bigdata113 software]$ tar -zxvf apache-flume1.8.0-bin.tar.gz -C /opt/module/ |

**改名：**

|  |
| --- |
| [itstar@bigdata113 conf]$ **mv flume-env.sh.template flume-env.sh** |

**flume-env.sh涉及修改项：**

|  |
| --- |
| export JAVA\_HOME=/opt/module/jdk1.8.0\_144 |

**Linux中文件上传命令是：rz，下载：sz**

**如果没有这两个命令使用yum下载：yum install lrzsz**

## **4.2、案例**

### **4.2.1、案例一：监控端口数据**

**目标：**Flume监控一端Console，另一端Console发送消息，使被监控端实时显示。

**分步实现：**

**1) 安装telnet工具**

|  |
| --- |
| 【联网状态】yum -y install telnet |
| 【安装完成】 |

**2) 创建Flume Agent配置文件flume-telnet.conf**

|  |
| --- |
| #定义Agent, 为三大组件起别名  a1.sources = r1 #a1为任意起的agent别名  a1.channels = c1  a1.sinks = k1  #定义netcatsource  a1.sources.r1.type = netcat #监控端口的数据源类型为netcat  a1.sources.r1.bind = bigdata111 #数据源组件侦听的IP地址  a1.sources.r1.port = 44445 #数据源侦听的端口  # 定义channel  a1.channels.c1.type = memory #定义通道存储数据的类型  a1.channels.c1.capacity = 1000 #定义通道的缓存容量,1000表示1000个事件  a1.channels.c1.transactionCapacity = 100 #定义了通道每次传输多少事件  # 定义sink  a1.sinks.k1.type = logger #数据落地的格式为日志文件  # 双向链接  a1.sources.r1.channels = c1 #source组件连接channel组件  a1.sinks.k1.channel = c1 #channel组件连接sink组件 |

**3) 判断44445 端口是否被占用**

|  |
| --- |
| $ netstat -tunlp | grep 44445 |

**4) 启动flume**

|  |
| --- |
| /opt/module/flume-1.8.0/bin/flume-ng agent \  --conf /opt/module/flume1.8.0/conf/ \ #指定flume的配置文件目录，最好是绝对路径  --name a1 \  --conf-file /opt/module/flume-1.8.0/jobconf/flume-telnet.conf \ #指定agent的配置文件  -Dflume.root.logger==INFO,console |
| bin/flume-ng agent --conf conf/ --name a1 --conf-file conf/flume-telnet.conf -Dflume.root.logger==INFO,console |

**5) 使用telnet工具向本机的44445端口发送内容**

|  |
| --- |
| $ telnet bigdata111 44445 |

### **4.2.2、案例二：实时读取本地文件到HDFS**

**1) 创建flume-hdfs.conf文件, 一个flume进程可以有多个配置文件来处理不同的agent**

|  |
| --- |
| # 1 agent  a2.sources = r2  a2.channels = c2  a2.sinks = k2  # 2 source  a2.sources.r2.type = exec # 数据源组件的类型是执行linux命令采集数据  a2.sources.r2.command = tail -f /opt/plus # 具体的Linux执行指令  a2.sources.r2.shell = /bin/bash -c  # 3 sink  a2.sinks.k2.type = hdfs # sink组件的类型  a2.sinks.k2.hdfs.path = hdfs://bigdata111:9000/flume/%Y%m%d/%H #hdfs文件路径  #上传文件的前缀，如logs-.12312312341  a2.sinks.k2.hdfs.filePrefix = logs-  #是否按照时间滚动文件夹  a2.sinks.k2.hdfs.round = true  #多少时间单位创建一个新的文件夹  a2.sinks.k2.hdfs.roundValue = 1  #重新定义时间单位,每小时创建一个数据文件夹  a2.sinks.k2.hdfs.roundUnit = hour  #是否使用本地时间戳  a2.sinks.k2.hdfs.useLocalTimeStamp = true  #积攒多少个Event才flush到HDFS一次  a2.sinks.k2.hdfs.batchSize = 1000  #设置文件类型，可支持压缩  a2.sinks.k2.hdfs.fileType = DataStream  #多久生成一个新的文件，单位秒  a2.sinks.k2.hdfs.rollInterval = 600  #设置每个文件的滚动大小，单位字节，可以设为略小于128M  a2.sinks.k2.hdfs.rollSize = 134217700  #文件的滚动与Event数量无关  a2.sinks.k2.hdfs.rollCount = 0  #最小副本数  a2.sinks.k2.hdfs.minBlockReplicas = 1  # Use a channel which buffers events in memory  a2.channels.c2.type = memory  a2.channels.c2.capacity = 1000  a2.channels.c2.transactionCapacity = 100  # Bind the source and sink to the channel  a2.sources.r2.channels = c2  a2.sinks.k2.channel = c2 |

**3) 执行监控配置**

|  |
| --- |
| /opt/module/flume1.8.0/bin/flume-ng agent \  --conf /opt/module/flume1.8.0/conf/ \  --name a2 \ #agent别名  --conf-file /opt/module/flume1.8.0/jobconf/flume-hdfs.conf |

### **4.2.3、案例三：实时读取目录文件到HDFS**

**目标：**使用flume监听整个目录的文件

**分步实现**：

1) 创建配置文件flume-dir.conf

|  |
| --- |
| #1 Agent  a3.sources = r3  a3.sinks = k3  a3.channels = c3  #2 source  #监控目录的类型  a3.sources.r3.type = spooldir  #监控目录的路径  a3.sources.r3.spoolDir = /opt/module/flume1.8.0/upload  #哪个文件上传hdfs，然后给这个文件添加一个后缀  a3.sources.r3.fileSuffix = .COMPLETED  a3.sources.r3.fileHeader = true  #忽略所有以.tmp结尾的文件，不上传（可选）  a3.sources.r3.ignorePattern = ([^ ]\*\.tmp)  # 3 sink  a3.sinks.k3.type = hdfs  a3.sinks.k3.hdfs.path = hdfs://bigdata111:9000/flume/%H  #上传文件的前缀  a3.sinks.k3.hdfs.filePrefix = upload-  #是否按照时间滚动文件夹  a3.sinks.k3.hdfs.round = true  #多少时间单位创建一个新的文件夹  a3.sinks.k3.hdfs.roundValue = 1  #重新定义时间单位  a3.sinks.k3.hdfs.roundUnit = hour  #是否使用本地时间戳  a3.sinks.k3.hdfs.useLocalTimeStamp = true  #积攒多少个Event才flush到HDFS一次  a3.sinks.k3.hdfs.batchSize = 100  #设置文件类型，可支持压缩  a3.sinks.k3.hdfs.fileType = DataStream  #多久生成一个新的文件  a3.sinks.k3.hdfs.rollInterval = 600  #设置每个文件的滚动大小大概是128M  a3.sinks.k3.hdfs.rollSize = 134217700  #文件的滚动与Event数量无关  a3.sinks.k3.hdfs.rollCount = 0  #最小副本数  a3.sinks.k3.hdfs.minBlockReplicas = 1  # Use a channel which buffers events in memory  a3.channels.c3.type = memory  a3.channels.c3.capacity = 1000  a3.channels.c3.transactionCapacity = 100  # Bind the source and sink to the channel  a3.sources.r3.channels = c3  a3.sinks.k3.channel = c3 |

2) 执行测试：执行如下脚本后，请向upload文件夹中添加文件试试

|  |
| --- |
| /opt/module/flume1.8.0/bin/flume-ng agent \  --conf /opt/module/flume1.8.0/conf/ \  --name a3 \  --conf-file /opt/module/flume1.8.0/jobconf/flume-dir.conf |

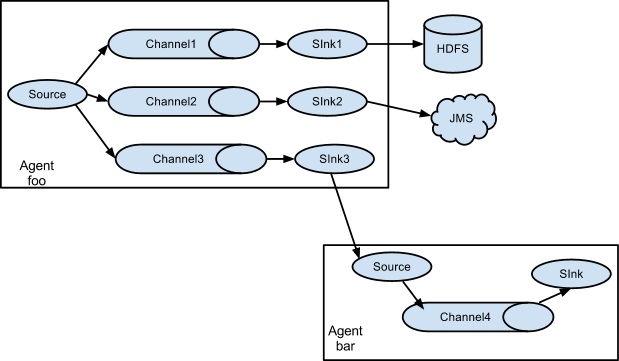
尖叫提示： 在使用Spooling Directory Source时

1) 不要在监控目录中创建并持续修改文件

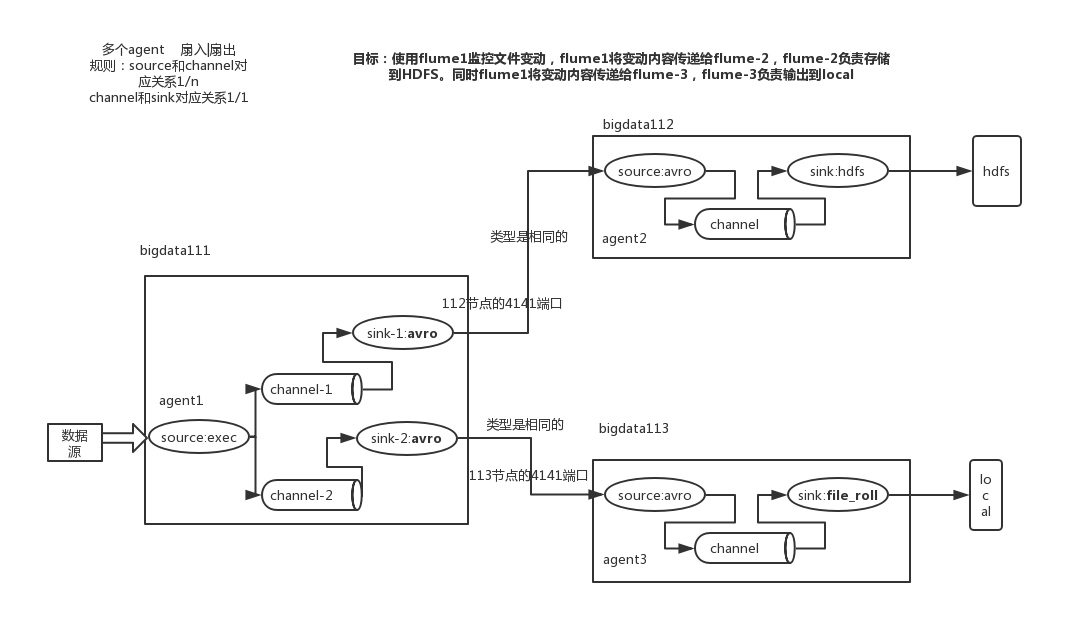
2) 上传完成的文件会以.COMPLETED结尾

3) 被监控文件夹每500毫秒扫描一次文件变动

### **4.2.4、案例四：Flume与Flume之间数据传递：单Flume多Channel、Sink**



**目标：**使用flume1监控文件变动，flume1将变动内容传递给flume-2，flume-2负责存储到HDFS。同时flume1将变动内容传递给flume-3，flume-3负责输出到local，如下图所示：

****

**分步实现：**

1) 创建flume1.conf，用于监控某文件的变动，同时产生两个channel和两个sink分别输送给flume2和flume3：

|  |
| --- |
| # 1.agent source->channel对应关系1/n sink->channel对应关系1/1  a1.sources = r1  a1.channels =c1 c2  a1.sinks = k1 k2  # 将数据流复制给多个channel  a1.sources.r1.selector.type = replicating  # 2.source  a1.sources.r1.type = exec  a1.sources.r1.command = tail -f /opt/plus  a1.sources.r1.shell = /bin/bash -c # 确定命令行可用  # 3.sink1  a1.sinks.k1.type = avro # 传递给下一个agent的数据格式类型是avro  a1.sinks.k1.hostname = bigdata112 #下一个agent的主机名  a1.sinks.k1.port = 4141 #下一个agent的端口  # sink2  a1.sinks.k2.type = avro  a1.sinks.k2.hostname = bigdata113  a1.sinks.k2.port = 4141  # 4.channel—1  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  # 4.channel—2  a1.channels.c2.type = memory  a1.channels.c2.capacity = 1000  a1.channels.c2.transactionCapacity = 100  # Bind the source and sink to the channel  a1.sources.r1.channels = c1 c2  a1.sinks.k1.channel = c1  a1.sinks.k2.channel = c2 |
|  |

2) 创建flume-2.conf，用于接收flume1的event，同时产生1个channel和1个sink，将数据输送给hdfs：

|  |
| --- |
| # 1 agent  a2.sources = r1  a2.sinks = k1  a2.channels = c1  # 2 source  a2.sources.r1.type = avro  a2.sources.r1.bind = bigdata112 // agent2所在的IP  a2.sources.r1.port = 4141 # agent2侦听数据的端口  # 3 sink  a2.sinks.k1.type = hdfs  a2.sinks.k1.hdfs.path = hdfs://bigdata111:9000/flume2/%H  #上传文件的前缀  a2.sinks.k1.hdfs.filePrefix = flume2-  #是否按照时间滚动文件夹  a2.sinks.k1.hdfs.round = true  #多少时间单位创建一个新的文件夹  a2.sinks.k1.hdfs.roundValue = 1  #重新定义时间单位  a2.sinks.k1.hdfs.roundUnit = hour  #是否使用本地时间戳  a2.sinks.k1.hdfs.useLocalTimeStamp = true  #积攒多少个Event才flush到HDFS一次  a2.sinks.k1.hdfs.batchSize = 100  #设置文件类型，可支持压缩  a2.sinks.k1.hdfs.fileType = DataStream  #多久生成一个新的文件  a2.sinks.k1.hdfs.rollInterval = 600  #设置每个文件的滚动大小大概是128M  a2.sinks.k1.hdfs.rollSize = 134217700  #文件的滚动与Event数量无关  a2.sinks.k1.hdfs.rollCount = 0  #最小副本数  a2.sinks.k1.hdfs.minBlockReplicas = 1  # 4 channel  a2.channels.c1.type = memory  a2.channels.c1.capacity = 1000  a2.channels.c1.transactionCapacity = 100  #5 Bind  a2.sources.r1.channels = c1  a2.sinks.k1.channel = c1 |

3) 创建flume-3.conf，用于接收flume1的event，同时产生1个channel和1个sink，将数据输送给本地目录：

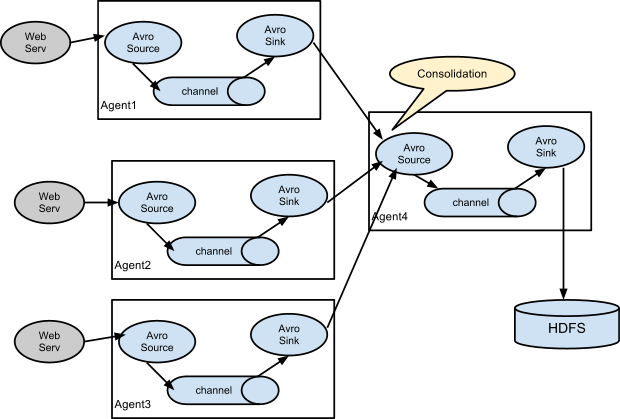
|  |
| --- |
| #1 agent  a3.sources = r1  a3.sinks = k1  a3.channels = c1  # 2 source  a3.sources.r1.type = avro  a3.sources.r1.bind = bigdata113  a3.sources.r1.port = 4141  #3 sink  a3.sinks.k1.type = file\_roll # 本地文件类型  #备注：此处的文件夹需要先创建好  a3.sinks.k1.sink.directory = /opt/flume3  # 4 channel  a3.channels.c1.type = memory  a3.channels.c1.capacity = 1000  a3.channels.c1.transactionCapacity = 100  # 5 Bind  a3.sources.r1.channels = c1  a3.sinks.k1.channel = c1 |

尖叫提示：输出的本地目录必须是已经存在的目录，如果该目录不存在，并不会创建新的目录。

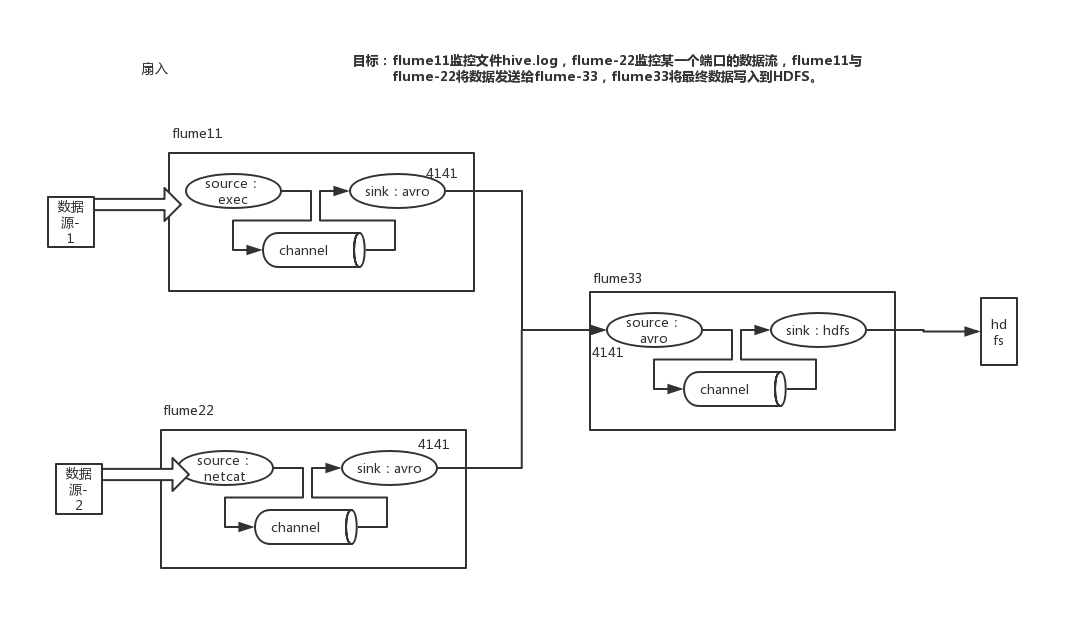
4) 执行测试：分别开启对应flume-job（依次启动flume1，flume-2，flume-3），同时产生文件变动并观察结果：

|  |
| --- |
| $ bin/flume-ng agent --conf conf/ --name a1 --conf-file jobconf/flume1.conf  $ bin/flume-ng agent --conf conf/ --name a2 --conf-file jobconf/flume2.conf  $ bin/flume-ng agent --conf conf/ --name a3 --conf-file jobconf/flume3.conf |

### **4.2.5、案例五：Flume与Flume之间数据传递，多Flume汇总数据到单Flume**



**目标：**flume11监控文件hive.log，flume-22监控某一个端口的数据流，flume11与flume-22将数据发送给flume-33，flume33将最终数据写入到HDFS。如下图所示：



**分步实现：**

1) 创建flume11.conf，用于监控hive.log文件，同时sink数据到flume-33：

|  |
| --- |
| # 1 agent  a1.sources = r1  a1.sinks = k1  a1.channels = c1  # 2 source  a1.sources.r1.type = exec  a1.sources.r1.command = tail -F /opt/plus  a1.sources.r1.shell = /bin/bash -c  # 3 sink  a1.sinks.k1.type = avro  a1.sinks.k1.hostname = bigdata113  a1.sinks.k1.port = 4141  # 4 channel  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  # 5. Bind  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1 |

2) 创建flume-22.conf，用于监控端口44444数据流，同时sink数据到flume-33：

|  |
| --- |
| # 1 agent  a2.sources = r1  a2.sinks = k1  a2.channels = c1  # 2 source  a2.sources.r1.type = netcat  a2.sources.r1.bind = bigdata112  a2.sources.r1.port = 44444  #3 sink  a2.sinks.k1.type = avro  a2.sinks.k1.hostname = bigdata113  a2.sinks.k1.port = 4141  # 4 channel  a2.channels.c1.type = memory  a2.channels.c1.capacity = 1000  a2.channels.c1.transactionCapacity = 100  # 5 Bind  a2.sources.r1.channels = c1  a2.sinks.k1.channel = c1 |

3) 创建flume33.conf，用于接收flume11与flume22发送过来的数据流，最终合并后sink到HDFS：

|  |
| --- |
| # 1 agent  a3.sources = r1  a3.sinks = k1  a3.channels = c1  # 2 source  a3.sources.r1.type = avro  a3.sources.r1.bind = bigdata113  a3.sources.r1.port = 4141  # 3 sink  a3.sinks.k1.type = hdfs  a3.sinks.k1.hdfs.path = hdfs://bigdata111:9000/flume3/%H  #上传文件的前缀  a3.sinks.k1.hdfs.filePrefix = flume3-  #是否按照时间滚动文件夹  a3.sinks.k1.hdfs.round = true  #多少时间单位创建一个新的文件夹  a3.sinks.k1.hdfs.roundValue = 1  #重新定义时间单位  a3.sinks.k1.hdfs.roundUnit = hour  #是否使用本地时间戳  a3.sinks.k1.hdfs.useLocalTimeStamp = true  #积攒多少个Event才flush到HDFS一次  a3.sinks.k1.hdfs.batchSize = 100  #设置文件类型，可支持压缩  a3.sinks.k1.hdfs.fileType = DataStream  #多久生成一个新的文件  a3.sinks.k1.hdfs.rollInterval = 600  #设置每个文件的滚动大小大概是128M  a3.sinks.k1.hdfs.rollSize = 134217700  #文件的滚动与Event数量无关  a3.sinks.k1.hdfs.rollCount = 0  #最小冗余数  a3.sinks.k1.hdfs.minBlockReplicas = 1  # 4 channel  a3.channels.c1.type = memory  a3.channels.c1.capacity = 1000  a3.channels.c1.transactionCapacity = 100  # 5 Bind  a3.sources.r1.channels = c1  a3.sinks.k1.channel = c1 |

4) 执行测试：分别开启对应flume-job（依次启动flume-33，flume-22，flume11），同时产生文件变动并观察结果：

|  |
| --- |
| $ bin/flume-ng agent --conf conf/ --name a3 --conf-file jobconf/flume33.conf  $ bin/flume-ng agent --conf conf/ --name a2 --conf-file jobconf/flume22.conf  $ bin/flume-ng agent --conf conf/ --name a1 --conf-file jobconf/flume11.conf |

数据发送

|  |
| --- |
| 1. telnet bigdata111 44444 打开后发送5555555 2. 在/opt/plus 中追加666666 |

### 监控落地数据

**hdfs dfs -cat /flume3/06/\***

### **4.2.6、案例六：Flume拦截器**

#### **时间戳拦截器**

Timestamp.conf

|  |
| --- |
| #1.定义agent名， source、channel、sink的名称  a4.sources = r1  a4.channels = c1  a4.sinks = k1  #2.具体定义source  a4.sources.r1.type = spooldir  a4.sources.r1.spoolDir = /opt/module/flume-1.8.0/upload  #定义拦截器，为文件最后添加时间戳  a4.sources.r1.interceptors = timestamp  a4.sources.r1.interceptors.timestamp.type= org.apache.flume.interceptor.TimestampInterceptor  $Builder  #具体定义channel  a4.channels.c1.type = memory  a4.channels.c1.capacity = 10000  a4.channels.c1.transactionCapacity = 100  #具体定义sink  a4.sinks.k1.type = hdfs  a4.sinks.k1.hdfs.path = hdfs://bigdata111:9000/flume-interceptors/%H  a4.sinks.k1.hdfs.filePrefix = events-  a4.sinks.k1.hdfs.fileType = DataStream  #不按照条数生成文件  a4.sinks.k1.hdfs.rollCount = 0  #HDFS上的文件达到128M时生成一个文件  a4.sinks.k1.hdfs.rollSize = 134217728  #HDFS上的文件达到60秒生成一个文件  a4.sinks.k1.hdfs.rollInterval = 60  #组装source、channel、sink  a4.sources.r1.channels = c1  a4.sinks.k1.channel = c1 |

启动命令

|  |
| --- |
| /opt/module/flume-1.8.0/bin/flume-ng agent -n a4 \  -f /opt/module/flume-1.8.0/jobconf/flume-interceptors.conf \  -c /opt/module/flume-1.8.0/conf \  -Dflume.root.logger=INFO,console |

#### **主机名拦截器**

Host.conf

|  |
| --- |
| #1.定义agent  a1.sources= r1  a1.sinks = k1  a1.channels = c1  #2.定义source  a1.sources.r1.type = exec  a1.sources.r1.channels = c1  a1.sources.r1.command = tail -F /opt/plus  #拦截器  a1.sources.r1.interceptors = i1  a1.sources.r1.interceptors.i1.type = host  #参数为true时用IP192.168.1.111，参数为false时用主机名，默认为true  a1.sources.r1.interceptors.i1.useIP = false  a1.sources.r1.interceptors.i1.hostHeader = agentHost # 主机名称  #3.定义sinks  a1.sinks.k1.type=hdfs  a1.sinks.k1.channel = c1  a1.sinks.k1.hdfs.path = hdfs://bigdata111:9000/flumehost/%{agentHost}  a1.sinks.k1.hdfs.filePrefix = plus\_%{agentHost} # hdfs文件前缀  #往生成的文件加后缀名.log  a1.sinks.k1.hdfs.fileSuffix = .log # hdfs文件后缀  a1.sinks.k1.hdfs.fileType = DataStream  a1.sinks.k1.hdfs.writeFormat = Text  a1.sinks.k1.hdfs.rollInterval = 10  a1.sinks.k1.hdfs.useLocalTimeStamp = true    a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c10.transactionCapacity = 100    a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1 |
|  |

启动命令：

|  |
| --- |
| bin/flume-ng agent -c conf/ -f jobconf/host.conf -n a1 -Dflume.root.logger=INFO,console |

#### **UUID拦截器**

uuid.conf

|  |
| --- |
| a1.sources = r1  a1.sinks = k1  a1.channels = c1  a1.sources.r1.type = exec  a1.sources.r1.channels = c1  a1.sources.r1.command = tail -F /opt/plus  a1.sources.r1.interceptors = i1  #type的参数不能写成uuid，得写具体，否则找不到类  a1.sources.r1.interceptors.i1.type = org.apache.flume.sink.solr.morphline.UUIDInterceptor$Builder  #如果UUID头已经存在,它应该保存  a1.sources.r1.interceptors.i1.preserveExisting = true  a1.sources.r1.interceptors.i1.prefix = UUID\_  #如果sink类型改为HDFS，那么在HDFS的文本中没有headers的信息数据  a1.sinks.k1.type = logger  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1  # bin/flume-ng agent -c conf/ -f jobconf/uuid.conf -n a1 -Dflume.root.logger==INFO,console |

#### **查询替换拦截器**

search.conf

|  |
| --- |
| #1 agent  a1.sources = r1  a1.sinks = k1  a1.channels = c1  #2 source  a1.sources.r1.type = exec  a1.sources.r1.channels = c1  a1.sources.r1.command = tail -F /opt/plus  a1.sources.r1.interceptors = i1  a1.sources.r1.interceptors.i1.type = search\_replace  #遇到数字改成itstar，A123会替换为Aitstar  a1.sources.r1.interceptors.i1.searchPattern = [0-9]+  a1.sources.r1.interceptors.i1.replaceString = \*\*\*  a1.sources.r1.interceptors.i1.charset = UTF-8  #3 sink  a1.sinks.k1.type = logger  #4 Chanel  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  #5 bind  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1  # bin/flume-ng agent -c conf/ -f jobconf/search.conf -n a1 -Dflume.root.logger=INFO,console |

#### **正则过滤拦截器**

filter.conf

|  |
| --- |
| #1 agent  a1.sources = r1  a1.sinks = k1  a1.channels = c1  #2 source  a1.sources.r1.type = exec  a1.sources.r1.channels = c1  a1.sources.r1.command = tail -F /opt/plus  a1.sources.r1.interceptors = i1  a1.sources.r1.interceptors.i1.type = regex\_filter  a1.sources.r1.interceptors.i1.regex = ^A.\*  #如果excludeEvents设为false,表示过滤掉不是以A开头的events。如果excludeEvents设为true，则表示过滤掉以A开头的events。  a1.sources.r1.interceptors.i1.excludeEvents = true  a1.sinks.k1.type = logger  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1  # bin/flume-ng agent -c conf/ -f jobconf/filter.conf -n a1 -Dflume.root.logger=INFO,console |

#### **正则抽取拦截器**

extractor.conf

|  |
| --- |
| #1 agent  a1.sources = r1  a1.sinks = k1  a1.channels = c1  #2 source  a1.sources.r1.type = exec  a1.sources.r1.channels = c1  a1.sources.r1.command = tail -F /opt/plus  a1.sources.r1.interceptors = i1  a1.sources.r1.interceptors.i1.type = regex\_extractor  # hostname is bigdata111 ip is 192.168.20.111  a1.sources.r1.interceptors.i1.regex = hostname is (.\*?) ip is (.\*)  a1.sources.r1.interceptors.i1.serializers = s1 s2  #hostname（自定义）= (.\*?)->bigdata111  a1.sources.r1.interceptors.i1.serializers.s1.name = hostname  #ip（自定义） = (.\*)->192.168.20.111  a1.sources.r1.interceptors.i1.serializers.s2.name = ip  a1.sinks.k1.type = logger  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1  # bin/flume-ng agent -c conf/ -f jobconf/extractor.conf -n a1 -Dflume.root.logger=INFO,console |

注：正则抽取拦截器的headers不会出现在文件名和文件内容中

### **4.2.7、案例七：Flume自定义拦截器**

字母小写变大写

#### **1.Pom.xml**

|  |
| --- |
| <dependencies>  <!-- flume核心依赖 -->  <dependency>  <groupId>org.apache.flume</groupId>  <artifactId>flume-ng-core</artifactId>  <version>1.8.0</version>  </dependency>  </dependencies>  <build>  <plugins>  <!-- 打包插件 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-jar-plugin</artifactId>  <version>2.4</version>  <configuration>  <archive>  <manifest>  <addClasspath>true</addClasspath>  <classpathPrefix>lib/</classpathPrefix>  <mainClass></mainClass>  </manifest>  </archive>  </configuration>  </plugin>  <!-- 编译插件 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <configuration>  <source>1.8</source>  <target>1.8</target>  <encoding>utf-8</encoding>  </configuration>  </plugin>  </plugins>  </build> |

#### **自定义实现拦截器**

注：拦截方法建议只做简单的数据转换，过滤等处理，不建议执行复杂的转换逻辑。

|  |
| --- |
| import org.apache.flume.Context;  import org.apache.flume.Event;  import org.apache.flume.interceptor.Interceptor;    import java.util.ArrayList;  import java.util.List;    public class MyInterceptor implements Interceptor {  @Override  public void initialize() {  }    @Override  public void close() {  }    /\*\*  \* 拦截source发送到通道channel中的消息  \*  \* @param event 接收过滤的event  \* @return event 根据业务处理后的event  \*/  @Override  public Event intercept(Event event) {  // 获取事件对象中的字节数据  byte[] arr = event.getBody();  // 将获取的数据转换成大写  event.setBody(new String(arr).toUpperCase().getBytes());  // 返回到消息中  return event;  }  // 接收被过滤事件集合  @Override  public List<Event> intercept(List<Event> events) {  List<Event> list = new ArrayList<>();  for (Event event : events) {  list.add(intercept(event));  }  return list;  }    public static class Builder implements Interceptor.Builder {  // 获取配置文件的属性  @Override  public Interceptor build() {  return new MyInterceptor();  }    @Override  public void configure(Context context) {    }  } |

使用Maven做成Jar包，在flume的目录下mkdir jar，上传此jar到jar目录中

#### **Flume配置文件**

ToUpCase.conf

|  |
| --- |
| #1.agent  a1.sources = r1  a1.sinks =k1  a1.channels = c1      # Describe/configure the source  a1.sources.r1.type = exec  a1.sources.r1.command = tail -F /opt/plus  a1.sources.r1.interceptors = i1  #全类名$Builder  a1.sources.r1.interceptors.i1.type = ToUpCase.MyInterceptor$Builder    # Describe the sink  a1.sinks.k1.type = hdfs  a1.sinks.k1.hdfs.path = /ToUpCase1  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.rollInterval = 3  a1.sinks.k1.hdfs.rollSize = 20  a1.sinks.k1.hdfs.rollCount = 5  a1.sinks.k1.hdfs.batchSize = 1  a1.sinks.k1.hdfs.useLocalTimeStamp = true  #生成的文件类型，默认是 Sequencefile，可用 DataStream，则为普通文本  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 |

运行命令：

|  |
| --- |
| bin/flume-ng agent -c conf/ -n a1 -f jar/ToUpCase.conf -C jar/Flume-1.0-SNAPSHOT.jar -Dflume.root.logger=DEBUG,console |

### **4.2.8、案例七：Flume对接kafka**

**配置flume(flume-kafka.conf)**

|  |
| --- |
| # define  a1.sources = r1  a1.sinks = k1  a1.channels = c1  # source  a1.sources.r1.type = exec  a1.sources.r1.command = tail -f -c +0 /opt/jars/calllog.csv  a1.sources.r1.shell = /bin/bash -c  # sink  a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink  a1.sinks.k1.brokerList = bigdata111:9092,bigdata112:9092,bigdata113:9092  a1.sinks.k1.topic = calllog  a1.sinks.k1.batchSize = 20  a1.sinks.k1.requiredAcks = 1  # channel  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  # bind  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1 |

**进入flume根目录下，启动flume**

|  |
| --- |
| /opt/module/flume-1.8.0/bin/flume-ng agent --conf /opt/module/flume-1.8.0/conf/ --name a1 --conf-file /opt/jars/flume2kafka.conf |

### **4.2.9、案例八：kafka对接Flume**

kafka2flume.conf

|  |
| --- |
| agent.sources = kafkaSource  agent.channels = memoryChannel  agent.sinks = hdfsSink  # The channel can be defined as follows.  agent.sources.kafkaSource.channels = memoryChannel  agent.sources.kafkaSource.type=org.apache.flume.source.kafka.KafkaSource  agent.sources.kafkaSource.zookeeperConnect=bigdata111:2181,bigdata112:2181,bigdata113:2181  agent.sources.kafkaSource.topic=calllog  #agent.sources.kafkaSource.groupId=flume  agent.sources.kafkaSource.kafka.consumer.timeout.ms=100  agent.channels.memoryChannel.type=memory  agent.channels.memoryChannel.capacity=10000  agent.channels.memoryChannel.transactionCapacity=1000  agent.channels.memoryChannel.type=memory  agent.channels.memoryChannel.capacity=10000  agent.channels.memoryChannel.transactionCapacity=1000  # the sink of hdfs  agent.sinks.hdfsSink.type=hdfs  agent.sinks.hdfsSink.channel = memoryChannel  agent.sinks.hdfsSink.hdfs.path=hdfs://bigdata111:9000/kafka2flume  agent.sinks.hdfsSink.hdfs.writeFormat=Text  agent.sinks.hdfsSink.hdfs.fileType=DataStream  #这两个不配置，会产生大量的小文件  agent.sinks.hdfsSink.hdfs.rollSize=0  agent.sinks.hdfsSink.hdfs.rollCount=0 |

**启动命令**

|  |
| --- |
| bin/flume-ng agent --conf conf --conf-file jobconf/kafka2flume.conf --name agent -Dflume.root.logger=INFO,console |

**注意：这个配置是从kafka过数据，但是需要重新向kafka的topic灌数据，他才会传到HDFS**