Flume

目录

[1. Flume 1](#_Toc31419)

[1.1. Flume概述 1](#_Toc229)

[1.1.1. Flume的运行时环境 3](#_Toc19091)

[1.1.2. Flume的版本变迁 3](#_Toc15670)

[1.1.3. Flume的特征与优势 4](#_Toc2484)

[1.2. Flume的基本概念 4](#_Toc15546)

[1.2.1. Flume总体架构 4](#_Toc2331)

[1.2.2. Flume的核心概念 5](#_Toc9933)

[1.2.3. Flume的运行机制 10](#_Toc27555)

[1.3. Flume的基本用法 11](#_Toc4111)

[1.3.1. Flume多级数据流动 11](#_Toc9203)

[1.3.2. 数据的扇入流 12](#_Toc5891)

[1.3.3. 数据的扇出流 12](#_Toc23312)

[1.4. Flume的特性 13](#_Toc31456)

[1.4.1. 可靠性 13](#_Toc19573)

[1.4.2. 可恢复性 14](#_Toc9683)

[1.4.3. 可扩展性 14](#_Toc19412)

[1.4.4. 可管理性 14](#_Toc11825)

[1.5. Flume的安装配置 15](#_Toc2540)

[1.5.1. Flume的安装步骤 15](#_Toc11892)

[1.5.2. Flume配置文件及其详解 15](#_Toc6433)

[1.5.3. 启动Flume agent 17](#_Toc4605)

[1.5.4. 访问Flume 18](#_Toc30642)

[2. Flume核心组件 20](#_Toc9446)

[2.1. Source组件 20](#_Toc21698)

[2.1.1. Avro source 21](#_Toc26480)

[2.1.2. Spooling Directory source 27](#_Toc11925)

[2.1.3. Http Source 29](#_Toc27084)

[2.2. Sink组件 32](#_Toc7076)

[2.2.1. Logger Sink 33](#_Toc23382)

[2.2.2. File Roll Sink 35](#_Toc23935)

[2.2.3. HDFS Sink 37](#_Toc4217)

[2.2.4. Avro Sink 40](#_Toc5014)

[2.3. Channel组件 42](#_Toc7378)

[2.3.1. File Channel 42](#_Toc21847)

[2.3.2. Memory Channel 44](#_Toc19277)

# Flume



## Flume概述

·Flume支持在日志系统中定制各类数据的发送方、以及收集数据操作。

Flume是一个高可用，高可靠，健壮性的分布式海量日志采集、聚合和传输系统。用于从不同来源有效收集，聚集和移动大量的日志数据用以集中式管理的系统，是apache的一个顶级项目。Flume支持在日志系统中定制各类数据发送方，用于收集数据；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。可以用一个简单的图例来描述一下Flume的工作流程，如果1-1所示。

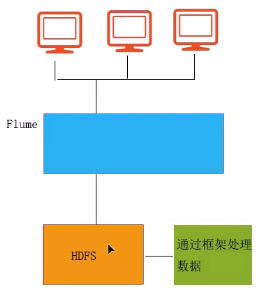


图 1-1 Flume工作流程

在Flume系统中其数据源可定制可扩展，其数据接收方也可定制可扩展。Flume提供了从console（控制台）、RPC（Thrift-RPC）、text（文件）、tail（UNIX tail）、syslog（syslog日志系统），支持TCP和UDP等2种模式），exec（命令执行）等数据源上收集数据的能力。

Flume的主要的应用场景：比如我们在做一个电子商务网站，然后我们想从消费用户中访问点特定的节点区域来分析消费者的行为或者购买意图。这样我们就可以更加快速的将他想要的推送到界面上，实现这一点，我们需要将获取到的它访问的页面以及点击的产品数据等日志数据信息收集移交给Hadoop平台上去分析。而Flume正是帮我们做到这一点。现在流行的内容推送，比如广告定点投放以及新闻私人定制，这些一定是使用FLume,毕竟优秀的产品很多，比如facebook的Scribe，还有Apache新出的另一个明星项目chukwa，还有淘宝Time Tunnel。

Flume作为大数据的一个重要组成部分，在世界500千企业中，乃至在很多小企业中都扮演着重要的角色，因此，无论是企业还是个人，对于flume的了解与掌握都应该成为一种必备的技能。

随着大数据发展速度的不断加快，其影响力在不断的上升，相信在未来数据驱动一切的时代，各个行业对于大数据的应用必将越来越深刻，相信，关于大数据中的flume等概念一定会达到更多的重视和关注度。

当前Flume有两个版本：Flume 0.9X版本统称为Flume-og，Flume1.X版本的统称为Flume-ng。由于Flume-ng经过重大重构，与Flume-og有很大不同，使用时需要注意区分，后续将会对两种版本的区别进行介绍。

### Flume的运行时环境

a.下载flume

flume是apache的顶级项目，如果需要下载Flume的安装包，可以到其官网上进行下载，下载的具体地址为: flume.apache.org。下载的时候要注意，flume主要有两个大的版本，分别是flume0.9x和flume1.x版本，这两个版本互不兼容，下载时要注意。我们学习的是最新的1.x版本，也叫flume-ng版本。

b. 准备前提

安装好jdk，并且版本要匹配。安装Flume的环境为jdk1.6及以上的版本，推荐使用jdk1.7版本。配置JAVA\_HOME环境变量。

c. 安装flume

将flume的安装包上传到linux中进行解压即可, flume可以通过JAVA\_HOME环境变量找到jdk使用，并不需要额外配置。

### Flume的版本变迁

在1.1.1小节中提到Flume有两个版本：Flume 0.9X版本，统称为Flume-og；Flume1.X版本，统称为Flume-ng。

Flume 1.x 版本即Flume-ng最明显的改动就是取消了集中管理配置的 Master 和 Zookeeper，变为一个纯粹的传输工具。Flume-ng另一个主要的不同点是读入数据和写出数据现在由不同的工作线程处理(称为 Runner)，而这种异步的设计使读入线程可以顺畅的工作而无需关注下游的任何问题。 在 Flume-og 中，读入线程同样做写出工作(除了故障重试)。如果写出很慢(不是完全失败)，它将阻塞 Flume 接收数据的能力。

### Flume的特征与优势

Flume的特征：

1. Flume可以高效率的将多个网站服务器中收集的日志信息存入HDFS/HBase中。

2. 使用Flume，我们可以将从多个服务器中获取的数据迅速的移交给Hadoop中。

3. 除了日志信息，Flume同时也可以用来接入收集规模宏大的社交网络节点事件数据，比如facebook,twitter,电商网站如亚马逊，flipkart等。

4. 支持各种接入资源数据的类型以及接出数据类型。

5. 支持多路径流量，多管道接入流量，多管道接出流量，上下文路由等。

6. 可以被水平扩展。

Flume的优势：

1. Flume可以将应用产生的数据存储到任何集中存储器中，比如HDFS,HBase。

2. 当收集数据的速度超过将写入数据的时候，也就是当收集信息遇到峰值时，这时候收集的信息非常大，甚至超过了系统的写入数据能力，这时候，Flume会在数据生产者和数据收容器间做出调整，保证其能够在两者之间提供平稳的数据。

3. 提供上下文路由特征。

4. Flume的管道是基于事务，保证了数据在传送和接收时的一致性。

5. Flume是可靠的，容错性高的，可升级的，易管理的,并且可定制的。

## Flume的基本概念

### Flume总体架构

Flume日志采集传输系统中核心的角色是agent，agent本身是一个java进程，一般运行在日志收集节点，当然也可以运行在数据下沉的节点。Source、Sink、Channel三个组件是Flume运行的核心。

Flume的总体架构如图1-2所示。

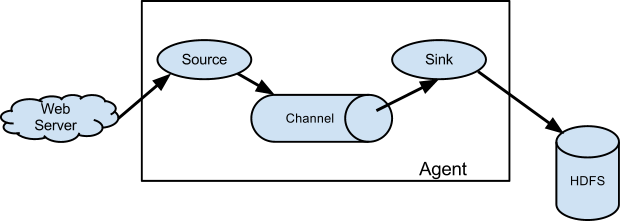


图 1-2 Flume总体架构

从图1-2 Flume总体架构图中可知，Flume作为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。简言之Flume就是用于日志收集的。

### Flume的核心概念

Flume的核心是把数据从数据源(source)收集过来，接着再将收集到的数据发送到指定的目的地(sink)。为了确保传输过程成功，在将数据发送到目的地(sink)之前，Flume会先缓存数据(channel),等数据真正到达目的地(sink)后，Flume再删除自己缓存的数据。

Flume Event - Flume事件

官方定义：被定义为一个具有有效荷载的字节数据流和可选的字符串属性集。

通俗定义：一条日志在flume中会被转换成一个JSON格式的串来传递，这个JSON

串就是一个FlumeEvent，具体的格式为{header:{头信息},body:日志内容}， 所以简单来说，一条日志在一个Flume中就对应一个JSON串，即一个FlumeEvent。

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是Flume传输数据的基本单位，它将对传输的数据进行封装。如果是文本文件，通常是一行记录。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。

Flume以事件的形式将数据从源头传送到最终的目的。Event由可选的header和载有数据的一个byte array 构成 , 载有的数据度flume是不透明的。Header是容纳了key-value字符串对的无序集合，key在集合内是唯一的。Header可以在上下文路由中使用扩展。

一个完整的Event包括：Event Headers、Event Body、Event信息。其中Event信息就是Flume收集到的日志记录。

Flume Agent- Flume 代理

官方定义：是一个进程承载从外部源事件流到下一个目的地的过程。包含Source

Channel和sink。

通俗定义：是Flume的基本组件，一个Agent就是一个基本的日志收集单元，由Source Channel和Sink组成。多个Agent之间还可以连接形成复杂的日志流动的网络。

一个Agent包含 source ，channel，sink 和其他组件。它利用这些组件将events从一个节点传输到另一个节点或最终目的地。agent是flume流的基础部分。flume为这些组件提供了配置，声明周期管理，监控支持。

Source - 数据源

官方定义：消耗外部传递给他的事件，外部源将数据按照Flume Source 能识别

的格式将Flume事件发送给Flume Source。

通俗定义：Agent中的组件，负责连接到数据源，获取数据原发送来的数据，将数据转换为FlumeEvent存储到Channel中。

Source组件是专门用来收集数据的，它可以处理各种类型、各种格式的日志数据，这里不妨把它理解为系统要处理数据的数据源。这些数据源包括avro、thrift、exec、jms、spooling directory、netcat、sequence generator、syslog、http、legacy、以及自定义的数据源等。Source负责接收event或通过特殊机制产生event，并将events批量的放到一个或多个Channel。source 必须至少和一个channel关联。

Flume Source 支持的类型

|  |  |
| --- | --- |
| Source类型 | 说明 |
| Avro Source | 支持Avro协议（实际上是Avro RPC），内置支持 |
| Thrift Source | 支持Thrift协议，内置支持 |
| Exec Source | 基于Unix的command在标准输出上的生产数据 |
| JMS Source | 从JMS系统（消息，主题）中读取数据，ActveMQ已经测试过 |
| Spooling Directory Source | 监控指定目录内数据变更 |
| Twitter 1% firehose Source | 通过API持续下载Twitter数据，试验性质 |
| Netcat Source | 监听某个端口，将流经端口的每一个文本数据作为Event输入 |
| Sequence Generator Source | 序列生成器数据源，生产序列数据 |
| Syslog Sources | 支持Syslog数据，产生Event，支持UDP和TCP两种协议 |
| Http Source | 基于HTTP POST或GET方式的数据源，支持JSON、BLOB表示形式 |
| Legacy Sources | 兼容老的Flume OG中的Source（0.9.x版本） |

Channel - 数据通道

官方定义：是一个被动的存储，用来保持事件，直到由一个Flume Sink消耗。

通俗定义：连接Source和Sink的通道，Source将Event写入Channel，Sink从Channel中消费Event。本质上是一种存储结构，用来缓冲Source和Sink之间速度不一致的问题，提供中间缓冲存储能力。

Channel位于Source和Sink之间，用于缓存进来的event。当sink成功的将event发送到下一个的channel或最终目的 event从channel删除。不同的channel提供的持久化水平也是不一样。Memory channel : 是一个不稳定的存储器，其原因是由于它在内存中存储所有的事件，如果Java进程死掉，任何存储在内存中的数据将会丢失。另外，内存空间受到RAM（一种存储器）大小的限制，而File Channel这方面是它的优势，只要磁盘空间足够大，他就可以将所有事件存储到磁盘。File Channel: 是一个持久化的存储器(Channel),它持久化所有的事件，并将其存储到磁盘中。因此，即使Java虚拟机当掉，或者操作系统崩溃或重启，再或者事件没有在管道中成功的传递到下一个代理（agent),这一切都不会造成数据的丢失。，JDBC channel：基于嵌入式database实现，channel支持事务，提供较弱的顺序保证，可以和任何数量的source和sink工作。

Flume Channel 支持的类型：

|  |  |
| --- | --- |
| Channel类型 | 说明 |
| Memory Channel | Event数据存储在内存中 |
| JDBC Channel | Event数据存储在持久化存储中，当前Flume Channel内置支持Derby |
| File Channel | Event数据存储在磁盘文件中 |
| Spillable Memory  Channel | Event数据存储在内存中和磁盘上，当内存队列满了，会持久化到磁盘文件（当前实验性的，不建议生产环境使用） |
| Pseudo TransationChannel | 测试用途 |
| Custom Channel | 自定义Channel实现 |

Sink - 数据汇聚点

官方定义：代表外部数据存放位置。发送flume中的事件到指定的外部目标。

通俗定义：连接到目的地，从Channel中消费Event，并将Event中的数据发送

给目的地。

Sink负责将event传输到下一个目的地或最终目的，成功完成后将event从Channel移除。Sink在设置存储数据时，可以向文件系统，数据库，hadoop存储据，在日志数据较少时，可以将数据存储在文件中，并且设定一定的时间间隔保存数据。在日志数据较多时，可以将相应的数据存储到Hadoop中，便于日后进行相应的数据分析。

有不同类型的Sink：存储event到最终目的的终端Sink。这些目的地包括hdfs、logger、avro、thrift、ipc、file、null、hbase、solr、自定义等。Sink必须作用于一个确切的Channel。

Flume Sink支持的类型

|  |  |
| --- | --- |
| Sink类型 | 说明 |
| HDFS Sink | 数据写入HDFS中 |
| Logger Sink | 数据写入数据文件中 |
| Avro Sink | 数据被转换成Avro Event，然后发送到配置的RPC端口上 |
| Thrift Sink | 数据被转换成Thrift Sink，然后发送到配置的RPC端口上 |
| IPC Sink | 数据在IPC上进行回收 |
| File Roll Sink | 存储数据到本地文件系统 |
| Null Sink | 丢弃到所有数据 |
| HBase Sink | 数据写入到HBase数据库 |
| Morphline Soir Sink | 数据发送到Soir搜索服务器（集群） |
| Elastic Search Sink | 数据发送到Elastic Search 搜索服务器（集群） |
| Kite Dataset Sink | 写数据到Kite Dataset，试验性质的 |
| Custom Sink | 自定义Sink实现 |

### 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当中包含3个核心的组件：Source、Channel、Sink。这种系统架构类似于生产者、仓库、消费者的架构模式。

Flume的一个完整的工作流程是这样的：Source组件不断的接收数据，将数据封装成一个一个的Event，然后将Event发送给Channel组件，Channel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。

需要注意的是只有在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的运行机制如果图1-3所示。

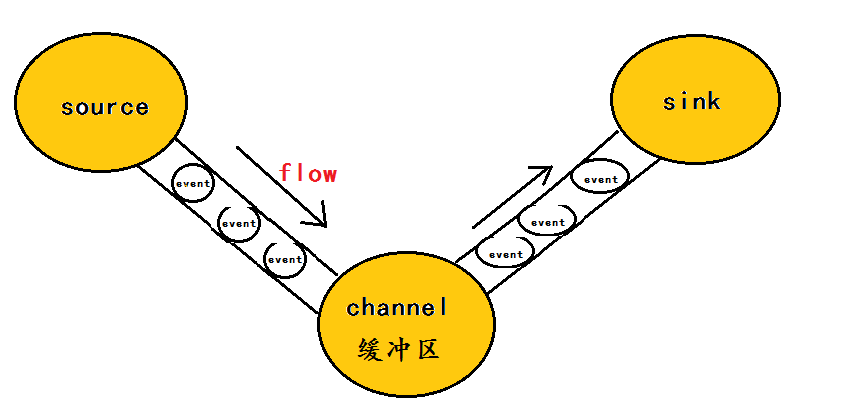


图 1-3 Flume运行机制

## Flume的基本用法

### Flume多级数据流动

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多级Agent，可以进行整体处理了。

多级流动的架构模式如图1-4所示。

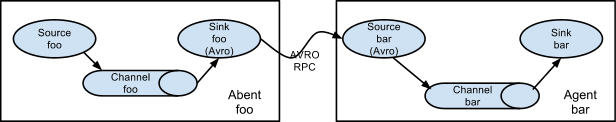


图 1-4Flume多级数据流动

此外，Flume还支持扇入(fan-in)流、扇出(fan-out)流。所谓扇入流就是Source可以接受多个输入，所谓扇出流就是Sink可以将数据输出到多个目的地中。下面具体来看一下扇入流和扇出流的架构方式。

### 数据的扇入流

在做日志收集的时候一个比较常见的场景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， 这种Flume系统架构方式被称为数据扇入流。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。这个场景在Flume中得以实现，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 ，当然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Agent合并到一个channel中，event被一个sink消费到它的最终的目的地。

这种扇入流的系统架构如图1-5所示。

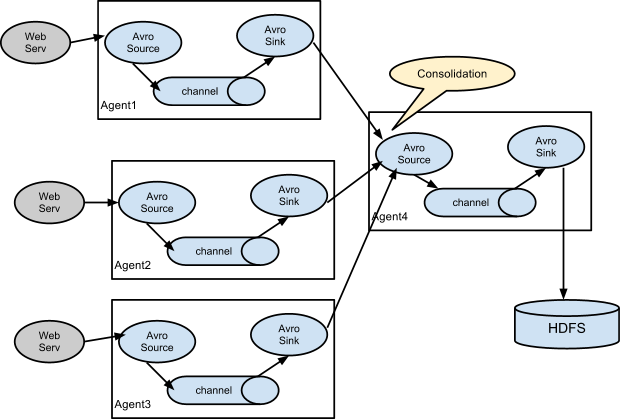


图 1-5Flume扇入流

### 数据的扇出流

Flume还支持多路输出Event流到一个或多个目的地，这种Flume系统架构方式称为数据扇出流。Flume数据扇出流架构是通过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。

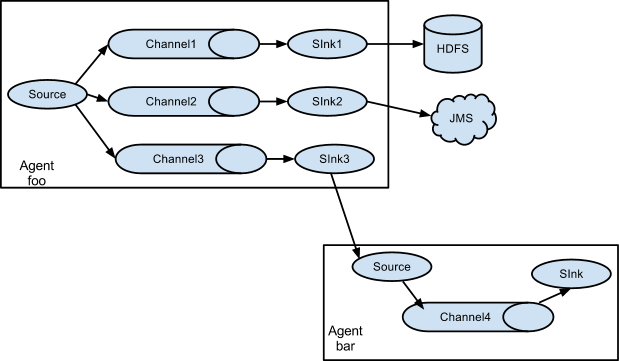


图 1-6 Flume扇出流

图1-6的架构展示了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被发送到所有的三个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编写在Agent的配置文件中。

Flume数据多级流动，数据扇入流，数据扇出流的具体实现会在后续的课程中有详细的介绍。

## Flume的特性

### 可靠性

Flume的核心是把数据源收集起来，再送到目的地。为了保证输送一定成功，再送到目的地之前，会先缓存数据，等到数据真正到达目的地后再删除缓存的数据Flume使用事务性的方式保证传送Event整个过程的可靠性。Sink必须在Event被存入Channel后，或者，已经被传达到下一站agent里，又或者，已经被存入外部数据目的地之后，才能把Event从Channel中remove掉。这样数据流里的event无论是在一个agent里还是多个agent之间流转，都能保证可靠，因为以上的事务保证了event成功存储起来。而Channel的多种实现可恢复性上有不同的保证。也保证了event不同程度的可靠性。比如Flume支持在本地保存一份channel作为备份，而memory channel 将event存在内存queue里，速度快，但丢失的话无法恢复。

当节点出现故障时，日志能够被传送到其他节点上而不会丢失。Flume提供了三种级别的可靠性保障，从强到弱依次分别为：end-to-end（收到数据agent首先将event写到磁盘上，当数据传送成功后，再删除；如果数据发送失败，可以重新发送。），Store on failure（这也是scribe采用的策略，当数据接收方crash时，将数据写到本地，待恢复后，继续发送），Besteffort（数据发送到接收方后，不会进行确认）。

### 可恢复性

Channel可以以内存(Memory)或文件(File)的方式实现，使用内存更快，但是不可恢复，而使用文件虽然比较慢但提供了可恢复性。

### 可扩展性

Flume采用了三层架构，分别为agent，collector和storage，每一层均可以水平扩展。其中，所有agent和collector由master统一管理，这使得系统容易监控和维护，且master允许有多个（使用ZooKeeper进行管理和负载均衡），这就避免了单点故障问题。

### 可管理性

所有agent和colletor由master统一管理，这使得系统便于维护。多master情况，Flume利用ZooKeeper和gossip，保证动态配置数据的一致性。用户可以在master上查看各个数据源或者数据流执行情况，且可以对各个数据源配置和动态加载。Flume提供了web 和shell script command两种形式对数据流进行管理。

## Flume的安装配置

### Flume的安装步骤

启动一台Linux虚拟机

安装jdk(1.6或以上版本)

配置JAVA\_HOME环境变量：vim /etc/profile，如图1-7所示



图1-7

执行命令：source /etc/profile，文件立即生效。

下载flume的安装包

下载的地址为：[http://flume.apache.org/download.html](http://flume.apache.org/download.html" \t "_blank)

上传flume的安装包

利用FZ软件将flume安装包上传指定的目录

解压安装flume 安装命令如图1-7所示:



图1-7 安装Flume

conf目录主要是存储文件的，在Flume安装目录下的conf目录中创建一个配置文件，如：source-nc.conf，这个文件名是自定义的，无论是主文件名还是扩展名都可自定义。

配置agent组件。

### Flume配置文件及其详解

在1.5.1小节的Flume的安装步骤中，第五步是在conf目录中创建一个配置文件，其操作步骤如图1-8所示。

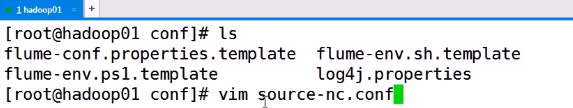


图1-8创建Flume配置文件

如果要配置的Flume其Agent的名字为a1, 则配置文件清单如下。

source-nc.conf文件示例：



具体配置参数可参照表1-2所示。

表 1-2 Flume配置文件详解

|  |  |
| --- | --- |
| **配置项** | **说明** |
| **channels** | 指定有几个通道并为每个通道起个名字。 |
| **type** | 类型名称，需要被设置为"netcat" |
| **port** | 指定要绑定到的端口号 |
| **max-line-length** | 512        单行最大字节数 |
| **ack-every-event        true** | 对于收到的每一个Event是否响应"OK" |

在表1-2所示的配置参数中前三项为必配项，其它各项均为可选配置项。

### 启动Flume agent

Flume配置文件创建完成后，需要执行如图1-9 所示的指令来启动Flume。

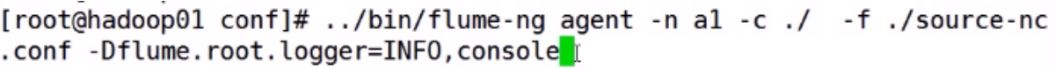


图 1-9 启动Flume

Agent将在配置目录中读取log4j.properties文件,并根据日志中的规范记录日志。Log4j配置的细节可以在log4j文档中找到【log4j】。一旦Agent启动，它继续运行，直到被命令杀死，导致Agent关闭。Agent记录的所有的日志到log4j日志(可能看起来像Agent没有运行，尽管它实际上在运行做它应该做的工作，没有信息被记录在控制台)。

在启动Flume的命令中有一些参数，这些参数所表示的含义可参照表1-2所示。

表1-3 启动 Flume参数含义

|  |  |
| --- | --- |
| **参数参数** | **描述描述** |
| **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 |
|  |  |

以上命令如果执行成功，会显示如图1-10所示的结果。

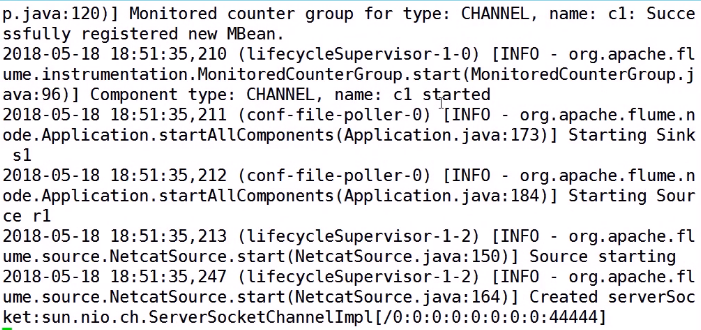


图 1-10 Flume启动成功

用jps查看进程，出现Application代表flume进程启动成功，如图1-11，



图1-11 进程查看

### 访问Flume

Flume启动成功后，就可以访问Flume了。这里通过Netcat客户端来访问Flume，Netcat客户端是一个网络通讯工具，可以通过Netcat客户端发起网络的通讯请求。首先需要安装Netcat客户端，Netcat客户端的安装文件可在本章提供的电子资料里查找。 Netcat客户端的安装过程如图1-11所示。

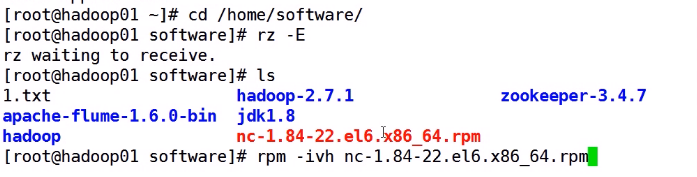


图1-11 安装nc客户端

在图1-11中使用cd 命令进入到/home/software目录中，在此目录下有Netcat的安装文件，该文件是一个rpm文件，因此我们使用 rpm命令运行该文件，这样Netcat客户端就安装成功了。Netcat客户端是否安装成功，可以使用Netcat命令来进行测试，如果1-12 所示。

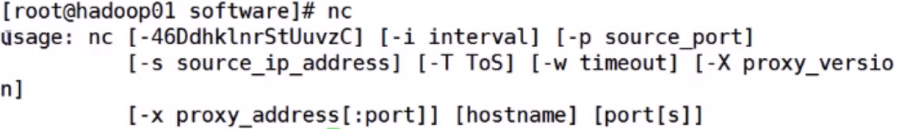


图1-12 测试Netcat是否安装成功

如果出现图1-12所示的结果，则说明Netcat客户端被安装成功。Netcat客户端被安装成功后，就可以访问Flume了。使用Netcat命令访问本机的44444端口，需要执行如图 1-13所示的命令。



图 1-13访问Flume

在图1-13中访问了运行于本机并对44444端口进行监听的Flume，同时发送了数据”hello1712“。数据发送成功后，Flume会接收到数据并打印在控制台，如图1-14所示。

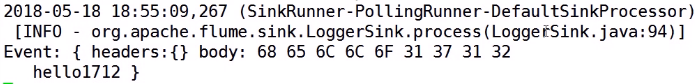


图1-14 Flume接收数据

如果数据发送成功，则Flume接收到数据并打印在控制台

# Flume核心组件

## Source组件

Source，像所有的Flume组件一样，需要在配置文件中指定的类型。可以是FQCN，或者内置Source的别名。所有的Source都需要至少一个用于写入的正确配置的Channel。因此，Channel的列表也是对于Source合理配置需要考虑的必须参数。

Source被命名为任何其他部件一样通过配置文件进行配置的组件。Flume的配置系统验证每个Source的配置和屏蔽错误配置的Source。通过配置系统完成的验证相当的少，不过，Flume的配置系统可以确保：

每个Source至少有一个正确的Channel连接它。

每个Source有一个定义的type参数。

Source是在Agent中Sources里面的活动表。

配置系统一旦验证通过一个Source，就会实例化并且由ConfigurationProvider进行配置。如果Source缺少配置或者缺少需要的参数，这个Source将被屏蔽。Source一旦成功配置之后，Flume的生命周期管理系统将会尝试启动Source。只有Agent自身停止或被杀死、或者Agent被用户重新配置，Source才会停止。

Source，像所有的Flume组件一样，需要在配置文件中指定它们的类型。可以是FQCN，或者内置Source的别名。所有的Source都需要至少一个用于写入正确的配置Channel。因此，Channel的列表也是对于Source合理配置需要考虑的必需参数。表2-1里展示了所需的配置的参数。

表2-1 Source的配置参数

|  |  |
| --- | --- |
| **参数** | **描述** |
| **Type** | Source的类型。可以是FQCN或者Source的别名（仅为Flume自带的Source）。类必须安装使用“plugins”一节中描述的plugins.d框架。每个Flume封装Source的别名的相关部分都会提及。 |
| **Channels** | Source写入Channel的时候应该以空格分隔。Channel选择器基于事件和路由参数会选择出用于写入的Channel，Source会按照配置文件中指定的顺序将事件写入这些Channel |

Source组件是专门用来收集数据的，它可以处理各种类型、各种格式的日志数据，这里不妨把它理解为系统要处理数据的数据源。这些数据源包括avro、thrift、exec、jms、spooling directory、netcat、sequence generator、syslog、http、legacy、以及自定义的数据源等。首先来看第一种类型的数据源:Avro Source。

### Avro source

Flume主要的RPC Source是Avro Source。Avro Source被设计为高扩展的RPC服务器端。能从其他的Flume Agent的Avro Sink或者使用Flume的SDK发送数据的客户端应用，接收到一个Flume Agent中。Avro Source和Avro Sink的组合代表了Flume内部的通信原理（Flume Agent之间）。Avro Source的可扩展性结合Channel担当了缓冲器的角色，是的Flume Agent能够处理重要的负载峰值。

Flume的Avro Source使用Netty-Avro inter-process的通信（IPC）协议来通信。因此可以用Java和JVM语言发送数据到Avro Source。如果你想从应用使用Avro Source发送给一个Agent，可以利用SDK来完成。

众所周知avro是一个序列化的框架，Flume通过监听avro 端口来接收外部avro客户端的事件流。与necat不同的是， avro-source接收到的是经过avro框架序列化后的数据，然后反序列化数据继续传输。所以，如果是avro-source的话，其源数据必须是经过avro序列化后的数据。而netcat接收的则只是字符串格式。

利用Avro source可以实现多级流动，如图2-2所示。

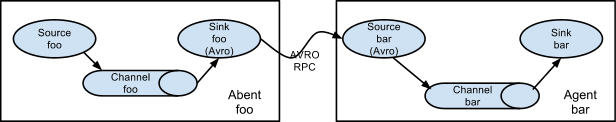


图 2-2 Flume多级流动

扇入流效果，如图2-3所示。

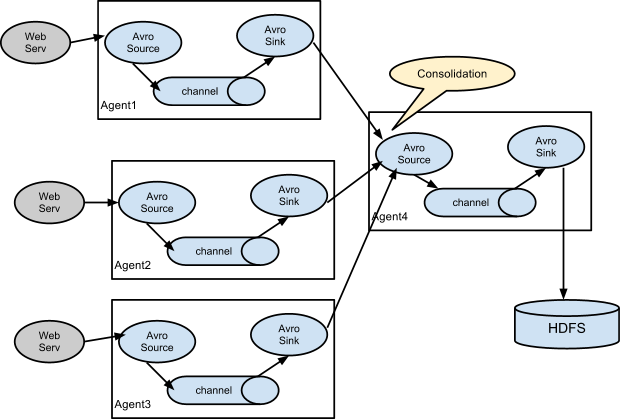


图2-3 Flume扇入流

扇出流效果，如图2-4所示。

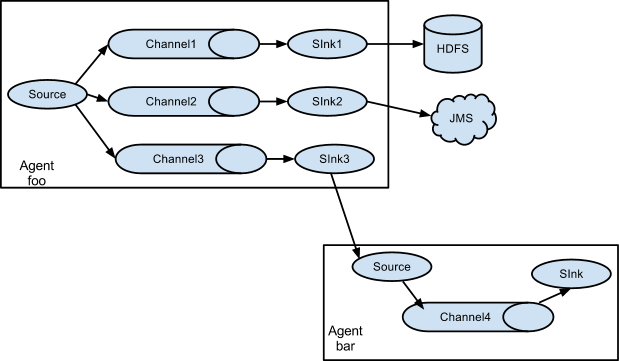


图2-4 Flume扇出流

对于数据的多级流动，数据的扇入与扇出等流动方式是其它source所无法实现的。此外，也可以接收通过flume提供的avro客户端发送的日志信息。

下面通过示例来完成avro source的配置。首先进入到conf目录中，创建一个source-avro.conf文件，其详细内容如下列代码段所示。

source-avro.conf 示例：



通过将以上配置文件与第一章中的source-nc.conf配置文件进行对比，会发现两者唯一的区别在于a1.sources.r1.type的值，前者为nccat,而avro sourcer的type则为avro。

avro source的详细的配置选项说明可参照表2-1所示。

表 2-1 avro source配置选项说明



在此配置参考表中只有前四项为必配项，其它各项均为可选项。以最简单的式配置Avro Source需要参数的最小集合。最少的情况下，Source需要两个除了必需参数以外的参数：bind和port这两个参数指明Source使用的socket地址。如果有多个网络接口，Avro Source可以绑定一个或所有接口。如果只是绑定其中的一个接口，简单地设置那个接口的IP地址或域地址为bind参数的值即可，如果要绑定所有的接口，设置bind参数的值为0.0.0.0。port参数为配置的绑定地址指明了Source应该监听的端口号。

配置文件创建完成后，就可以启动Flume了，启动方式如图2-5所示。



图2-5启动Flume

运行启动命令之后如果看到如图2-6所示的结果，则说明启动成功。

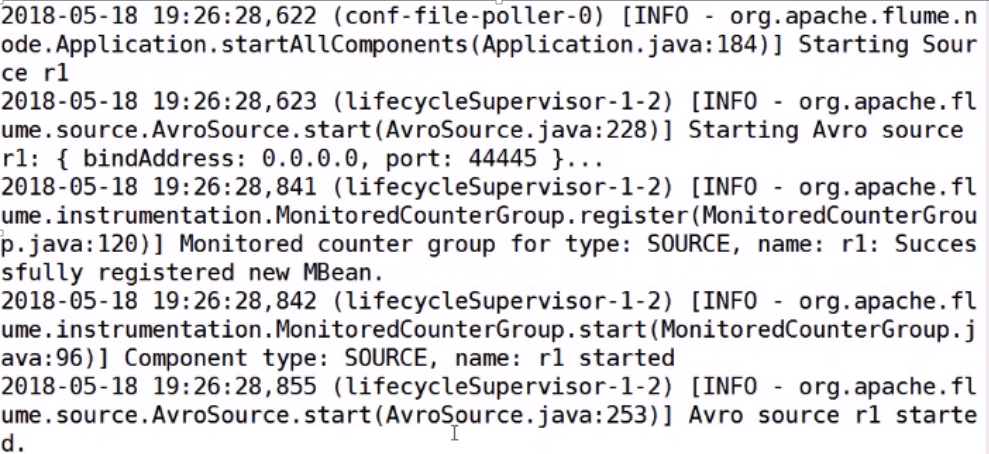


图2-6 Flume启动成功

具有avro source的Flume启动成功后就可以接收avro客户端的访问了。为了便于用户的测试，Flume在其安装目录的bin文件夹下提供了一个flume-ng指令。用户可按照图2-7所示的步骤进入到bin文件夹。

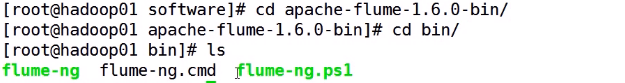


图 2-7 flume-ng指令

进入bin目录后，在此目录下创建一个txt文件，作为访问Flume时传递序列化的数据，可参照图2-8所示的方式来创建文件。



图2-8 创建文件

文件创建完成后，使用flume-ng指令模拟avro客户端，对1.txt进行序列化并发送给flume。具体操作指令如图2-9所示。



图2-9 flume-ng访问flume

请求发送成功后在flume端会接收到avro客户端发送的数据，如图2-10所示。

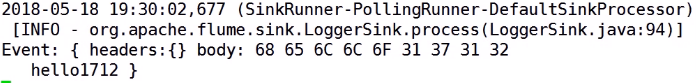


图 2-10 flume接收avro客户端数据

对于avro source，它可以实现数据的多级流动，数据的扇入，数据的扇出等架构方式，但是需要结合avro sink来实现。在后续内容中，我们会根据sink知识点的完善，进一步实现这三种复杂的数据流架构。接下来，继续学习Flume第二种source，即 Spooling Directory source。

### Spooling Directory source

Spooling Directory Source监视读取事件的目录。尽管新文件可以被实时地添加到该目录，Source还是期望目录中的文件是不变的。文件一旦被移入到该目录，它不应该被写入。如果你在处理日志文件， 这样做的一个好方法是配置日志系统当日志滚动时移动文件。Source也期望文件名是不重名的。如果这两种情况发生一种，Source 将会抛出异常并终止。这时重启Source的唯一方式是重启Agent自身。

Spooling Directory Source唯一的缺点是数据不是实时跟踪的，并且只能在文件关闭或者移入到相关目录时才能读取文件。文件一旦被Source完全使用且所有的事件被成功写入Source的Channel中，Source就可以基于配置重命名文件或者删除文件。当文件重命名，Source只是给文件添加一个后缀，而不是完全改变它。这个后缀也是可以配置的。

Spooling Directory Source的特点是将指定的文件加入到“自动搜集”目录中。而Flume则会持续监听这个目录，目录中一旦有新文件添加进来，该文件会被当作source来处理。需要注意的是一旦文件被添加到“自动收集”目录，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。

接下来演示如何创建一个Spooling Directory Source。

首先进入到conf目录，创建一个名为source-spooldir.conf的文件，其详细内容如下列代码段所示。

source-spooldir.conf配置文件示例:



在Spooling Directory Source的配置文件中，有两行代码是加粗显示的，a1.sources.r1.type=spooldir配置数据源的类型为监听目录类型，a1.sources.r1.spoolDir=/home/work控制所监听的目录此，此监听目录在访问Flume之间一定要事先创建好，否则会报错。

### Http Source

Flume自带的HTTP Source可以通过HTTP POST。对于应用程序环境，它可能不能部署Flume SDK及它的依赖项，或者客户端代码倾向于通过HTTP而不是Flume的RPC发送数据的情况，HTTP Source可以用来将数据接收到Flume中。从客户端的角度来看，HTTP Source表现得像web服务器一样能接收Flume事件。

这种类型的source能够接收HTTP的GET和POST请求作为Flume的事件，其中GET方式只用于测试。

首先我们来创建 Flume Http Source的配置文件，进入到Flume的安装目录conf下面，创建一个名为source-http.conf文件。

此文件的具体配置信息参考以下source-http.conf代码示例。

source-http.conf配置文件示例:



参照以上示例创建了配置文件后，就可以启动Flume了，启动命令如图2-20所示。



图2-20启动Flume

Flume启动成功后显示如果2-21所示的结果。

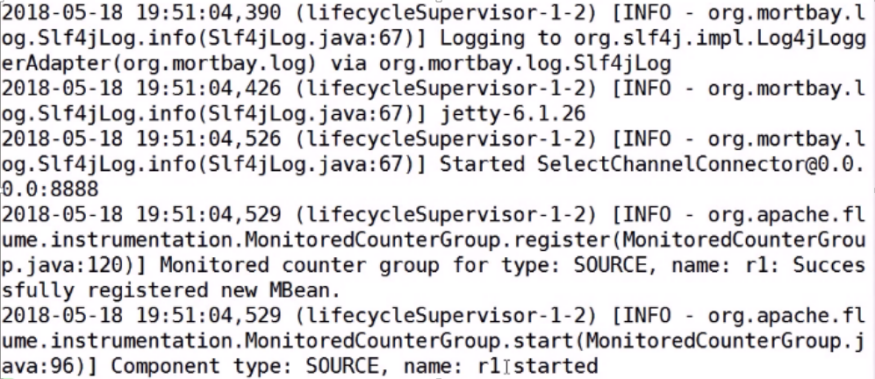


图2-21 Flume启动成功

接下来就可以向该配置文件下的Flume发送Http的请求了，请求方式可以参考图2-22。

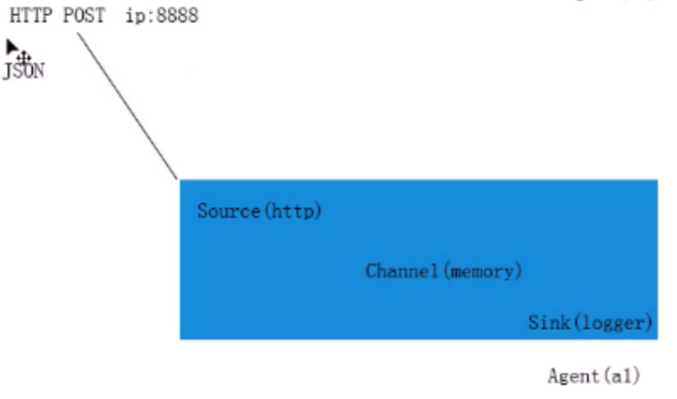


图2-22发送Http请求

从图2-22中可知当向Flume发送Http请求时需要提供请求的主机端口，以及向Flume发送的数据，而且数据需要以JSON格式的字符串进行封装，只有这样Flume才可以接收到请求并且收集到数据，具体命令及发送数据的内容如图2-23所示。



图2-23发送http请求

从图2-23中可知，在发送POST请求时通过-d参数设置要发送的数据，在JSON字符串中需要通过headers键来设置数据的头信息，通过body键来设置具体的数据内容。请求发送成功后我们可以看到图2-24所示的结果。

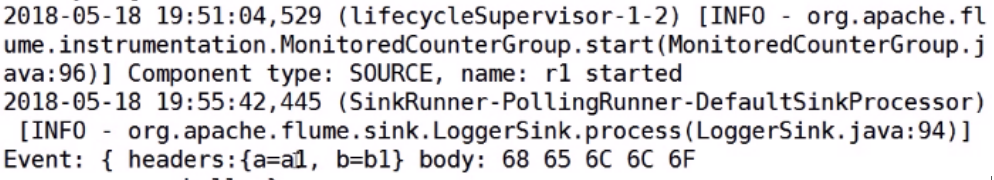


图2-24 Http请求结果

在请求结果中看到Http请求通过 JSON封装的数据 headers及body被Flume收集并打印输出。至此我们介绍了Flume中的六种Source，接下来介绍Flume中另一种比较重要的组件:Sink。

## Sink组件

Sink组件在Flume中的作用是用于把数据发送到指定的目的地，这些目的地包括hdfs、logger、avro、thrift、ipc、file、null、hbase、solr、自定义等。首先来学习第一种Sink组件：Logger Sink。

Sink使用标准的Flume配置系统进行配置。每个Agent可以没有Sink或若干Sink,每个Sink只能从一个Channel中读取事件。如果Sink没有配置Channel，那么Sink 从Agent中被移除。配置系统保证:

每个Sink至少有一个正确配置的Channel连接它。

每个Sink有一个定义的type参数。

Sink 是在Agent中Sink活跃列表中的。

类似于Source, 每个Sink (如果配置正确)被ConfigurationProvider实例化然后被配置。配置错误和configure方法抛出异常的Sink会从Agent中被移除。Sink一旦配置好，就会被配置系统启动。Flume 可以聚合线程到Sink组，这个我们将在后面讨论。每个Sink组可以包含一个或多个Sink。如果一个Sink没有定义Sink组，那么该Sink可以被认为是在一个组内，且该Sink是组内的唯一成员。

每个Sink运行器运行一个Sink组，Sink 运行器是个针对Sink组循环调用 process 方法的线程，依次转发调用到组里其中一个Sink的process方法。如果Sink可以立即处理更多的事件，通常是当前调用至少处理了一个事件的时候，process方法就返回Status . READY。如果Sink不能处理任何事件，Sink返回Status . BACKOFF，表明运行线程在再次尝试之前应该回退一段时间。为Sink定义Sink组不是强制的。如果一个 Sink是活跃的，但是不在Sink组里，那么它就获取了一个针对自身的Sink运行器，调用Sink的process方法。针对该Sink的回退语义和Sink组里的相同。

Sink可以在配置中指定它们的FQCN或内置Sink的别名来指定。每个Sink必须有一个配置的Channel。只有当Channel配置正确且启动成功，Sink才会启动。表2-2出了需要的参数。

表2-2 所有Sink必须配置参数

|  |  |
| --- | --- |
| **参数** | **描述** |
| **type** | Sink的类型。可以是FQCN或Sink的别名（这只是针对Flume内置的Sink）。该类必须在Flume的环境变量中。Flume内置Sink的别名在相关部分将被提及 |
| **channels** | 读取事件的Channel |

类似于其他所有的组件（除了Memory Channel），当Flume被重新配置，Sink的实例是不会被重用的。因此，Sink不应该缓冲任何已被读取和提交的事件。这是因为提交事务表明Channel可以删除在事务上下文中读取的事件。当Agent重启时，Sink中缓冲的这些事件可能会造成数据丢失，因为数据没有在持久的存储或缓冲中，而是在可能不能重用的Sink的内部数据结构中。一旦提交，Sink必须保证事件已经被提交到下一个Agent的Channel（在RPC Sink的情况下）或已经持久化到最终目的地。

### Logger Sink

Logger Sink用于记录指定级别（例如INFO，DEBUG，ERROR等）的日志，通常用于调试。

在使用Logger Sink时要求在--conf参数指定的目录下有log4j的配置文件。可参照图 2-25所示。

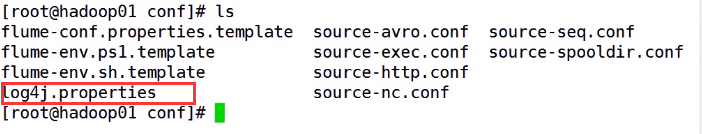


图2-25 Flume conf 目录

以编辑方式打开log4j.properties文件，可看到如图2-26所示的内容。

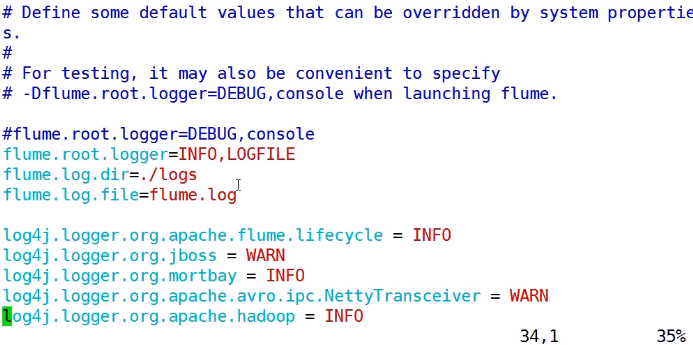


图2-26 log4j配置文件

根据设计，logger sink将要处理的内容限制为16个字节，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，如可以使file\_roll sink，它会将日志写到本地文件系统中。

在Flume的配置文件中如何设置 logger类型的sink，可参照以下代码。

sink-logger.conf参考示例:



### File Roll Sink

File Roll Sink能够在本地系统中存储事件，并且可以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。其运行模式如图2-27所示。

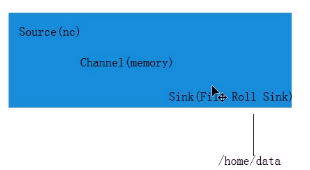


图2-27 File Roll Sink 运行模式

File Roll Sink的配置方式可参照以下sink-fileroll.conf代码。

sink-fileroll.conf参考示例：



在sink-fileroll.conf配置文件中通过a1.sinks.s1.type=file\_roll将sink的类型配置为file\_roll类型，通过a1.sinks.s1.sink.directory=/home/rolldata指定产生的文件存放的目录，通过a1.sinks.s1.sink.rollInterval=60设置产生文件的时间间隔。

配置文件创建完成后基于sink-fileroll.conf文件启动Flume。启动命令可参照图2-28所示。



图2-28 启动Flume

访问Flume并且发送数据，操作过程如图2-29所示。

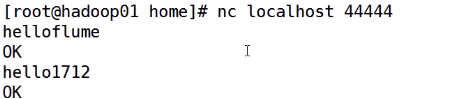


图2-29 访问Flume

访问Flume的指令发出后，进行入/home/fileroll目录，发现在该目录下多了一个文件，该文件就是在收集数据时产生的文件，如图2-30所示。

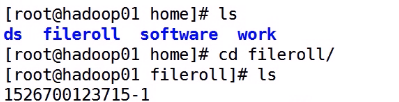


图2-30 file roll文件

以编辑方式打开该文件，发现在文件在保存了我们在访问Flume时发送的数据。如图2-31所示。

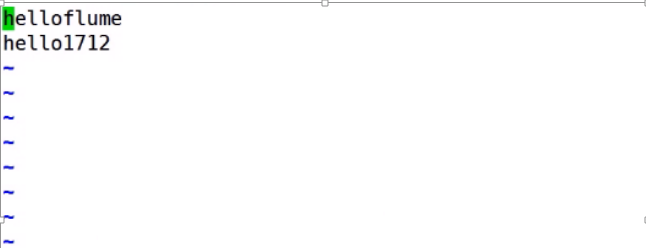


图2-31 file roll 存储的数据

当间隔一定的时间，如果我们再查看/home/fileroll文件夹会发现在此文件夹下又多了一个文件，这个新文件就是在指定时间间隔后又生成的新文件，此时如果我们再往Flume发送数据，数据将会被新的文件收集。这就是File Roll Sink的特点。接下来介绍Flume的第三种Sink: HDFS Sink。

### HDFS Sink

HDFS Sink将事件(event)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。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。

首先创建HDFS Sink的配置文件：sink-hdfs.conf，参考代码如下所示。

sink-hdfs.conf示例代码。



在上述配置文件中通过a1.sinks.s1.type=hdfs 配置sink的类型为hdfs，通过a1.sinks.s1.hdfs.path=hdfs://192.168.234.21:9000/flume配置分布式文件系统的路径为hdfs://192.168.234.21:9000/flume,通过a1.sinks.s1.hdfs.fileType=DataStream配置文件的类型为文本文件。

配置文件创建完成之后，接着启动Hadoop 伪分布式文件系统。如图2-32所示。



图2-32 启动Hadoop 伪分布式系统

Hadoop伪分布式系统启动之后，接下来再启动Flume,如图2-33所示。



图2-33 启动Flume

然后通过NetCat客户端向Flume发送数据，如图2-34所示。

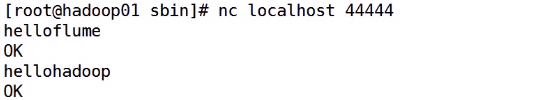


图2-34向Flume发送数据

数据发送成功后可通过HDFS插件查看到Flume把收集到的数据保存到了分布式文件系统的flume目录下。如图2-35所示。

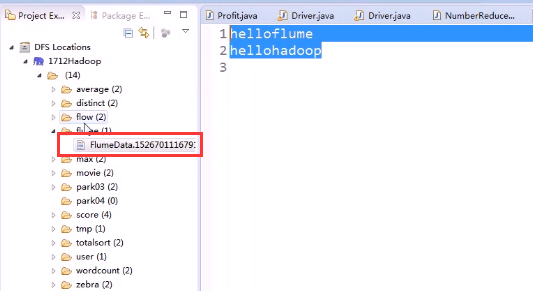


图2-35 HDFS 数据

以上是关于HDFS Sink的使用步骤。在2.1.1小节中介绍Avro Source时曾谈到Flume在收集数据时，根据不同的Source和Sink能够实现数据的多级流动，数据的扇入流动及扇出流动等构架，那么要想实现以上的三种架构模式需要将Avro Source和Avro Sink结合在一起使用。而在2.1.1小节中我们已经介绍了Avro Source，接下来介绍Avro Sink。

### Avro Sink

Avro Sink使用Avro的Netty-based RPC 协议发送数据到Avro Source。它被实现为一个感知事务的Flume NettyAvroRpcClient的包装器。因此，它们共享一些配置参数。Avro Sink可以批量发送事件到Avro Source。这很重要，因为Avro Source不控制每个事务写出事件的数量----它将写出整个批次的事件。对于File Channel，重要的是要以合理的批量写出，因为Channel将同步每个数据事务的数据文件到磁盘。为了避免太多的同步操作，批量大小必须适度大。

大型批量大小的另一个原因是，发送的每个批处理作为RPC调用。小批量可能意味着RPC开销在有效负载占据了很大比例，这应该避免。最好的批处理大小因为每个部署会有所不同。与HDFS Sink 类似，它将取决于所使用的硬件、网络，甚至配置(如果在特有Sink和持有写入Source的两个Agent上批量大小都是大的，File Channel将表现得更好)。

创建配置文件前我们需要再启动第二台服务器:hadoop02，第二台服务启动后还需要在第二台服务器上配置Flume的环境，因为在第一台服务器hadoop0上已经有了Flume的环境，所以我们可通远程复制的方式将第一台服务器的Flume环境复制到第二台服务器上，具体操作方式可参考图2-38所示的命令。

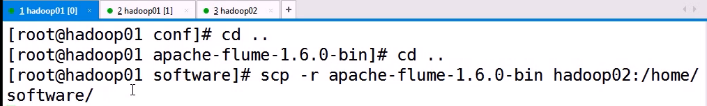


图2-38远程复制Flume环境

将两台Linux虚拟机的Flume环境准备完成后，就可以分别为两台Flume编写配置文件了，第一台Flume Agent配置文件具体可参考以下示例内容。

sink-avro.conf代码示例:



先后启动两台Flume Agent之后，通过NetCat客户端向第一台Agent发送数据，如图2-39所示。



图2-39 发送多级流动数据

请求发出之后如果第二台Flume agent能够接收到数据，说明多级流动架构配置成功了。如图2-40所示。

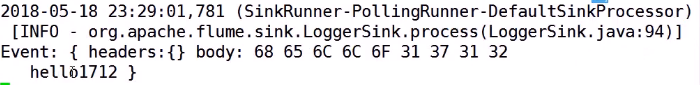


图2-40 多级流动接收数据

在以上的示例中所介绍的多级流动架构共有两级Agent，如果想要更多级别的多级流动的Agent，则配置思想是一样的，只需要更改当前最下游的配置文件中的sink的类型，并添加其下游Agent的IP地址和监听的端口号即可

## Channel组件

Channel是位于Source和Sink之间的缓冲区。因此，Channel允许Source 和Sink在不同速率上运作。Channel是Flume可以保证数据不丢失(当然，这是在正确配置的情况下)。Source 写入数据到一个或多个Channel中,再由一个或多个Sink读取，但是Sink只能从一个Channel读取数据， 而多个Sink可以从相同的Channel读取数据。Channel 本质上是事务性的，允许Flume为写进Channel的数据提供明确的保证。

位于Source和Sink之间作为缓冲区的Channel操作，有几个优点。Channel允许

Source在同一Channel上操作，以拥有自己的线程模型而不必担心Sink从Channel读取数据，反之亦然。位于Source和Sink之间的缓冲区也允许它们以不同的速率进行工作，因为写操作发生在缓冲区的尾部，读取发生在缓冲区的头部。这也使得Flume Agent能处理Source“高峰小时”的负载，即使Sink无法立即读取Channel。Channel允许多个Source 和Sink在它们上面进行操作。

### File Channel

File Channel是Flume持久的Channel。它将所有事件写到磁盘中，因此在程序关闭或机器宕机的情况下不会丢失数据。File Channel保证了即使机器或Agent宕机或重启，只有当Sink取走了事件并提交给事务时，任何提交到Channel的事件才从Channel移除。它被设计为高并发且可以同时处理多个Source和Sink。

channel-file.conf代码示例。



在上述channel-file.conf配置文件中通过a1.channels.c1.type=file将Channel的类型配置为file类型，通过a1.channels.c1.dataDirs=/home/data 将文件保存的目录配置为/home/data。配置文件创建好了以后就可以启动Flume以进行测试了，启动命令如图2-51所示。



图2-51启动Flume

Flume启动成功后，进入到虚拟机home目录，发现在该目录下会自动成功一个子目录，即data目录，如图2-52所示。

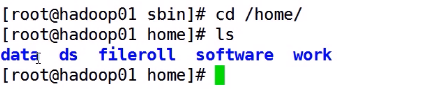


图2-52 data目录

然后向本机的44444端口发送数据，如图2-53所示

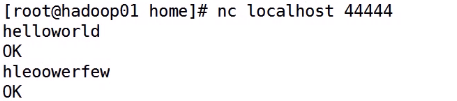


图2-53发送数据

数据发送完成后入到/home/data目录，会发现在该目录了产生了目录文件，如图2-54所示。

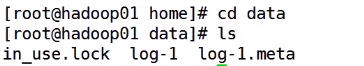


图2-54日志文件

### Memory Channel

Memory Channel 是内存中的Channel，在堆上存储写入的事件。实际上，Memory

Channel是内存中的队列---Source从它的尾部写入，Sink从它的头部读取。Memory支持很高的吞吐量，因为它在内存中保存了所有的数据。如前面所说，Channel是线程安全的，可以同时处理几个Source的写入操作和几个Sink的读取操作。Memory Channel在不需要关心数据丢失的情景下适用，因为该类Channel中没有将数据持久化到磁盘。如果需要关心数据的丢失，那么Memory Channel就不应该使用，因为如果程序死亡，机器宕机或者重新启动，都会造成数据的丢失。