一 内容回顾（列举前一天重点难点内容）

1.1 教学重点

```
窗口函数
序列函数
排名函数
UDF自定义函数
自定义函数三种方式
自定义函数案例
数据导入导出
序列化和反序列化SerDe
文件存储格式
优化
```

1.2 教学难点

```
常用窗口函数
UDF自定义函数
序列化和反序列化SerDe
优化
```

二 教学目标

```
Flume简介
Flume安装与测试
Flume部署
Flume source相关配置
Flume sink相关配置及测试
Flume selector相关配置与案例分析
Flume Sink Processors相关配置和案例分析
Flume Interceptors相关配置和案例分析(选讲)
Flume AVRO Client开发(选讲)
```

三 教学导读

3.1 为什么学习Flume?

Flume是一种分布式的，可靠的、高可用的服务，用于有效地收集，聚合和移动大量日志数据。它具有基于流数据流的简单灵活的体系结构。它具有可调整的可靠性机制以及许多故障转移和恢复机制，具有强大的功能和容错能力。它使用一个简单的可扩展数据模型，允许在线分析应用程序。

四 教学内容

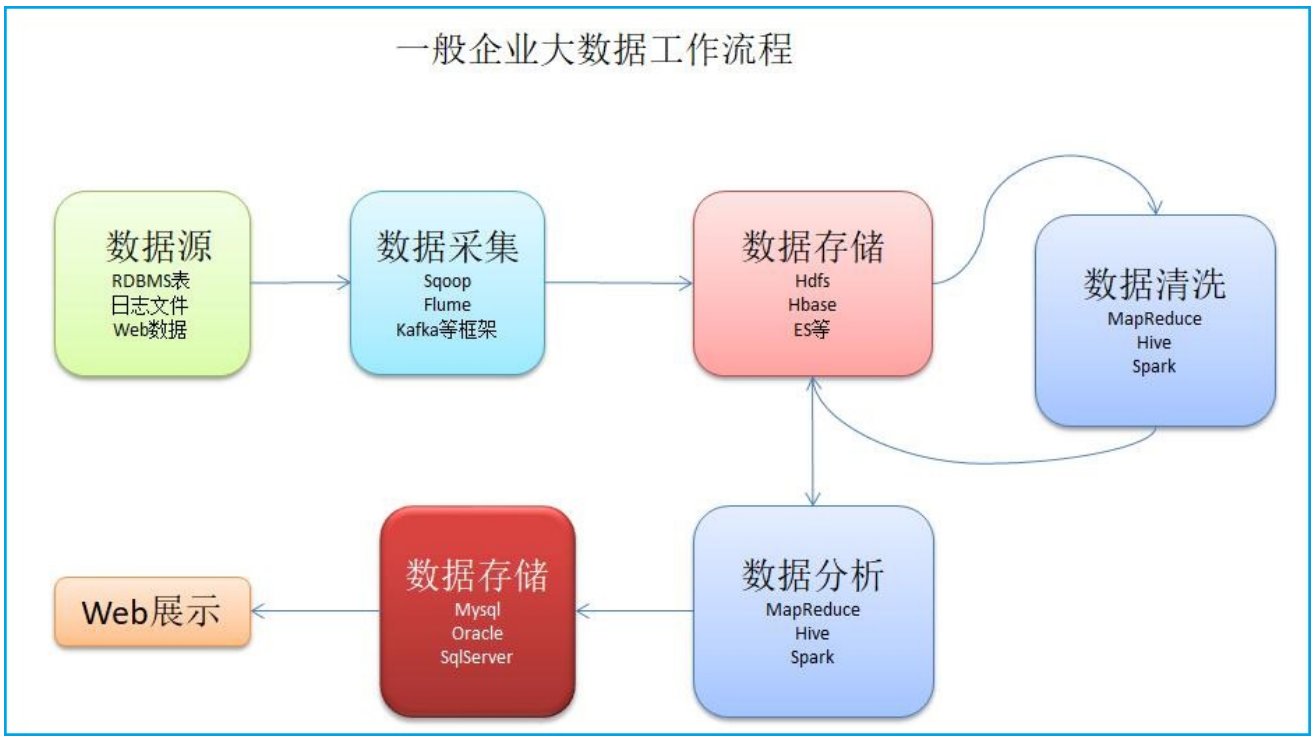
4.1 Flume的简介

4.1.1 大数据处理流程

在企业中，大数据的处理流程一般是：

- 1.数据采集
- 2.数据存储
- 3.数据清洗
- 4.数据分析
- 5.数据展示

参考下图：



在数据采集和搜集的工具中，Flume框架占有一定的市场份量。

4.1.2 Flume的简介

Flume是一种分布式的，可靠的、高可用的服务，用于有效地收集，聚合和移动大量日志数据。它具有基于流数据流的简单灵活的体系结构。它具有可调整的可靠性机制以及许多故障转移和恢复机制，具有强大的功能和容错能力。它使用一个简单的可扩展数据模型，允许在线分析应用程序。

> 参考官网： http://flume.apache.org/
>
> Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

Flume最开始是由 cloudera 开发的实时日志收集系统，受到了业界的认可与广泛应用。但随着 Flume功能的扩展，Flume的代码工程臃肿、核心组件设计不合理、核心配置不标准等缺点渐渐暴露出来，尤其是在发行版本0.9.4中，日志传输不稳定的现象尤为严重。

为了解决这些问题，2011 年 10 月 22 号，cloudera 对 Flume进行了里程碑式的改动：重构核心组件、核心配置以及代码架构，并将 Flume纳入 apache 旗下，从cloudera Flume改名为 Apache Flume。

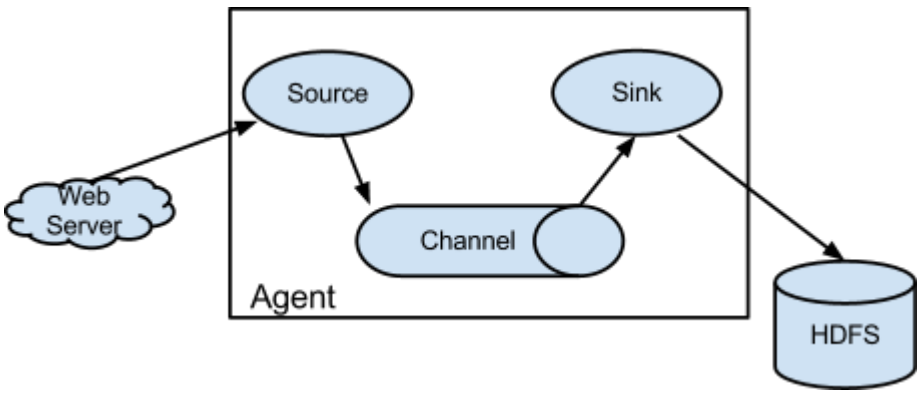
4.1.3 版本区别

为了与之前版本区分开，重构后的版本统称为 **FlumeNG（next generation）**，重构前的版本被统称为 **FlumeOG（original generation）**，Flume目前只有Linux系统的启动脚本，没有Windows环境的启动脚本。

4.2 Flume的体系结构

4.2.1 体系结构简介

Flume运行的核心是 Agent。Flume是以agent为最小的独立运行单位。一个agent就是一个JVM。它是一个完整的数据收集工具，含有三个核心组件，分别是source、 channel、 sink。通过这些组件， Event 可以从一个地方流向另一个地方。如下图所示：



4.2.2 组件及其作用

```
- Client:
    客户端，Client生产数据，运行在一个独立的线程中

- Event:
    一个数据单元，消息头和消息体组成。（Events可以是日志记录、 avro 对象等。）

- Flow:
    Event从源点到达目的点的迁移的抽象。

- Agent:
    一个独立的Flume进程，运行在JVM中，包含组件Source、 Channel、 Sink。
    每台机器运行一个agent，但是一个agent中可以包含多个sources和sinks。

- Source:
    数据收集组件。source从Client收集数据，传递给Channel

- Channel:
    管道，负责接收source端的数据，然后将数据推送到sink端。

- Sink:
```

```
    负责从channel端拉取数据，并将其推送到持久化系统或者是下一个Agent。

- selector:
    选择器，作用于source端，然后决定数据发往哪个目标。

- interceptor:
    拦截器，Flume允许使用拦截器拦截数据。允许使用拦截器链，作用于source和sink阶段。
```

4.3 Flume的安装

4.3.1 安装和配置环境变量

4.3.1.1 准备软件包

```
将apache-flume-1.9.0-bin.tar.gz 上传到linux系统中的/root/softwares/目录中
```

4.3.1.2 解压软件包

```
[root@qianfeng01 softwares]# pwd
/root/local
[root@qianfeng01 softwares]# tar -zxvf apache-flume-1.9.0-bin.tar.gz -C /usr/local/
```

4.3.1.3 更名操作

```
[root@qianfeng01 softwares]# cd /usr/local/
[root@qianfeng01 local]# mv apache-flume-1.9.0-bin/ flume-1.9.0
```

4.3.1.4 配置环境变量

```
[root@qianfeng01 local]# vi /etc/profile
.........省略..........
export FLUME_HOME=/usr/local/flume-1.9.0
export PATH=$FLUME_HOME/bin:$PATH

# 加载环境变量
[root@qianfeng01 local]# source /etc/profile
```

4.3.1.5 验证环境变量

```
[root@qianfeng01 local]# flume-ng version
Flume1.9.0
Source code repository: https://git-wip-us.apache.org/repos/asf/flume.git
Revision: 99f591994468633fc6f8701c5fc53e0214b6da4f
Compiled by denes on Fri Sep 15 14:58:00 CEST 2017
From source with checksum fbb44c8c8fb63a49be0a59e27316833d
```

4.3.2 配置文件

```
[root@qianfeng01 local]# cd flume/conf/
[root@qianfeng01 conf]# ll    #查看里面是否有一个flume-env.sh.template文件
[root@qianfeng01 conf]# cp flume-env.sh.template flume-env.sh
[root@qianfeng01 conf]# vi flume-env.sh
........省略..........
export JAVA_HOME=/usr/local/jdk-1.8.0_45
........省略..........
```

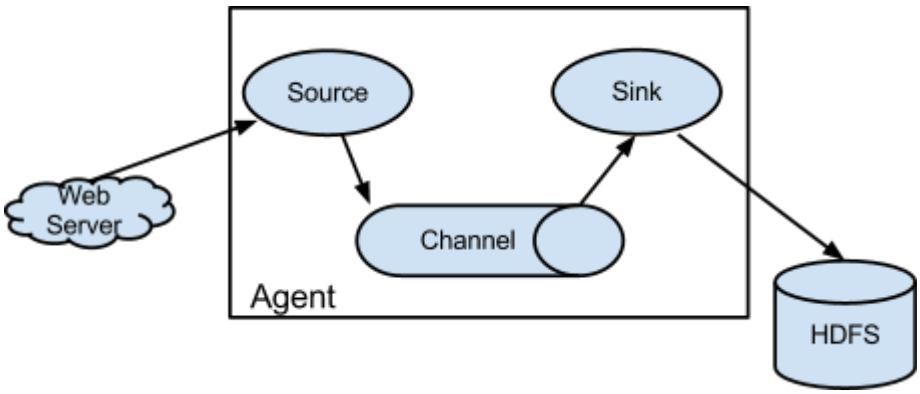
4.4 Flume的部署

4.4.1 数据模型

```
- 单一数据模型
- 多数据流模型
```

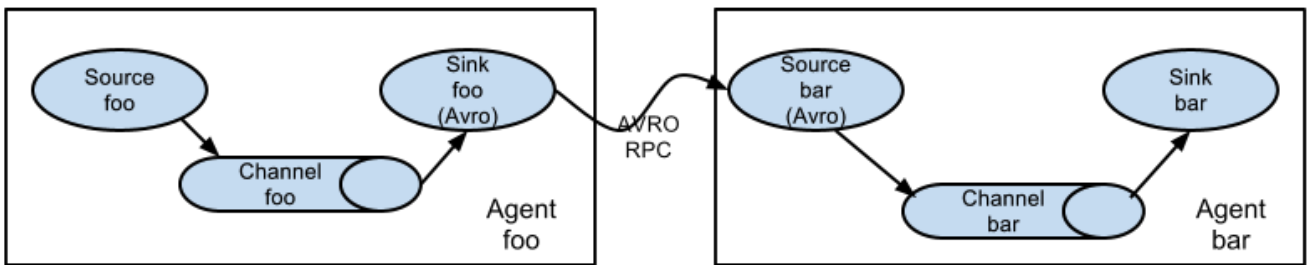
4.4.1.1 单一数据模型

在单个 Agent 内由单个 Source, Channel, Sink 建立一个单一的数据流模型，如下图所示，整个数据流为 Web Server --> Source --> Channel --> Sink --> HDFS。

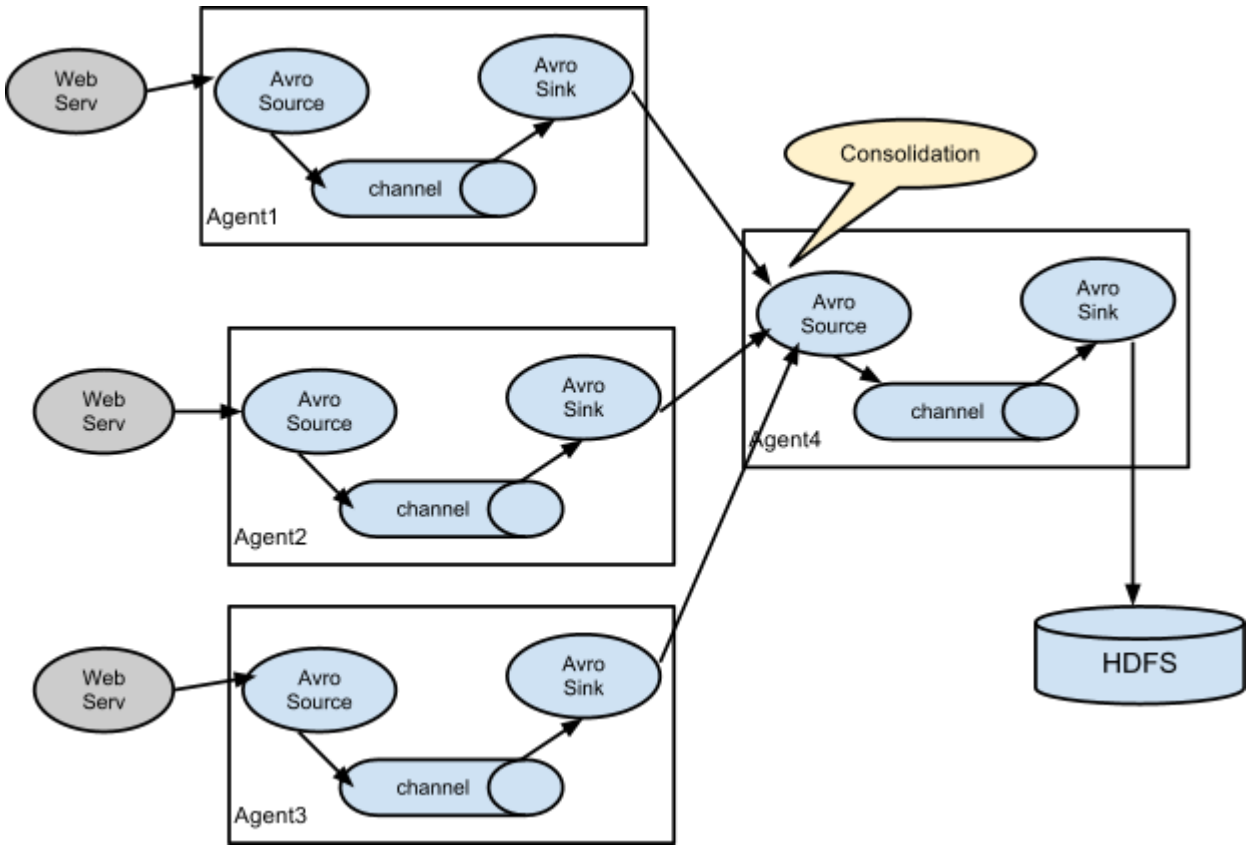


4.4.1.2 多数据流模型

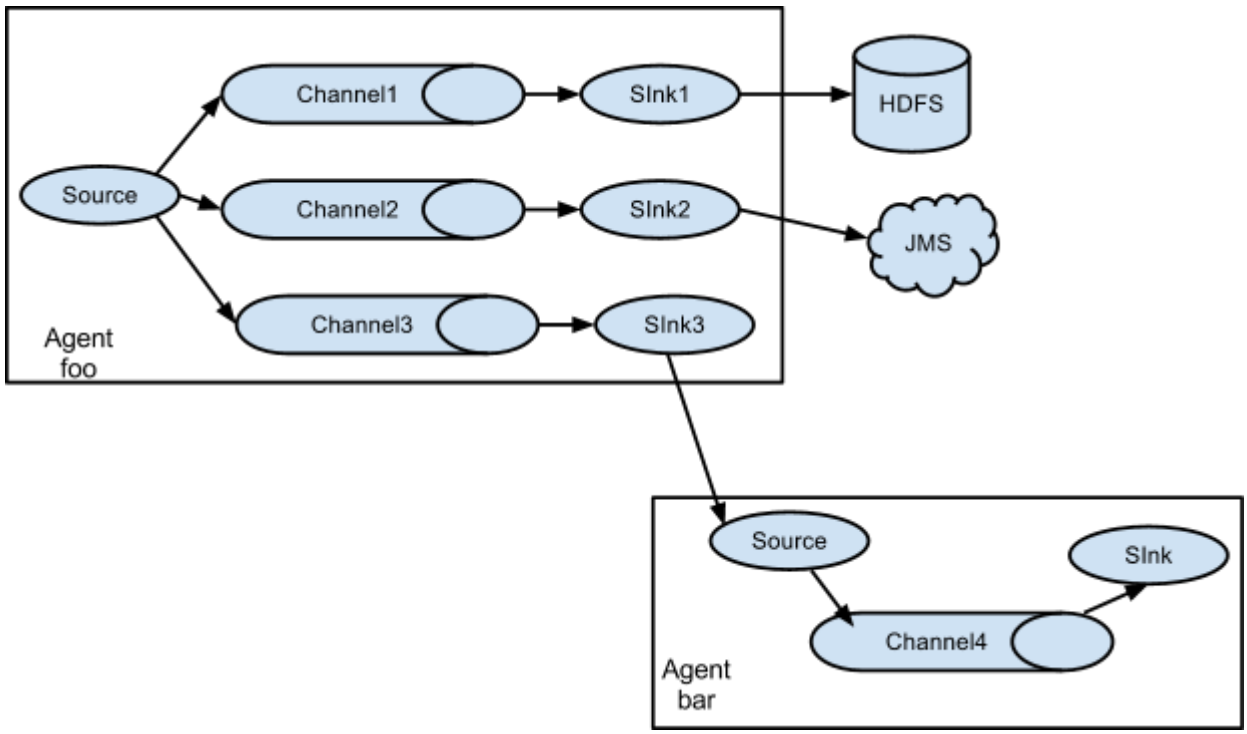
1）多 Agent 串行传输数据流模型



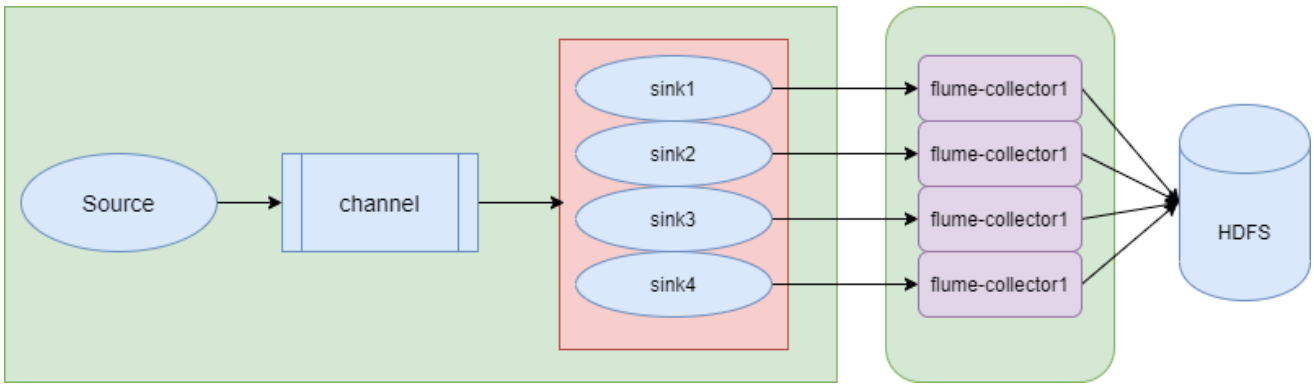
2）多 Agent 汇聚数据流模型



3）单 Agent 多路数据流模型



4）Sinkgroups 数据流模型



4.4.1.3 小总结

```
在flume提供的数据流模型中，几个原则很重要。

Source--> Channel
  1.单个Source组件可以和多个Channel组合建立数据流，既可以replicating(复制) 和 multiplexing(多路复用)。
  2.多个Sources可以写入单个 Channel

Channel-->Sink
  1.多个Sinks又可以组合成Sinkgroups从Channel中获取数据，既可以loadbalancing(负载均衡)和failover(故障转移)机制。
  2.多个Sinks也可以从单个Channel中取数据。
  3.单个Sink只能从单个Channel中取数据

根据上述 5 个原则，你可以设计出满足你需求的数据流模型。
```

4.4.2 配置介绍

4.4.2.1 定义组件名称

要定义单个代理中的流，您需要通过通道链接源和接收器。您需要列出给定代理的源，接收器和通道，然后将源和接收器指向一个通道。一个源实例可以指定多个通道，但是一个接收器实例只能指定一个通道。格式如下：

```
# list the sources, sinks and channels for the agent
<Agent>.sources = <Source>
<Agent>.channels = <Channel1> <Channel2>
<Agent>.sinks = <Sink>

# set channel for source
<Agent>.sources.<Source>.channels = <Channel1> <Channel2> ...

# set channel for sink
<Agent>.sinks.<Sink>.channel = <Channel1>
```

案例如下：

```
# list the sources, sinks and channels for the agent
agent_foo.sources = avro-appserver-src-1
agent_foo.channels = mem-channel-1
agent_foo.sinks = hdfs-sink-1

# set channel for source
agent_foo.sources.avro-appserver-src-1.channels = mem-channel-1

# set channel for sink
agent_foo.sinks.hdfs-sink-1.channel = mem-channel-1
```

4.4.2.2 配置组件属性

```
# properties for sources
<Agent>.sources.<Source>.<someProperty> = <someValue>

# properties for channels
<Agent>.channel.<Channel>.<someProperty> = <someValue>

# properties for sinks
<Agent>.sources.<Sink>.<someProperty> = <someValue>
```

案例如下:

```
agent_foo.sources = avro-AppSrv-source
agent_foo.sinks = hdfs-Cluster1-sink
agent_foo.channels = mem-channel-1

# set channel for sources, sinks

# properties of avro-AppSrv-source
agent_foo.sources.avro-AppSrv-source.type = avro
agent_foo.sources.avro-AppSrv-source.bind = localhost
agent_foo.sources.avro-AppSrv-source.port = 10000

# properties of mem-channel-1
agent_foo.channels.mem-channel-1.type = memory
agent_foo.channels.mem-channel-1.capacity = 1000
agent_foo.channels.mem-channel-1.transactionCapacity = 100

# properties of hdfs-Cluster1-sink
agent_foo.sinks.hdfs-Cluster1-sink.type = hdfs
agent_foo.sinks.hdfs-Cluster1-sink.hdfs.path = hdfs://namenode/flume/webdata

#...
```

4.4.3 常用的Source和Sink种类

4.4.3.1 常用的Flume Sources

```
# Avro source:
    avro
# Syslog TCP source:
    syslogtcp
# Syslog UDP Source:
    syslogudp
# HTTP Source:
    http
# Exec source:
    exec
# JMS source:
    jms
# Thrift source:
    thrift
# Spooling directory source: (监控目录)
    spooldir
# Kafka source:
    org.apache.flume.source.kafka,KafkaSource
.....
```

4.4.3.2 常用的Flume hannels

```
# Memory Channel
    memory
# JDBC Channel
    jdbc
# Kafka Channel
    org.apache.flume.channel.kafka.KafkaChannel
# File Channel
    file
```

4.4.3.3 常用的Flume Sinks

```
# HDFS Sink
    HDFS
# HIVE Sink
    Hive
# Logger Sink
    Logger
# Avro Sink
    Avro
# Kafka Sink
    org.apache.flume.sink.kafka.KafkaSink
# Hbase Sink
    Hbase
```

4.5 案例演示

4.5.1 案例演示：avro+memory+logger

> Avro Source：监听一个指定的Avro端口，通过Avro端口可以获取到Avro client发送过来的文件，即只要应用程序通过Avro端口发送文件，source组件就可以获取到该文件中的内容,输出位置为Logger

4.5.1.1 编写采集方案

```
[root@qianfeng01 flume-1.9.0]# mkdir flumeconf
[root@qianfeng01 flume-1.9.0]# cd flumeconf
[root@qianfeng01 flumeconf]# vi avro-logger.conf
#定义各个组件的名字
a1.sources=avro-sour1
a1.channels=mem-chan1
a1.sinks=logger-sink1

#定义sources组件的相关属性
a1.sources.avro-sour1.type=avro
a1.sources.avro-sour1.bind=qianfeng01
a1.sources.avro-sour1.port=9999

#定义channels组件的相关属性
a1.channels.mem-chan1.type=memory

#定义sinks组件的相关属性
a1.sinks.logger-sink1.type=logger
a1.sinks.logger-sink1.maxBytesToLog=100

#组件之间进行绑定
a1.sources.avro-sour1.channels=mem-chan1
a1.sinks.logger-sink1.channel=mem-chan1
```

4.5.1.2 启动Agent

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./avro-logger.conf -n a1 -Dflume.root.logger=INFO,console
```

4.5.1.3 测试数据

```
[root@qianfeng01 ~]# mkdir flumedata
[root@qianfeng01 ~]# cd flumedata/
[root@qianfeng01 flumedata]#
[root@qianfeng01 flumedata]# date >> test.data
[root@qianfeng01 flumedata]# cat test.data
2019年 11月 21日 星期四 21:22:36 CST
[root@qianfeng01 flumedata]# ping qianfeng01 >> test.data
[root@qianfeng01 flumedata]# cat test.data
....省略....
[root@qianfeng01 flumedata]# flume-ng avro-client -c /usr/local/flume-1.9.0/conf/ -H qianfeng01 -p 9999 -F ./test.data
```

4.5.2 实时采集(监听文件)：exec+memory+hdfs

> Exec Source:监听一个指定的命令，获取一条命令的结果作为它的数据源 #常用的是tail -F file指令，即只要应用程序向日志（文件）里面写数据，source组件就可以获取到日志（文件）中最新的内容
>
> memory:传输数据的Channel为Memory
>
> hdfs 是输出目标为Hdfs

4.5.2.1 配置方案

```
[root@qianfeng01 flumeconf]# vi exec-hdfs.conf
a1.sources=r1
a1.sources.r1.type=exec
a1.sources.r1.command=tail -F /root/flumedata/test.data

a1.channels=c1
a1.channels.c1.type=memory
#通道中可以保存的最大事件数量
a1.channels.c1.capacity=1000
#通道从一个source可以获取的最大事件数量或者每个事务中给一个sink的最大事件数量
a1.channels.c1.transactionCapacity=100

a1.sinks=k1
a1.sinks.k1.type=hdfs
a1.sinks.k1.hdfs.path=hdfs://qianfeng01:8020/flume/tailout/%y-%m-%d/%H%M/
#设置文件的前缀
a1.sinks.k1.hdfs.filePrefix=events-
#时间戳是否四舍五入
a1.sinks.k1.hdfs.round=true
#时间戳舍入的最高位数
a1.sinks.k1.hdfs.roundValue=10
#时间戳舍入的单位
a1.sinks.k1.hdfs.roundUnit=second
#设置滚动的条件(关闭当前文件,开启新文件)---3秒钟滚动一次
a1.sinks.k1.hdfs.rollInterval=3
#设置滚动的条件---20字节
a1.sinks.k1.hdfs.rollSize=20
#设置滚动的条件---5个事件
a1.sinks.k1.hdfs.rollCount=5
#刷新进hdfs的事件数量
a1.sinks.k1.hdfs.batchSize=100
#是否使用本地时间戳(自定义拦截器中)---true是使用本地的
a1.sinks.k1.hdfs.useLocalTimeStamp=true
a1.sinks.k1.hdfs.fileType=DataStream

a1.sources.r1.channels=c1
a1.sinks.k1.channel=c1
```

4.5.2.2 启动Agent

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./exec-hdfs.conf -n a1 -Dflume.root.logger=INFO,console
```

报错解决:

```
报错:
(SinkRunner-PollingRunner-DefaultSinkProcessor) [ERROR - org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:459)] process failed
java.lang.NoSuchMethodError: com.google.common.base.Preconditions.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)V

原因:com.google.common.base.Preconditions.checkArgument 这是因为flume-1.9.0内依赖的guava-11.02.jar和hadoop内的(guava-27.0-jre.jar)版本不一致造成的。

检验方法:
查看hadoop安装目录下share/hadoop/common/lib内guava.jar版本
查看Flume安装目录下lib内guava.jar的版本
如果两者不一致，删除版本低的，并拷贝高版本过去
```

4.5.2.3 测试数据

```
[root@qianfeng01 flumedata]# ping qianfeng01 >> test.data
```

4.5.3 实时采集(监听文件) exec+memory+logger

> Exec Source:监听一个指定的命令，获取一条命令的结果作为它的数据源 #常用的是tail -F file指令，即只要应用程序向日志（文件）里面写数据，source组件就可以获取到日志（文件）中最新的内容，
>
> logger为日志格式输出

4.5.3.1 配置方案

```
[root@qianfeng01 flumeconf]# vi exec-logger.conf
a2.sources = r1
a2.channels = c1
a2.sinks = s1

a2.sources.r1.type = exec
a2.sources.r1.command = tail -F /root/flumedata/log.01

a2.channels.c1.type=memory
a2.channels.c1.capacity=1000
a2.channels.c1.transactionCapacity=100
a2.channels.c1.keep-alive=3
#通道中的事件总容量(byteCapacity)和预估总事件容量的百分比
a2.channels.c1.byteCapacityBufferPercentage=20
a2.channels.c1.byteCapacity=800000

a2.sinks.s1.type=logger
a2.sinks.s1.maxBytesToLog=16
```

```
19
20 a2.sources.r1.channels=c1
21 a2.sinks.s1.channel=c1
```

4.5.3.2 启动agent

```
1 [root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./exec-logger.conf -n a2 -
  Dflume.root.logger=INFO,console
```

4.5.3.3 测试：

```
1 [root@qianfeng01 ~]# echo "nice" >> /root/flumedata/log.01
```

4.5.4 实时采集(监听目录)：Spool +file + hdfs

> Spool 是Source来源于目录，有文件进入目录就摄取，File Channel将它暂存到磁盘，最终目的地是HDFS 即只要某个目录不断有文件，HDFS上也会同步到所有数据。

4.5.4.1 配置方案

```
1  [root@qianfeng01 flumeconf]# vi spool-hdfs.conf
2  a1.sources = r1
3  a1.channels = c1
4  a1.sinks = k1
5
6  a1.sources.r1.type = spooldir
7  a1.sources.r1.spoolDir = /root/flumedata/input/2020/01/
8
9
10 a1.channels.c1.type = file
11 a1.channels.c1.checkpointDir = /root/flumedata/checkpoint
12 a1.channels.c1.dataDirs = /root/flumedata/data
13
14
15 a1.sinks.k1.type = hdfs
16 a1.sinks.k1.hdfs.path = hdfs://qianfeng01:8020/flume/spooldir
17 a1.sinks.k1.hdfs.filePrefix =
18 a1.sinks.k1.hdfs.round = true
19 a1.sinks.k1.hdfs.roundValue = 10
20 a1.sinks.k1.hdfs.roundUnit = minute
21 a1.sinks.k1.hdfs.fileSuffix= .log
22 a1.sinks.k1.hdfs.rollInterval=60
23 a1.sinks.k1.hdfs.fileType=DataStream
24 a1.sinks.k1.hdfs.writeFormat=Text
25
26
27 a1.sources.r1.channels = c1
28 a1.sinks.k1.channel = c1
```

4.5.4.2 启动Agent

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf/ -f ./spool-hdfs.conf -n a1 -Dflume.root.logger=INFO,console
```

4.5.4.3 测试

```
--1. 向/home/flume/input/2020/01/目录里添加几个文件
[root@qianfeng01 ~]# cd /home/flume/input/2020/01/
[root@qianfeng01 01]# echo "hello world" >>a1.log
[root@qianfeng01 01]# echo "no zuo no die" >>a2.log
[root@qianfeng01 01]# echo "ao li gei" >>a3.sh
--2. 查看hdfs的目录内容
--3. 在查看一下home/flume/input/2020/01/目录里的内容
[root@qianfeng01 01]# ll
```

4.5.5 实时采集(监听目录)：Spool+ mem+logger

> Spool: Source来源于目录，有文件进入目录就摄取，File Channel将它暂存到磁盘，最终目的地是HDFS 即只要某个目录不断有文件，HDFS上也会同步到所有数据。
>
> mem:通过内存来传输数据
>
> logger:是传送数据到日志

4.5.5.1 配置方案

```
[root@qianfeng01 flumeconf]# vi spool-logger.conf
a1.sources = r1
a1.channels = c1
a1.sinks = s1

a1.sources.r1.type=spooldir
a1.sources.r1.spoolDir = /root/flumedata/spool
#设置了前缀的代表已经采集完成
a1.sources.r1.fileSuffix = .COMPLETED
#never代表采集完不删除
a1.sources.r1.deletePolicy=never
#是否添加header中的文件绝对路径显示,false代表不显示
a1.sources.r1.fileHeader=false
#文件路径对应的fileHeaderKey
a1.sources.r1.fileHeaderKey=file
#是否添加header中的文件名字,false代表不显示
a1.sources.r1.basenameHeader=false
#文件名字对用的basenameHeaderKey
a1.sources.r1.basenameHeaderKey=basename
a1.sources.r1.batchSize=100
a1.sources.r1.inputCharset=UTF-8
a1.sources.r1.bufferMaxLines=1000

a1.channels.c1.type=memory
a1.channels.c1.capacity=1000
```

```
a1.channels.c1.transactionCapacity=100
a1.channels.c1.keep-alive=3
a1.channels.c1.byteCapacityBufferPercentage=20
a1.channels.c1.byteCapacity=800000

a1.sinks.s1.type=logger
a1.sinks.s1.maxBytesToLog=16

a1.sources.r1.channels=c1
a1.sinks.s1.channel=c1
```

4.5.5.2 启动Agent

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./spool-logger.conf -n a1 -Dflume.root.logger=INFO,console
```

4.5.5.3 测试

```
[root@qianfeng01 ~]# for i in `seq 1 10` ; do echo $i >> /root/flume/spool/$i;done
```

4.5.6 案例演示：http+ mem+logger

> http: 表示数据来源是http网络协议,一般接收的请求为get或post请求. 所有的http请求会通过插件格式的Handle转化为一个Flume的Event数据.
>
> mem:表示用内存传输通道
>
> logger:表示输出格式为Logger格式

4.5.6.1 配置方案

```
[root@qianfeng01 flumeconf]# vi http-logger.conf
a1.sources = r1
a1.channels = c1
a1.sinks = s1

a1.sources.r1.type=http
a1.sources.r1.bind = qianfeng01
a1.sources.r1.port = 6666
a1.sources.r1.handler = org.apache.flume.source.http.JSONHandler
a1.sources.r1.handler.nickname = JSON props

a1.channels.c1.type=memory
a1.channels.c1.capacity=1000
a1.channels.c1.transactionCapacity=100
a1.channels.c1.keep-alive=3
a1.channels.c1.byteCapacityBufferPercentage=20
a1.channels.c1.byteCapacity=800000

a1.sinks.s1.type=logger
a1.sinks.s1.maxBytesToLog=16
```

```
21
22 a1.sources.r1.channels=c1
23 a1.sinks.s1.channel=c1
```

4.5.6.2 启动agent的服务

```
1 [root@qianfeng01 ~]# flume-ng agent -c ../conf -f ./http-logger.conf -n a1 -
  Dflume.root.logger=INFO,console
```

4.5.6.3 测试

```
1 curl -X 指定请求方式 -d 指定发送内容  服务器地址
2
3 [root@qianfeng01 ~]# curl -X POST -d '[{"headers":
  {"name":"zhangsan","pwd":"123456"},"body":"this is my content"}]' http://qianfeng01:6666
```

4.5.7 案例演示：Syslogtcp+Mem+Logger

> Syslogtcp: syslog广泛应用于系统日志。syslog日志消息既可以记录在本地文件中，也可以通过网络发送到接收syslog的服务器。接收syslog的服务器可以对多个设备的syslog消息进行统一的存储，或者解析其中的内容做相应的处理。本数据源指的是syslog的通过tcp端口来传送数据
>
> mem:通过内存选择器来处理
>
> logger:输出目的地为logger

4.5.7.1 配置方案

```
1  [root@qianfeng01 flumeconf]# vi syslogtcp-logger.conf
2  a1.sources = r1
3  a1.channels = c1
4  a1.sinks = s1
5
6  a1.sources.r1.type=syslogtcp
7  a1.sources.r1.host = qianfeng01
8  a1.sources.r1.port = 6666
9
10 a1.channels.c1.type=memory
11 a1.channels.c1.capacity=1000
12 a1.channels.c1.transactionCapacity=100
13 a1.channels.c1.keep-alive=3
14 a1.channels.c1.byteCapacityBufferPercentage=20
15 a1.channels.c1.byteCapacity=800000
16
17 a1.sinks.s1.type=logger
18 a1.sinks.s1.maxBytesToLog=16
19
20 a1.sources.r1.channels=c1
21 a1.sinks.s1.channel=c1
```

4.5.7.2 启动agent

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./syslogtcp-logger.conf -n a1 -Dflume.root.logger=INFO,console
```

4.5.7.3 测试：需要先安装nc

```
[root@qianfeng01 ~]# yum -y install nmap-ncat
[root@qianfeng01 ~]# echo "hello world" | nc qianfeng01 6666
```

4.5.8 案例演示：Syslogtcp+mem+hdfs

> Syslogtcp:同上
>
> mem:表示从内存传送
>
> hdfs:表示目的地为HDFS服务器

4.5.8.1 配置方案

```
[root@qianfeng01 flumeconf]# vi syslogtcp-hdfs.conf
a1.sources = r1
a1.channels = c1
a1.sinks = s1

a1.sources.r1.type=syslogtcp
a1.sources.r1.host = qianfeng01
a1.sources.r1.port = 6666

a1.channels.c1.type=memory
a1.channels.c1.capacity=1000
a1.channels.c1.transactionCapacity=100
a1.channels.c1.keep-alive=3
a1.channels.c1.byteCapacityBufferPercentage=20
a1.channels.c1.byteCapacity=800000

a1.sinks.s1.type=hdfs
a1.sinks.s1.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/%H%M
a1.sinks.s1.hdfs.filePrefix=flume-hdfs
a1.sinks.s1.hdfs.fileSuffix=.log
a1.sinks.s1.hdfs.inUseSuffix=.tmp
a1.sinks.s1.hdfs.rollInterval=60
a1.sinks.s1.hdfs.rollSize=1024
a1.sinks.s1.hdfs.rollCount=10
a1.sinks.s1.hdfs.idleTimeout=0
a1.sinks.s1.hdfs.batchSize=100
a1.sinks.s1.hdfs.fileType=DataStream
a1.sinks.s1.hdfs.writeFormat=Text
a1.sinks.s1.hdfs.round=true
a1.sinks.s1.hdfs.roundValue=1
a1.sinks.s1.hdfs.roundUnit=second
a1.sinks.s1.hdfs.useLocalTimeStamp=true

a1.sources.r1.channels=c1
```

```
a1.sinks.s1.channel=c1
```

4.5.8.2 启动Agent的服务

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./syslogtcp-hdfs.conf -n a1 -Dflume.root.logger=INFO,console
```

4.5.8.3 测试

```
[root@qianfeng01 ~]# echo "hello world hello qianfeng" | nc qianfeng01 6666
```

4.5.9 案例演示：Syslogtcp+file+hdfs

> syslogtcp 同上
>
> file:表示用file作为channel传送介质
>
> hdfs:表示目的地为HDFS

4.5.9.1 配置方案

```
[root@qianfeng01 flumeconf]# vi syslogtcp-fh.conf
a1.sources = r1
a1.channels = c1
a1.sinks = s1

a1.sources.r1.type=syslogtcp
a1.sources.r1.host = qianfeng01
a1.sources.r1.port = 6666

a1.channels.c1.type=file
a1.channels.c1.dataDirs=/root/flumedata/filechannel/data
a1.channels.c1.checkpointDir=/root/flumedata/filechannel/point
a1.channels.c1.transactionCapacity=10000
a1.channels.c1.checkpointInterval=30000
a1.channels.c1.capacity=1000000
a1.channels.c1.keep-alive=3

a1.sinks.s1.type=hdfs
a1.sinks.s1.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/%H%M
a1.sinks.s1.hdfs.filePrefix=flume-hdfs
a1.sinks.s1.hdfs.fileSuffix=.log
a1.sinks.s1.hdfs.inUseSuffix=.tmp
a1.sinks.s1.hdfs.rollInterval=60
a1.sinks.s1.hdfs.rollSize=1024
a1.sinks.s1.hdfs.rollCount=10
a1.sinks.s1.hdfs.idleTimeout=0
a1.sinks.s1.hdfs.batchSize=100
a1.sinks.s1.hdfs.fileType=DataStream
a1.sinks.s1.hdfs.writeFormat=Text
a1.sinks.s1.hdfs.round=true
```

```
a1.sinks.s1.hdfs.roundValue=1
a1.sinks.s1.hdfs.roundUnit=second
a1.sinks.s1.hdfs.useLocalTimeStamp=true

a1.sources.r1.channels=c1
a1.sinks.s1.channel=c1
```

4.5.9.2 启动Agent的服务

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./syslogtcp-fh.conf -n a1 -Dflume.root.logger=INFO,console
```

4.5.9.3 测试

```
[root@qianfeng01 ~]# echo "hello world hello qianfeng" | nc qianfeng01 6666
```

4.6 拦截器的使用

在Flume运行过程中 ,Flume有能力在运行阶段修改/删除Event，这是通过拦截器（Interceptors）来实现的。拦截器有下面几个特点:

- 拦截器需要实现org.apache.flume.interceptor.Interceptor接口。
- 拦截器可以修改或删除事件基于开发者在选择器中选择的任何条件。
- 拦截器采用了责任链模式，多个拦截器可以按指定顺序拦截。
- 一个拦截器返回的事件列表被传递给链中的下一个拦截器。
- 如果一个拦截器需要删除事件，它只需要在返回的事件集中不包含要删除的事件即可。

4.6.1 常用拦截器

1. Timestamp Interceptor :时间戳拦截器，将当前时间戳（毫秒）加入到events header中，key名字为：timestamp，值为当前时间戳。用的不是很多
2. Host Interceptor:主机名拦截器。将运行Flume agent的主机名或者IP地址加入到events header中，key名字为：host（也可自定义）
3. Static Interceptor:静态拦截器，用于在events header中加入一组静态的key和value。

4.6.2 案例演示：Timestamp+Syslogtcp+file+hdfs

> 通过时间拦截器,数据源为SyslogTcp,传送的通道模式是FileChannel,最后输出的目的地为HDFS

4.6.2.1 配置方案

```
[root@qianfeng01 flumeconf]# vi ts.conf
a1.sources = r1
a1.channels = c1
a1.sinks = s1

a1.sources.r1.type=syslogtcp
a1.sources.r1.host = qianfeng01
a1.sources.r1.port = 6666
a1.sources.r1.interceptors=i1 i2 i3
```

```
a1.sources.r1.interceptors.i1.type=timestamp
#如果拦截器中已经有了时间戳,直接替换成现在的
a1.sources.r1.interceptors.i1.preserveExisting=false
a1.sources.r1.interceptors.i2.type=host
a1.sources.r1.interceptors.i2.preserveExisting=false
a1.sources.r1.interceptors.i2.useIP=true
a1.sources.r1.interceptors.i2.hostHeader=hostname
a1.sources.r1.interceptors.i3.type=static
a1.sources.r1.interceptors.i3.preserveExisting=false
a1.sources.r1.interceptors.i3.key=hn
a1.sources.r1.interceptors.i3.value=qianfeng01

a1.channels.c1.type=memory
a1.channels.c1.capacity=1000
a1.channels.c1.transactionCapacity=100
a1.channels.c1.keep-alive=3
a1.channels.c1.byteCapacityBufferPercentage=20
a1.channels.c1.byteCapacity=800000

a1.sinks.s1.type=hdfs
a1.sinks.s1.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/%H%M
a1.sinks.s1.hdfs.filePrefix=%{hostname}
a1.sinks.s1.hdfs.fileSuffix=.log
a1.sinks.s1.hdfs.inUseSuffix=.tmp
a1.sinks.s1.hdfs.rollInterval=60
a1.sinks.s1.hdfs.rollSize=1024
a1.sinks.s1.hdfs.rollCount=10
a1.sinks.s1.hdfs.idleTimeout=0
a1.sinks.s1.hdfs.batchSize=100
a1.sinks.s1.hdfs.fileType=DataStream
a1.sinks.s1.hdfs.writeFormat=Text
a1.sinks.s1.hdfs.round=true
a1.sinks.s1.hdfs.roundValue=1
a1.sinks.s1.hdfs.roundUnit=second
a1.sinks.s1.hdfs.useLocalTimeStamp=true

a1.sources.r1.channels=c1
a1.sinks.s1.channel=c1
```

4.6.2.2 启动Agent的服务

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./ts.conf -n a1 -
Dflume.root.logger=INFO,console
```

4.6.2.3 测试

```
[root@qianfeng01 ~]# echo "hello world hello qianfeng" | nc qianfeng01 6666
```

4.6.3 案例演示：regex+Syslogtcp+file+hdfs

> 拦截器为正则表达式拦截器, 数据源为Syslogtcp格式,传送通道为FileChannel,最后传送的目的地是HDFS

4.6.3.1 配置方案

```
[root@qianfeng01 flumeconf]# vi regex-ts.conf
a1.sources = r1
a1.channels = c1
a1.sinks = s1

a1.sources.r1.type=syslogtcp
a1.sources.r1.host = qianfeng01
a1.sources.r1.port = 6666
a1.sources.r1.interceptors=i1
a1.sources.r1.interceptors.i1.type=regex_filter
#不要加引号包裹正则
a1.sources.r1.interceptors.i1.regex=^[0-9].*$
a1.sources.r1.interceptors.i1.excludeEvents=false


a1.channels.c1.type=memory
a1.channels.c1.capacity=1000
a1.channels.c1.transactionCapacity=100
a1.channels.c1.keep-alive=3
a1.channels.c1.byteCapacityBufferPercentage=20
a1.channels.c1.byteCapacity=800000

a1.sinks.s1.type=hdfs
a1.sinks.s1.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/%H%M
a1.sinks.s1.hdfs.filePrefix=%{hostname}
a1.sinks.s1.hdfs.fileSuffix=.log
a1.sinks.s1.hdfs.inUseSuffix=.tmp
a1.sinks.s1.hdfs.rollInterval=60
a1.sinks.s1.hdfs.rollSize=1024
a1.sinks.s1.hdfs.rollCount=10
a1.sinks.s1.hdfs.idleTimeout=0
a1.sinks.s1.hdfs.batchSize=100
a1.sinks.s1.hdfs.fileType=DataStream
a1.sinks.s1.hdfs.writeFormat=Text
a1.sinks.s1.hdfs.round=true
a1.sinks.s1.hdfs.roundValue=1
a1.sinks.s1.hdfs.roundUnit=second
a1.sinks.s1.hdfs.useLocalTimeStamp=true

a1.sources.r1.channels=c1
a1.sinks.s1.channel=c1
```

4.6.3.2 启动agent的服务

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./regex-ts.conf -n a1 -
Dflume.root.logger=INFO,console
```

4.6.3.3 测试

```
[root@qianfeng01 ~]# echo "hello world hello qianfeng" | nc qianfeng01 6666
[root@qianfeng01 ~]# echo "123123123 hello world hello qiangeng" | nc qianfeng01 6666
```

4.6.4 自定义拦截器

4.6.4.0 需求

```
为了提高Flume的扩展性,用户可以自己定义一个拦截器，对每一组的item_type和active_time都过滤出相应的HOST和USERID
```

4.6.4.1 pom.xml

可以参考`/code/pom.xml`

```xml
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.apache.flume/flume-ng-core -->
    <dependency>
        <groupId>org.apache.flume</groupId>
        <artifactId>flume-ng-core</artifactId>
        <version>1.9.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
        <version>1.2.48</version>
    </dependency>
</dependencies>
```

4.6.4.2 代码

具体代码可以参考`code/MyInterceptor`

```
/**
 * @Author 千锋大数据教学团队
 * @Company 千锋好程序员大数据
 * @Description  自定义拦截器:对每一组的item_type和active_time都过滤出相应的HOST和USERID
 */
public class MyInterceptor implements Interceptor {
    @Override
    public void initialize() {
        //初始化方法,写拦截器初始化时的业务
    }

    @Override
    public void close() {
        //关闭方法,写拦截器关闭时的代码

    }

```

```java
    /**
     * 解析单条event
     * @param event
     * @return
     */
    @Override
    public Event intercept(Event event) {
        //输入
        String inputeBody=null;
        //输出
        byte[] outputBoday=null;
        //解析---这里定义对单条Event处理规则
        try {
            inputeBody=new String(event.getBody(), Charsets.UTF_8);
            ArrayList<String> temp = new ArrayList<>();

            JSONObject bodyObj = JSON.parseObject(inputeBody);

            //1)公共字段
            String host = bodyObj.getString("host");
            String user_id = bodyObj.getString("user_id");
            JSONArray data = bodyObj.getJSONArray("items");

            //2)Json数组=>every json obj
            for (Object item : data) {
                JSONObject itemObj = JSON.parseObject(item.toString());
                HashMap<String, Object> fields = new HashMap<>();
                fields.put("host",host);
                fields.put("user_id",user_id);
                fields.put("item_type",itemObj.getString("item_type"));
                fields.put("active_time",itemObj.getLongValue("active_time"));
                temp.add(new JSONObject(fields).toJSONString());
            }
            //3)Json obj 拼接
            outputBoday=String.join("\n",temp).getBytes();
        }catch (Exception e){
            System.out.println("输入数据:"+inputeBody);
            e.printStackTrace();
        }
        event.setBody(outputBoday);
        return event;
    }

    /**
     * 解析一批event
     * @param events
     * @return
     */
    @Override
    public List<Event> intercept(List<Event> events) {
        //输出---一批Event
        ArrayList<Event> result = new ArrayList<>();
        //输入---一批Event
```

```
        try{
            for (Event event : events) {
                //一条条解析
                Event interceptedEvent = intercept(event);
                byte[] interceptedEventBody = interceptedEvent.getBody();
                if(interceptedEventBody.length!=0){
                    String multiEvent = new String(interceptedEventBody, Charsets.UTF_8);
                    String[] multiEventArr = multiEvent.split("\n");
                    for (String needEvent : multiEventArr) {
                        SimpleEvent simpleEvent = new SimpleEvent();
                        simpleEvent.setBody(needEvent.getBytes());
                        result.add(simpleEvent);
                    }

                }
            }

        }catch (Exception e){
            e.printStackTrace();
        }
        return result;
    }



    /**
     * 实现内部类接口
     */
    public static class Builder implements Interceptor.Builder{

        @Override
        public Interceptor build() {
            return new MyInterceptor();
        }

        @Override
        public void configure(Context context) {

        }
    }

}
```

4.6.4.3 打包上传

```
使用maven将拦截器打包,然后把此包和依赖的fastjson一起上传到Flume lib目录下
```

4.6.4.4 编写方案

```
[root@qianfeng01 flumeconf]# vi mytest.conf
a1.sources = s1
```

```
a1.channels = c1
a1.sinks = r1

a1.sources.s1.type = TAILDIR
#文件以JSON格式记录inode、绝对路径和每个跟踪文件的最后位置
a1.sources.s1.positionFile = /root/flume/taildir_position.json
#以空格分隔的文件组列表。每个文件组表示要跟踪的一组文件
a1.sources.s1.filegroups = f1
#文件组的绝对路径
a1.sources.s1.filegroups.f1=/root/flume/data/.*log
#是否添加存储绝对路径文件名的标题
a1.sources.s1.fileHeader = true
#使用自定义拦截器
a1.sources.s1.interceptors = i1
a1.sources.s1.interceptors.i1.type = flume.MyInterceptor$Builder

a1.channels.c1.type = file
a1.channels.c1.dataDirs = /root/flume/filechannle/dataDirs
a1.channels.c1.checkpointDir = /root/flume/filechannle/checkpointDir
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100


a1.sinks.r1.type = hdfs
a1.sinks.r1.hdfs.path = hdfs://qianfeng01:8020/flume/spooldir
a1.sinks.r1.hdfs.filePrefix =
a1.sinks.r1.hdfs.round = true
a1.sinks.r1.hdfs.roundValue = 10
a1.sinks.r1.hdfs.roundUnit = minute
a1.sinks.r1.hdfs.fileSuffix= .log
a1.sinks.r1.hdfs.rollInterval=60
a1.sinks.r1.hdfs.fileType=DataStream
a1.sinks.r1.hdfs.writeFormat=Text


a1.sources.s1.channels = c1
a1.sinks.r1.channel = c1
```

4.6.4.5 启动agent

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf/ -f ./mytest.conf -n a1 -
Dflume.root.logger=INFO,console
```

4.6.4.6 测试

```
[root@qianfeng01 ~]# vi my.sh
#!/bin/bash
log='{
"host":"www.baidu.com",
"user_id":"13755569427",
"items":[
```

```
    {
        "item_type":"eat",
        "active_time":156234
    },
    {
        "item_type":"car",
        "active_time":156233
    }
 ]
}'
echo $log>> /root/flume/data/test.log

[root@qianfeng01 ~]# bash my.sh

执行后我们希望得到是数据格式:
{"active_time":156234,"user_id":"13755569427","item_type":"eat","host":"www.baidu.com"}
{"active_time":156233,"user_id":"13755569427","item_type":"car","host":"www.baidu.com"}
```

4.7 选择器的使用

4.7.1 说明

Flume中的Channel选择器作用于source阶段,是决定Source接受的特定事件写入到哪个Channel的组件，他们告诉Channel处理器，然后由其将事件写入到Channel。

Agent中各个组件的交互 由于Flume不是两阶段提交，事件被写入到一个Channel，然后事件在写入下一个Channel之前提交，如果写入一个Channel出现异常，那么之前已经写入到其他Channel的相同事件不能被回滚。当这样的异常发生时，Channel处理器抛出ChannelException异常，事务失败，如果Source试图再次写入相同的事件（大多数情况下，会再次写入，只有Syslog，Exec等Source不能重试，因为没有办法生成相同的数据），重复的事件将写入到Channel中，而先前的提交是成功的，这样在Flume中就发生了重复。

Channel选择器的配置是通过Channel处理器完成的，Channel选择器可以指定一组Channel是必须的，另一组的可选的。

Flume分类两种选择器，如果Source配置中没有指定选择器，那么会自动使用复制Channel选择器.

- **replicating**:该选择器复制每个事件到通过Source的Channels参数指定的所有Channel中。
- **multiplexing**:是一种专门用于动态路由事件的Channel选择器，通过选择事件应该写入到哪个Channel，基于一个特定的事件头的值进行路由

4.7.2 案例演示：replicating selector

4.7.2.1 配置方案

```
[root@qianfeng01 flumeconf]# vi rep.conf
a1.sources = r1
a1.channels = c1 c2
a1.sinks = s1 s2

a1.sources.r1.type=syslogtcp
a1.sources.r1.host = qianfeng01
```

```
a1.sources.r1.port = 6666
a1.sources.r1.selector.type=replicating


a1.channels.c1.type=memory
a1.channels.c1.capacity=1000
a1.channels.c1.transactionCapacity=100
a1.channels.c1.keep-alive=3
a1.channels.c1.byteCapacityBufferPercentage=20
a1.channels.c1.byteCapacity=800000

a1.channels.c2.type=memory
a1.channels.c2.capacity=1000
a1.channels.c2.transactionCapacity=100


a1.sinks.s1.type=hdfs
a1.sinks.s1.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/rep
a1.sinks.s1.hdfs.filePrefix=s1sink
a1.sinks.s1.hdfs.fileSuffix=.log
a1.sinks.s1.hdfs.inUseSuffix=.tmp
a1.sinks.s1.hdfs.rollInterval=60
a1.sinks.s1.hdfs.rollSize=1024
a1.sinks.s1.hdfs.rollCount=10
a1.sinks.s1.hdfs.idleTimeout=0
a1.sinks.s1.hdfs.batchSize=100
a1.sinks.s1.hdfs.fileType=DataStream
a1.sinks.s1.hdfs.writeFormat=Text
a1.sinks.s1.hdfs.round=true
a1.sinks.s1.hdfs.roundValue=1
a1.sinks.s1.hdfs.roundUnit=second
a1.sinks.s1.hdfs.useLocalTimeStamp=true

a1.sinks.s2.type=hdfs
a1.sinks.s2.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/rep
a1.sinks.s2.hdfs.filePrefix=s2sink
a1.sinks.s2.hdfs.fileSuffix=.log
a1.sinks.s2.hdfs.inUseSuffix=.tmp
a1.sinks.s2.hdfs.rollInterval=60
a1.sinks.s2.hdfs.rollSize=1024
a1.sinks.s2.hdfs.rollCount=10
a1.sinks.s2.hdfs.idleTimeout=0
a1.sinks.s2.hdfs.batchSize=100
a1.sinks.s2.hdfs.fileType=DataStream
a1.sinks.s2.hdfs.writeFormat=Text
a1.sinks.s2.hdfs.round=true
a1.sinks.s2.hdfs.roundValue=1
a1.sinks.s2.hdfs.roundUnit=second
a1.sinks.s2.hdfs.useLocalTimeStamp=true

a1.sources.r1.channels=c1 c2
a1.sinks.s1.channel=c1
a1.sinks.s2.channel=c2
```

4.7.2.2 启动agent的服务

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./rep.conf -n a1 -
Dflume.root.logger=INFO,console
```

4.7.2.3 测试

```
[root@qianfeng01 ~]# echo "hello world hello qianfeng" | nc qianfeng01 6666
```

4.7.3 案例演示：Multiplexing selector

4.7.3.1 配置方案

```
[root@qianfeng01 flumeconf]# vi mul.conf
a1.sources = r1
a1.channels = c1 c2
a1.sinks = s1 s2

a1.sources.r1.type=http
a1.sources.r1.bind = qianfeng01
a1.sources.r1.port = 6666

a1.sources.r1.selector.type=multiplexing
a1.sources.r1.selector.header = state
a1.sources.r1.selector.mapping.USER = c1
a1.sources.r1.selector.mapping.ORDER = c2
a1.sources.r1.selector.default = c1

a1.channels.c1.type=memory
a1.channels.c1.capacity=1000
a1.channels.c1.transactionCapacity=100
a1.channels.c1.keep-alive=3
a1.channels.c1.byteCapacityBufferPercentage=20
a1.channels.c1.byteCapacity=800000

a1.channels.c2.type=memory
a1.channels.c2.capacity=1000
a1.channels.c2.transactionCapacity=100


a1.sinks.s1.type=hdfs
a1.sinks.s1.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/mul
a1.sinks.s1.hdfs.filePrefix=s1sink
a1.sinks.s1.hdfs.fileSuffix=.log
a1.sinks.s1.hdfs.inUseSuffix=.tmp
a1.sinks.s1.hdfs.rollInterval=60
a1.sinks.s1.hdfs.rollSize=1024
a1.sinks.s1.hdfs.rollCount=10
a1.sinks.s1.hdfs.idleTimeout=0
a1.sinks.s1.hdfs.batchSize=100
a1.sinks.s1.hdfs.fileType=DataStream
```

```
a1.sinks.s1.hdfs.writeFormat=Text
a1.sinks.s1.hdfs.round=true
a1.sinks.s1.hdfs.roundValue=1
a1.sinks.s1.hdfs.roundUnit=second
a1.sinks.s1.hdfs.useLocalTimeStamp=true

a1.sinks.s2.type=hdfs
a1.sinks.s2.hdfs.path=hdfs://qianfeng01:8020/flume/%Y/%m/%d/mul
a1.sinks.s2.hdfs.filePrefix=s2sink
a1.sinks.s2.hdfs.fileSuffix=.log
a1.sinks.s2.hdfs.inUseSuffix=.tmp
a1.sinks.s2.hdfs.rollInterval=60
a1.sinks.s2.hdfs.rollSize=1024
a1.sinks.s2.hdfs.rollCount=10
a1.sinks.s2.hdfs.idleTimeout=0
a1.sinks.s2.hdfs.batchSize=100
a1.sinks.s2.hdfs.fileType=DataStream
a1.sinks.s2.hdfs.writeFormat=Text
a1.sinks.s2.hdfs.round=true
a1.sinks.s2.hdfs.roundValue=1
a1.sinks.s2.hdfs.roundUnit=second
a1.sinks.s2.hdfs.useLocalTimeStamp=true

a1.sources.r1.channels=c1 c2
a1.sinks.s1.channel=c1
a1.sinks.s2.channel=c2
```

4.7.3.2 启动Agent的服务

```
[root@qianfeng01 flumeconf]# flume-ng agent -c ../conf -f ./mul.conf -n a1 -
Dflume.root.logger=INFO,console
```

4.7.3.3 测试

```
[root@qianfeng01 ~]# curl -X POST -d '[{"headers":{"state":"ORDER"},"body":"this my multiplex
to c2"}]' http://qianfeng01:6666
[root@qianfeng01 ~]# curl -X POST -d '[{"headers":{"state":"ORDER"},"body":"this is my
content"}]' http://qianfeng01:6666
```

4.7.4 Flume的集群

其实slave配置差不多。

配置192.168.10.101：

```
a1.sources=r1
a1.channels=c1
a1.sinks=s1

a1.sources.r1.type=syslogtcp
a1.sources.r1.host=192.168.10.101
```

```
a1.sources.r1.port=6666

a1.channels.c1.type=memory

a1.sinks.s1.type=avro
a1.sinks.s1.hostname=192.168.10.103
a1.sinks.s1.port=6666

a1.sources.r1.channels=c1
a1.sinks.s1.channel=c1
```

配置192.168.10.102：

```
a1.sources=r1
a1.channels=c1
a1.sinks=s1

a1.sources.r1.type=http
a1.sources.r1.bind=192.168.10.102
a1.sources.r1.port=6666

a1.channels.c1.type=memory

a1.sinks.s1.type=avro
a1.sinks.s1.hostname=192.168.10.103
a1.sinks.s1.port=6666

a1.sources.r1.channels=c1
a1.sinks.s1.channel=c1
```

配置192.168.10.103：

```
a1.sources=r1
a1.channels=c1
a1.sinks=s1

a1.sources.r1.type=avro
a1.sources.r1.bind=192.168.10.103
a1.sources.r1.port=6666

a1.channels.c1.type=memory

a1.sinks.s1.type=logger

a1.sources.r1.channels=c1
a1.sinks.s1.channel=c1
```

启动qianfeng01：

```
flume-ng agent -c ./conf -f ./conf/qianfeng01.conf -n a1 -Dflume.root.logger=INFO,console
```

再启动slave：

```
flume-ng agent -c ./conf -f ./conf/qianfeng02.conf -n a1 -Dflume.root.logger=INFO,console
flume-ng agent -c ./conf -f ./conf/qianfeng03.conf -n a1 -Dflume.root.logger=INFO,console
```

测试：

```
echo "i am slave1" | nc qianfeng01 6666
curl -X POST -d '[{"headers":{"param1":"kkk"},"body":"i am slave2"}]' http://qianfeng01:6666
```

五 实战应用

六 教学总结

6.1 教学重点

```
Flume配置
Flume Agant结构
Flume Source
Flume Channel
Flume Sink
拦截器
```

七 课后作业

八 解决方案

8.1 应用场景

8.2 核心面试题