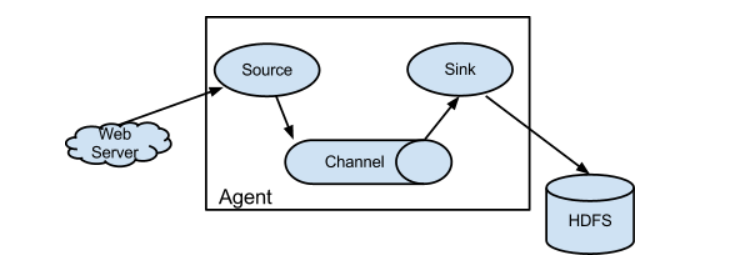
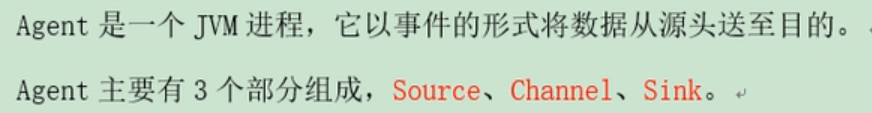
#### 学习资源

<https://www.bilibili.com/video/BV1yz411i74z?p=1>

#### Flume的核心概念



##### Agent



##### Source

Source读取数据。

数据源：日志文件、网络端口等

用户需要根据自己的数据源类型，选择合适的Source对象。

##### Channel

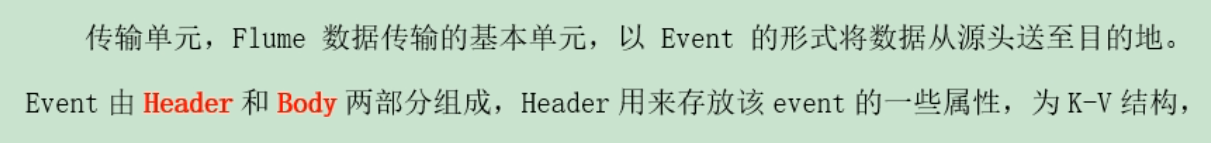
托管Event，Source读数据到Channel和Sink从Channel取数据可以按照自己的速率来。

有Memory Channel(Event 存内存，有丢数据的风险，处理效率高)和File Channel(存磁盘)

##### Event

Header:k-v属性说明

Body:数据



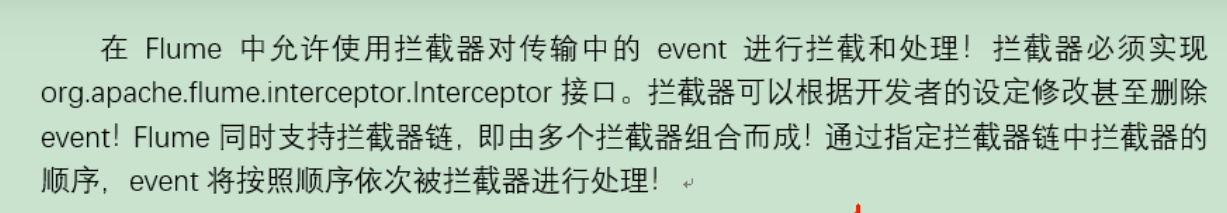


##### Sink

用户需要根据自己数据存储目的地类型选择合适的Sink对象。

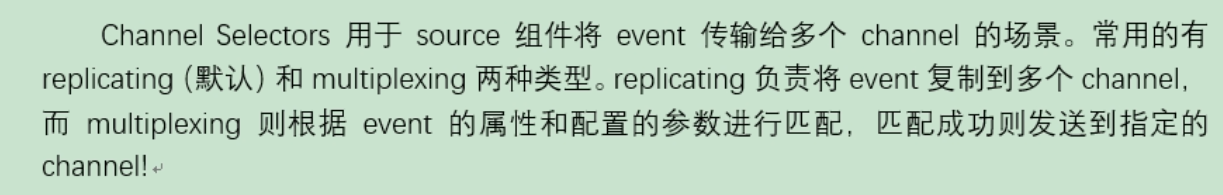
##### Interceptors 拦截器

对传输中的event进行拦截和处理。可以根据开发者设定修改甚至删除Event。Source 到 Channel之间进行拦截。



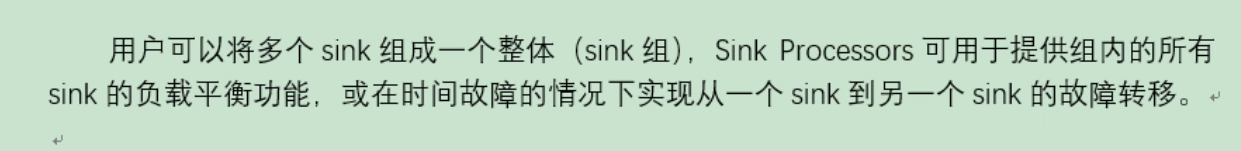
##### Channel Selectors

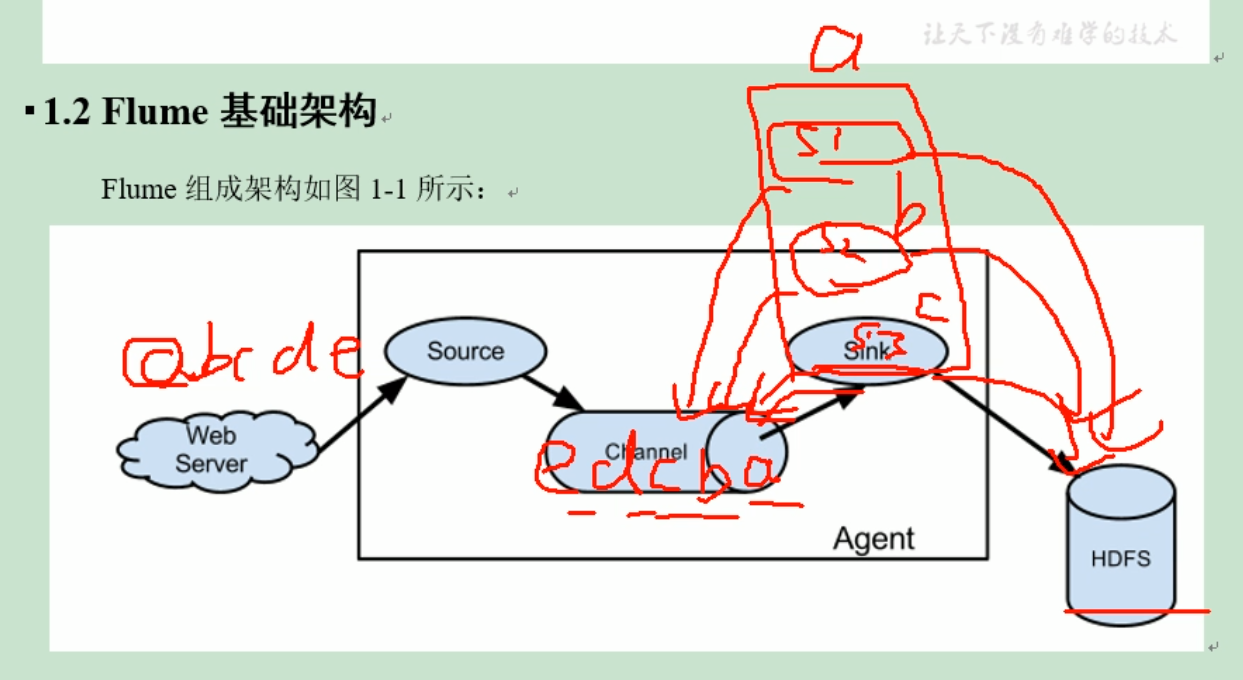
当一个source对接多个channel时，由Channel Selectors选取channel将event存入。



##### Sink Processors

多个Sink从一个Channel取数据，为了保证数据的顺序，由Sink Proccessors来挑选一个sink,由这个sink干活。





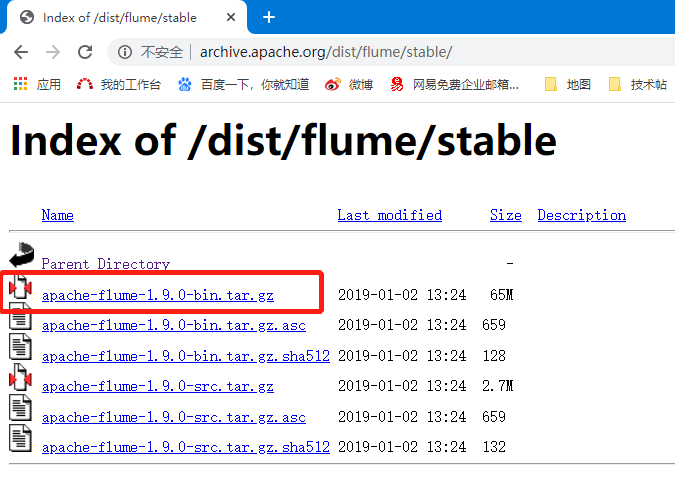
#### Flume安装部署

##### 安装地址

官网地址：<http://flume.apache.org/>

文档地址: <http://flume.apache.org/FlumeUserGuide.html>

下载地址：<http://archive.apache.org/dist/flume/>



##### 安装部署

条件：

1. 保证有JAVA\_HOME
2. 解压即可

Flume是一个客户端，哪个机器有数据就安装在哪里。

#### Flume使用

##### 启动agent

flume-ng agent -n agent的名称 -f agent的配置文件 -c 其他配置文件所在位置,如log4j.properties(可选)

cd到flume文件夹再执行下列命令：

|  |
| --- |
| flume-ng agent -c conf/ -n a1 -f E:\study\trunk\Flume\netcat\_logger\_memory.conf -property "flume.root.logger=INFO,console" |

测试时，Log输出至控制台参数：

-property "flume.root.logger=INFO,console"：适用于windows

-Dflume.root.logger=DEBUG,console ：linux平台适用

若报错按下面帖子解决问题。

<https://blog.csdn.net/wateryouyo/article/details/82057082>

##### agent的配置文件

本质是一个Porperties文件！格式为 属性名=属性值

1. 定义当前配置文件中angent名称 与-n后的agent名称一致，还要定义source 、sink 、channel的别名；
2. 指定source、sink、channel等组件类型；
3. 指定source、sink、channel等组件的配置，配置参数需要参考flume官方用户手册；
4. 指定source和channel的对应关系，以及sink和channel的对应关系；连接组件！

##### 案例1：监听netcat发给某个tcp端口数据

###### netcat安装指南

<https://blog.csdn.net/BoomLee/article/details/102563472>

###### 使用的组件类型

netcat source

作用就是监听某个tcp端口收到的数据，将每一行数据封装为event；工作原理类似与nc -l 端口

配置：Required properties are in **bold**.

| **Property Name** | **Default** | **Description** |
| --- | --- | --- |
| **channels** | – |  |
| **type** | – | The component type name, needs to be netcat |
| **bind** | – | Host name or IP address to bind to |
| **port** | – | Port # to bind to |
| max-line-length | 512 | Max line length per event body (in bytes) |
| ack-every-event | true | Respond with an “OK” for every event received |
| selector.type | replicating | replicating or multiplexing |
| selector.\* |  | Depends on the selector.type value |
| interceptors | – | Space-separated list of interceptors |
| interceptors.\* |  |  |

Example for agent named a1:

a1.sources **=** r1

a1.channels **=** c1

a1.sources.r1.type **=** netcat

a1.sources.r1.bind **=** 0.0.0.0

a1.sources.r1.port **=** 6666

a1.sources.r1.channels **=** c1

logger sink

用logger（日志输出器）将event输出到文件或控制台。

配置：Required properties are in **bold**.

| **Property Name** | **Default** | **Description** |
| --- | --- | --- |
| **channel** | – |  |
| **type** | – | The component type name, needs to be logger |
| maxBytesToLog | 16 | Maximum number of bytes of the Event body to log |

Example for agent named a1:

a1.channels **=** c1

a1.sinks **=** k1

a1.sinks.k1.type **=** logger

a1.sinks.k1.channel **=** c1

Memory Channnel

The events are stored in an in-memory queue with configurable max size. It’s ideal for flows that need higher throughput and are prepared to lose the staged data in the event of a agent failures. Required properties are in **bold**.

| **Property Name** | **Default** | **Description** |
| --- | --- | --- |
| **type** | – | The component type name, needs to be memory |
| capacity | 100 | The maximum number of events stored in the channel |
| transactionCapacity | 100 | The maximum number of events the channel will take from a source or give to a sink per transaction |
| keep-alive | 3 | Timeout in seconds for adding or removing an event |
| byteCapacityBufferPercentage | 20 | Defines the percent of buffer between byteCapacity and the estimated total size of all events in the channel, to account for data in headers. See below. |
| byteCapacity | see description | Maximum total **bytes** of memory allowed as a sum of all events in this channel. The implementation only counts the Event body, which is the reason for providing the byteCapacityBufferPercentage configuration parameter as well. Defaults to a computed value equal to 80% of the maximum memory available to the JVM (i.e. 80% of the -Xmx value passed on the command line). Note that if you have multiple memory channels on a single JVM, and they happen to hold the same physical events (i.e. if you are using a replicating channel selector from a single source) then those event sizes may be double-counted for channel byteCapacity purposes. Setting this value to 0 will cause this value to fall back to a hard internal limit of about 200 GB. |

Example for agent named a1:

a1.channels **=** c1

a1.channels.c1.type **=** memory

a1.channels.c1.capacity **=** 10000

a1.channels.c1.transactionCapacity **=** 10000

a1.channels.c1.byteCapacityBufferPercentage **=** 20

a1.channels.c1.byteCapacity **=** 800000

###### 编写配置文件

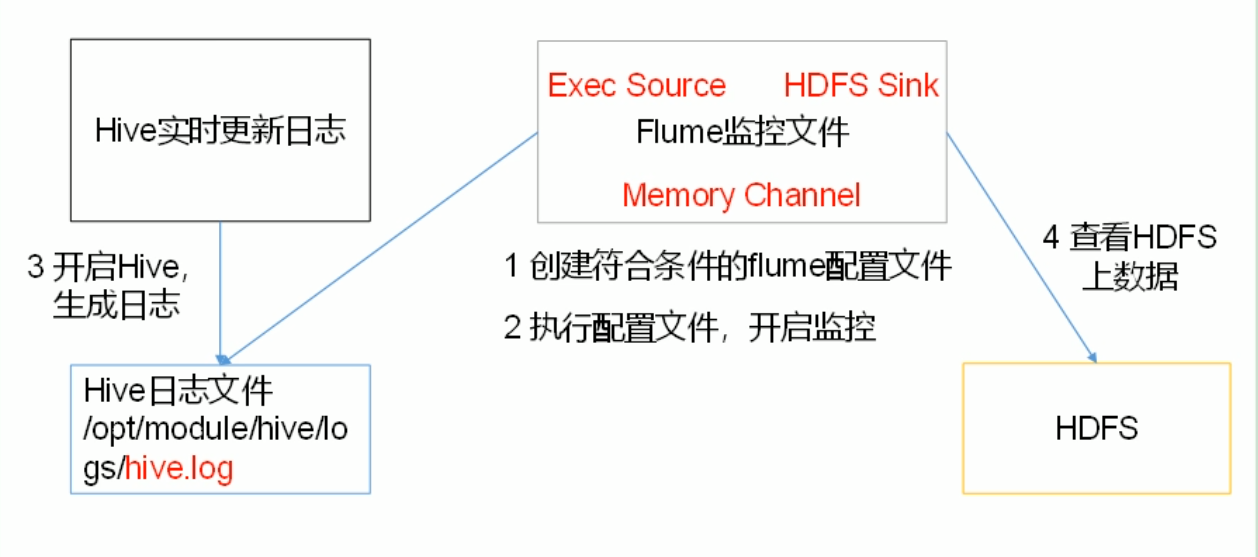
从文档上copy “A simple example”内容至本地修改。

##### 案例2：实时读取本地文件到HDFS案例

**从本地到集群要经过一轮avro**

**参考：https://blog.csdn.net/HANLIPENGHANLIPENG/article/details/78181706?utm\_source=blogxgwz8**

**flume通过公网ip采集到hdfs上**



###### Exec Source

Windows上avro方式没有成功。

**exec source,会在agent启动时，运行一个linux命令，**运行Linux命令的进程要求是一个可以持续产生数据的进程，将标准输出的数据封装位event!

通常情况下，如果指定的命令退出了，那么source也会退出并且不会再封装任何的数据！

所以使用这个source一般推荐类似cat,tail -f 这种命令，而不是date这种只会返回一个数据，并且执行完就退出的命令！

配置信息：

Required properties are in **bold**.

| **Property Name** | **Default** | **Description** |
| --- | --- | --- |
| **channels** | – |  |
| **type** | – | The component type name, needs to be exec |
| **command** | – | The command to execute |
| shell | – | A shell invocation used to run the command. e.g. /bin/sh -c. Required only for commands relying on shell features like wildcards, back ticks, pipes etc. |
| restartThrottle | 10000 | Amount of time (in millis) to wait before attempting a restart |
| restart | false | Whether the executed cmd should be restarted if it dies |
| logStdErr | false | Whether the command’s stderr should be logged |
| batchSize | 20 | The max number of lines to read and send to the channel at a time |
| batchTimeout | 3000 | Amount of time (in milliseconds) to wait, if the buffer size was not reached, before data is pushed downstream |
| selector.type | replicating | replicating or multiplexing |
| selector.\* |  | Depends on the selector.type value |
| interceptors | – | Space-separated list of interceptors |
| interceptors.\* |  |  |

Exec Source: channel是有容量限制的，会发生故障。

和异步的source一样，无法在source和channel中放入event故障时即使通知客户端，暂停生成数据！容易造成数据丢失！

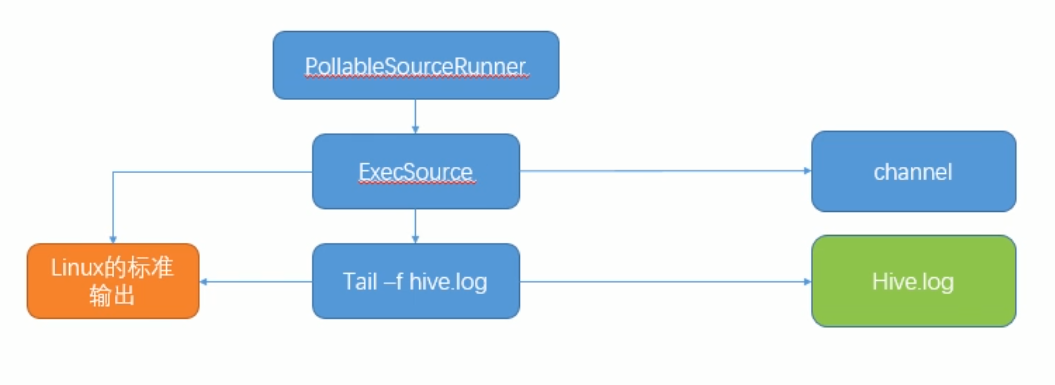
Tail -f hive.log是Linux线程

PollableSourceRunner 是Java线程，来处理event ，根据拦截器，收到channel

以上为两个异步进程。

解决方案：发生故障时，需要及时通知客户端！如果客户端无法暂停，必须有一个数据的缓存机制！

如果希望数据有强的可靠性保证，可以考虑使用SpoolingDirSource或TailDirSource或者自己写source自己控制。



###### HDFS Sink

--将event写入到HDFS！目前只支持生成两种类型的文件：text | sequenceFile，这两种文件都可以使用压缩！写入到HDFS的文件可以自动滚动（关闭当前正在写的文件，创建一个新文件）。基于时间、events数量、数据大小进行周期性的滚动。

--支持基于时间和采集数据的机器进行分桶和分区操作！

--HDFS数据所上传的目录或文件名可以包含一个格式化的转义序列，这个路径或文件名会在上传event时，被自动替换，替换从完整的路径。

--使用次sink要求本机已经安装了Hadoop，或持有Hadoop的jar包！

配置信息：

Required properties are in **bold**.

**Note**

For all of the time related escape sequences, a header with the key “timestamp” must exist among the headers of the event (unless hdfs.useLocalTimeStamp is set to true). One way to add this automatically is to use the TimestampInterceptor.

| **Name** | **Default** | **Description** |
| --- | --- | --- |
| **channel** | – |  |
| **type** | – | The component type name, needs to be hdfs |
| **hdfs.path** | – | HDFS directory path (eg hdfs://namenode/flume/webdata/) |
| hdfs.filePrefix | FlumeData | Name prefixed to files created by Flume in hdfs directory |
| hdfs.fileSuffix | – | Suffix to append to file (eg .avro - *NOTE: period is not automatically added*) |
| hdfs.inUsePrefix | – | Prefix that is used for temporal files that flume actively writes into |
| hdfs.inUseSuffix | .tmp | Suffix that is used for temporal files that flume actively writes into |
| hdfs.emptyInUseSuffix | false | If false an hdfs.inUseSuffix is used while writing the output. After closing the output hdfs.inUseSuffix is removed from the output file name. If true the hdfs.inUseSuffix parameter is ignored an empty string is used instead. |
| hdfs.rollInterval | 30 | Number of seconds to wait before rolling current file (0 = never roll based on time interval) |
| 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.idleTimeout | 0 | Timeout after which inactive files get closed (0 = disable automatic closing of idle files) |
| hdfs.batchSize | 100 | number of events written to file before it is flushed to HDFS |
| hdfs.codeC | – | Compression codec. one of following : gzip, bzip2, lzo, lzop, snappy |
| hdfs.fileType | SequenceFile | File format: currently SequenceFile, DataStream or CompressedStream (1)DataStream will not compress output file and please don’t set codeC (2)CompressedStream requires set hdfs.codeC with an available codeC |
| hdfs.maxOpenFiles | 5000 | Allow only this number of open files. If this number is exceeded, the oldest file is closed. |
| hdfs.minBlockReplicas | – | Specify minimum number of replicas per HDFS block. If not specified, it comes from the default Hadoop config in the classpath. |
| hdfs.writeFormat | Writable | Format for sequence file records. One of Text or Writable. Set to Text before creating data files with Flume, otherwise those files cannot be read by either Apache Impala (incubating) or Apache Hive. |
| hdfs.threadsPoolSize | 10 | Number of threads per HDFS sink for HDFS IO ops (open, write, etc.) |
| hdfs.rollTimerPoolSize | 1 | Number of threads per HDFS sink for scheduling timed file rolling |
| hdfs.kerberosPrincipal | – | Kerberos user principal for accessing secure HDFS |
| hdfs.kerberosKeytab | – | Kerberos keytab for accessing secure HDFS |
| hdfs.proxyUser |  |  |
| hdfs.round | false | Should the timestamp be rounded down (if true, affects all time based escape sequences except %t) |
| hdfs.roundValue | 1 | Rounded down to the highest multiple of this (in the unit configured using hdfs.roundUnit), less than current time. |
| hdfs.roundUnit | second | The unit of the round down value - second, minute or hour. |
| hdfs.timeZone | Local Time | Name of the timezone that should be used for resolving the directory path, e.g. America/Los\_Angeles. |
| hdfs.useLocalTimeStamp | false | Use the local time (instead of the timestamp from the event header) while replacing the escape sequences. |
| hdfs.closeTries | 0 | Number of times the sink must try renaming a file, after initiating a close attempt. If set to 1, this sink will not re-try a failed rename (due to, for example, NameNode or DataNode failure), and may leave the file in an open state with a .tmp extension. If set to 0, the sink will try to rename the file until the file is eventually renamed (there is no limit on the number of times it would try). The file may still remain open if the close call fails but the data will be intact and in this case, the file will be closed only after a Flume restart. |
| hdfs.retryInterval | 180 | Time in seconds between consecutive attempts to close a file. Each close call costs multiple RPC round-trips to the Namenode, so setting this too low can cause a lot of load on the name node. If set to 0 or less, the sink will not attempt to close the file if the first attempt fails, and may leave the file open or with a ”.tmp” extension. |
| serializer | TEXT | Other possible options include avro\_event or the fully-qualified class name of an implementation of the EventSerializer.Builder interface. |
| serializer.\* |  |  |

###### 启用agent

|  |
| --- |
| flume-ng agent -c conf/ -n a1 -f E:\study\trunk\Flume\exec\_hdfs\_memory.conf -property "flume.root.logger=INFO,console" |

##### Spooling Directory Source

Windows avro方式成功

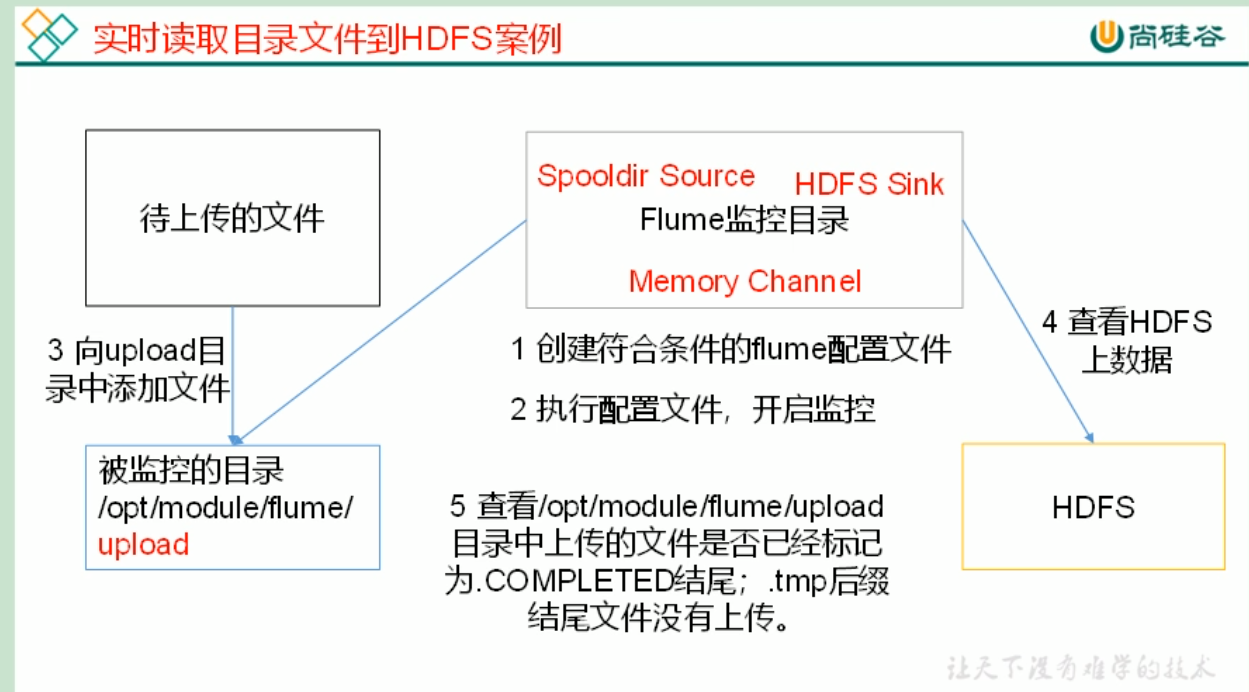
SpoolingDirSource指定本地磁盘的一个目录为“Spooling（自动收集）”的目录！这个source可以读取目录中新增的文件，将文件的内容封装为event！

SpoolingDirSource在读取一整个文件到channel之后，它会采取策略，要么删除文件（取决于配置），要么对文件进行一个完成状态的重命名，这样可以保证source持续监控新的文件！

与execsource不同，SpoolingDirSource是可靠的！即使flume被杀死或重启，依然不丢数据！但是为了保证这个特性，付出的代价是，一旦flume发现以下情况，flume会报错，停止！

1. 一个文件已经被放入目录，放入后又被重新写入新数据；
2. 文件的名称在放入目录后又被重新使用（出现了重名的文件）。

要求：必须已经封闭的文件才能放入到SpoolingDirSource,在同一个SpoolingDirSource中都不能出现重名的文件！



##### TailDirSource

Windows上avro方式不成功，路径与Linux系统不同导致。

解法参考：<https://www.cnblogs.com/laoqing/p/12836826.html>

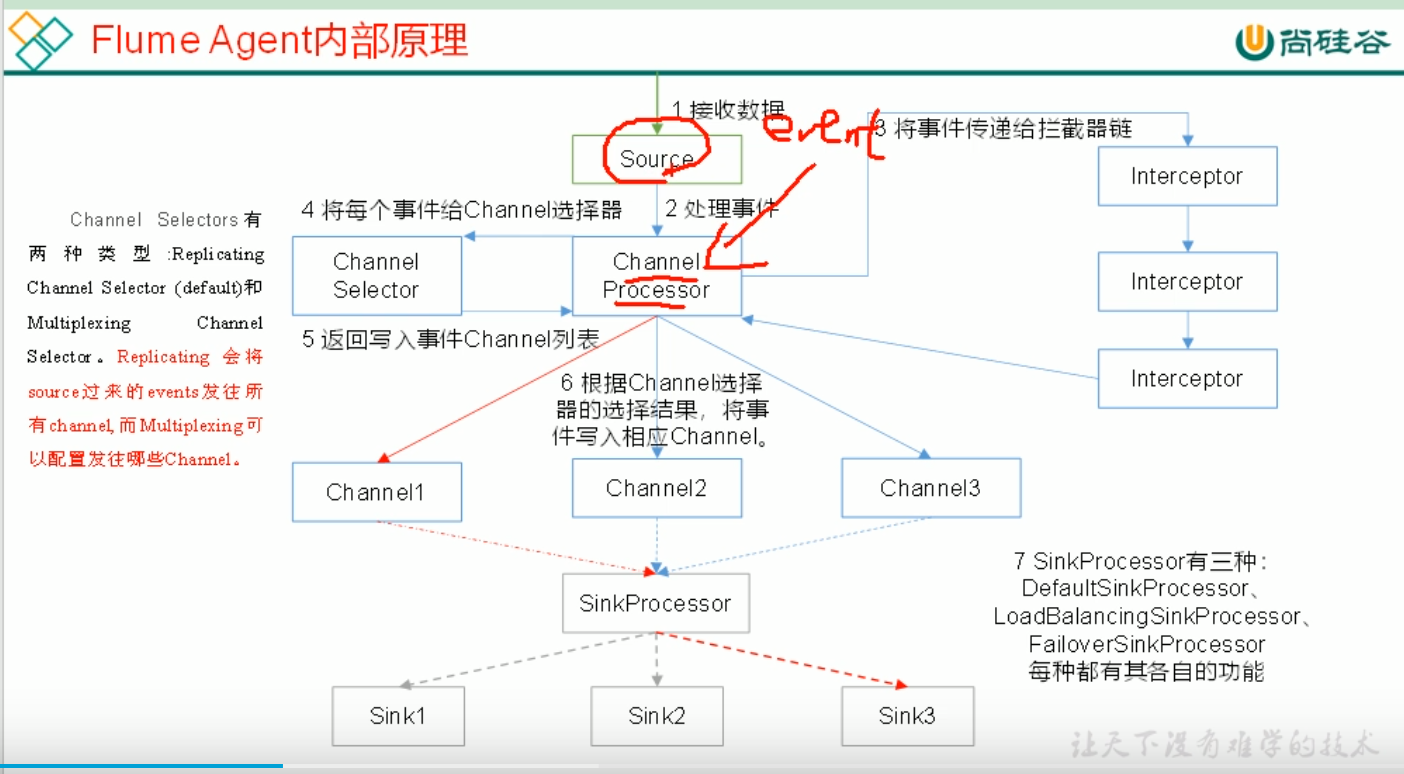
Flume1.7之后才有的。监控文件。它只能一行一行读文本文件。

可以读取多个文件最新追加写入的内容！

可靠；在工作时会将读取到最后位置记录在一个json文件中，一旦agent重启，会从之前已经记录的位置，继续执行tail操作！

目前只能读文本文件！

##### Agent的内部原理



同一时间只能有一个sink来干活。