# flume介绍

## 概述

Flume最早是Cloudera提供的日志收集系统，后贡献给Apache。所以目前是Apache下的项目，Flume支持在日志系统中定制各类数据发送方，用于收集数据。

Flume是一个高可用的，高可靠的 鲁棒性（robust 健壮性），**分布式**的海量日志采集、聚合和传输的系统，Flume支持在日志系统中定制各类数据发送方，用于收集数据(source);同时，Flume提供对数据进行简单处理，并写到各种数据接受方(可定制)的能力(sink)。

当前Flume有两个版本Flume 0.9X版本的统称Flume-og，Flume1.X版本的统称Flume-ng。由于Flume-ng经过重大重构，与Flume-og有很大不同，使用时请注意区分。

## 系统需求

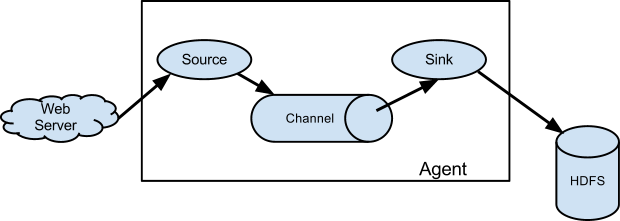
jdk1.6以上。推荐1.7

## 下载地址

flume.apache.org

# flume基本概念

## flume总体架构



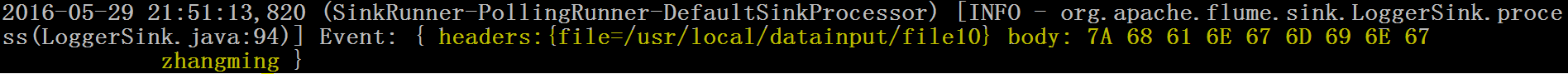
flume是分布式的日志收集系统，它将各个服务器中的数据收集起来并送到指定的地方去，比如说送到图中的HDFS，简单来说flume就是收集日志的。

## event 事件

event的相关概念：flume的核心是把数据从数据源(source)收集过来，在将收集到的数据送到指定的目的地(sink)。为了保证输送的过程一定成功，在送到目的地(sink)之前，会先缓存数据(channel),待数据真正到达目的地(sink)后，flume在删除自己缓存的数据。

在整个数据的传输的过程中，流动的是event，即事务保证是在event级别进行的。那么什么是event呢？—–event将传输的数据进行封装，是flume传输数据的基本单位，如果是文本文件，通常是一行记录，event也是事务的基本单位。event从source，流向channel，再到sink，本身为一个字节数组，并可携带headers(头信息)信息。event代表着一个数据的最小完整单元，从外部数据源来，向外部的目的地去。

一个完整的event包括：event headers、event body、event信息(即文本文件中的单行记录)，如下所以：



其中event信息就是flume收集到的日记记录。

## flume的运行机制

flume运行的核心就是agent，agent本身是一个Java进程，

agent里面包含3个核心的组件：source—->channel—–>sink,类似生产者、仓库、消费者的架构。

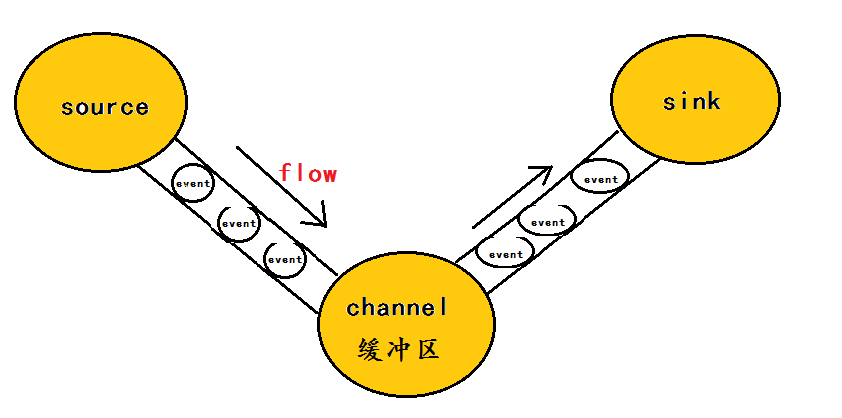
source：source组件是专门用来收集数据的，可以处理各种类型、各种格式的日志数据,包括avro、thrift、exec、jms、spooling directory、netcat、sequence generator、syslog、http、legacy、自定义。

channel：source组件把数据收集来以后，临时存放在channel中，即channel组件在agent中是专门用来存放临时数据的——对采集到的数据进行简单的缓存，可以存放在memory、jdbc、file等等。

sink：sink组件是用于把数据发送到目的地的组件，目的地包括hdfs、logger、avro、thrift、ipc、file、null、hbase、solr、自定义。

一个完整的工作流程：source不断的接收数据，将数据封装成一个一个的event，然后将event发送给channel，chanel作为一个缓冲区会临时存放这些event数据，随后sink会将channel中的event数据发送到指定的地方—-例如HDFS等。

注：只有在sink将channel中的数据成功发送出去之后，channel才会将临时event数据进行删除，这种机制保证了数据传输的可靠性与安全性。

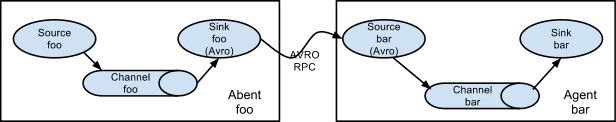


## flume的用法

flume之所以这么神奇—-其原因也在于flume可以支持多级flume的agent，即flume可以前后相继形成多级的复杂流动，例如sink可以将数据写到下一个agent的source中，这样的话就可以连成串了，可以整体处理了。

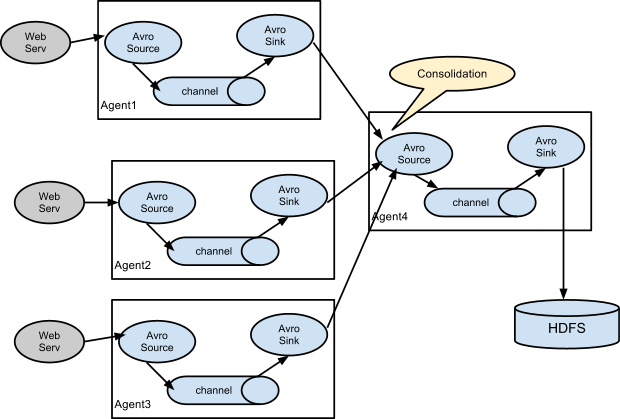
此外，flume还支持扇入(fan-in)、扇出(fan-out)。所谓扇入就是source可以接受多个输入，所谓扇出就是sink可以将数据输出多个目的地中。

**置多个agent的数据流（多级流动）**



**数据流合并（扇入流）**

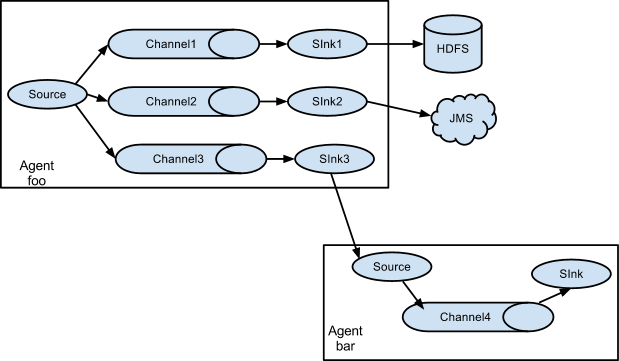
在做日志收集的时候一个常见的场景就是，大量的生产日志的客户端发送数据到少量的附属于存储子系统的消费者agent。例如，从数百个web服务器中收集日志，它们发送数据到十几个负责将数据写入HDFS集群的agent。



这个可在Flume中可以实现，需要配置大量第一层的agent，每一个agent都有一个avro sink，让它们都指向同一个agent的avro source（强调一下，在这样一个场景下你也可以使用thrift source/sink/client）。在第二层agent上的source将收到的event合并到一个channel中，event被一个sink消费到它的最终的目的地。

**数据流复用（扇出流）**

Flume支持多路输出event流到一个或多个目的地。这是靠定义一个多路数据流实现的，它可以实现复制和选择性路由一个event到一个或者多个channel。



上面的例子展示了agent foo中source扇出数据流到三个不同的channel，这个扇出可以是复制或者多路输出。在复制数据流的情况下，每一个event被发送所有的三个channel；在多路输出的情况下，一个event被发送到一部分可用的channel中，它们是根据event的属性和预先配置的值选择channel的。 这些映射关系应该被填写在agent的配置文件中。

## Flume的特性

### 可靠性

事务型的数据传递，保证数据的可靠性

一个日志交给flume来处理，不会出现此日志丢失或未被处理的情况。

### 可恢复性

通道可以以内存或文件的方式实现，内存更快，但不可恢复。文件较慢但提供了可恢复性。

# flume安装和配置

## 实现步骤：

1.安装jdk，1.6版本以上

2.上传flume的安装包

3.解压安装

4.在conf目录下，创建一个配置文件，比如：template.conf（名字可以不固定,后缀也可以不固定）

5.配置agent组件

**相关配置：**

#配置Agent a1 的组件

a1.sources=r1

a1.channels=c1 (可以配置多个，以空格隔开，名字自己定）

a1.sinks=s1 (可以配置多个，以空格隔开，名字自己定）

#描述/配置a1的r1

a1.sources.r1.type=netcat (netcat表示通过指定端口来访问)

a1.sources.r1.bind=0.0.0.0 (表示本机)

a1.sources.r1.port=44444 （指定的端口，此端口不固定，但是不要起冲突）

#描述a1的s1

a1.sinks.s1.type=logger (表示数据汇聚点的类型是logger日志）

#描述a1的c1

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1 （一个source是可以对应多个通道的）

a1.sinks.s1.channel=c1 (一个sink,只能对应一个通道）

6.根据指定的配置文件，来启动flume

进入flume的bin目录

执行： ./flume-ng agent -n a1 -c ../conf -f ../conf/template.conf -Dflume.root.logger=INFO,console

计算机生成了可选文字:
HANNEL, n a me ： channe11 started 
2316 一 39 一 29 22 ： 43 ： 34 ， 741 （ conf 一 file 一 p0119r 一 3 ） 
[ INFO - org.apache.fIume.node.AppIicati 
0 n 。 startAUComponents (Application 。 j a V a ： 173 ） ] 
Starting Sink sinkl 
2316 一 39 一 29 22 ： 43 ： 34 ， 742 （ conf 一 file 一 p0119r 一 3 ） 
[ INFO - org.apache.fIume.node.AppIicati 
0 n 。 startAIIComponents (Application 。 j a V a ： 184 ） ] 
Starting Source sourcel 
2316 一 39 一 29 22 ： 43 ： 34 ， 756 (lifecycleSupervisor-1-0) [ INFO - org.apache.fIume.source.Ne 
tcatSource.start(NetcatSource.java:150)] Source starting 
2316 一 39 一 29 22 ： 43 ： 34 ， 835 (lifecycleSupervisor-1-0) [ INFO - org.apache.fIume.source.Ne 
tcatSource.start(NetcatSource.java:164)] Created serverSocket:sun.nio.ch.ServerSocke 
tChanneIImpI [ / ： ： ： ： ： ： ： ： 44444 ] 

如果出现上图所示，证明配置和启动成功

7.通过nc来访问：

nc localhost 44444

hello flume

计算机生成了可选文字:
[ root@localhost software]# nc localhost 8888 
hs110 flume 

或者：

通过外部http请求访问对应的ip和端口

比如：http://192.168.234.163:44444/hello

在虚拟机这边，会出现如下提示：

计算机生成了可选文字:
2316 一 39 一 29 22 ： 46 ： 34 ， 869 (SinkRunner-PoUingRunner-DefauItSinkProcessor) [INFO 
apache.flume.sink.LoggerSink.process(LoggerSink.java:94)] Event: { headers:{} 
47 45 54 23 2F 68 65 6C 6C 6F 23 48 54 54 53 2F GET / h9110 HTTP/ } 
2316 一 39 一 29 22 ： 46 ： 34 ， 873 (SinkRunner-PoUingRunner-DefauItSinkProcessor) [INFO 
apache.flume.sink.LoggerSink.process(LoggerSink.java:94)] Event: { headers:{} 
41 63 63 65 73 74 3A 23 2A 2F 2A OD 
Accept: * / * 
2316 一 39 一 29 22 ： 46 ： 34 ， 873 (SinkRunner-PoUingRunner-DefauItSinkProcessor) [INFO 
apache.flume.sink.LoggerSink.process(LoggerSink.java:94)] Event: { headers:{} 
41 63 63 65 73 74 2D 4C 61 6E 67 75 61 67 65 3A Accept-Language: } 
2316 一 39 一 29 22 ： 46 ： 34 ， 873 (SinkRunner-PoUingRunner-DefauItSinkProcessor) [INFO 
0 rg ． 
body: 
0 rg ． 
body: 
0 rg ． 
body: 
0 rg ． 

**启动命令解释**

|  |  |
| --- | --- |
| 参数 | 描述 |
| agent | 运行一个Flume Agent |
| --conf,-c <conf> | 指定配置文件放在什么目录 |
| --conf-file,-f <file> | 指定配置文件，这个配置文件必须在全局选项的--conf参数定义的目录下 |
| --name,-n <name> | Agent的名称，注意：要和配置文件里的名字一致。 |
| -Dproperty=value | 设置一个JAVA系统属性值。常见的：-Dflume.root.logger=INFO,console |

# flume的Source

source学习网址：

<http://flume.apache.org/FlumeUserGuide.html>

## 一、Avro 类型的Source

监听Avro 端口来接收外部avro客户端的事件流。和netcat不同的是，avro-source接收到的是经过avro序列化后的数据，然后反序列化数据继续传输。所以，如果是avro-source的话，源数据必须是经过avro序列化后的数据。而netcat接收的是字符串格式。

**利用Avro source可以实现多级流动、扇出流、扇入流等效果**。

另外，也可以接收通过flume提供的avro客户端发送的日志信息。

### Avro Source可配选项说明

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channels | 指定有几个通道并为每个通道起个名字 |
| type | 如果是Avro类型，则type属性为AVRO |
| bind | 需要监听的主机名或IP |
| port | 要监听的端口 |
| threads | 工作线程最大线程数 |
| selector.type |  |
| selector.\* |  |
| interceptors | 空格分隔的拦截器列表 |
| interceptors.\* |  |

### 实现步骤：

1.修改配置文件，source的type属性为avro

2.根据指定的配置文件启动flume

**格式代码，template-avro.conf为例子：**

#配置Agent a1 的组件

a1.sources=r1

a1.channels=c1

a1.sinks=s1

#描述/配置a1的source1

a1.sources.r1.type=avro

a1.sources.r1.bind=0.0.0.0

a1.sources.r1.port=44444

#描述sink

a1.sinks.s1.type=logger

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

3.执行启动指令，如果出现如下提示，证明启动成功

计算机生成了可选文字:
3 ． 3 ． 3 ， port: 44445 } ． 
2316 一 39 一 33 13 ： 32 ： 13 ， 543 (lifecycleSupervisor-1-4) [ INFO - org.apache.flume.instrumen 
tation.MonitoredCounterGroup. register(MonitoredCounterGroup.java:120)] Monitored C 0 u 
nter group for type: SOURCE, n a me ： sourcel: Successfully 
凹 ． 
2316 一 39 一 33 13 ： 32 ： 13 ， 543 (lifecycleSupervisor-1-4) [INFO 
0 rg 。 apache 。 flume 。 instrumen 
tation.MonitoredCounterGroup.start(MonitoredCounterGroup.java:96)] Component type: S 
OURCE, n a me ： sourcel started 
2316 一 39 一 33 13 ： 32 ： 13 ， 544 (lifecycleSupervisor-1-4) [INFO 
0 rg 。 apache. flume 。 S 0 u rc ． Av 
roSource.start(AvroSource.java:253)] Av ro S 0 u rc 9 sourcel 
started. 

4.执行agent-avro客户端指令：

./flume-ng avro-client -H 0.0.0.0 -p 44444 -F ../mydata/1.txt -c ../conf/

**（注：可以通过执行：./flume-ng help 来查看指令）**

如果出现：

计算机生成了可选文字:
2316 一 39 一 33 13 ： 52 ： 21 ， 928 (New I/O S 9 rv 9 r boss # 1 （ [ i d ： Oxc5ea8fcd, / 3 ： 3 ： 3 ： 3 ： 3 ： 3 ： 3 ： 3 ： 4 
0 rg 。 apache 。 a V ro 。 ipc .NettyServer$NettyServerAv roHandIer 。 handleUps trea 
4445 ] 月 [INFO 
m(NettyServer.java:171)] 巨 d ： 3x558b4979 ， / 127 ． 3 ． 3 ． 1 ： 63164 = > / 127 ． 3 ． 3 ． 1 ： 44445 ] OPEN 
2316 一 39 一 33 13 ： 52 ： 21 ， 928 (New I/O WO rke r # 3 ） [INFO 
0 rg 。 apache 。 a V ro 。 ipc .NettyServer 
$NettyServerAvroHandIer.handIeUpstream(NettyServer.java:171)] [ i d ： 3x558b4979 ， / 127 ． 
3 ． 3 ． 1 ： 63164 = > / 127 ． 3 ． 3 ． 1 ： 44445 ] 80 刂 ND ： / 127 ． 3 ． 3 ． 1 ： 44445 
2316 一 39 一 33 13 ： 52 ： 21 ， 928 (New I/O WO rke r # 3 ） [INFO 
0 rg 。 apache 。 a V ro 。 ipc .NettyServer 
$NettyServerAvroHandIer.handIeUpstream(NettyServer.java:171)] [ i d ： 3x558b4979 ， / 127 ． 
3 ． 3 ． 1 ： 63164 = > / 127 ． 3 ． 3 ． 1 ： 44445 ] CONNECTED: / 127 ． 3 ． 3 ． 1 ： 63164 
2316 一 39 一 33 13 ： 52 ： 22 ， 262 (SinkRunner-PoUingRunner-DefauItSinkProcessor) [ INFO - 0 rg ． 
apache.flume.sink.LoggerSink.process(LoggerSink.java:94)] Event: { headers:{} body: 
68 65 6C 6C 6F 23 66 6C 
75 6D 65 
2316 一 39 一 33 13 ： 52 ： 22 ， 438 
3 ． 3 ． 1 ： 63164 ： > / 127 ． 3 ． 3 ． 1 ： 44445 ] DISCONNECTED 
h9110 flume } 
(New I/O WO rke r # 3 ） [ INFO - org.apache.avro.ipc.NettyServer 
$NettyServerAvroHandIer.handIeUpstream(NettyServer.java:171)] [ i d ： 3x558b4979 ， / 127 ． 

## 二、Exec类型的Source

可以将命令产生的输出作为源

### Exec Source 可配置选项说明

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channels |  |
| type | 类型名称，需要是"exec" |
| command | 要执行的命令 |
| selector.type | 复制还是多路复用 |
| selector.\* | Depends on the selector.type value |
| interceptors | 空格分隔的拦截器列表 |
| interceptors.\* |  |

### 实现步骤：

1.修改配置文件，source的type属性为avro

配置代码：

#配置Agent a1 的组件

a1.sources=r1

a1.channels=c1

a1.sinks=s1

#描述/配置a1的source1

**a1.sources.r1.type=exec**

**a1.sources.r1.command=ping 192.168.234.163**

#描述sink

a1.sinks.s1.type=logger

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

2.进入到bin目录执行指令：

./flume-ng agent --conf ../conf --conf-file ../conf/template-exec.conf --name a1 -Dflume.root.logger=INFO,console

计算机生成了可选文字:
2316 一 31 一 29 39 ： 51 ： 52 ， 728 
31 39 64 bytes from 19 
2316 一 31 一 29 39 ： 51 ： 52 ， 728 
31 39 64 bytes from 19 
2316 一 31 一 29 39 ： 51 ： 55 ， 728 
31 39 64 bytes from 19 
2316 一 31 一 29 39 ： 51 ： 55 ， 729 
31 39 64 bytes from 19 
2316 一 31 一 29 39 ： 51 ： 55 ， 729 
31 39 64 bytes from 19 
(SinkRunner-PoUingRunner-DefauItSinkProce 
(SinkRunner-PoUingRunner-DefauItSinkProce 
(SinkRunner-PoUingRunner-DefauItSinkProce 
(SinkRunner-PoUingRunner-DefauItSinkProce 
(SinkRunner-PoUingRunner-DefauItSinkProce 

结果为ping 当前主机返回的结果

（也可以尝试其他的linux指令，比如 cat 1.txt 查看文件内容）

## 三、Spooling Directory类型的 Source

将指定的文件加入到“自动搜集”目录中。flume会持续监听这个目录，把文件当做source来处理。注意：一旦文件被放到“自动收集”目录中后，便不能修改，如果修改，flume会报错。此外，也不能有重名的文件，如果有，flume也会报错。

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channels |  |
| type | 类型，需要指定为"spooldir" |
| spoolDir | 读取文件的路径，即"搜集目录" |
| fileSuffix | .COMPLETED        对处理完成的文件追加的后缀 |
| deletePolicy | never        处理完成后是否删除文件，需是"never"或"immediate" |
| selector.type | replicating        replicating or multiplexing |
| selector.\* | Depends on the selector.type value |
| interceptors | Space-separated list of interceptors |
| interceptors.\* |  |

### 实现步骤：

1.配置示例：

#配置Agent a1 的组件

a1.sources=r1

a1.channels=c1

a1.sinks=s1

#描述/配置a1的source1

a1.sources.r1.type=spooldir

a1.sources.r1.spoolDir=/home/work/data

#描述sink

a1.sinks.s1.type=logger

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

2.创建相关的文件夹

3.根据指定的配置文件，启动flume

出现下图，证明flume正在监听指定的文件目录：

计算机生成了可选文字:
2316 一 39 一 33 14 ： 56 ： 26 ， 236 （ conf 一 file 一 p0119r 一 3 ） 
[ INFO - org.apache.fIume.node.AppIicati 
0 n 。 startAUComponents (Application 。 j a V a ： 173 ） ] 
Starting Sink sinkl 
2316 一 39 一 33 14 ： 56 ： 26 ， 237 （ conf 一 file 一 p0119r 一 3 ） 
[ INFO - org.apache.fIume.node.AppIicati 
0 n 。 startAIIComponents (Application 。 j a V a ： 184 ） ] 
Starting Source sourcel 
2316 一 39 一 33 14 ： 56 ： 26 ， 213 (lifecycleSupervisor-1-0) [ INFO - org.apache.fIume.source.Sp 
001DirectorySource.start(SpooIDirectorySource.java:78)] SpoolDirectorySource S 0 u rc 9 
starting with directory: /home/work/data 
2316 一 39 一 33 14 ： 56 ： 26 ， 344 (lifecycleSupervisor-1-8) [ INFO - org.apache.flume.instrumen 
tation.MonitoredCounterGroup. register(MonitoredCounterGroup.java:120)] Monitored C 0 u 
nter group for type: SOURCE, n a me ： s 0 u rc 91 ： Successfully registered new 卜 汜 ean ． 
2316 一 39 一 33 14 ： 56 ： 26 ， 344 (lifecycleSupervisor-1-0) [ INFO - org.apache.flume.instrumen 
tation.MonitoredCounterGroup.start(MonitoredCounterGroup.java:96)] Component type: S 
OURCE, n a me ： sourcel started 

4.向指定的文件目录下传送一个日志文件，发现flume的控制台打印相关的信息

此外，会发现被处理的文件，会追加一个后缀：completed，表示已处理完。

（重名文件包括已加后缀的文件）

## 四、NetCat Source

一个NetCat Source用来监听一个指定端口，并接收监听到的数据。

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channels |  |
| type | 类型名称，需要被设置为"netcat" |
| port | 指定要绑定到的端口号 |
| selector.type |  |
| selector.\* |  |
| interceptors |  |
| interceptors.\* |  |

配置示例：

#配置Agent a1 的组件

a1.sources=r1

a1.channels=c1

a1.sinks=s1

#描述/配置a1的r1

a1.sources.r1.type=netcat

a1.sources.r1.bind=0.0.0.0

a1.sources.r1.port=44444

#描述a1的s1

a1.sinks.s1.type=logger

#描述a1的c1

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

## 五、Sequence Generator Source --序列发生源

一个简单的序列发生器，不断的产生事件，值是从0开始每次递增1。主要用来测试。

可配置选项

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channels |  |
| type | seq |
| selector.type |  |
| selector.\* |  |
| interceptors |  |
| interceptors.\* |  |
| batchSize | 1 递增步长， 默认是1 |

**配置示例**

#配置Agent a1 的组件

a1.sources=r1

a1.sinks=s1

a1.channels=c1

#描述/配置a1的source1

a1.sources.r1.type=seq

#描述sink

a1.sinks.s1.type=logger

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

## 六、HTTP source

此Source接受HTTP的GET和POST请求作为Flume的事件。其中GET方式应该只用于试验。

如果想让flume正确解析Http协议信息，比如解析出请求头、请求体等信息，需要提供一个可插拔的"处理器"来将请求转换为事件对象，这个处理器必须实现HTTPSourceHandler接口。

这个处理器接受一个 HttpServletRequest对象，并返回一个Flume Envent对象集合。

flume提供了一些常用的Handler（处理器）：

**JSONHandler**

可以处理JSON格式的数据，并支持UTF-8 UTF-16 UTF-32字符集

该handler接受Evnet数组，并根据请求头中指定的编码将其转换为Flume Event

如果没有指定编码，默认编码为UTF-8.

JSON格式如下：

[{

"headers" : {

"timestamp" : "434324343",

"host" : "random\_host.example.com"

},

"body" : "random\_body"

},

{

"headers" : {

"namenode" : "namenode.example.com",

"datanode" : "random\_datanode.example.com"

},

"body" : "really\_random\_body"

}]

**BlobHandler**

BlobHandler是一种将请求中上传文件信息转化为event的处理器。

参数说明，加！为必须属性：

！handler        –        The FQCN of this class: org.apache.flume.sink.solr.morphline.BlobHandler

handler.maxBlobLength        100000000        The maximum number of bytes to read and buffer for a given request

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channels |  |
| type | HTTP |
| selector.type |  |
| selector.\* |  |
| interceptors |  |
| interceptors.\* |  |

**配置示例：**

#配置Agent a1 的组件

a1.sources=r1

a1.sinks=s1

a1.channels=c1

#描述/配置a1的source1

a1.sources.r1.type=http

a1.sources.r1.port=8888

#描述sink

a1.sinks.s1.type=logger

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

2.执行启动命令

3.执行curl 命令，模拟一次http的Post请求

curl -X POST -d '[{"headers":{"a":"a1","b":"b1"},"body":"hello http-flume"}]' <http://0.0.0.0:8890>

注：这种格式是固定的，因为我们用的是flume自身提供的Json格式的Handler。此外，需要包含header 和body两关键字，这样，handler在解析时才能拿到对应的数据

# flume的Sink

## 一、Logger Sink

记录指定级别（比如INFO，DEBUG，ERROR等）的日志，通常用于调试

要求，在 --conf参数指定的目录下有log4j的配置文件

根据设计，logger sink将体内容限制为16字节，从而避免屏幕充斥着过多的内容。如果想要查看调试的完整内容，那么你应该使用其他的sink，也许可以使用file\_roll sink，它会将日志写到本地文件系统中。

可配置项说明

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channel |  |
| type | logger |
| 补充说明 | 要求必须在 --conf 参数指定的目录下有 log4j的配置文件  可以通过-Dflume.root.logger=INFO,console在命令启动时手动指定log4j参数 |

配置示例：

#配置Agent a1 的组件

a1.sources=r1

a1.channels=c1

a1.sinks=s1

#描述/配置a1的r1

a1.sources.r1.type=netcat

a1.sources.r1.bind=0.0.0.0

a1.sources.r1.port=44444

#描述a1的s1

a1.sinks.s1.type=logger

#描述a1的c1

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

## 二、File Roll Sink

在本地系统中存储事件。

每隔指定时长生成文件保存这段时间内收集到的日志信息。

可配置参数说明

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channel |  |
| type | file\_roll |
| sink.directory | 文件被存储的目录 |
| sink.rollInterva | 30 记录日志到文件里，每隔30秒生成一个新日志文件。如果设置为0，则禁止滚动，从而导致所有数据被写入到一个文件中。 |

配置示例：

#配置Agent a1 的组件

a1.sources=r1

a1.sinks=s1

a1.channels=c1

#描述/配置a1的source1

a1.sources.r1.type=netcat

a1.sources.r1.bind=0.0.0.0

a1.sources.r1.port=8888

#描述sink

a1.sinks.s1.type=file\_roll

a1.sinks.s1.sink.directory=/home/work/rolldata

a1.sinks.s1.sink.rollInterval=60

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

创建指定的文件目录 /home/work/rolldata

启动测试

../bin/flume-ng agent -c ./ -f ./template.conf -n a1

## 三、Avro Sink

是实现多级流动、扇出流(1到多) 扇入流(多到1) 的基础。

可配置项说明

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channel |  |
| type | avro |
| hostname | The hostname or IP address to bind to |
| port | The port # to listen on |

## 四、HDFS Sink

此Sink将事件写入到Hadoop分布式文件系统HDFS中。

目前它支持创建文本文件和序列化文件。

对这两种格式都支持压缩。

这些文件可以分卷，按照指定的时间或数据量或事件的数量为基础。

它还通过类似时间戳或机器属性对数据进行 buckets/partitions 操作 It also buckets/partitions data by attributes like timestamp or machine where the event originated.

HDFS的目录路径可以包含将要由HDFS替换格式的转移序列用以生成存储事件的目录/文件名。

使用这个Sink要求haddop必须已经安装好，以便Flume可以通过hadoop提供的jar包与HDFS进行通信。

|  |  |
| --- | --- |
| 配置项 | 说明 |
| channel |  |
| type | hdfs |
| hdfs.path | HDFS 目录路径 （hdfs://namenode/flume/webdata/) |
| hdfs.inUseSuffix | .tmp        Flume正在处理的文件所加的后缀 |
| hdfs.rollInterval | 30        Number of seconds to wait before  举例：如果flume需要40s，30s=>1个文件 10s=>30s 1个文件 |
| hdfs.rollSize | 1024        File size to trigger roll, in bytes (0: never roll based on file size) |
| hdfs.rollCount | 10        Number of events written to file before it rolled (0 = never roll based on number of events) |
| hdfs.fileType | SequenceFile        File format: currently SequenceFile, DataStream or CompressedStream |

**配置文件**

#配置Agent a1 的组件

a1.sources=r1

a1.sinks=s1

a1.channels=c1

#描述/配置a1的source1

a1.sources.r1.type=netcat

a1.sources.r1.bind=0.0.0.0

a1.sources.r1.port=8888

#描述sink

a1.sinks.s1.type=hdfs

a1.sinks.s1.hdfs.path=hdfs://192.168.234.21:9000/flume

a1.sinks.s1.hdfs.fileType=DataStream

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

如果报错，是因为flume缺少相关hadoop的依赖jar包，找到以下的jar包，放到flume的lib目录下即可。

commons-configuration-1.6.jar

hadoop-auth-2.5.2.jar

hadoop-common-2.5.2.jar

hadoop-hdfs-2.5.2.jar

hadoop-mapreduce-client-core-2.5.2.jar

但是一个一个找特别麻烦，所以解决办法是将hadoop的jar包都拷贝到flume的lib目录下：

进入到hadoop安装目录的share目录下的hadoop目录

执行：scp common/\* common/lib/\* hdfs/\* hdfs/lib/\* mapreduce/\* mapreduce/lib/\* tools/lib/\* 192.168.234.163:/home/software/flume/lib/

# flume的Channel

## 一、Memory Channel

事件将被存储在内存中（指定大小的队列里）

**非常适合那些需要高吞吐量且允许数据丢失的场景下**

属性说明：

|  |  |
| --- | --- |
| 配置项 | 说明 |
| type | memory |
| capacity | 100        事件存储在信道中的最大数量 the maximum number of events stored in the channel 建议实际工作调节：10万 |
| transactionCapacity | 100        每个事务中的最大事件数 the maximum number of events the channel wile take from a source or give  to a sink per transaction 建议实际工作调节：1000~3000 |

配置示例略

## 二、JDBC Channel

事件会被持久化（存储）到可靠的数据库里，目前支持嵌入式Derby数据库。即source=》channel=》sink。在传输的过程中，会先把事件存到关系型数据库里。但是Derby数据库不太好用，所以JDBC Channel目前仅用于测试，不能用于生产环境。

## 三、FileChannel

性能比较低，但是即使程序出错数据不会丢失

性能会比较低下，但是即使程序出错数据不会丢失

|  |  |
| --- | --- |
| 配置项 | 说明 |
| type | file |
| dataDirs | ~/.flume/file-channel/data        逗号分隔的目录列表，用以存放日志文件。使用单独的磁盘上的多个目录可以提高文件通道效率。 |

**配置示例:**

a1.sources=r1

a1.channels=c1

a1.sinks=s1

a1.sources.r1.type=netcat

a1.sources.r1.bind=0.0.0.0

a1.sources.r1.port=8888

a1.sinks.s1.type=logger

a1.channels.c1.type=file

a1.channels.c1.dataDirs=/home/filechannel

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

## 四、Spillable Memory Channel -- 内存溢出通道

事件被存储在内存队列和磁盘中。

内存队列作为主存储，而磁盘作为溢出内容的存储。

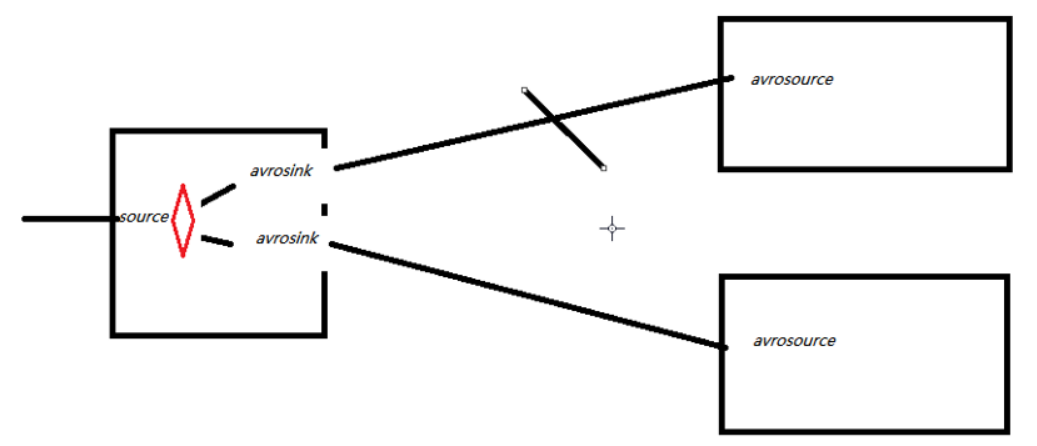
此通道处于试验阶段，不建议在生产环境中使用。

# flume的Selector

## 一、Selector的复制模式

选择器可以工作在复制、多路复用(路由) 模式下

Selector 默认是复制模式(replicating)，即把source复制，然后分发给多个sink



|  |  |
| --- | --- |
| 配置项 | 说明 |
| selector.type | replicating 表示复制模式，source的selector如果不配置，默认就是这种模式  针对上图，在复制模式下，当source接收到数据后，会复制成两份，分别发给两个avro sink |
| selector.optional | 标志通道为可选 |

针对每个Source，都有一个配置参数，即selector.type

配置示例：

a1.sources = r1

a1.channels = c1 c2 c3

a1.source.r1.selector.type = replicating(这个是默认的)

a1.source.r1.channels = c1 c2 c3

a1.source.r1.selector.optional = c3

## 二、多路复用模式

在这种模式下，用户可以指定转发的规则。selector根据规则进行数据的分发

**可配置参数**

|  |  |
| --- | --- |
| 配置项 | 说明 |
| selector.type | multiplexing 表示路由模式 |
| selector.header | 指定要监测的头的名称 |
| selector.mapping.\* | 匹配规则 |
| selector.default | 如果未满足匹配规则，则默认发往指定的通道 |

### 案例需求说明：

01机利用http resource接收数据，根据路由规则，发往02，03机。02，03通过avro source接收数据，通过logger sink 打印数据

说明：

检测的头信息是state关键字，如果state=CN，进c1通道。如果state=US，进c2,c3通道。如果不满足以上的情况，进c4通道。

配置示例：

#配置Agent a1 的组件

a1.sources=r1

a1.sinks=s1 s2

a1.channels=c1 c2

#描述/配置a1的source1

a1.sources.r1.type=http

a1.sources.r1.port=8888

a1.sources.r1.selector.type=multiplexing

a1.sources.r1.selector.header=state

a1.sources.r1.selector.mapping.cn=c1

a1.sources.r1.selector.mapping.us=c2

a1.sources.r1.selector.default=c2

#描述sink

a1.sinks.s1.type=avro

a1.sinks.s1.hostname=192.168.234.212

a1.sinks.s1.port=9999

a1.sinks.s2.type=avro

a1.sinks.s2.hostname=192.168.234.213

a1.sinks.s2.port=9999

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

a1.channels.c2.type=memory

a1.channels.c2.capacity=1000

a1.channels.c2.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1 c2

a1.sinks.s1.channel=c1

a1.sinks.s2.channel=c2

02,03配置示例：

#配置Agent a1 的组件

a1.sources=r1

a1.sinks=s1

a1.channels=c1

#描述/配置a1的source1

a1.sources.r1.type=avro

a1.sources.r1.bind=0.0.0.0

a1.sources.r1.port=9999

#描述sink

a1.sinks.s1.type=logger

#描述内存channel

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

#位channel 绑定 source和sink

a1.sources.r1.channels=c1

a1.sinks.s1.channel=c1

然后执行：

curl -X POST -d '[{"headers":{"state":"cn"},"body":"hello china" }]' <http://192.168.150.137:8890>

（01机的ip和端口）