实验三：多路复用实时数据采集实验【重要】

一、实验目的

通过本实验，学生将能够：

1. 理解 **“一份数据，多路输出”** 的采集架构思想（实时 + 离线备份）。
2. 掌握 **TAILDIR Source** 的作用与配置方法，理解它与 exec / spooldir 的区别。
    - **Exec Source**：实时监听命令输出（如 `tail -F`），但不具备断点续传，会丢数据。
    - **SpoolDir Source**：专门处理“完整文件”，一次性读完，不支持实时追加，也不能读未写完的文件。
    - **TAILDIR Source**：实时监听文件新增内容，并支持断点续传，是企业最常用的实时日志采集方式。
3. 能够编写 Flume 配置，实现：
    - 监听日志文件的实时新增内容；
    - 同时写入 **Kafka 主题** 和 **HDFS 目录**。
4. 理解 **Kafka 实时消费** 与 **HDFS 离线存储** 的配合作用，形成完整的数据采集链路。

二、实验背景与原理

1. 实验场景

在企业生产环境中，**业务日志** 往往既需要：

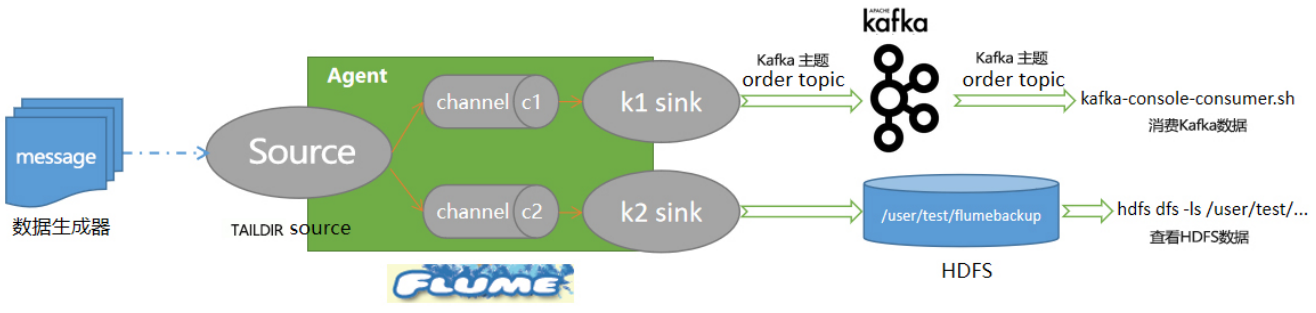
- 实时发送到 Kafka，供 **实时计算 / 风险预警 / 实时监控** 使用；
- 又需要落地到 HDFS，做 **离线报表 / 历史分析 / 模型训练。**

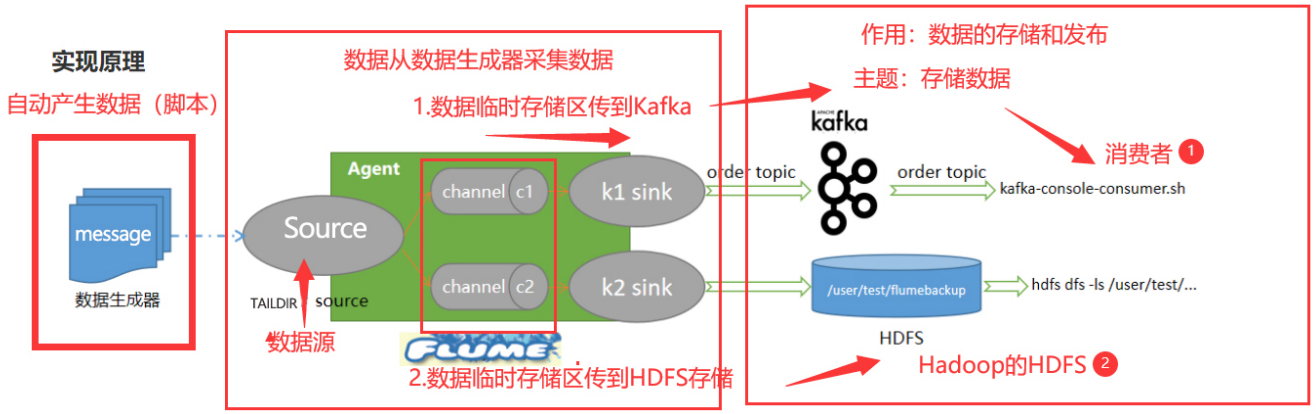
本实验模拟如下场景：

> 将招聘信息日志 **实时追加** 到一个文件中，由 Flume **TAILDIR Source** 实时监听，将每一行数据：
>
> - 一路写入 Kafka 主题 `job_info_stream`；
> - 一路写入 HDFS 目录 `/user/flumebackup/job_info/`。

2. 数据流架构

实现原理：





- **TAILDIR Source**
  - 类似 `tail -F`，实时监听文件新增内容。
  - 支持 **断点续传**：重启后从上次读取位置继续，不会漏数据。
- **Channel c1、c2**
  - 两个内存 Channel，用于缓存要写入 HDFS / Kafka 的数据。
- **Sink k1（HDFS）**：负责把数据落地 HDFS 做备份。
- **Sink k2（Kafka）**：负责把数据写入 Kafka 做实时消费。

三、实验环境与前置条件

1. 三节点集群：`master、slave1、slave2`
2. 已安装并配置：
   - Hadoop 集群（HDFS + YARN）
   - ZooKeeper 集群
   - Kafka 集群
   - Flume（安装在 slave1）
3. 三台机器之间可以通过主机名互通（建议已配置 SSH 免密）。
4. HDFS NameNode 地址：`hdfs://master:9000`

四、实验步骤

> 建议顺序：**先启动 Hadoop / ZK / Kafka → 再配置 Flume → 最后启动数据追加脚本。**

步骤 1：准备实时数据源 和脚本 （slave2）

1）在 slave2 的 `/opt/data/flume/logs` 目录下创建日志源文件messages和脚本。

```
mkdir -p /opt/data/flume/logs
```

> **复制messages 文件（日志内容）**

```
1,玛纳斯分公司（白酒销售）销售员,新疆五谷传奇酒业...,玛纳斯,3800-10000元/月
2,仪表工,新疆雅澳科技有限责任公司,玛纳斯,5000-7000元/月
3,加液工,石河子市物源盛通商贸有限公司,北泉镇,3500-4500元/月
4,大锅饭师傅,石河子市天石肉类加工有限公司,石河子,3000-3300元/月
5,维修工,石河子市天石肉类加工有限公司,北泉镇,5000-6000元/月
......
```

2）创建数据产生脚本 append_data.sh（模拟实时写日志）

> 脚本作用： 每隔 1 秒，从slave2 的本地 messages （与脚本在同一路径下）中读一行，加上时间戳后，通过 ssh 追加到 slave1 的 /opt/apps/flume/logs/messages 中。

```
#!/bin/bash

# 本地要读取的文件（在 slave2 上）
LOGFILE="./messages"

# 远程目标主机（slave1）和目标文件路径
TARGET_HOST="slave1"
TARGET_FILE="/opt/apps/flume/logs/messages"

COUNT=1   # 计数器，从第1条开始

while true; do
    # 从本地 LOGFILE 一行一行读取数据
    while IFS= read -r line || [ -n "$line" ]; do

        # ① 生成当前时间戳：YYYY-MM-DD HH:MM:SS
        timestamp=$(date "+%Y-%m-%d %H:%M:%S")

        # ② 拼接“时间戳 + 内容”，并发送到 slave1
        newline="$timestamp,$line"
        printf '%s\n' "$newline" | ssh "$TARGET_HOST" "cat >> '$TARGET_FILE'"

        # ③ 在 slave2 终端输出你想要的格式
        echo "成功向 slave1 发送第 ${COUNT} 条数据：$newline"

        # ④ 累计条数 +1
        COUNT=$((COUNT + 1))

        # 每写一行停 1 秒，模拟实时输入
        sleep 1
    done < "$LOGFILE"
done
```

赋执行权限：

```
cd /opt/data/flume/logs
chmod +x append_data.sh
```

测试脚本，模拟生成数据

```
./append_data.sh
```

步骤 2：启动 Hadoop 与 ZooKeeper

(1) 在 master 启动 HDFS

```
start-dfs.sh
```

(2) 在 slave1 启动 YARN

```
start-yarn.sh
```

(3) 在 master / slave1 / slave2 启动 ZooKeeper

```
zkServer.sh start
```

步骤 3：启动 Kafka 集群并创建 Topic

(1) 在 master / slave1 / slave2 启动 Kafka

```
kafka-server-start.sh -daemon /opt/apps/kafka/config/server.properties
```

(2) 在 master 创建 Kafka 主题 job_info_stream

```
kafka-topics.sh --create \
--bootstrap-server master:9092,slave1:9092,slave2:9092 \
--topic job_info_stream \
--replication-factor 1 \
--partitions 3
```

(3) 在 master 查看主题（验证用）

```
kafka-topics.sh --list \
--bootstrap-server master:9092,slave1:9092,slave2:9092
# 应能看到 job_info_stream 主题
```

步骤 4：编写 Flume 多路复用配置（slave1）

在 **slave1** 上编写 `taildir-kafka-hdfs.conf`。

```
cd /opt/apps/flume/conf
touch taildir-kafka-hdfs.conf
```

写入配置内容：

```
## 定义 Agent 名称
# 为 Flume 配置的 Agent 定义各组件的名称
a1.sources = r1
a1.sinks   = k1 k2
a1.channels = c1 c2

## 配置 Source（TAILDIR 实时监听文件）
# 配置 Source 组件，类型为 TAILDIR，用于实时监听文件内容的新增部分（类似 tail -F）
# TAILDIR Source 支持断点续传，Flume 重启后可从上次读取的位置继续读取，更适合企业日志采集
a1.sources.r1.type = TAILDIR
# positionFile 用于记录文件的读取位置（偏移量 offset）
# 可保证断电、意外中断后能够继续读取，而不会漏数据或重复数据
a1.sources.r1.positionFile = /opt/apps/flume/taildir_position.json
# filegroups：定义要监听的文件组名称
a1.sources.r1.filegroups = fg1
# filegroups.fg1: 定义 fg1 组具体要监听的文件（本实验监听 messages 文件的实时新增内容）
a1.sources.r1.filegroups.fg1 = /opt/apps/flume/logs/messages

## 配置 Channel（数据临时存储）
# 配置 Channel c1，类型为 memory，临时存储待写入 HDFS 的数据
a1.channels.c1.type = memory
# Channel 的容量最多存储 1000 条 Event
# 当 Event 到达容量上限时，Channel 会暂时停止接收新数据
a1.channels.c1.capacity = 1000
# Source 和 Sink 从 Channel 每次事务传输的最大事件数量为 100
a1.channels.c1.transactionCapacity = 100

# 配置 Channel c2，类型为 memory，临时存储待写入 Kafka 的数据
a1.channels.c2.type = memory
a1.channels.c2.capacity = 1000
a1.channels.c2.transactionCapacity = 100

## 配置 Sink 1（写入 HDFS）
# 配置 Sink k1，类型为 hdfs，用于将数据写入 Hadoop 分布式文件系统
a1.sinks.k1.type = hdfs
# 指定写入 HDFS 的目标目录
a1.sinks.k1.hdfs.path = hdfs://master:9000/data/flume/job_info/
# 设置写入方式为 DataStream（流式写入）
a1.sinks.k1.hdfs.fileType = DataStream
# 写入格式为 Text（纯文本格式）
a1.sinks.k1.hdfs.writeFormat = Text

# 配置文件滚动策略：数据达到 100 条时滚动生成新文件
```

```
a1.sinks.k1.hdfs.rollCount = 100
# 不按时间自动滚动
a1.sinks.k1.hdfs.rollInterval = 0
# 不按文件大小滚动
a1.sinks.k1.hdfs.rollSize = 0


## 配置 Sink 2（写入 Kafka）
# 配置 Sink k2，用于将数据实时写入 Kafka 主题
a1.sinks.k2.type = org.apache.flume.sink.kafka.KafkaSink
# Kafka Broker 地址列表
a1.sinks.k2.brokerList = master:9092,slave1:9092,slave2:9092
# 数据写入的 Kafka 主题名称
a1.sinks.k2.topic = job_info_stream
# batchSize 默认为 1，一条一条实时发送，生产环境可根据吞吐量需求调整为 10~100
a1.sinks.k2.batchSize = 1

## 将 Source、Sink 与 Channel 连接起来
# Source r1 同时绑定 Channel c1 和 c2，实现数据多路复用：一份进入 HDFS，一份进入 Kafka
a1.sources.r1.channels = c1 c2
# Sink k1 从 Channel c1 获取数据，将数据写入 HDFS
a1.sinks.k1.channel = c1
# Sink k2 从 Channel c2 获取数据，将数据写入 Kafka
a1.sinks.k2.channel = c2
```

步骤 5：启动 Kafka 消费者 和 Flume Agent

(1) 在master启动 Kafka 消费者

```
kafka-console-consumer.sh \
--bootstrap-server master:9092,slave1:9092,slave2:9092 \
--topic job_info_stream
```

(2) 在slave1 启动 Flume Agent

```
flume-ng agent \
-n a1 \
-c /opt/apps/flume/conf \
-f /opt/apps/flume/conf/taildir-kafka-hdfs.conf \
-Dflume.root.logger=INFO,console
```

> 保持窗口开启，方便观察 Flume 日志输出是否正常。

(3) 在slave2 启动实时数据追加脚本

```
# 切换目录
cd  /opt/data/flume/logs

# 执行脚本
./append_data.sh
```

持续输出：

```
成功向 slave1 发送第 1 条数据：2025-11-21 10:35:32,1,玛纳斯分公司（白酒销售）销售员,新疆五谷传奇酒业...,玛纳斯,3800-10000元/月
成功向 slave1 发送第 2 条数据：2025-11-21 10:35:33,2,仪表工,新疆雅澳科技有限责任公司,玛纳斯,5000-7000元/月
...
```

五、结果验证

1. 验证 Kafka 实时消费（master）

回到 **master** 的 Kafka 消费者窗口，应看到类似输出：

```
2025-11-21 10:35:32,1,玛纳斯分公司（白酒销售）销售员,新疆五谷传奇酒业...,玛纳斯,3800-10000元/月
2025-11-21 10:35:33,2,仪表工,新疆雅澳科技有限责任公司,玛纳斯,5000-7000元/月
...
```

说明 **实时链路：messages → Flume → Kafka 正常。**

---

2. 验证 HDFS 离线备份

在 **master** 上执行：

```
hdfs dfs -ls /user/flumebackup/job_info/
```

可以看到滚动生成的文件，如：

```
/user/flumebackup/job_info/FlumeData.172xxxxxxx.tmp
/user/flumebackup/job_info/FlumeData.172xxxxxxx
...
```

查看部分内容：

```
hdfs dfs -cat /user/flumebackup/job_info/* | head -5
```

若能看到与 Kafka 消费同样格式的记录，则说明 **HDFS 备份链路** 正常。

六、重要注意事项（新手必看）

注意点	说明
📝 positionFile 位置文件	必须有写权限，用于记录读取进度，避免重复或漏读
📂 被监听文件路径	`a1.sources.r1.filegroups.fg1` 指向的是 **slave1 的文件路径**，不要写错到 slave2
🧠 TAILDIR 适用于持续追加	更适合长期运行的业务日志采集
❗ HDFS 目录需存在	如 `/user/flumebackup` 不存在，需要提前创建：`hdfs dfs -mkdir -p /user/flumebackup/job_info/`
⚠️ SSH 写入依赖网络	append_data.sh 依赖 `ssh root@slave1`，需确保免密和网络畅通

七、实验小结

通过本实验，同学们完成了一个**企业级味道很强**的采集方案：

- 使用 **TAILDIR Source** 实时监听日志文件新增内容；
- 使用 **两个 Channel + 两个 Sink** 实现“一份数据两个去向”；
- 一路写入 **Kafka**，用于实时分析和监控；
- 一路写入 **HDFS**，用于离线统计和历史分析。

这是后续构建 **Flink 实时处理 / Spark 离线分析** 的基础采集模块，也是大数据平台中的核心套路之一。

那我们就在这个实验基础上，**改编出一个“二段式”的新实验**：
原来是：`文件 → Flume(TAILDIR) → Kafka + HDFS`（一跳到位）
现在改成：

> **文件 → Flume①（TAILDIR）→ Kafka → Flume②（KafkaSource）→ HDFS**

这样一来，同学不仅会用 **KafkaSink**，还会学到 **KafkaSource**，真正体会到“Kafka 做总线、Flume 只负责采集和落地”的架构思想，Flume+Kafka 的认知就更完整了。

实验四：基于 Kafka 总线的 Flume 双向采集实验【进阶】

一、实验目的

通过本实验，学生将能够：

1. 理解 **“Kafka 作为统一数据总线”** 的架构思想：
    - 上游 Flume 把数据推到 Kafka；

    - 下游 Flume 再从 Kafka 拉数据落地 HDFS。
2. 掌握 **KafkaSink** 与 **KafkaSource** 的基本配置和区别：
    - KafkaSink：作为“生产者”，把数据写入 Kafka 主题；
    - KafkaSource：作为“消费者”，从 Kafka 主题订阅数据。
3. 能够配置 **两个 Flume Agent**：
    - Agent1：`TAILDIR Source → KafkaSink`
    - Agent2：`KafkaSource → HDFSSink`
4. 理解 **“解耦”** 的好处：
   采集（Flume①）和落地（Flume②）互不影响，中间由 Kafka 负责缓存和转发。

二、实验背景与原理

在很多企业里，架构是这样的：

> 各种业务系统 → 一堆 Flume Agent → **统一写入 Kafka** → 下游多个系统（Flink / Spark / 另一个 Flume / 监控系统）从 Kafka 订阅数据。

也就是说，Kafka像一个“**总线 / 数据中转站**”，
上游谁要发数据，就往 Kafka 丢；
下游谁要用数据，就从 Kafka 拿。

本实验做一个 **简化版双段链路**：

1. **第一段：采集到总线**
    - slave2 上的脚本模拟实时写招聘信息到 `messages` 文件；
    - slave1 的 Flume① 使用 **TAILDIR Source** 监听 `/opt/apps/flume/logs/messages`；
    - 使用 **KafkaSink** 把日志写入 Kafka 主题 `job_info_stream`。
2. **第二段：从总线落地**
    - master 上的 Flume② 使用 **KafkaSource** 订阅 `job_info_stream`；
    - 使用 **HDFSSink** 把数据写到 HDFS 目录 `/user/flumebackup/job_info_from_kafka/`。

最终数据链路：

> slave2:messages → Flume①(TAILDIR) → Kafka:job_info_stream → Flume②(KafkaSource) → HDFS

三、实验环境与前置条件

与上一实验基本相同：

1. 三节点集群：`master、slave1、slave2`
2. 已安装并配置好：
    - Hadoop 集群（HDFS + YARN）
    - ZooKeeper 集群
    - Kafka 集群

- Flume（假设安装在 master、slave1，至少这两台要有 Flume）

3. 三台机器主机名互通（建议 SSH 免密已配置）。
4. HDFS NameNode 地址：`hdfs://master:9000`

> ⚠️ 如果你目前只在 slave1 装了 Flume，也可以 **在 slave1 上同时跑两个 Agent**（a1、a2），配置里路径稍微改一下即可，整体思路不变。

四、实验步骤

> 总思路：
> ① 准备数据源（和上个实验基本一样，继续用招聘信息）
> ② 启动 Hadoop / ZK / Kafka
> ③ 配置并启动 Flume①：文件 → Kafka
> ④ 配置并启动 Flume②：Kafka → HDFS
> ⑤ 同时验证 Kafka 和 HDFS 中的数据

步骤 1：准备实时数据源和脚本（slave2）

仍然使用上一实验的方式，从本地 `messages` 文件 **一行一行加时间戳**，通过 SSH 追加到 slave1 的日志文件。

1）在 slave2 创建目录和文件

```
mkdir -p /opt/data/flume/logs
cd /opt/data/flume/logs
```

准备 `messages` 文件（内容与之前相同，例如）：

```
1,玛纳斯分公司（白酒销售）销售员,新疆五谷传奇酒业...,玛纳斯,3800-10000元/月
2,仪表工,新疆雅澳科技有限责任公司,玛纳斯,5000-7000元/月
3,加液工,石河子市物源盛通商贸有限公司,北泉镇,3500-4500元/月
4,大锅饭师傅,石河子市天石肉类加工有限公司,石河子,3000-3300元/月
5,维修工,石河子市天石肉类加工有限公司,北泉镇,5000-6000元/月
......
```

2）创建 append_data.sh 脚本

```
vi append_data.sh
```

内容：

```
#!/bin/bash

# 本地要读取的文件（在 slave2 上）
LOGFILE="./messages"

# 远程目标主机（slave1）和目标文件路径
TARGET_HOST="slave1"
TARGET_FILE="/opt/apps/flume/logs/messages"
```

```
COUNT=1   # 计数器，从第 1 条开始

while true; do
    # 从本地 LOGFILE 一行一行读取数据
    while IFS= read -r line || [ -n "$line" ]; do
        # 1）生成当前时间戳：YYYY-MM-DD HH:MM:SS
        timestamp=$(date "+%Y-%m-%d %H:%M:%S")

        # 2）拼接“时间戳 + 原始内容”
        newline="$timestamp,$line"

        # 3）通过 ssh 追加到 slave1 的目标文件
        printf '%s\n' "$newline" | ssh "$TARGET_HOST" "cat >> '$TARGET_FILE'"

        # 4）在本机终端打印提示
        echo "成功向 slave1 发送第 ${COUNT} 条数据：$newline"

        # 5）累计条数 +1
        COUNT=$((COUNT + 1))

        # 每写一行停 1 秒，模拟实时输入
        sleep 1
    done < "$LOGFILE"
done
```

赋执行权限：

```
chmod +x append_data.sh
```

> ⚠️ 在 slave1 上提前创建目录和空文件：

```
mkdir -p /opt/apps/flume/logs
touch /opt/apps/flume/logs/messages
```

步骤 2：启动 Hadoop、ZooKeeper、Kafka

在 master 启动 HDFS：

```
start-dfs.sh
```

在 slave1 启动 YARN：

```
start-yarn.sh
```

在 master / slave1 / slave2 启动 ZooKeeper：

```
zkServer.sh start
```

在 master / slave1 / slave2 启动 Kafka：

```
kafka-server-start.sh -daemon /opt/apps/kafka/config/server.properties
```

步骤 3：创建 Kafka 主题 job_info_stream

在 **master** 上创建主题：

```
kafka-topics.sh --create \
--bootstrap-server master:9092,slave1:9092,slave2:9092 \
--topic job_info_stream \
--replication-factor 1 \
--partitions 3
```

验证主题存在：

```
kafka-topics.sh --list \
--bootstrap-server master:9092,slave1:9092,slave2:9092
# 应看到 job_info_stream
```

步骤 4：配置 Flume① —— 文件 → Kafka（slave1）

Flume① 的任务：
监听 `/opt/apps/flume/logs/messages` 文件的实时新增内容，将日志写入 Kafka 主题 `job_info_stream`。

1）编写配置文件

在 `slave1` 上：

```
cd /opt/apps/flume/conf
touch taildir-to-kafka.conf
```

内容（简化版，只写 Kafka，不写 HDFS）：

```
## 定义 Agent 名称
a1.sources  = r1
a1.sinks    = k1
a1.channels = c1

## 1. 配置 Source：TAILDIR 实时监听文件
a1.sources.r1.type = TAILDIR

# positionFile：记录文件读取位置（偏移量），用于断点续传
a1.sources.r1.positionFile = /opt/apps/flume/taildir_position.json

# 要监听的文件组
a1.sources.r1.filegroups = fg1
# 监听 slave1 本机上的 messages 文件
a1.sources.r1.filegroups.fg1 = /opt/apps/flume/logs/messages
```

```
## 2. 配置 Channel：内存 Channel
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

## 3. 配置 Sink：写入 Kafka
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink

# Kafka 集群地址
a1.sinks.k1.brokerList = master:9092,slave1:9092,slave2:9092

# 写入的 Kafka 主题
a1.sinks.k1.topic = job_info_stream

# 一条一条实时发送（教学用，便于观察）
a1.sinks.k1.batchSize = 1

## 4. 绑定关系
# Source r1 使用 Channel c1
a1.sources.r1.channels = c1
# Sink k1 也使用 Channel c1
a1.sinks.k1.channel = c1
```

步骤 5：配置 Flume② —— Kafka → HDFS（master）

Flume② 的任务：
从 Kafka 主题 `job_info_stream` 订阅数据，写入 HDFS `/user/flumebackup/job_info_from_kafka/`。

1）在 master 创建 HDFS 目录

```
hdfs dfs -mkdir -p /user/flumebackup/job_info_from_kafka/
```

2）编写 Flume② 配置文件

在 **master** 上：

```
cd /opt/apps/flume/conf
touch kafka-to-hdfs.conf
```

内容：

```
## 定义 Agent 名称
a2.sources  = r1
a2.sinks    = k1
a2.channels = c1

## 1. 配置 Source：KafkaSource，从 Kafka 订阅数据
a2.sources.r1.type = org.apache.flume.source.kafka.KafkaSource

# Kafka 集群地址
```

```
a2.sources.r1.kafka.bootstrap.servers = master:9092,slave1:9092,slave2:9092

# 要订阅的主题（可以是多个，这里只用一个）
a2.sources.r1.kafka.topics = job_info_stream

# 消费组 ID，用于管理消费进度
a2.sources.r1.kafka.consumer.group.id = job_info_flume_group

# 每次从 Kafka 拉取的最大记录数（教学中保持默认即可，也可显式写出）
a2.sources.r1.kafka.consumer.max.poll.records = 100

## 2. 配置 Channel：内存 Channel
a2.channels.c1.type = memory
a2.channels.c1.capacity = 1000
a2.channels.c1.transactionCapacity = 100

## 3. 配置 Sink：写入 HDFS
a2.sinks.k1.type = hdfs

# 指定 HDFS 目标路径
a2.sinks.k1.hdfs.path = hdfs://master:9000/user/flumebackup/job_info_from_kafka/

# 使用 DataStream 流式写入
a2.sinks.k1.hdfs.fileType    = DataStream
a2.sinks.k1.hdfs.writeFormat = Text

# 文件滚动策略：按条数滚动
a2.sinks.k1.hdfs.rollCount    = 100
a2.sinks.k1.hdfs.rollInterval = 0
a2.sinks.k1.hdfs.rollSize     = 0

## 4. 绑定 Source、Channel、Sink
a2.sources.r1.channels = c1
a2.sinks.k1.channel    = c1
```

> 🔎 课堂讲解要点：
>
> - 这里使用的是 **KafkaSource**，而不是之前的 TAILDIR Source；
> - 配置中多了 `kafka.bootstrap.servers`、`kafka.topics`、`kafka.consumer.group.id` 这类 Kafka 专用参数；
> - 整体结构和之前的“普通 Flume”一样：Source → Channel → Sink，只是 Source 换成了 KafkaSource。

步骤 6：依次启动 Flume 和脚本并观察数据流

1）在 slave1 启动 Flume①（文件 → Kafka）

```
flume-ng agent \
-n a1 \
-c /opt/apps/flume/conf \
-f /opt/apps/flume/conf/taildir-to-kafka.conf \
-Dflume.root.logger=INFO,console
```

保持窗口开着，观察日志输出是否有 ERROR。

2）在 master 启动 Flume②（Kafka → HDFS）

另开一个终端：

```
flume-ng agent \
-n a2 \
-c /opt/apps/flume/conf \
-f /opt/apps/flume/conf/kafka-to-hdfs.conf \
-Dflume.root.logger=INFO,console
```

同样保持窗口开着。

3）在 master 启动一个 Kafka 消费者（仅用来"旁路观察"）

```
kafka-console-consumer.sh \
--bootstrap-server master:9092,slave1:9092,slave2:9092 \
--topic job_info_stream \
--group test_console_group
```

> 这个消费者不是必须的，但非常适合作为"窗口"，观察 Flume① 往 Kafka 写进来的数据。

4）在 slave2 启动实时数据追加脚本

```
cd /opt/data/flume/logs
./append_data.sh
```

终端会持续输出类似：

```
成功向 slave1 发送第 1 条数据：2025-11-21 10:35:32,1,玛纳斯分公司（白酒销售）销售员,新疆五谷传奇酒业...,玛纳斯,3800-10000元/月
成功向 slave1 发送第 2 条数据：2025-11-21 10:35:33,2,仪表工,新疆雅澳科技有限责任公司,玛纳斯,5000-7000元/月
...
```

此时：

- slave1 上 Flume① 的控制台应显示事件被发送到 Kafka；
- master 上 Kafka 控制台消费者会实时打印招聘数据；
- master 上 Flume② 也在悄悄把这些数据写入 HDFS。

五、结果验证

1. 验证 Kafka 中的数据（实时）

在 **master** 的 Kafka 消费者窗口，应该持续看到类似输出：

```
2025-11-21 10:35:32,1,玛纳斯分公司（白酒销售）销售员,新疆五谷传奇酒业...,玛纳斯,3800-10000元/月
2025-11-21 10:35:33,2,仪表工,新疆雅澳科技有限责任公司,玛纳斯,5000-7000元/月
...
```

说明 **文件 → Flume① → Kafka** 这一段是通的。

---

2. 验证 HDFS 中的数据（离线）

在 **master** 上查看 HDFS 目录：

```
hdfs dfs -ls /user/flumebackup/job_info_from_kafka/
```

应看到一个或多个 Flume 生成的文件，例如：

```
/user/flumebackup/job_info_from_kafka/FlumeData.1730xxxxx
/user/flumebackup/job_info_from_kafka/FlumeData.1730xxxxx.tmp
...
```

查看部分内容：

```
hdfs dfs -cat /user/flumebackup/job_info_from_kafka/* | head -5
```

看到的数据格式应与 Kafka 消费者窗口中看到的格式一致。
 说明 **Kafka → Flume② → HDFS** 这一段也是通的。

六、重要注意事项（与实验三的对比）

点	实验三	新实验（实验四）
Source 类型	TAILDIR（从文件读）	Flume①：TAILDIR；Flume②：**KafkaSource**
Sink 类型	HDFSSink + KafkaSink（同一个 Agent）	Flume①：KafkaSink；Flume②：HDFSSink
HDFS 写入位置	直接由 Flume 把文件写 HDFS	**先进 Kafka，再由另一 Flume 从 Kafka 写入 HDFS**
架构特点	“一跳到位”：文件 → Flume → Kafka+HDFS	“两段式”：采集 → Kafka 总线 → 落地
学习重点	多路复用 + TAILDIR	**Kafka 作为“总线”、KafkaSource 消费机制、采集与落地解耦**

七、实验小结（给学生的话）

通过这个“改编版”实验，同学要能说清楚三句话：

1. **Kafka 是总线**：
   采集端和落地端不用直接绑在一起，它们只需要都和 Kafka 打交道。
2. **Flume 既可以“写 Kafka”，也可以“读 Kafka”**：
   - 写：用 **KafkaSink**；
   - 读：用 **KafkaSource**。
3. **解耦的好处**：
   - 采集端出了问题，只影响写 Kafka，不影响已经在 Kafka 里的数据；
   - 落地端（Flume② 或 Spark/Flink）可以暂停、重启、重跑，只要 Kafka 还在，数据就还在。

> 你可以把本实验作为 **“实验四：Flume+Kafka 双向采集与总线架构”** 放在实验三后面，前后连起来讲，Flume+Kafka 这一块体系就非常完整了。