

分布式数据采集工具Flume

庞磊 | 2020年10月

目录 > CONTENTS

- 1 Flume简介
- 2 Flume原理
- 3 Flume使用



1 chapter

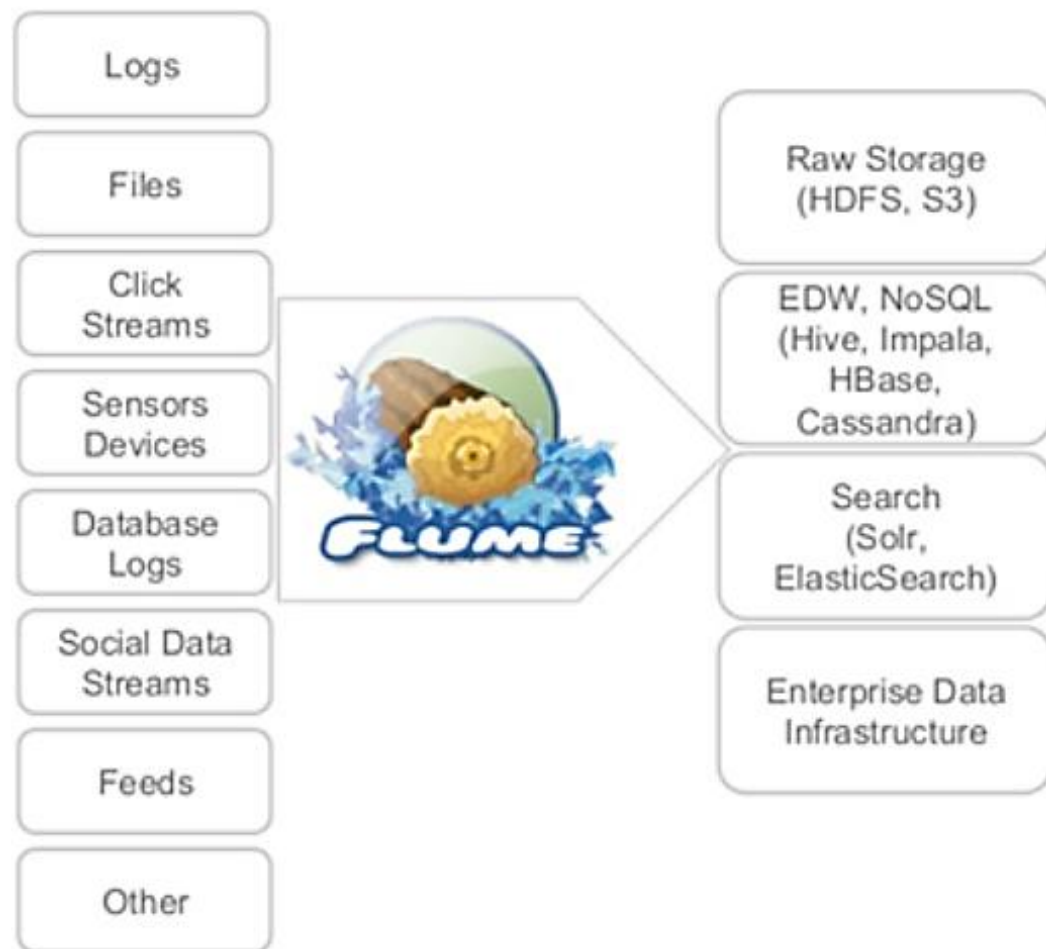
Flume简介

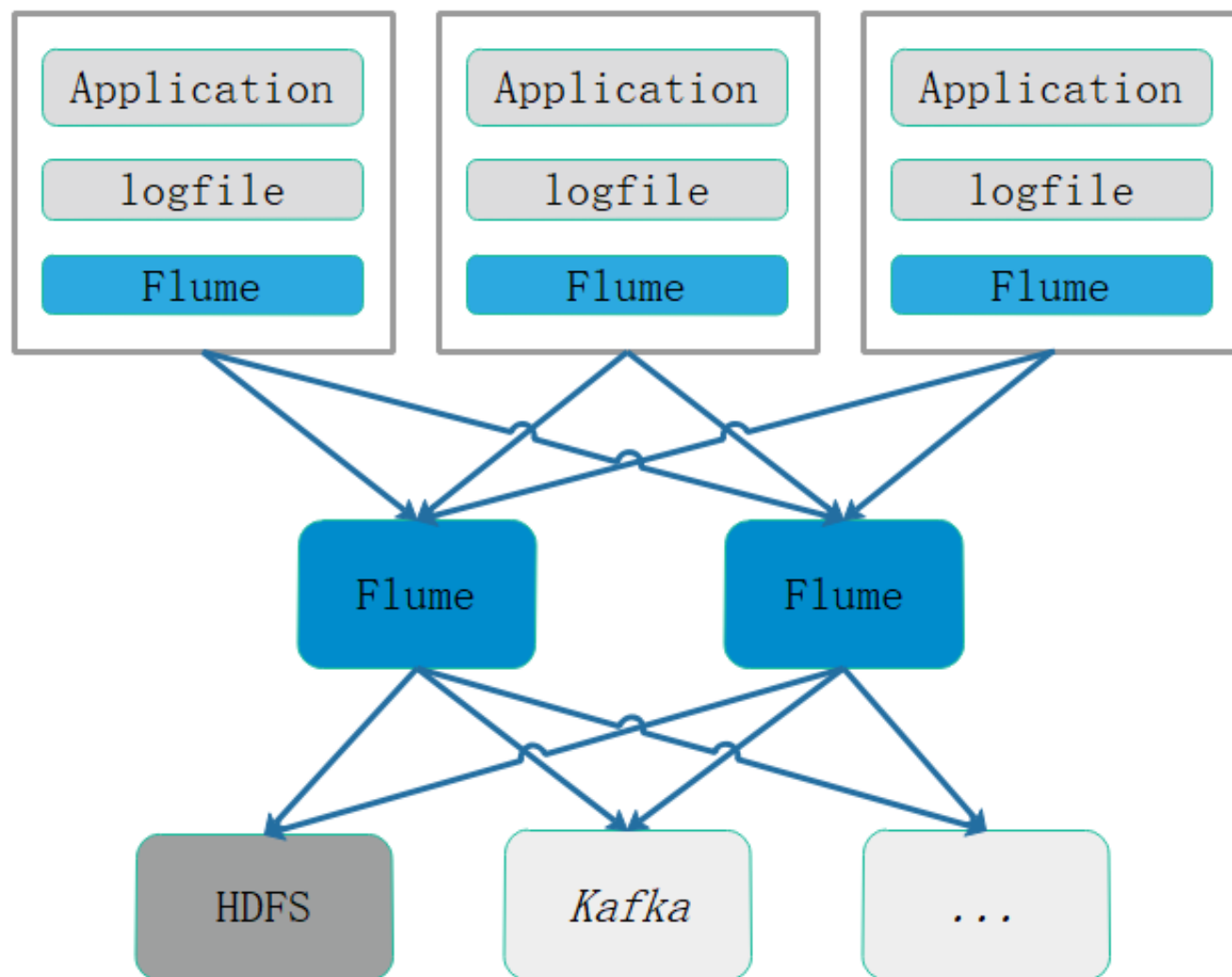
- ✓ 什么是Flume
- ✓ 应用场景


➤ Flume是一个分布式海量数据采集、聚合和传输系统

➤ 特点

- 基于事件的海量数据采集
- 数据流模型：Source→Channel→Sink
- 事务机制：支持重读重写，保证消息传递的可靠性
- 内置丰富插件：轻松与各种外部系统集成
- 高可用：Agent主备切换
- Java实现：开源，优秀的系统设计





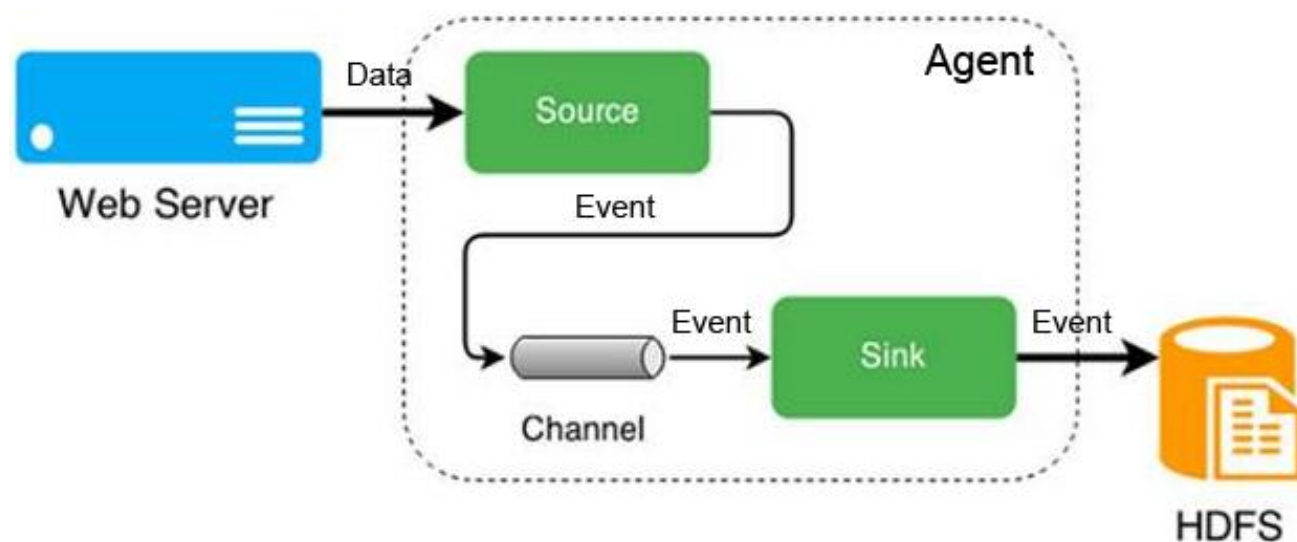


2 chapter

Flume原理

- ✓ 基本概念
- ✓ Flume基本组件
- ✓ Flume数据流
- ✓ Flume架构

- Event: 事件, 最小数据传输单元, 由Header和Body组成
- Agent: 代理, JVM进程, 最小运行单元, 由Source、Channel、Sink三个基本组件构成, 负责将外部数据源产生的数据以Event的形式传输到目的地
 - Source: 负责对接各种外部数据源, 将采集到的数据封装成Event, 然后写入Channel
 - Channel: Event暂存容器, 负责保存Source发送的Event, 直至被Sink成功读取
 - Sink: 负责从Channel读取Event, 然后将其写入外部存储, 或传输给下一阶段的Agent
 - 映射关系: 1个Source → 多个Channel, 1个Channel → 多个Sink, 1个Sink → 1个Channel



➤ Source组件

- 对接各种外部数据源，将采集到的数据封装成Event，然后写入Channel
- 一个Source可向多个Channel发送Event
- Flume内置类型丰富的Source，同时用户可自定义Source

类型	Type（参数）	说明
Exec Source	exec	监听Linux命令的标准输出
Spooling Directory Source	spooldir	监听目录下的新文件，不支持断点续传和嵌套目录
Taildir Source	TAILDIR	监听目录或文件，支持断点续传
Avro Source	avro	启动Avro Server，通过RPC接收Avro数据，可与上一级Agent连接
HTTP Source	http	启动Http Server，通过Http Post接收数据
Kafka Source	org.apache.flume.source.kafka.KafkaSource	从Kafka中读取数据
JMS Source	jms	从JMS源中读取数据

➤ Channel组件

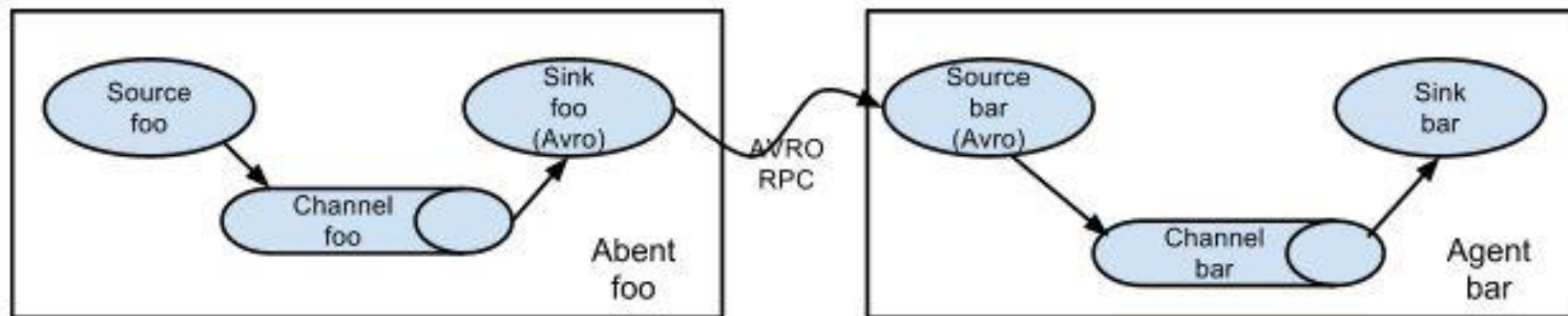
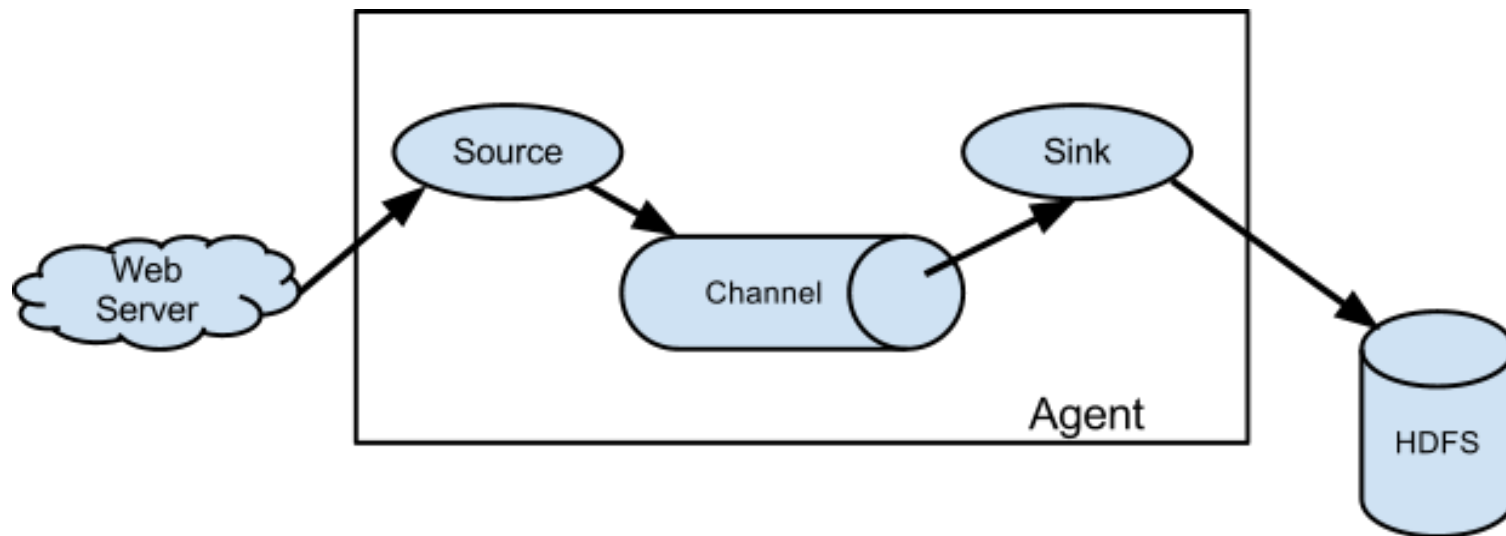
- Event中转暂存区，存储Source采集但未被Sink读取的Event
- 为了平衡Source采集、Sink读取的速度，可视为Flume内部的消息队列
- 线程安全并具有事务性，支持Source写失败重写和Sink读失败重读

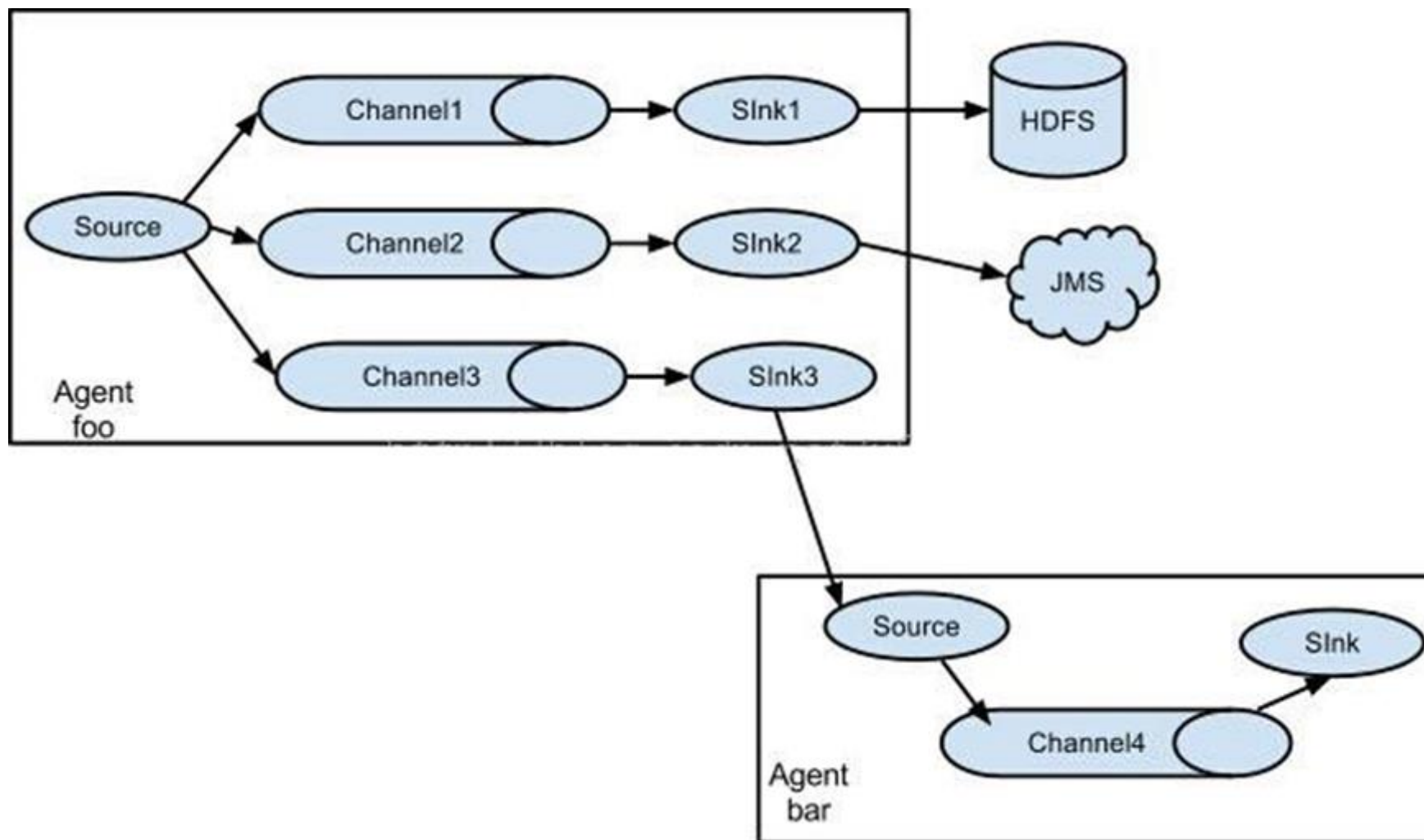
类型	Type（参数）	说明
Memory Channel	memory	内存作为Channel，读写速度快，但存储容量小，数据易丢失
File Channel	file	本地磁盘文件作为Channel，存储容量大，数据较安全，但读写速度较慢
JDBC Channel	jdbc	关系数据库作为Channel，内置Derby，数据安全可靠恢复
Kafka Channel	org.apache.flume.channel.kafka.KafkaChannel	Kafka作为Channel，读写速度快，存储容量大，容错能力强

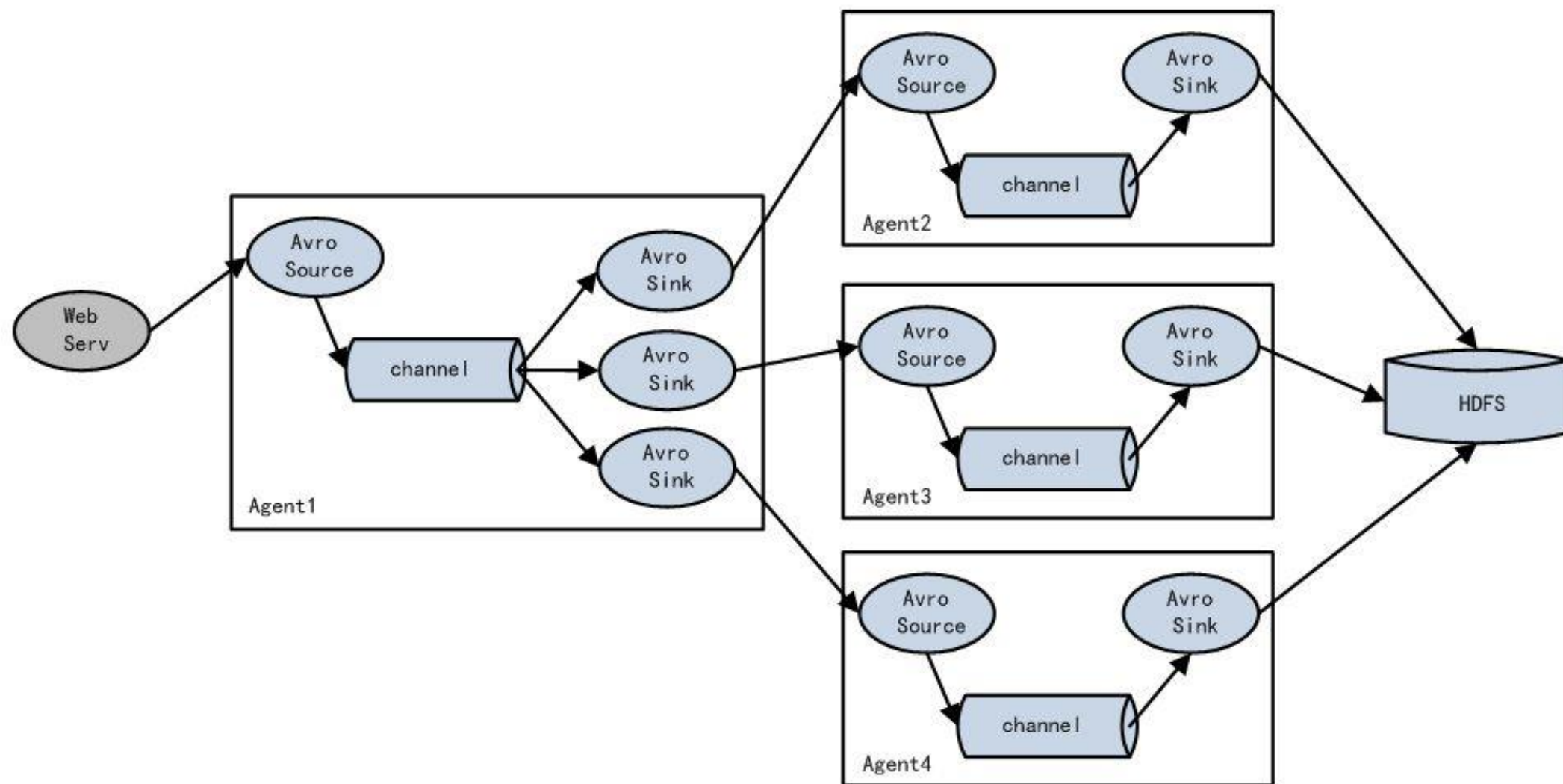
➤ Sink组件

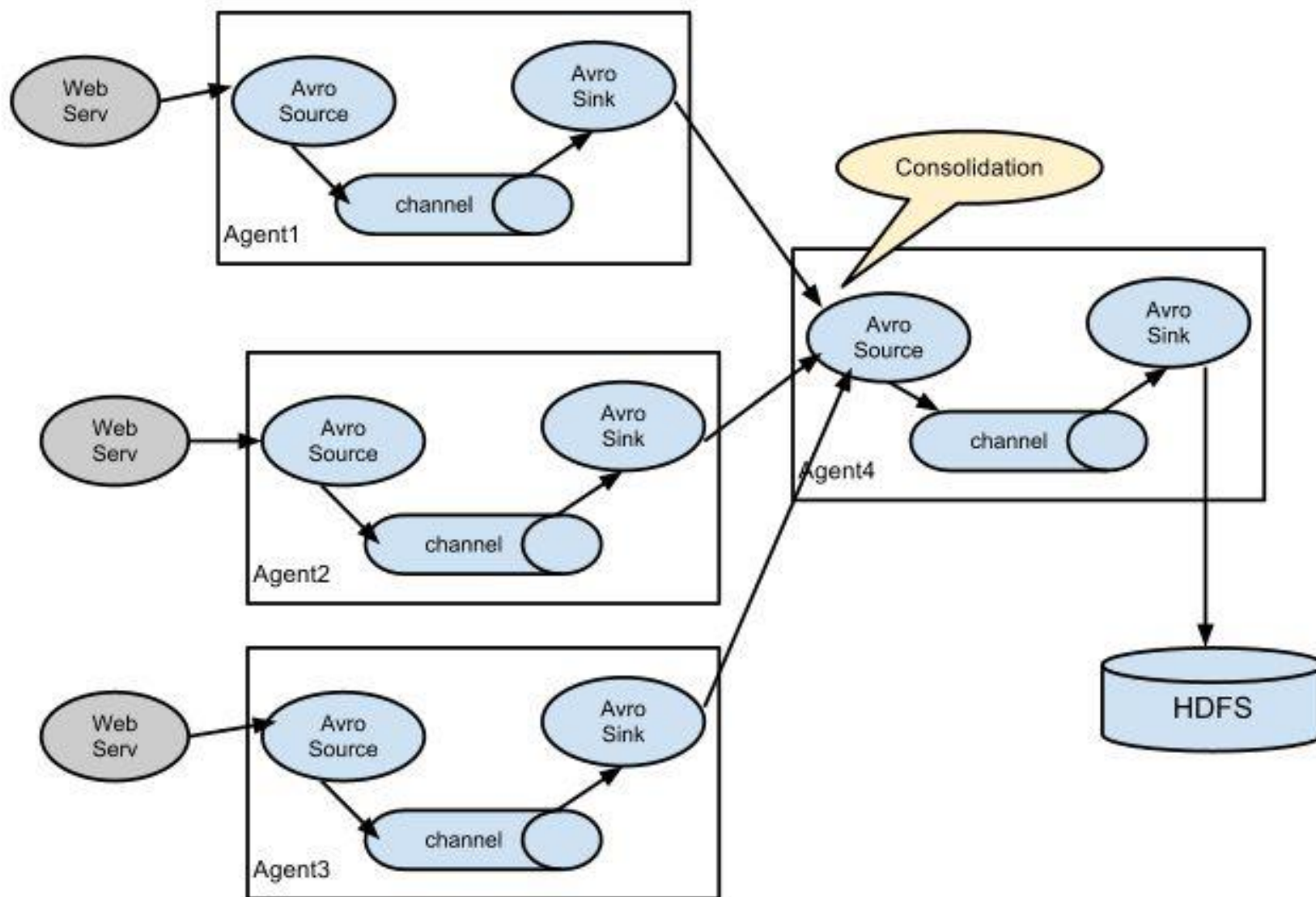
- 从Channel读取Event，将其写入外部存储，或传输到下一阶段的Agent
- 一个Sink只能从一个Channel中读取Event
- Sink成功读取Event后，向Channel提交事务，Event被删除，否则Channel会等待Sink重新读取

类型	Type（参数）	说明
Avro Sink	avro	常用于对接下一阶段的Avro Source，通过RPC实现端到端的批量压缩数据传输
HDFS Sink	hdfs	将Event写入HDFS，可生成文本文件，支持数据压缩
Kafka Sink	org.apache.flume.sink.kafka.KafkaSink	将Event写入Kafka指定主题
File Roll Sink	file_roll	将Event写入本地文件系统





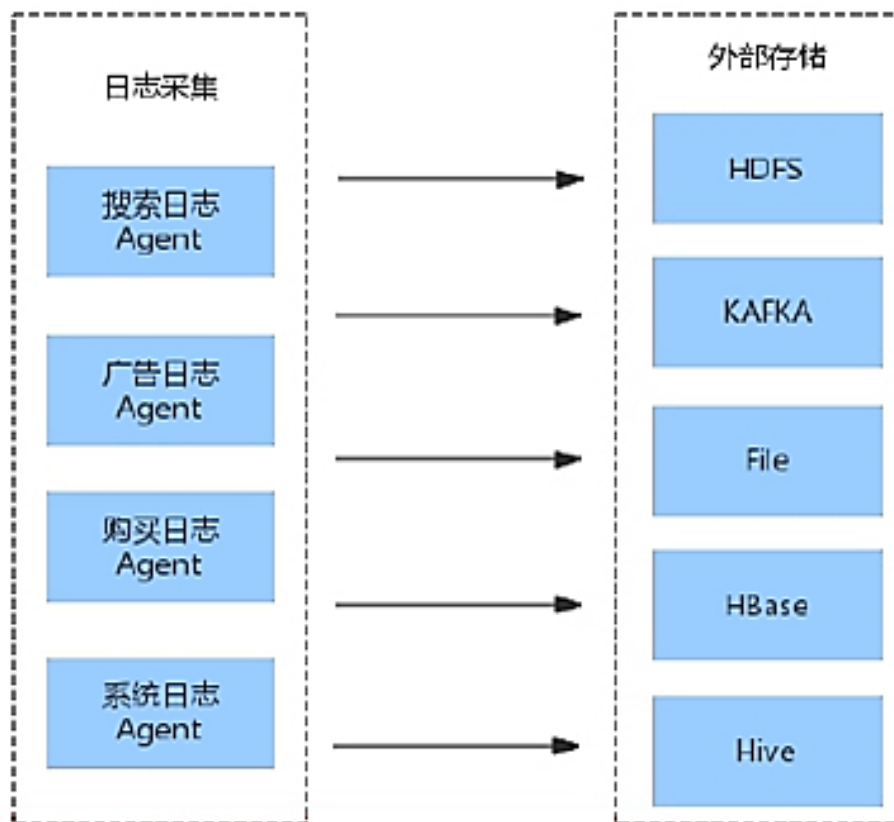




➤ 优点：架构简单，使用方便，占用资源较少

➤ 缺点

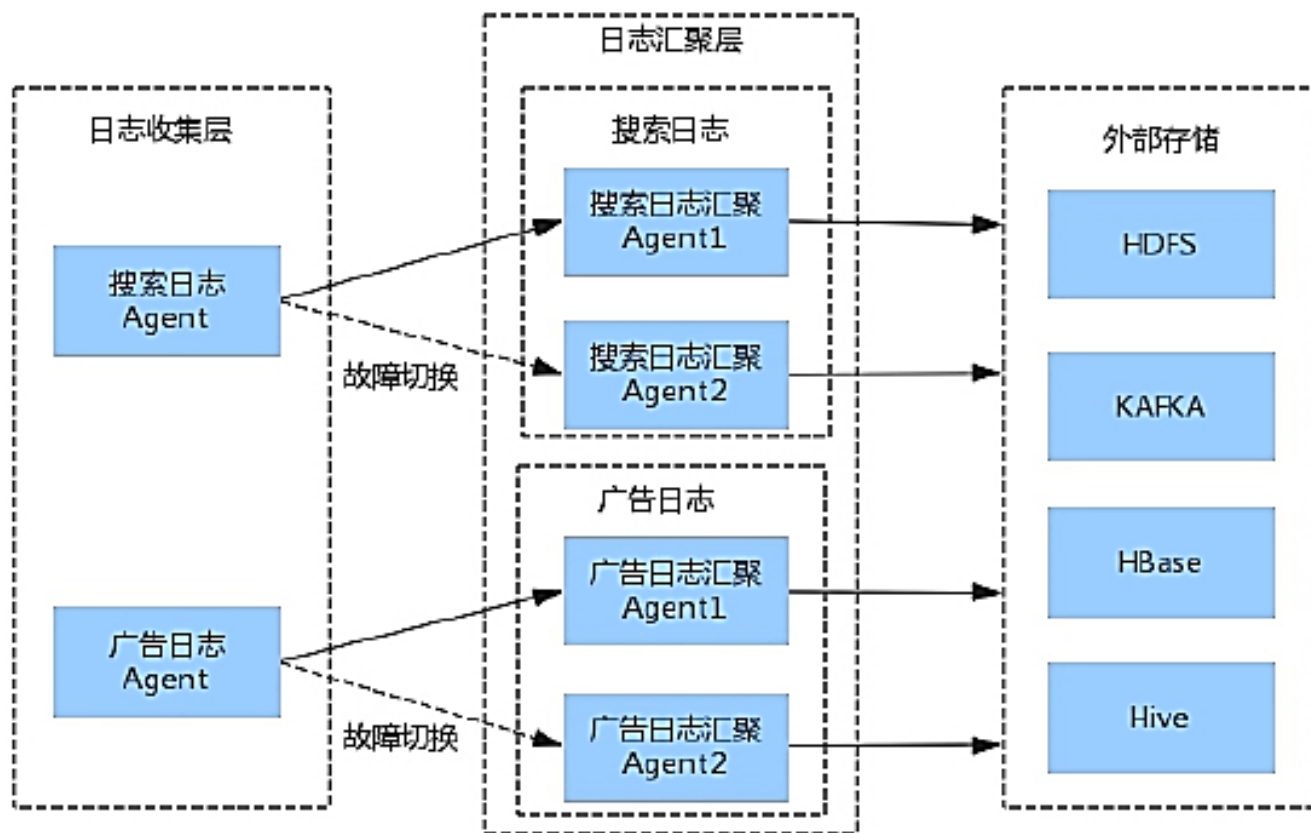
- 如果采集的数据源或Agent较多，将Event写入到HDFS会产生很多小文件
- 外部存储升级维护或发生故障，需对采集层的所有Agent做处理，人力成本较高，系统稳定性较差
- 系统安全性较差
- 数据源管理较混乱




➤ 优点

- 各类日志数据分层处理，架构清晰，运维高效，降低人工误操作风险
- 避免产生过多小文件，提高系统稳定性和处理能力
- 对外不会暴露系统关键信息，降低攻击风险，显著提升安全性
- 各关联系统易于升级维护

➤ 缺点：部署相对复杂，占用资源较多





3 chapter

Flume使用

- ✓ Flume安装
- ✓ Agent配置
- ✓ Flume运行

➤ 运行环境

- JDK1.7及以上版本

➤ 安装Flume

- 从官网下载Flume安装包
- 解压安装包
- 修改conf/flume-env.sh文件中的JDK配置，添加JAVA_HOME、HADOOP_HOME变量
 - TDH平台可省略

➤ 验证是否安装成功

- 在Flume安装目录下，运行命令./bin/flume-ng version，查看Flume版本信息

➤ 示例：Agent配置文件

```
/* 定义Agent组件名 */
a1.sources = r1
a1.sinks = k1
a1.channels = c1

/* 配置Souce */
a1.sources.r1.type = netcat // Source类型为netcat，监听指定的Socket端口
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

/* 配置Sink */
a1.sinks.k1.type = logger // Sink类型为logger，将Event输出到控制台

/* 配置Channel */
a1.channels.c1.type = memory // Channel类型为memory，Event缓存在内存中
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

/* 设置Source、Sink和Channel的关系 */
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1 // 一个Sink只能连接一个Channel
```

➤ Exec Source

- 功能：采集Linux命令的标准输出，如：通过tail -f file监听指定文件
- 缺点：虽然可以实时传输消息，但并不记录文件读取的位置，不支持断点续传，如果Exec Source重启或挂掉会造成后续新增消息的丢失，一般在测试环境使用
- 关键参数说明
 - type: Source类型为exec
 - command: Linux命令
 - channels: Source连接的Channel

```
a1.sources = s1 // 定义Agent a1的Source  
a1.channels = c1 // 定义Agent a1的Channel  
a1.sources.s1.type = exec  
a1.sources.s1.command = tail -f /var/log/secure  
a1.sources.s1.channels = c1
```


➤ Spooling Directory Source

- 功能：采集目录下新增文件的所有数据，完成后将文件后缀改为.COMPLETED
- 缺点：不能采集已有文件的新增数据，不能对嵌套文件夹进行递归监听
- 关键参数说明
 - type：Source类型为spoolDir
 - spoolDir：Source监听的文件夹
 - fileHeader：是否将文件绝对路径添加到Event Header中，默认值false
 - fileHeaderKey：Event Header中文件绝对路径的键值，默认值file
 - fileSuffix：采集完成后，给文件添加后缀，默认值.COMPLETED

```
a1.sources = s1
a1.channels = c1
a1.sources.s1.type = spoolDir
a1.sources.s1.spoolDir = /var/log/apache/flumeSpool
a1.sources.s1.fileHeader = true
a1.sources.s1.channels = c1
```

➤ Kafka Source

- 功能：作为Kafka的消费者，Flume持续从中读取数据。若多个Source消费同一个Kafka Topic，则kafka.consumer.group.id应设置成相同的组id，确保不会重复消费数据
- 关键参数说明
 - type: Source类型设置为Kafka Souce的类路径org.apache.flume.source.kafka.KafkaSource
 - kafka.bootstrap.servers: Kafka Broker列表，格式为ip1:port1, ip2:port2..., 建议配置多个，防止单个Broker发生故障导致连接失败
 - kafka.topics: 消费的Kafka Topic名称
 - kafka.consumer.group.id: Kafka Souce所属组的id，默认值flume
 - batchSize: 批量写入Channel的最大消息数，默认值1000
 - batchDurationMillis: 等待批量写入Channel的最长时间，该参数和batchSize只要有一个先满足就会触发批量写入Channel操作，默认值1000毫秒
 - kafka.consumer.timeout.ms: Kafka消费超时时间

➤ Memory Channel

- 功能: 用内存作为Channel, 读写速度快, 但存储容量小, 数据丢失风险较高
- 适用场景: Agent所在服务器的内存资源充足, 且不关心数据丢失
- 关键参数说明
 - type: Channel类型为memory
 - capacity: 存储Event的最大数量, 默认值100
 - transactionCapacity: 一次事务中写入和读取Event的最大数量, 默认值100
 - keep-alive: 在Channel中写入或读取Event等待完成的超时时间, 默认值3秒
 - byteCapacityBufferPercentage: 缓冲空间占Channel容量的百分比, 默认值20
 - byteCapacity: Channel占用内存的最大容量, 默认为Flume堆内存的80%

```
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 10000
a1.channels.c1.byteCapacity = 800000
```

➤ File Channel

- 用磁盘文件作为Channel，与Memory Channel相比，存储容量大，无数据丢失风险
- Event被顺序写到文件末尾，maxFileSize参数设置数据文件大小上限
- 可配置多磁盘文件路径，提高文件写入性能
- 当File Channel文件中的Event被完全读取，并且Sink已提交事务，Flume将删除该文件
- 通过设置检查点，Agent重启之后能快速将File Channle中的数据按顺序恢复到内存中
- 关键参数说明
 - type: Channel类型为file
 - checkpointDir: 检查点目录，默认在Flume用户目录下创建，建议单独配置磁盘路径
 - useDualCheckpoints: 是否开启备份检查点，默认值false，建议设置为true开启备份检查点
 - backupCheckpointDir: 备份检查点目录，建议不要和检查点目录放在同一块磁盘上
 - checkpointInterval: 写检查点的时间间隔，默认值30000毫秒

➤ File Channel

- 关键参数说明

- dataDirs: 数据文件存储路径, 建议配置多块盘的多个路径, 通过磁盘的并行写入来提高性能, 路径用逗号隔开
- transactionCapacity: 一次事务中写入和读取Event的最大数量, 默认值10000
- maxFileSize: 数据文件的最大容量, 默认值2146435071字节
- minimumRequiredSpace: 磁盘目录最小剩余空间, 如果剩余空间小于设置值, 则不再写入
- capacity: 存储Event的最大数量
- keep-alive: 写入或读取Event等待完成的超时时间, 默认值3秒

```
a1.channels.c1.type = file  
a1.channels.c1.checkpointDir = /mnt/flume/checkpoint  
a1.channels.c1.dataDirs = /mnt/flume/data
```

➤ HDFS Sink

- 功能: 将Event写入HDFS文件, 同时提供了强大的时间戳转义功能, 将Event Header中的timestamp转义成日期时间格式, 在HDFS中按时间分区存储
- 关键参数说明
 - type: Sink类型为hdfs
 - hdfs.path: HDFS存储路径, 支持按时间分区
 - hdfs.filePrefix: HDFS文件名前缀, 默认值FlumeData
 - hdfs.fileSuffix: HDFS文件名后缀
 - hdfs.rollInterval: 临时文件滚动生成时间间隔, 默认值30秒, 0表示不滚动生成
 - hdfs.rollSize: 临时文件滚动生成大小, 默认值1024B, 0表示不滚动生成
 - hdfs.rollCount: 临时文件滚动生成的Event数, 默认值10, 0表示不滚动生成
 - hdfs.idleTimeout: 临时文件等待Event写入的超时时间, 超时后文件自动关闭, 重命名为目标文件名。
默认值0秒, 表示禁用此功能, 不自动关闭临时文件

➤ HDFS Sink

- 关键参数说明

- `hdfs.batchSize`: 批量写入HDFS的Event数量, 默认为100
- `hdfs.callTimeout`: 操作HDFS文件的超时时间
- `hdfs.round`: 用于HDFS文件按时间分区, 时间戳向下取整, 默认值false
- `hdfs.roundUnit`: 按时间分区使用的时间单位, 支持second、minute、hour三种单位, 默认值second
- `hdfs.roundValue`: 当round为true时, 配合roundUnit一起使用, 默认值1。例如: roundUnit为minute、该值为1, 表示将1分钟内的数据写到一个文件, 即每分钟生成一个文件
- `hdfs.timeZone`: 写入HDFS文件使用的时区, 默认值Local Time
- `hdfs.useLocalTimeStamp`: 是否使用本地时间替换Event头信息中的时间戳
- `hdfs.minBlockReplicas`: HDFS文件块的最小副本数
- `hdfs.writeFormat`: HDFS文件格式, 支持Text、Writable两种格式, 默认值Writable

➤ HDFS Sink

- 关键参数说明

- `hdfs.codeC`: 文件压缩格式, 支持gzip、bzip2、lzo、lzop、snappy, 默认不压缩
- `hdfs.fileType`: 文件类型, `DataStream`表示不压缩文件, `CompressedStream`表示按`codeC`指定格式压缩文件, 默认值`SequenceFile`

```
a1.sinks = k1
a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path = /flume/events/%y-%m-%d/%H%M/%S // %y%m%d年月日, %H%M%S时分秒
a1.sinks.k1.hdfs.filePrefix = events-
a1.sinks.k1.hdfs.round = true
a1.sinks.k1.hdfs.roundUnit = minute
a1.sinks.k1.hdfs.roundValue = 10
a1.sinks.k1.hdfs.minBlockReplicas = 1
a1.sinks.k1.hdfs.rollInterval = 0
a1.sinks.k1.hdfs.rollSize = 1024000000
```

➤ Kafka Sink

- 功能：将Event写入Kafka的指定主题中
- 关键参数说明
 - type: Sink类型为org.apache.flume.sink.kafka.KafkaSink
 - kafka.bootstrap.servers: Kafka Broker列表，格式为ip1:port1, ip2:port2, ...，建议配置多个，防止单个Broker发生故障导致连接失败
 - kafka.topic: Kafka主题名，默认值flume-topic
 - flumeBatchSize: Producer端单次批量发送的消息条数，默认值100，该值应根据实际环境适当调整，增加批量发送条数能在一定程度上提高性能，但是同时也增加了延迟和Producer端数据丢失的风险
 - useFlumeEventFormat: 默认值false，仅将Event Body内容发送到Kafka；若设置为true，则把Event完整发送到Kafka

➤ Kafka Sink

- 关键参数说明

- `kafka.producer.acks`: Kafka Sink发送消息后是否等待Broker返回成功接收信号，默认值1。0表示不等待，这种方式吞吐量高，但存在数据丢失风险。1表示Broker收到消息并成功写本地log文件后，立刻向Producer返回成功接收信号，不需要等待所有的Follower同步完消息后再做回应，这种方式在数据丢失风险和吞吐量之间做了平衡。`all`（或-1）表示Broker收到消息并成功写log后，必须等待所有的Follower成功写log后，才能向Producer返回成功接收的信号，这种方式能够保证消息不丢失，但是性能最差

```
a1.sinks = k1
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.topic = test_topic
a1.sinks.k1.brokerList = localhost:9092
a1.sinks.k1.requiredAcks = 0
a1.sinks.k1.batchSize = 20
```


➤ 运行Flume的基本步骤

- 第1步：编写Agent配置文件 (*.conf)
- 第2步：在命令行运行Agent
 - 进入Flume安装目录
 - 运行以下命令

```
# bin/flume-ng agent --conf conf --conf-file example.conf
--name a1 -D flume.root.logger = INFO, console
```

Command Options	Description
--conf	Agent配置文件目录
--conf-file	Agent配置文件名 (*.conf)
--name	Agent名称
-D	JVM参数

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

a1.sources.r1.type = spooldir
a1.sources.r1.spoolDir = /opt/vdb/log
a1.sources.r1.fileHeader = true
a1.sources.r1.deserializer.outputCharset = UTF-8

a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path = hdfs://nameservice1/user/root/file1
a1.sinks.k1.hdfs.fileType = DataStream
a1.sinks.k1.hdfs.writeFormat = Text
a1.sinks.k1.hdfs.maxOpenFiles = 1
a1.sinks.k1.hdfs.rollCount = 0
a1.sinks.k1.hdfs.rollInterval = 0
a1.sinks.k1.hdfs.rollSize = 1000000
a1.sinks.k1.hdfs.batchSize = 10000

a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 100

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





Q&A

TRANSWARP
星环科技