# 知识回顾

## Hadoop基础

Hadoop：分布式海量离线数据存储处理平台

重要组成部分：

HDFS：海量数据的存储（起源于Google论文《Google File System》）

MapReduce：海量离线数据的处理（起源于Google论文《MapReduce》）

Yarn：集群资源协调管理（使Hadoop.HDFS可以对接更多的应用（如Hbase、Spark等））

Hadoop1.0：HDFS、MapReduce

Hadoop2.0：HDFS、MapReduce、Yarn

Hadoop3.0：HDFS、MapReduce、Yarn

## HDFS详解

纵向拓展缺点：不能无限拓展，不安全，成本增长非线性，读写速度慢

横向拓展的优点：理论上无限拓展，相对安全，成本增长友好，读写速度相对快

横向拓展也就是集群的理念，不光在存储，在处理上也一样。（某种意义上讲，分布式是硬件条件达不到的妥协）

### HDFS组成

NameNode：HDFS中的老大，负责记录元数据信息，分派任务。

SecondaryNameNode：HDFS中的小秘，帮助NameNode完成文件合并的工作。

DataNode：HDFS中的小弟，负责存储文件块，记录存放文件块的部分信息。

Block：文件块，由原文件切割备份而来，1.0版本默认64M，2.0版本128M，对于不满128M的文件不会占用128M空间，只会占用实际大小的空间。

# Flume

分布式日志收集系统

支持接收多种数据来源，可以对日志信息进行简单处理，然后写出到数据存储系统中。

## 重要概念

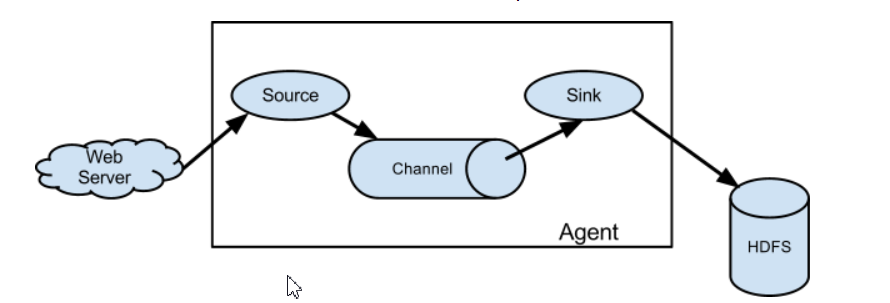
Event：事件，数据载体，flume将一批日志数据包装成为一个event进行传输处理，其结构非常简单，就是json串。Eg：{“headers”:info,”body”:info}其中header中的数据是我们自定义的一些内容，或者默认的内容。Body中的数据就是日志本身。

Agent：代理，就是flume包装、承载、传输event到目的地的过程。这个过程中包含三部分（source、channel、sink）。

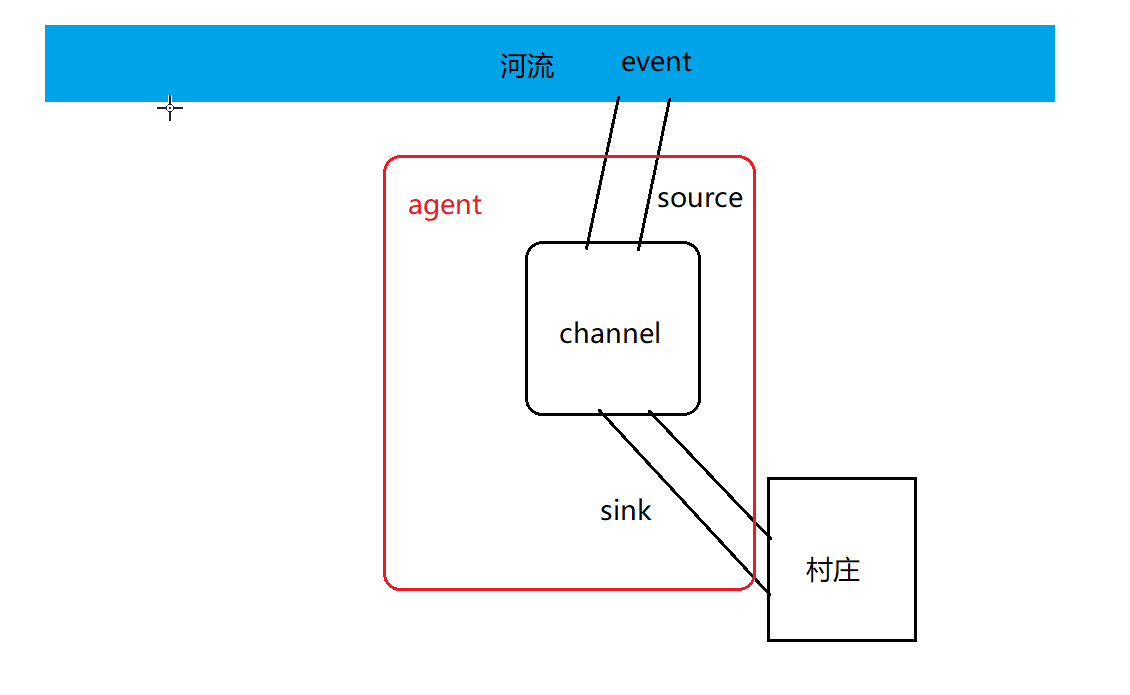
Source：数据源，接收日志信息并将其包装为一个一个的event，传到channel中。

Channel：通道，被动接收source传来的数据，并将其暂时存储，等待sink的消费。

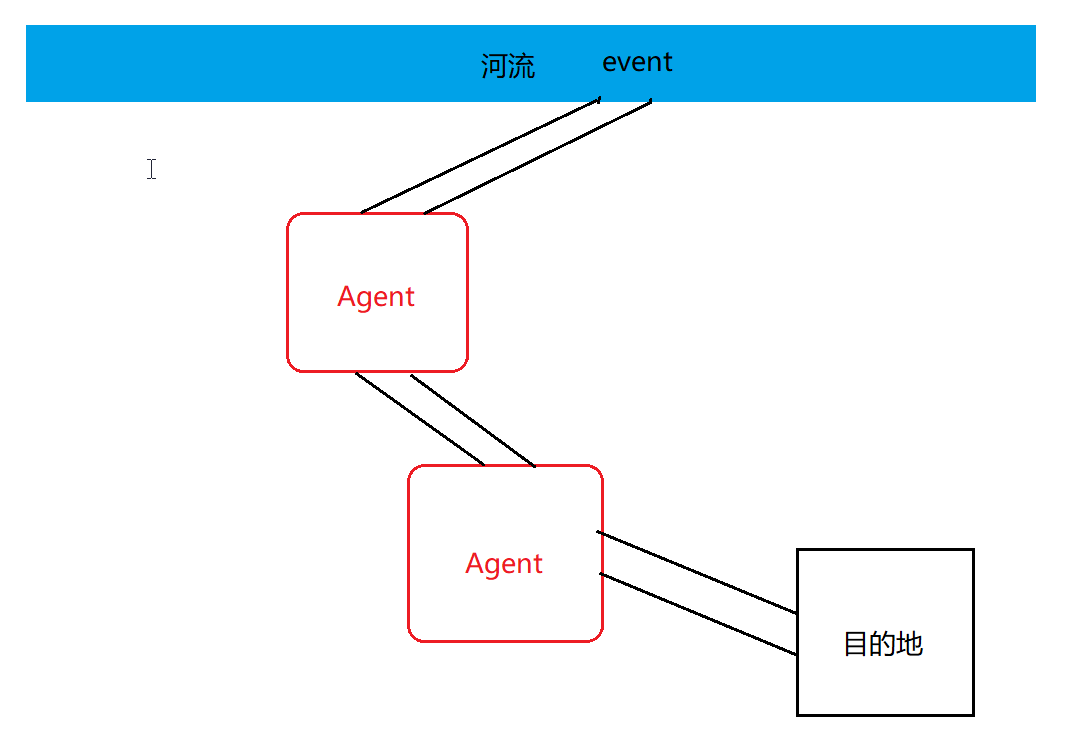
Sink：水槽，指定该agent数据传输的目的地，消费channel中的数据加载到目的地。



上图表示：利用flume将webserver产生的日志，传输到HDFS中保存的过程。

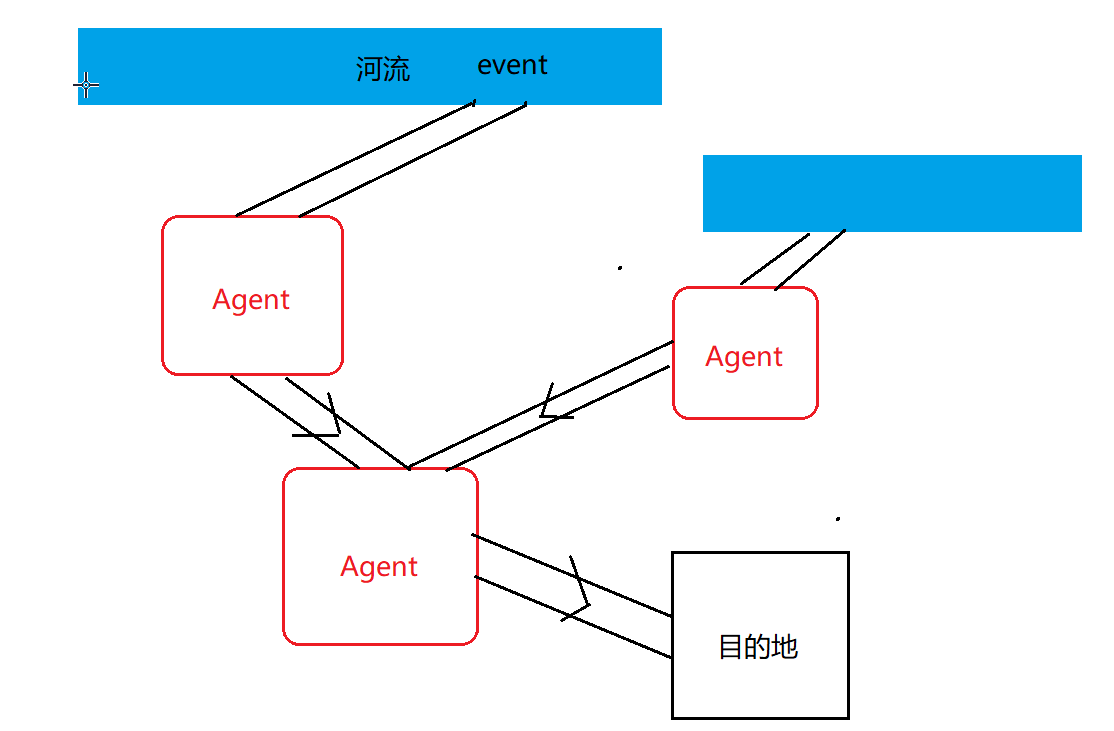


## Flume多级流动

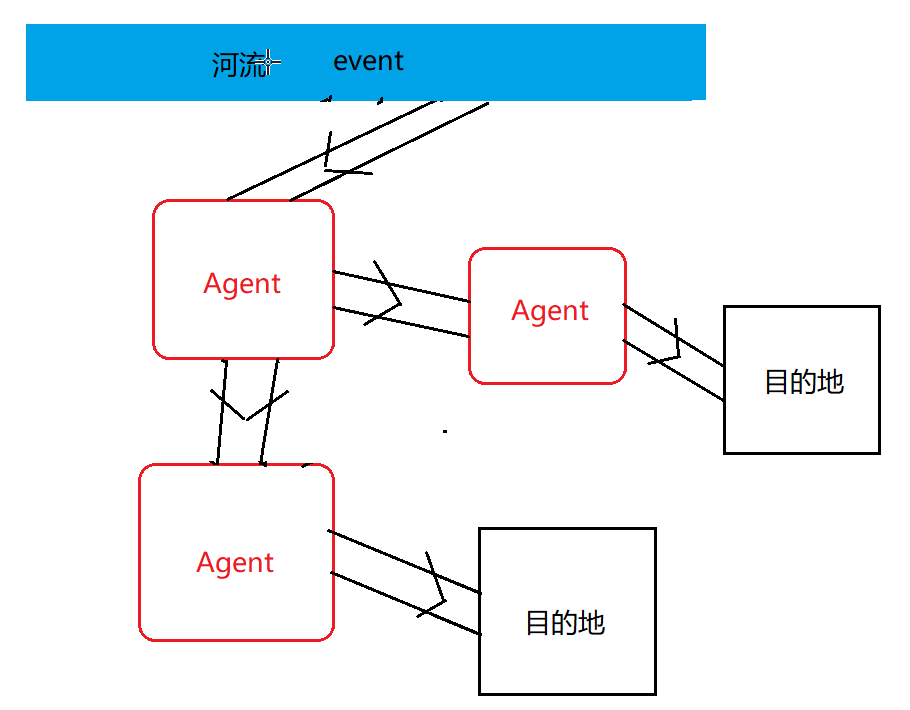


## Flume的扇入扇出

扇入：多个数据来源一起流进同一个agent中。

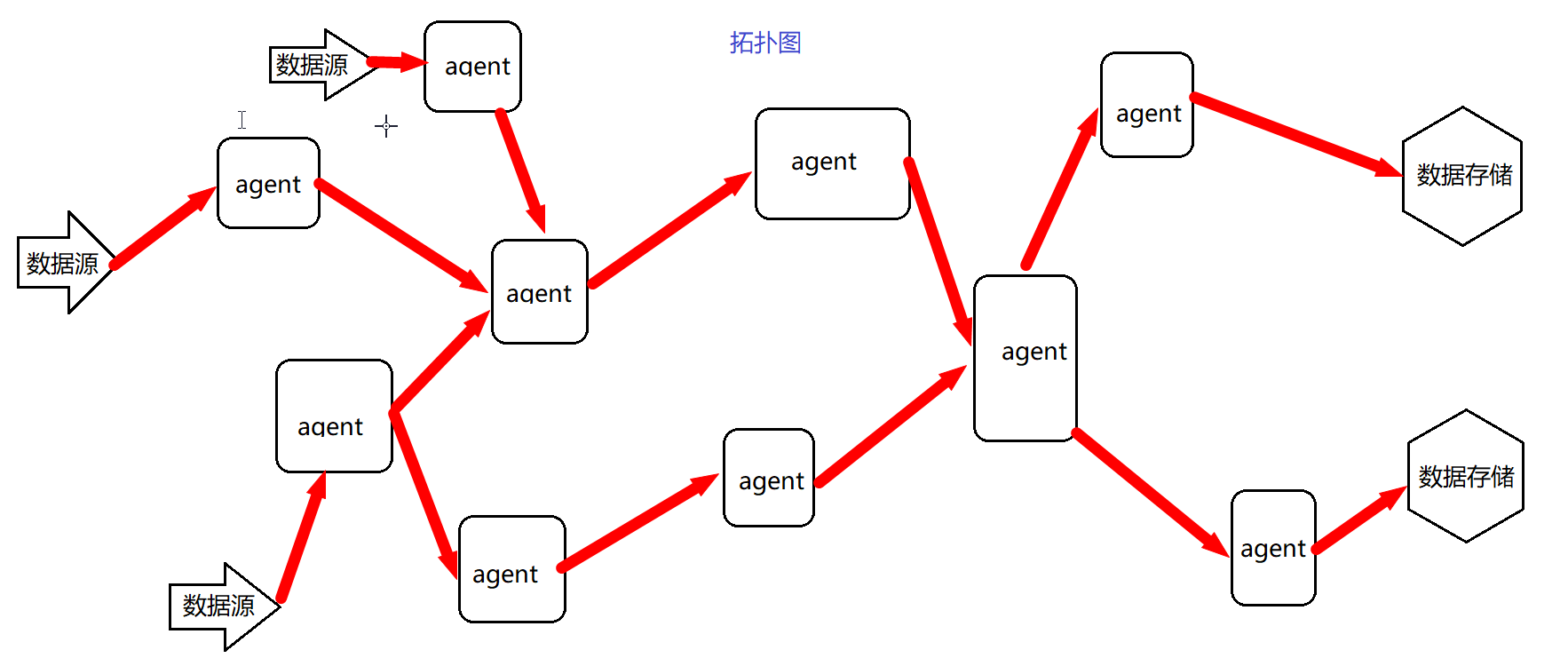


扇出：一个agent中的数据流向多个数据出处。



## Flume的优势

基于flume的多级流动，扇入扇出，可以实现非常复杂的拓扑结构，适应处理几乎任何场景的数据。



## Flume的安装

上传解压即可。

## Flume.properties详解（需要把我给的flume.properties上传到conf目录下）

Flume配置文件主要为配置agent内容。若值为多个，空格隔开

#a1为自定义的agent名字，与启动命令中的-n属性对应

a1.sources = r1 #定义agent的数据源source，可以有多个。

a1.sinks = k1 #定义agent的数据出处，可以有多个。

a1.channels = c1 #定义agent的通道，一般有多少个sink就有多少个channel

a1.sources.r1.type = avro #指定source的类型为avro

a1.sources.r1.bind = 0.0.0.0 #指定source的来源。一般为本机，被动接收。

a1.sources.r1.port = 22222 #指定端口

a1.sinks.k1.type = avro #指定sink的类型为avro

a1.sinks.k1.hostname = 192.168.65.162 #指定sink的目标节点IP

a1.sinks.k1.port = 22222 #指定目标端口

a1.channels.c1.type = memory #指定channel的类型为 内存

a1.channels.c1.capacity = 1000 #指定存储容量，避免强制抢占内存影响其他进程的正常运行

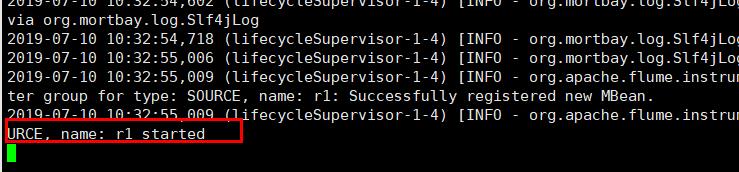
a1.channels.c1.transactionCapacity = 100 #指定事务容量

a1.sources.r1.channels = c1 #绑定source

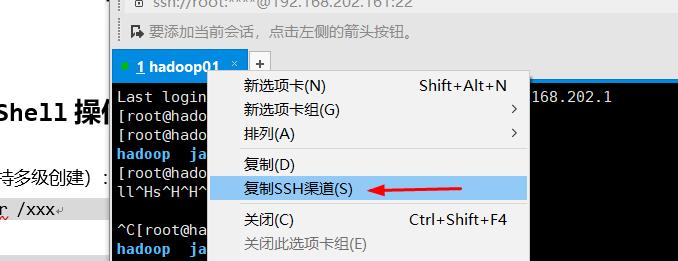
a1.sinks.k1.channel = c1 #绑定sink

### 启动（\* 退出停止命令为ctrl+C \*）

[root@localhost conf]# ../bin/flume-ng agent -c ./ -f ./flume.properties -n a1 -Dflume.root.logger=INFO,console



## 案例（以下三个案例需要打开俩个渠道窗口：一个输入，一个输出）



### Source练习

Source三种类型：

2. avro

1. http
2. spooldir

**avro**

1.在/usr/local/src/flume下创建（vim命令）文件log.txt并编辑添加数据

2.在flume安装目录下的conf目录下执行命令启动agent

3.模拟发送avro在flume的bin目录下执行：

./flume-ng avro-client -c ../conf -H 0.0.0.0 -p 22222 -F ../../log.txt

案例执行完后ctrl+C停止

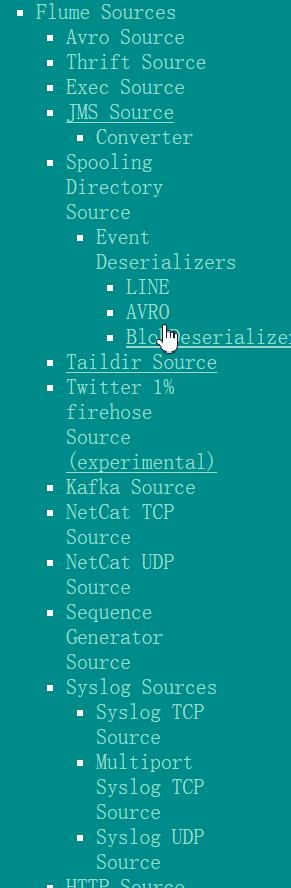
**Spooldir**

1. 在/usr/local/src/flume下创建文件夹data
2. 启动agent
3. 在data中vim aaa文件并添加内容111。发现flume日志中打印编辑内容。

**http**

1. 启动agent
2. 模拟发送http请求，在任意目录下执行：

curl -X POST -d '[{"headers":{"tester":"tony"},"body":"hello http flume"}]' <http://0.0.0.0:22222>



1．Avro

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = avro

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

2．Spooldir（\*此案例需要在flume下创建data目录，启动后再创建一个文件输入内容并测设\*）

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = spooldir

a1.sources.r1.spoolDir = /usr/local/src/flume/data

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

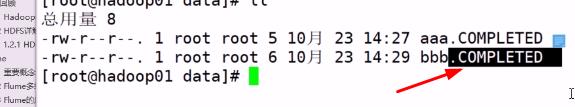
a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

读取后的效果



1. http

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

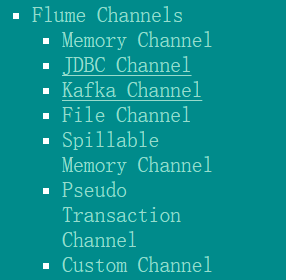
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

（\*补充：以上案例有失败后有需要创建目录的要删除目录后再创建测试\*）

### Channel练习



Memory

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

### Sink练习



Logger

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

Avro

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = avro

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = avro

a1.sinks.k1.hostname = hadoop02

a1.sinks.k1.port = 22222

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

HDFS（最后的案例会用到，这里不做测试）

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = hdfs

a1.sinks.k1.hdfs.path = hdfs://hadoop01:9000/flume/data

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

## 集群部署

（在src目录下： scp -r flume/ root@hadoop02:/usr/local/src）

Hadoop01：JDK、Hadoop、Flume

Hadoop02：JDK、Flume

Hadoop03：JDK、Flume

### 多级（\*都是在conf目录下编辑flume.properties\*）

Hadoop01（\*以下192.168.62.16可替换为hadoop02因配过hosts \*）

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = avro

a1.sinks.k1.hostname = 192.168.65.162

a1.sinks.k1.port = 22222

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

hadoop02

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = avro

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = avro

a1.sinks.k1.hostname = 192.168.65.163

a1.sinks.k1.port = 22222

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

hadoop03

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = avro

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

启动顺序为hadoop03，hadoop02,hadoop01

复制hadoop01测试模拟发送http请求后hadoop03有值为正确。

### 扇入

Hadoop01

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = avro

a1.sinks.k1.hostname = 192.168.65.163

a1.sinks.k1.port = 22222

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

Hadoop02

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = avro

a1.sinks.k1.hostname = 192.168.65.163

a1.sinks.k1.port = 22222

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

Hadoop03

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = avro

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

### 扇出

Hadoop01

a1.sources = r1

a1.sinks = k1 k2

a1.channels = c1 c2

a1.sources.r1.type = http

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = avro

a1.sinks.k1.hostname = 192.168.65.162

a1.sinks.k1.port = 22222

a1.sinks.k2.type = avro

a1.sinks.k2.hostname = 192.168.65.163

a1.sinks.k2.port = 22222

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.channels.c2.type = memory

a1.channels.c2.capacity = 1000

a1.channels.c2.transactionCapacity = 100

a1.sources.r1.channels = c1 c2

a1.sinks.k1.channel = c1

a1.sinks.k1.channel = c2

Hadoop02

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = avro

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

Hadoop03

a1.sources = r1

a1.sinks = k1

a1.channels = c1

a1.sources.r1.type = avro

a1.sources.r1.bind = 0.0.0.0

a1.sources.r1.port = 22222

a1.sinks.k1.type = logger

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

# 项目、Flume、HDFS整合

## log4j和flume整合

配置log4j.properties

log4j.rootLogger = info,stdout,flume

log4j.appender.stdout = org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target = System.out

log4j.appender.stdout.layout = org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern = %m%n

# appender flume

log4j.appender.flume = org.apache.flume.clients.log4jappender.Log4jAppender

log4j.appender.flume.Hostname = hadoop01

log4j.appender.flume.Port = 22222

log4j.appender.flume.UnsafeMode = true

## 配置flume.properties（\*/jt/data目录会自动创建\*）

#命名Agent a1的组件

a1.sources = r1

a1.sinks = k1

a1.channels = c1

#描述/配置Source

a1.sources.r1.type = avro

a1.sources.r1.bind = hadoop01

a1.sources.r1.port = 22222

#描述Sink

a1.sinks.k1.type = hdfs

a1.sinks.k1.hdfs.path = hdfs://hadoop01:9000/jt/data

a1.sinks.k1.hdfs.fileType=DataStream

#描述内存Channel

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

#为Channle绑定Source和Sink

a1.sources.r1.channels = c1

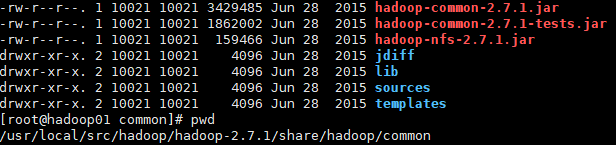
a1.sinks.k1.channel = c1

启动hadoop01 启动flume 启动java项目测试下载

## 复制依赖jar文件（低版本）

进入共享目录

cd /usr/local/src/hadoop/hadoop-2.7.1/share/hadoop/common



复制文件

cp \*.jar /usr/local/src/flume/apache-flume-1.6.0-bin/lib

cd /usr/local/src/hadoop/hadoop-2.7.1/share/hadoop/common/lib

**cp commons-configuration-1.6.jar** /usr/local/src/flume/apache-flume-1.6.0-bin/lib

**cp hadoop-auth-2.7.1.jar** /usr/local/src/flume/apache-flume-1.6.0-bin/lib

**cp htrace-core-3.1.0-incubating.jar** /usr/local/src/flume/apache-flume-1.6.0-bin/lib

**cp commons-io-2.4.jar** /usr/local/src/flume/apache-flume-1.6.0-bin/lib

cd /usr/local/src/hadoop/hadoop-2.7.1/share/hadoop/hdfs

cp **hadoop-hdfs-2.7.1.jar** /usr/local/src/flume/apache-flume-1.6.0-bin/lib

否则启动和运行时会报错：

java.lang.NoClassDefFoundError: org/apache/hadoop/io/SequenceFile$CompressionType

java.io.IOException: No FileSystem for scheme: hdfs