Flume：最初设计目的是为了把数据传入hdfs，并不是为了采集数据而设计

Logstash:侧重于数据的预处理，因日志的字段需要大量的预处理，为解析做铺垫

nginx的日志文件没有rotate功能。如果你不处理，日志文件将变得越来越大，还好我们可以写一个nginx日志切割脚本来自动切割日志文件。  
 第一步就是重命名日志文件，不用担心重命名后nginx找不到日志文件而丢失日志。在你未重新打开原名字的日志文件前，nginx还是会向你重命名的文件写日志，linux是靠文件描述符而不是文件名定位文件。  
 第二步向nginx主进程发送USR1信号。  
 nginx主进程接到信号后会从配置文件中读取日志文件名称，重新打开日志文件(以配置文件中的日志名称命名)，并以工作进程的用户作为日志文件的所有者。  
 重新打开日志文件后，nginx主进程会关闭重名的日志文件并通知工作进程使用新打开的日志文件。  
 工作进程立刻打开新的日志文件并关闭重名名的日志文件。  
 然后你就可以处理旧的日志文件了

**Nginx\_log\_division.sh代码：**

 Shell代码

#!/bin/bash

#设置日志文件存放目录

logs\_path="/usr/local/nginx/nginxlog/"

#设置pid文件

pid\_path="/usr/local/nginx/nginx-1.7.3/logs/nginx.pid"

#日志文件

filepath=${logs\_path}"access.log"

# Source function library.

#重命名日志文件

mv ${logs\_path}access.log ${logs\_path}access\_$(date -d '-1 day' '+%Y-%m-%d').log

#向nginx主进程发信号重新打开日志

kill -USR1 `cat ${pid\_path}`

**Oracle\_log\_division.sh代码：**

 Shell代码

#!/bin/bash

#设置日志文件存放目录

logs\_path="/usr/local/nginx/nginxlog/"

#设置pid文件

pid\_path="/usr/local/nginx/nginx-1.7.3/logs/nginx.pid"

#日志文件

filepath=${logs\_path}"access.log"

# Source function library.

#重命名日志文件

mv ${logs\_path}access.log ${logs\_path}access\_$(date -d '-1 day' '+%Y-%m-%d').log

#向nginx主进程发信号重新打开日志

kill -USR1 `cat ${pid\_path}`

**flumeng配置：**

Xml代码  [IMG_256](https://www.cnblogs.com/tonychai/p/javascript:void())

# A single-node Flume configuration

# Name the components on this agent

agent1.sources = source1

agent1.sinks = sink1

agent1.channels = channel1

# Describe/configure source1

agent1.sources.source1.type = exec

agent1.sources.source1.command = tail -n +0 -F /logs/access.log

agent1.sources.source1.channels = channel1

# Describe sink1

agent1.sinks.sink1.type = file\_roll

agent1.sinks.sink1.sink.directory=/var/log/data

# Use a channel which buffers events in memory

agent1.channels.channel1.type = file

agent1.channels.channel1.checkpointDir=/var/checkpoint

agent1.channels.channel1.dataDirs=/var/tmp

agent1.channels.channel1.capacity = 1000

agent1.channels.channel1.transactionCapactiy = 100

# Bind the source and sink to the channel

agent1.sources.source1.channels = channel1

agent1.sinks.sink1.channel = channel1

flume-ng agent -n a1 -c conf -f conf/example.file -Dflume.root.logger=DEBUG,console

参数说明：

-n 指定agent名称(与配置文件中代理的名字相同)

-c 指定flume中配置文件的目录

-f 指定配置文件

-Dflume.root.logger=DEBUG,console 设置日志等级

agent1.channels.channel1.transactionCapactiy = 100 sink端收集到100条数据后提交事务

Sink.batchsize sink端一次性从channel取的数据量(sink和channel使用一个双向队列维护事务，进行回滚。LinkedBlockingQuene)

og与ng的区别：

og是单线程  主从结构，原本由zookeeper管理

ng是双线程    已经取消master管理机制和zookeeper管理机制，变成了纯粹的传输工具

* source：负责日志的流入，eg：从文件、网络、kafka等数据源流入数据。   
  数据的流入有两种方式：
  + 轮询拉去
  + 事件驱动
* channel:数据聚合/暂存，暂存到内存、本地文件、数据库、kafka等，日志数据不会在管道停留很长的时间，很快被sink消费掉
* sink：负责数据转移存储，从channel中拿到日志，直接存储到HDFS、Hbase、kafka、ElasticSearch等

source和sink是异步的。

# flumeSource

Source支持Avro、Exec、Http、Spooling Director、Kafka、Syslog

* Exec Source 以运行linux命令的方式，持续的输出最新的数据
* Spool Source 支持监视一个目录或文件，解析其中新生成的事件，并将文件中的数据读取出来   
  拷贝到spool目录下的文件不可以再打开编辑，spool目录不可包含相应的子目录

Exec source和Spool Source的比较：

1. Exec source可以实现对日志的实时收集，但是存在Flume不运行或指令出错时将无法手机到日志数据，无法保证日志数据的完整性。
2. Spool Source 虽然无法实现实时的收集数据，但是可以以分钟的方式切割文件，趋近于实时。
3. 如果应用无法以分钟分割日志文件，可以两种收集方式同时使用。

Flume 过滤器（拦截器）：在App（应用程序日志）和source之间，对App日志进行拦截处理，官方已有的拦截器有：

* Timestamp Interceptor：在event的header中添加一个key作为timestamp，value为当前的事件戳。
* Host Interceprot：在event的header中添加一个的key为host，value为当前机器的hostname或IP
* Static Interceptor：在event的header中添加自动以的key和value
* Regex Filtering Interceptor：通过正则来清晰或包含匹配的events
* Regex Extrator Inteneptor：通过正则来在header中添加指定的key，value则为正则匹配的部分。

# Flume Channel

缓存从Source 到Sink的中间数据，可用下列配置作为channel：

* 内存
* 文件
* JDBC
* kafka

Flume Channel选择器：数据通过不同的Channel路由到不同的Sink中

* Replication Channel Selectot：将source过来的events发往所有的Channel
* Multiplexing Channel Selectot：可以选择将数据发往哪些Channel

# Flume Sink

负责从Channel中读取数据并发送给下一个Agent或最终目的地。   
目的地：HDFS、Hbase、Avro、ElasticSearch、File、Logger

Flume 序列化器

* Body Text Serializer：忽略event header ，只将event value写到输出流。
* Avro Event Serializer：数据以Avro序列化，然后写到输出流。

fileHeader = true

# 或者

basenameHeader = true会将文件名加在数据之前

# kafka Sink 配置坑

AGENT\_NAME.sinks.malwrK.kafka.bootstrap.servers= serv1:PORT[,serv2:PORT,serv3:PORT...]

这项配置中，如果只指定单个的 Kafka 服务器地址，Flume Agent 会正常运行，但如果把 Kafka集群里的多个服务器地址都写上，并且Kafka的默认配置不修改，则会报以下错误：

WARN Error while fetching metadata with correlation id 39 : {4-3-16-topic1=LEADER\_NOT\_AVAILABLE}

并且数据无法正常发送到 Kafka 集群。该错误的意思是集群中的服务器，没有一个可用的 LEADER，导致数据无法正常 Fetch。那么解决方法也很简单，在 Kafka 集群的每个服务器 server.properties 配置文件中，开启（取消注释）以下配置，并把配置项的值改成统一指定的一台 Kafka 服务器做 LEADER：

|  |  |
| --- | --- |
| 1 | advertised.listeners=PLAINTEXT://your.host.name:9092 |

# http json source 配置坑

Flume 的 Event 只有两部分组成：headers 和 body，所以传输的 JSON Array 里的 Flume Event，每个 Event 也只可以包括 headers 和 body 两部分，而且 headers的值是 JSON 对象（object），而body的值只能是 JSON 字符串（string），不能是其他 JSON 值类型，比如下图所示的 JSON 形式，对 Flume 来说算是合法的

IMG_256

最后总结一下，传给 flume HTTP Source 的 JSON 数据：

必须是 Array 形式（用方括号括起来），即使 Array 里只有 1 个元素（每个元素到了 Flume 那儿就代表一个 Event），在 Python 里就要用 dict 外面嵌套 list 或者 tuple 生成 JSON Array；

JSON Array 里每个元素必须由 headers 和 body 两个 field 构成;

JSON Array 中元素的 headers 的值是 JSON Object(具体可以自定义)，body 的值必须是字串（JSON String），而不能是其他形式（JSON Object/JSON Array等，对应 Python 中的 dict/list/tuple 等）

**TAILDIR**flume 1.7目前最新版新增类型，支持目录变化的文件，如遇中断，并以json数据记录目录下的每个文件的收集状态.

flume 1.7.0 推出了 taildirSource 组件。tail 监控 目录下匹配上正则表达式的 的所有文件，实现断点续传。

# ****问题****

但是官方的 taildirSource 组件貌似是支持nginx 日志，对于log4j 日志好像不太管用。   
 因为log4j 日志会自动切分，log4j 切分日志其实就是新建一个文件，然后把原来的日志文件都改名。但是 taildirSource 组件时不支持文件改名的。如果文件改名会认为是新文件，就会重新读取，这就导致了日志文件重读。

# ****解决****

读了 taildirSource 的源码,发现了两个地方 需要修改

# ****1.修改 ReliableTaildirEventReader****

修改ReliableTaildirEventReader 类的 updateTailFiles 方法。

将其中的 tf.getPath().equals(f.getAbsolutePath()) 判断条件去除。   
 只用判断文件不为空即可，不用判断文件的名字，因为log4j 日志切分文件被重命名了。

// if (tf == null || !tf.getPath().equals(f.getAbsolutePath())) {

if (tf == null) {//文件不存在 position 中则全读。

# ****2.修改TailFile****

修改TailFile 类的 updatePos 方法

此处同样的原因，inode 已经能够确定唯一的 文件了，所以不用加 path 作为判定条件了。所以去掉该条件就支持了文件重命名情况。

// if (this.inode == inode && this.path.equals(path))

if (this.inode == inode)

修改这两个地方就支持了文件重命名 的问题，实现了目录下多文件监控，断点续传。

添加自定义source入口，也就是将源码拷贝过来，然后将修改过的代码打包为自定义source的jar 包运行flume 。

# ****3.配置文件****

taildirsource 的配置文件如下

a1.sources = r1

a1.sources.r1.type = com.xx.xx.source.taildir.TaildirSource

a1.sources.r1.channels = c1

a1.sources.r1.positionFile = /opt/apps/log4j/taildir\_position.json

a1.sources.r1.filegroups = f1

a1.sources.r1.filegroups.f1 = /opt/apps/log4j/logs/\*.log.\*

a1.sources.r1.fileHeader = true

通过这些修改就 实现了多文件监控和断点续传。支持log4j 日志监控。

# ****4. 监控多个文件夹下的多个文件****

监控多个文件夹下的多个文件，只用 配置多个filegroups 即可。

a1.sources = r1

a1.sources.r1.type = com.xx.xx.source.taildir.TaildirSource

a1.sources.r1.channels = c1

a1.sources.r1.positionFile = /opt/apps/log4j/taildir\_position.json

a1.sources.r1.filegroups = f1 f2

a1.sources.r1.filegroups.f1=/opt/apps/log4j/test\*.log

a1.sources.r1.filegroups.f2=/opt/apps/log4j2/test\*.log

a1.sources.r1.fileHeader = true

**通道复用：** 因为复用的流的事件要声明一个头部，然后我们检查头部对应的值

1. #配置文件：multi\_source\_case12.conf
2. a1.sources= r1
3. a1.sinks= k1 k2
4. a1.channels= c1 c2
6. #Describe/configure the source
7. a1.sources.r1.type= org.apache.flume.source.http.HTTPSource
8. a1.sources.r1.port= 50000
9. a1.sources.r1.host= 192.168.233.128
10. a1.sources.r1.selector.type= multiplexing
11. a1.sources.r1.channels= c1 c2
13. a1.sources.r1.selector.header= state
14. a1.sources.r1.selector.mapping.CZ= c1
15. a1.sources.r1.selector.mapping.US= c2
16. a1.sources.r1.selector.default= c1
18. #Describe the sink
19. a1.sinks.k1.type= avro
20. a1.sinks.k1.channel= c1
21. a1.sinks.k1.hostname= 192.168.233.129
22. a1.sinks.k1.port= 50000
24. a1.sinks.k2.type= avro
25. a1.sinks.k2.channel= c2
26. a1.sinks.k2.hostname= 192.168.233.130
27. a1.sinks.k2.port= 50000
28. # Usea channel which buffers events in memory
29. a1.channels.c1.type= memory
30. a1.channels.c1.capacity= 1000
31. a1.channels.c1.transactionCapacity= 100
33. a1.channels.c2.type= memory
34. a1.channels.c2.capacity= 1000
35. a1.channels.c2.transactionCapacity= 100

打开另一个终端输入，往侦听端口送数据

curl -X POST -d '[{"headers" :{"state" : "CZ"},"body" :"TEST1"}]' http://192.168.233.128:50000

curl -X POST -d '[{"headers" :{"state" : "US"},"body" :"TEST2"}]' http://192.168.233.128:50000

curl -X POST -d '[{"headers" :{"state" : "SH"},"body" :"TEST3"}]' http://192.168.233.128:50000

**故障转移和负载均衡：**

FailoverSink Processor会通过配置维护了一个优先级列表。保证每一个有效的事件都会被处理。

故障转移的工作原理是将连续失败sink分配到一个池中，在那里被分配一个冷冻期，在这个冷冻期里，这个sink不会做任何事。一旦sink成功发送一个event，sink将被还原到live 池中。

在这配置中，要设置sinkgroups processor为failover，需要为所有的sink分配优先级，所有的优先级数字必须是唯一的，这个得格外注意。此外，failover time的上限可以通过maxpenalty 属性来进行设置。

Failover测试

1. #配置文件：failover\_sink\_case13.conf
2. #Name the components on this agent
3. a1.sources= r1
4. a1.sinks= k1 k2
5. a1.channels= c1 c2
7. a1.sinkgroups= g1
8. a1.sinkgroups.g1.sinks= k1 k2
9. a1.sinkgroups.g1.processor.type= failover
10. a1.sinkgroups.g1.processor.priority.k1= 5
11. a1.sinkgroups.g1.processor.priority.k2= 10
12. a1.sinkgroups.g1.processor.maxpenalty= 10000
14. #Describe/configure the source
15. a1.sources.r1.type= syslogtcp
16. a1.sources.r1.port= 50000
17. a1.sources.r1.host= 192.168.233.128
18. a1.sources.r1.channels= c1 c2
20. #Describe the sink
21. a1.sinks.k1.type= avro
22. a1.sinks.k1.channel= c1
23. a1.sinks.k1.hostname= 192.168.233.129
24. a1.sinks.k1.port= 50000
26. a1.sinks.k2.type= avro
27. a1.sinks.k2.channel= c2
28. a1.sinks.k2.hostname= 192.168.233.130
29. a1.sinks.k2.port= 50000
30. # Usea channel which buffers events in memory
31. a1.channels.c1.type= memory
32. a1.channels.c1.capacity= 1000
33. a1.channels.c1.transactionCapacity= 100

下面是第一个接受复制事件代理配置

1. #配置文件：replicate\_sink1\_case11.conf
2. # Name the components on this agent
3. a2.sources = r1
4. a2.sinks = k1
5. a2.channels = c1
7. # Describe/configure the source
8. a2.sources.r1.type = avro
9. a2.sources.r1.channels = c1
10. a2.sources.r1.bind = 192.168.233.129
11. a2.sources.r1.port = 50000
13. # Describe the sink
14. a2.sinks.k1.type = logger
15. a2.sinks.k1.channel = c1
17. # Use a channel which buffers events inmemory
18. a2.channels.c1.type = memory
19. a2.channels.c1.capacity = 1000
20. a2.channels.c1.transactionCapacity = 100

下面是第二个接受复制事件代理配置：

1. #配置文件：replicate\_sink2\_case11.conf
2. # Name the components on this agent
3. a3.sources = r1
4. a3.sinks = k1
5. a3.channels = c1
7. # Describe/configure the source
8. a3.sources.r1.type = avro
9. a3.sources.r1.channels = c1
10. a3.sources.r1.bind = 192.168.233.130
11. a3.sources.r1.port = 50000
13. # Describe the sink
14. a3.sinks.k1.type = logger
15. a3.sinks.k1.channel = c1
17. # Use a channel which buffers events inmemory
18. a3.channels.c1.type = memory
19. a3.channels.c1.capacity = 1000
20. a3.channels.c1.transactionCapacity = 100

****#敲命令****

首先先启动2个接受复制事件代理，如果先启动源发送的代理，会报他找不到sinks的绑定，因为2个接事件的代理还未起来。

flume-ng agent -cconf -f conf/replicate\_sink1\_case11.conf -n a1 -Dflume.root.logger=INFO,console

flume-ng agent -cconf -f conf/replicate\_sink2\_case11.conf -n a1 -Dflume.root.logger=INFO,console

在启动源发送的代理

flume-ng agent -cconf -f conf/failover\_sink\_case13.conf -n a1 -Dflume.root.logger=INFO,console

启动成功后，打开另一个终端输入，往侦听端口送数据

echo "hello failoversink" | nc 192.168.233.128 50000

**Load balancing SinkProcessor**

下面是测试例子

1. #配置文件：load\_sink\_case14.conf
2. # Name the components on this agent
3. a1.sources = r1
4. a1.sinks = k1 k2
5. a1.channels = c1
7. a1.sinkgroups = g1
8. a1.sinkgroups.g1.sinks = k1 k2
9. a1.sinkgroups.g1.processor.type =load\_balance
10. a1.sinkgroups.g1.processor.backoff = true
11. a1.sinkgroups.g1.processor.selector =round\_robin
13. # Describe/configure the source
14. a1.sources.r1.type = syslogtcp
15. a1.sources.r1.port = 50000
16. a1.sources.r1.host = 192.168.233.128
17. a1.sources.r1.channels = c1
19. # Describe the sink
20. a1.sinks.k1.type = avro
21. a1.sinks.k1.channel = c1
22. a1.sinks.k1.hostname = 192.168.233.129
23. a1.sinks.k1.port = 50000
25. a1.sinks.k2.type = avro
26. a1.sinks.k2.channel = c1
27. a1.sinks.k2.hostname = 192.168.233.130
28. a1.sinks.k2.port = 50000
29. # Use a channel which buffers events inmemory
30. a1.channels.c1.type = memory
31. a1.channels.c1.capacity = 1000
32. a1.channels.c1.transactionCapacity = 100

**拦截器**

**Body TextSerializer**，别名：text。这个拦截器将把事件的body部分写入到输出流中而不需要任何转换或者修改。事件的header将直接被忽略。

1. #配置文件：body\_case15.conf
2. # Name the components on this agent
3. a1.sources = r1
4. a1.sinks = k1
5. a1.channels = c1
7. # Describe/configure the source
8. a1.sources.r1.type = http
9. a1.sources.r1.port = 50000
10. a1.sources.r1.host = 192.168.233.128
11. a1.sources.r1.channels = c1
13. # Describe the sink
14. a1.sinks.k1.type = file\_roll
15. a1.sinks.k1.channel = c1
16. a1.sinks.k1.sink.directory = /tmp/logs
17. a1.sinks.k1.sink.serializer = text
18. a1.sinks.k1.sink.serializer.appendNewline =false
20. # Use a channel which buffers events inmemory
21. a1.channels.c1.type = memory
22. a1.channels.c1.capacity = 1000
23. a1.channels.c1.transactionCapacity = 100

#敲命令

flume-ng agent -c conf -fconf/body\_case15.conf -n a1 -Dflume.root.logger=INFO,console

启动成功后，打开另一个终端输入，往侦听端口送数据

curl -X POST -d '[{"headers":{"looklook1" : "looklook1 isheader","looklook2": "looklook2 isheader"},"body" : "hellolooklook5"}]' http://192.168.233.128:50000

**Timestamp Interceptor**

官网说Flume 可以在事件传输过程中对它进行修改与删除，而这个都是通过Interceptor进行实现的，实际都是往事件的header里插数据。而Timestamp Interceptor拦截器就是可以往event的header中插入关键词为timestamp的时间戳。

**下面是测试例子**

1. #配置文件：timestamp\_case16.conf
2. # Name the components on this agent
3. a1.sources = r1
4. a1.sinks = k1
5. a1.channels = c1
7. # Describe/configure the source
8. a1.sources.r1.type = syslogtcp
9. a1.sources.r1.port = 50000
10. a1.sources.r1.host = 192.168.233.128
11. a1.sources.r1.channels = c1
13. **a1.sources.r1.interceptors = i1**
14. **a1.sources.r1.interceptors.i1.preserveExisting= false**
15. **a1.sources.r1.interceptors.i1.type = timestamp**
16. # Describe the sink
17. a1.sinks.k1.type = hdfs
18. a1.sinks.k1.channel = c1
19. a1.sinks.k1.hdfs.path =hdfs://carl:9000/flume/%Y-%m-%d/%H%M
20. a1.sinks.k1.hdfs.filePrefix = looklook5.
21. a1.sinks.k1.hdfs.fileType=DataStream
23. # Use a channel which buffers events inmemory
24. a1.channels.c1.type = memory
25. a1.channels.c1.capacity = 1000
26. a1.channels.c1.transactionCapacity = 100

## **Host Interceptor**

该拦截器可以往event的header中插入关键词默认为host主机名或者ip地址（注意是agent运行的机器的主机名或者ip地址）

**下面是测试例子**

1. #配置文件：time\_host\_case17.conf
2. # Name the components on this agent
3. a1.sources = r1
4. a1.sinks = k1
5. a1.channels = c1
7. # Describe/configure the source
8. a1.sources.r1.type = syslogtcp
9. a1.sources.r1.port = 50000
10. a1.sources.r1.host = 192.168.233.128
11. a1.sources.r1.channels = c1
13. a1.sources.r1.interceptors = i1 i2
14. a1.sources.r1.interceptors.i1.preserveExisting= false
15. a1.sources.r1.interceptors.i1.type =timestamp
16. a1.sources.r1.interceptors.i2.type = host
17. a1.sources.r1.interceptors.i2.hostHeader =hostname
18. a1.sources.r1.interceptors.i2.useIP = false
20. # Describe the sink
21. a1.sinks.k1.type = hdfs
22. a1.sinks.k1.channel = c1
23. a1.sinks.k1.hdfs.path =hdfs://carl:9000/flume/%Y-%m-%d/%H%M
24. a1.sinks.k1.hdfs.filePrefix = %{hostname}
25. a1.sinks.k1.hdfs.fileType=DataStream
27. # Use a channel which buffers events inmemory
28. a1.channels.c1.type = memory
29. a1.channels.c1.capacity = 1000
30. a1.channels.c1.transactionCapacity = 100

增加一个拦截器，类型是host,h将hostname作为文件前缀。

#敲命令

flume-ng agent -c conf -f conf/time\_host\_case17.conf-n a1 -Dflume.root.logger=INFO,console

启动成功后

打开另一个终端输入，往侦听端口送数据

echo "Time&hostInterceptor1" | nc 192.168.233.128 50000

echo "Time&hostInterceptor2" | nc 192.168.233.128 50000

## **Static Interceptor**

Static Interceptor拦截器允许用户增加一个static的header并为所有的事件赋值。范围是所有事件

1. a1.sources.r1.interceptors = i1
2. a1.sources.r1.interceptors.i1.type = static
3. a1.sources.r1.interceptors.i1.key = looklook5
4. a1.sources.r1.interceptors.i1.value =looklook10

## **Regex FilteringInterceptor**

Regex Filtering Interceptor拦截器用于过滤事件，筛选出与配置的正则表达式相匹配的事件。可以用于包含事件和排除事件。常用于数据清洗，通过正则表达式把数据过滤出来。

excludeEvents 为true的时候为排除所有匹配正则表达式的数据。

下面是测试例子

1. #配置文件：regex\_filter\_case19.conf
2. # Name the components on this agent
3. a1.sources = r1
4. a1.sinks = k1
5. a1.channels = c1
7. # Describe/configure the source
8. a1.sources.r1.type = syslogtcp
9. a1.sources.r1.port = 50000
10. a1.sources.r1.host = 192.168.233.128
11. a1.sources.r1.channels = c1
12. a1.sources.r1.interceptors = i1
13. a1.sources.r1.interceptors.i1.type =regex\_filter
14. a1.sources.r1.interceptors.i1.regex =^[0-9]\*$
15. a1.sources.r1.interceptors.i1.excludeEvents =true
17. # Describe the sink
18. a1.sinks.k1.type = logger
20. # Use a channel which buffers events inmemory
21. a1.channels.c1.type = memory
22. a1.channels.c1.capacity = 1000
23. a1.channels.c1.transactionCapacity = 100
25. # Bind the source and sink to the channel
26. a1.sources.r1.channels = c1
27. a1.sinks.k1.channel = c1

我们对开头字母是数字的数据，全部过滤。

****#敲命令****

flume-ng agent -c conf -f conf/regex\_filter\_case19.conf-n a1 -Dflume.root.logger=INFO,console

启动成功后，打开另一个终端输入，往侦听端口送数据

echo "a" | nc192.168.233.128 50000

echo "1222" |nc 192.168.233.128 50000

echo "a222" |nc 192.168.233.128 50000

**taildir配置：**

agent1.sources=source1

agent1.sinks=sink1

agent1.channels=channel1

# Describe/configure source1

agent1.sources.source1.type=TAILDIR

agent1.sources.source1.positionFile=/opt/apache-flume-1.8.0-bin/taildir\_position.json

agent1.sources.source1.filegroups=f1

agent1.sources.source1.filegroups.f1=/opt/dirlog.log

agent1.sources.source1.filegroups.f1.headerKey1=value1

# Describe sink1

agent1.sinks.sink1.type=file\_roll

agent1.sinks.sink1.sink.directory=/opt/apache-flume-1.8.0-bin/logs

# Use a channel which buffers events in memory，filechannel延迟两三秒

agent1.channels.channel1.type=file

agent1.channels.channel1.checkpointDir=/opt/apache-flume-1.8.0-bin/checkpoint

agent1.channels.channel1.dataDirs=/opt/apache-flume-1.8.0-bin/data

agent1.channels.channel1.capacity=1000

agent1.channels.channel1.transactionCapactiy=100

# Bind the source and sink to the channel

agent1.sources.source1.channels=channel1

agent1.sinks.sink1.channel=channel1

flume-ng agent -n a1 -c conf -f conf/example.file -Dflume.root.logger=DEBUG,console