

# Data Ingestion Tools

211870287 丁旭

September 2023

## 1 任务一使用 Apache Kafka 进行数据流

### 1. 预置安装 Zookeeper-3.9.0

0、下载zookeeper

```
wget https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/zookeeper-3.9.0-bin.tar.gz
```

1、解压到 /usr/local/ 目录下

```
tar -zxvf ./apache-zookeeper-3.9.0-bin.tar.gz -C /usr/local/
```

2、进入目录并修改名称

```
cd /usr/local && mv apache-zookeeper-3.9.0-bin/ zookeeper-3.9.0
```

3、进入 zookeeper 创建两个文件

```
cd zookeeper-3.9.0
```

```
mkdir data
```

```
mkdir logs
```

4、创建配置文件

```
vim conf/zoo.cfg
```

5、写入配置信息

```
tickTime = 2000
```

```
dataDir = /usr/local/zookeeper-3.9.0/data
```

```
dataLogDir = /usr/local/zookeeper-3.9.0/logs
```

```
tickTime = 2000  
clientPort = 2181  
initLimit = 5  
syncLimit = 2
```

## 启动 zookeeper

启动服务：

```
/usr/local/zookeeper-3.9.0/bin/zkServer.sh start
```

连接服务：

/usr/local/zookeeper-3.9.0/bin/zkCli.sh

查看服务状态：

```
/usr/local/zookeeper-3.9.0/bin/zkServer.sh status
```

停止服务：

```
/usr/local/zookeeper-3.9.0/bin/zkServer.sh stop
```

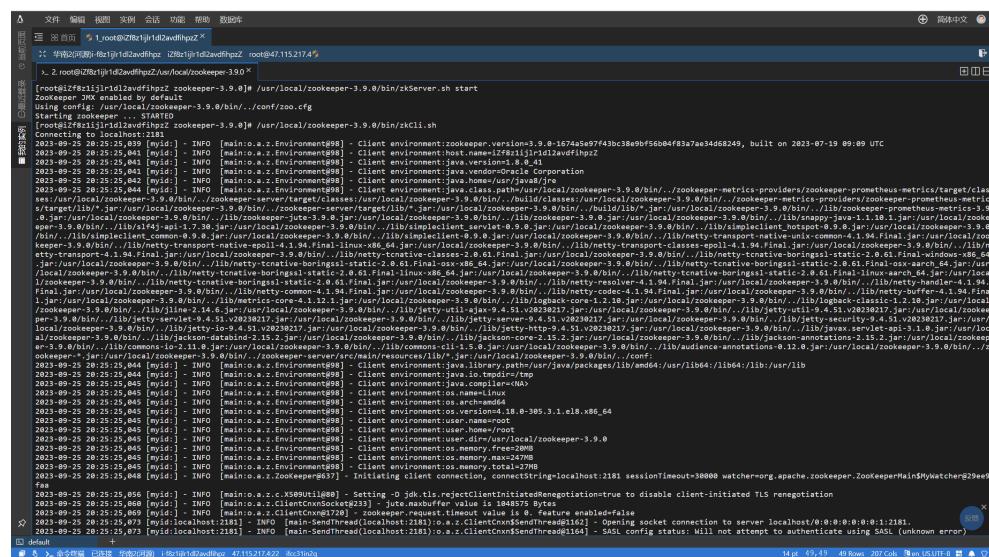


图 1: zookeeper 启动

```
[root@iZf8zijlrlid12avdfihpzZ zookeeper-3.9.0]# /usr/local/zookeeper-3.9.0/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.9.0/bin/../conf/zoo.cfg
myid could not be determined, will not able to locate clientPort in the server configs.
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: standalone
[root@iZf8zijlrlid12avdfihpzZ zookeeper-3.9.0]# /usr/local/zookeeper-3.9.0/bin/zkServer.sh stop
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.9.0/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
[root@iZf8zijlrlid12avdfihpzZ zookeeper-3.9.0]#
```

图 2: zookeeper status 命令

```
[root@iZf8zijlrlid12avdfihpzZ zookeeper-3.9.0]# /usr/local/zookeeper-3.9.0/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.9.0/bin/../conf/zoo.cfg
myid could not be determined, will not able to locate clientPort in the server configs.
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: standalone
[root@iZf8zijlrlid12avdfihpzZ zookeeper-3.9.0]# /usr/local/zookeeper-3.9.0/bin/zkServer.sh stop
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.9.0/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
[root@iZf8zijlrlid12avdfihpzZ zookeeper-3.9.0]#
```

图 3: zookeeper stop 命令

## 2. 安装 Kafka

0、linux 中使用 wget 命令，远程下载 kafka

```
wget https://archive.apache.org/dist/kafka/2.8.1/kafka_2.12-2.8.1.tgz
--no-check-certificate
```

1、解压压缩包到 /usr/local/ 目录

```
tar -zxvf ./kafka_2.12-2.8.1.tgz -C /usr/local/
```

2、进入 /usr/local/ 目录

```
cd /usr/local/
```

3、修改原始名称

```
mv ./kafka_2.12-2.8.1/ kafka2.12
```

4、进入 Kafka 并修改配置文件

```
cd kafka2.12
```

```
vim config/server.properties
```

修改其中的：

```

broker.id=1
log.dir=/usr/local/kafka/kafka-logs

#配置 kafka 的监听端口
listeners=PLAINTEXT://:9092

#把 kafka 的地址端口注册给 zookeeper , 如果是远程访问要改成外网 IP
advertised.listeners=PLAINTEXT://:9092
-----
```

kafka 启动命令(必须先启动 zookeeper ! ):

```

/usr/local/zookeeper-3.9.0/bin/zkServer.sh start
/usr/local/kafka2.12/bin/kafka-server-start.sh -daemon config/server.properties
注：可以将 bin 文件路径加到 .bashrc 中
例：
export PATH="/usr/local/kafka/bin:$PATH"
source ~/.bashrc
kafka-server-start.sh -daemon config/server.properties
```

kafka 停止命令：

```
bin/kafka-server-stop.sh
```

```
[root@iZf8z1ijlrl1d12avdfihpz ~]# /usr/local/zookeeper-3.9.0/bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.9.0/bin/../conf/zoo.cfg
Starting zookeeper ... already running as process 1037135.
[root@iZf8z1ijlrl1d12avdfihpz ~]# /usr/local/kafka2.12/bin/kafka-server-start.sh -daemon config/server.properties
[root@iZf8z1ijlrl1d12avdfihpz ~]#
```

图 4: kafka version 命令

```
[root@iZf8z1ijlrl1d12avdfihpz ~]# /usr/local/zookeeper-3.9.0/bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper-3.9.0/bin/../conf/zoo.cfg
Starting zookeeper ... already running as process 1037135.
[root@iZf8z1ijlrl1d12avdfihpz ~]# /usr/local/kafka2.12/bin/kafka-server-start.sh -daemon config/server.properties
[root@iZf8z1ijlrl1d12avdfihpz ~]#
```

图 5: 启动 kafka

3. 生产和消费一个简单的消息。

### 创建 Topic

1、进入到 Kafka 目录下， 创建一个名为 testTopic 的 Topic。

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor
```

2、查看创建的 Topic 列表。

```
bin/kafka-topics.sh --list --zookeeper localhost:2181
```

### 测试通信

1、连接生产者， 命令后面需要对应 Topic 名称。

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic testTopic
```

```
>hhh
```

```
>hello
```

```
>actually you are
```

2 连接消费者， 需要连接消费者的 Topic

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic testTopic
```

```
>hhh
```

```
>hello
```

```
>actually you are
```

```
[root@iZf8ziijlr1dl2avdfihpzz kafka2.12]# bin/kafka-server-start.sh -daemon config/server.properties
[root@iZf8ziijlr1dl2avdfihpzz kafka2.12]# bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic testTopic
Created topic testTopic.
[root@iZf8ziijlr1dl2avdfihpzz kafka2.12]# bin/kafka-topics.sh --list --zookeeper localhost:2181
testTopic
[root@iZf8ziijlr1dl2avdfihpzz kafka2.12]# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic testTopic
>hhh you are a big shaniao
>what?
>actually you are!
><[root@iZf8ziijlr1dl2avdfihpzz kafka2.12]# bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic testTopic --from-beginning
hhh you are a big shaniao
what?
actually you are!
```

图 6: 任务一

#### 4. 验证

输入与输出一致, 任务完成

## 2 任务三使用 Apache NiFi 进行数据流管理

#### 1. 安装 nifi

1. 使用 wget 命令 安装 nifi ( 这里我是从清华源安装下载的 )

```
wget https://mirrors.tuna.tsinghua.edu.cn/apache/nifi/1.23.2/nifi-1.23.2-
```

2. 解压 nifi-1.23.2-bin.tar.gz 到 /opt/ 目录下面

```
tar -zxvf nifi-1.23.2-bin.tar.gz -C /opt/
```

3. 改名 nifi

```
mv nifi-1.23.2-bin nifi
```

4. 配置 conf/nifi.properties

```
https.host: 127.0.0.1
```

```
https.port:8443
```

```
# web properties #
#####
# For security, NiFi will present the UI on 127.0.0.1 and only be accessible through this loopback interface.
# Be aware that changing these properties may affect how your instance can be accessed without any restriction.
# We recommend configuring HTTPS instead. The administrators guide provides instructions on how to do this.

nifi.web.http.host=
nifi.web.http.port=
nifi.web.http.network.interface.default=

#####
nifi.web.https.host=127.0.0.1
nifi.web.https.port=8443
nifi.web.https.network.interface.default=
nifi.web.https.application.protocols=http/1.1
nifi.web.jetty.working.directory=../work/jetty
nifi.web.jetty.threads=200
nifi.web.max.header.size=16 KB
nifi.web.proxy.context.path=
nifi.web.proxy.host=
nifi.web.max.content.size=
nifi.web.max.requests.per.second=30000
nifi.web.max.access.token.requests.per.second=25
nifi.web.request.timeout=60 secs
nifi.web.request.ip.whitelist=
nifi.web.should.send.server.version=true
nifi.web.request.log.format=%{client}a - %u %t "%r" %s %o "%{Referer}i" "%{User-Agent}i"

# Filter JMX MBeans available through the System Diagnostics REST API
```

图 7: conf/nifi.properties

5. 运行 nifi

```
nifi.sh start
```

6. 端口映射（因为部署在云服务器的私有 ip 上 但是不能直接访问网页 需要映射到这里 使用 ssh 来实现

```
ssh -i D:/NJU/tmp/nifi.pem -L 8443:localhost:8443 root@47.115.217.4
```

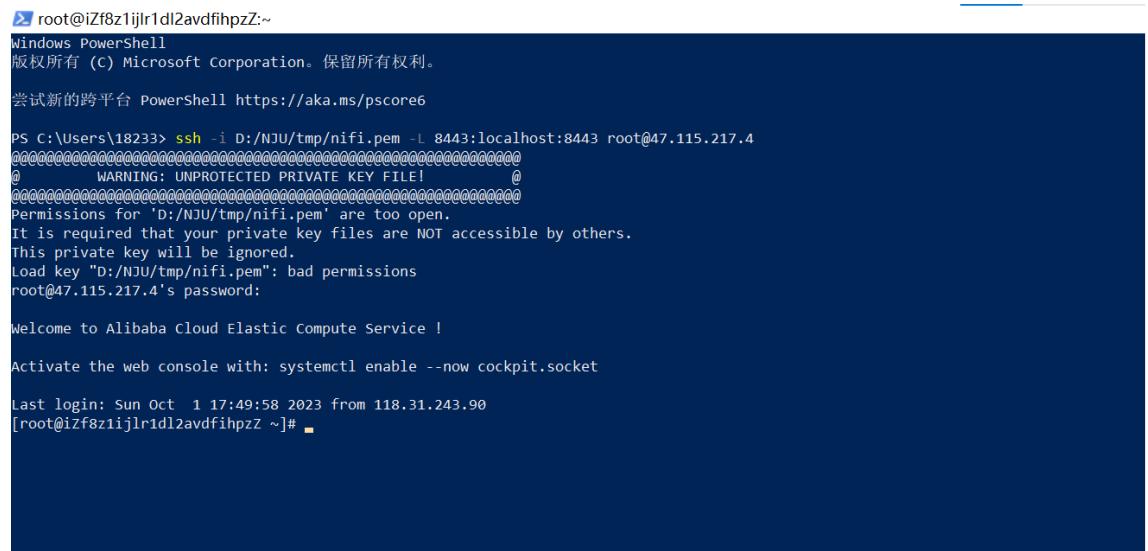
参数说明：

D:/NJU/tmp/nifi.pem 是存放密钥的路径

8443 映射端口号

root 用户名

47.115.217.4 公网 ip



```
PS C:\Users\18233> ssh -i D:/NJU/tmp/nifi.pem -L 8443:localhost:8443 root@47.115.217.4
@@@@@@@WARNING: UNPROTECTED PRIVATE KEY FILE! @@@@
Permissions for 'D:/NJU/tmp/nifi.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "D:/NJU/tmp/nifi.pem": bad permissions
root@47.115.217.4's password:

Welcome to Alibaba Cloud Elastic Compute Service !

Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Oct  1 17:49:58 2023 from 118.31.243.90
[root@izf8z1ijlrl1dl2avdfihpz ~]#
```

图 8: ssh 端口映射

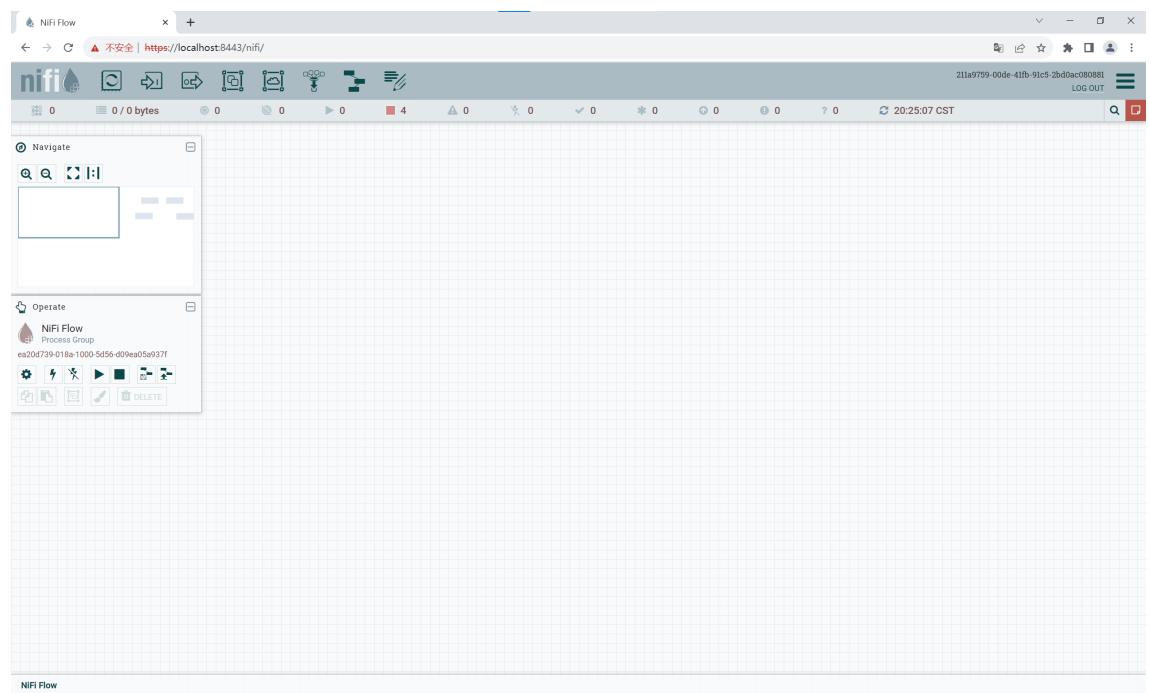


图 9: nifi 的 web 界面

## 2. 安装 mysql

1. 下载和安装yum源：

```
wget https://dev.mysql.com/get/mysql80-community-release-el8-5.noarch.rpm  
rpm -ivh mysql80-community-release-el8-5.noarch.rpm
```

## 2. 安装 mysql 服务

```
yum -y install mysql-server
```

```
[root@izf8z1ijlr1dl2avdfihpzz ~]# mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 26  
Server version: 8.0.34 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2023, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> select version();  
+-----+  
| version() |  
+-----+  
| 8.0.34 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

图 10: mysql-version

```
mysql> show databases;  
+--------------------+  
| Database          |  
+--------------------+  
| information_schema |  
| mysql              |  
| performance_schema |  
| sys                |  
| test_nifi          |  
+--------------------+  
5 rows in set (0.02 sec)  
  
mysql>
```

图 11: database

### 3. 安装 MySQL 连接驱动文件 mysql-connector-java

#### 1. 下载

```
wget https://downloads.mysql.com/archives/get/p/3/file/mysql-connector-j
```

#### 2. 安装依赖

```
yum -y install java-headless
```

#### 3. 安装 mysql-connector-java

```
[root@study1 opt]# rpm -ivh mysql-connector-java-8.0.22-1.el7.noarch.rpm
```

#### 4. 将 mysql-connector-java.jar 移至 /usr/java8/jre/lib/ext 中

```
mv mysql-connector-java.jar /usr/java8/jre/lib/ext
```

```
[root@izf8zijlrlrd12awdfihpz ~]# cd /usr/java8/
[root@izf8zijlrlrd12awdfihpz java8]# ls
ADDITIONAL_LICENSE_INFO_ASSEMBLY_EXCEPTION bin demo include jre lib LICENSE man release sample src.zip THIRD_PARTY_README
[root@izf8zijlrlrd12awdfihpz java8]# cd jre/
[root@izf8zijlrlrd12awdfihpz jre]# ls
ADDITIONAL_LICENSE_INFO_ASSEMBLY_EXCEPTION bin lib LICENSE THIRD_PARTY_README
[root@izf8zijlrlrd12awdfihpz jre]# cd lib/
[root@izf8zijlrlrd12awdfihpz lib]# ls
andd4 charsets.jar content-types.properties flavormap.properties jce.jar jse.jar management net.properties resources.jar sound.properties
applet classlist currency.data hijrah-config-umalqura.properties jexec jvm.hprof.txt management-agent.jar psfontj2d.properties rt.jar tzdb.dat
calendars.properties cmm ext images jexec.diz logging.properties meta-index psfont.properties.ja security
[root@izf8zijlrlrd12awdfihpz lib]# cd ext/
[root@izf8zijlrlrd12awdfihpz ext]# ls
[root@izf8zijlrlrd12awdfihpz ext]# ls
[root@izf8zijlrlrd12awdfihpz ext]# pwd
/usr/java8/jre/lib/ext
[root@izf8zijlrlrd12awdfihpz ext]#
```

图 12: /usr/java8/jre/lib/ext

### 4. 实现平面文件.json 存储到 mysql

#### 1. 创建 json 数据

```
cd ~/file
vim data.json
```

#数据如下

```
[{
    {
        "name": "张三",
        "age": 23,
        "gender": 1
    },
}
```

```

        "name": "李四",
        "age": 24,
        "gender": 1
    }, {
        "name": "小红",
        "age": 18,
        "gender": 0
    }
]

```

The screenshot shows a terminal window with the following content:

```

文件 编辑 视图 实例 会话 功能 帮助 数据库
简体中文
[ 1.root@iZbx1jrl1d2avdfhpz ~ ] 2. root@iZbx1jrl1d2avdfhpz ~ /file
{
    "name": "张三",
    "age": 23,
    "gender": 1
}, {
    "name": "李四",
    "age": 24,
    "gender": 1
}, {
    "name": "小红",
    "age": 18,
    "gender": 0
}

```

The terminal window has a dark theme. The title bar says "简体中文". The command prompt is at the top. The main area displays the JSON data. The bottom status bar shows the file path "/file", the user "root", the host "iZbx1jrl1d2avdfhpz", the port "47.115.217.42", and the command "4egox2kf".

图 13: data.json

## 2. 在数据库中创建表

```

CREATE TABLE `sys_user` (
    `id` bigint NOT NULL AUTO_INCREMENT COMMENT '用户ID',
    `name` varchar(50) NOT NULL DEFAULT '' COMMENT '姓名',
    `age` int NOT NULL DEFAULT 0 COMMENT '年龄',
    `gender` tinyint NOT NULL COMMENT '性别,1:男,0:女',
)

```

```
'create_time' timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
'is_deleted' tinyint NOT NULL DEFAULT '0' COMMENT '是否已删除',
PRIMARY KEY ('id') USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci ROW_
```

### 3. 创建文件流

添加处理器：GetFile

点击工具栏的 Processor , 拖拽到画布中

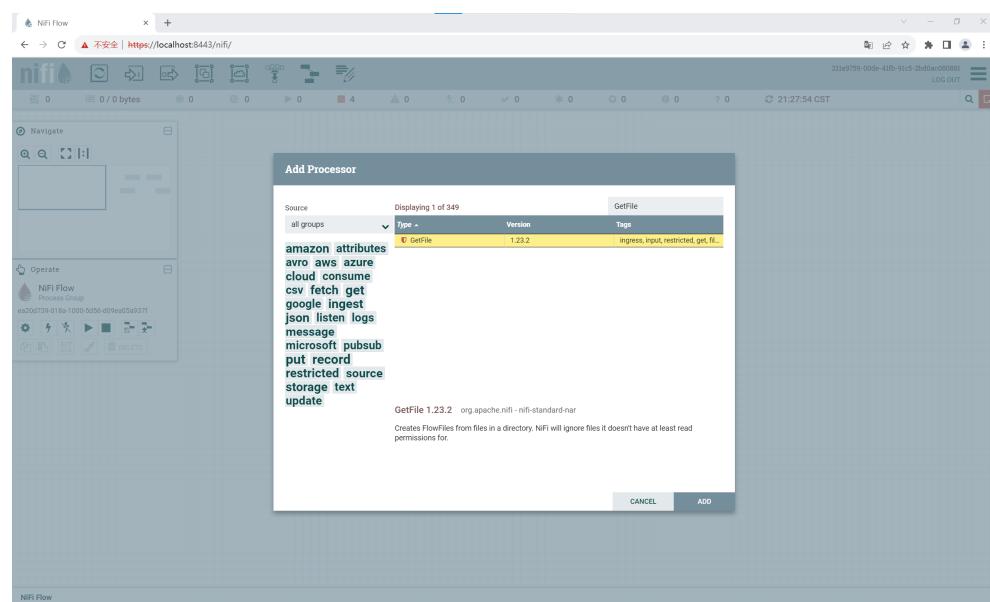


图 14: 筛选 GetFile, 点击 ADD 添加到画布中

配置 GetFile 处理器

双击添加的处理器，弹出对应的配置界面

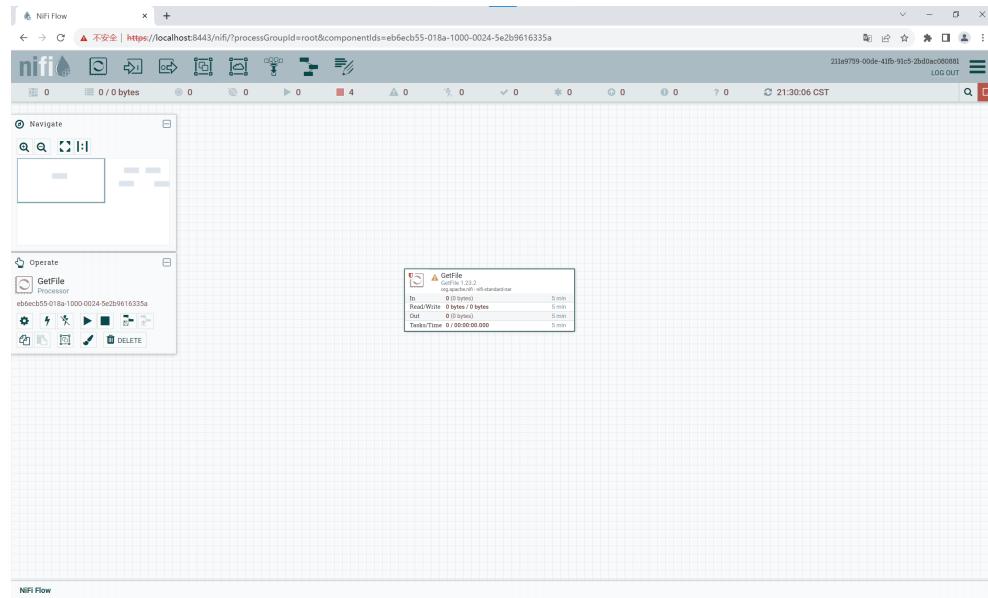


图 15: GetFile

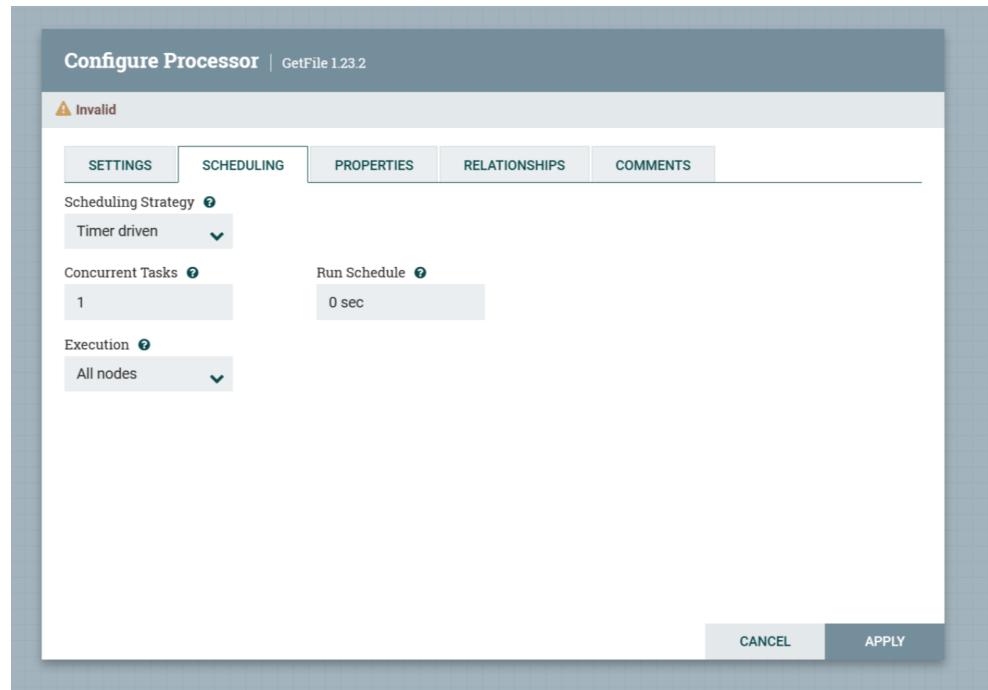


图 16: 配置界面

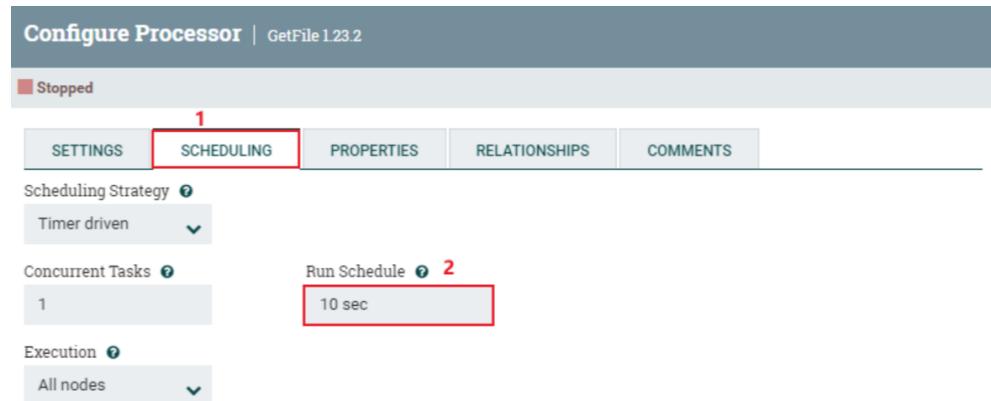


图 17: 设置 10s 定时器

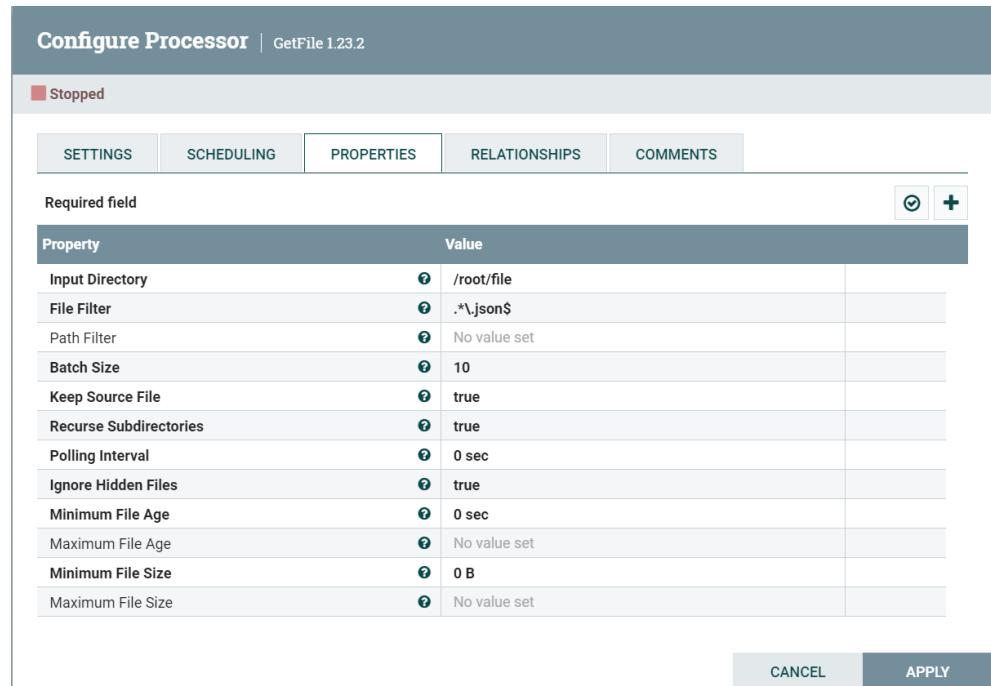


图 18: 设置文件路径

## 添加和配置处理器：SplitJson

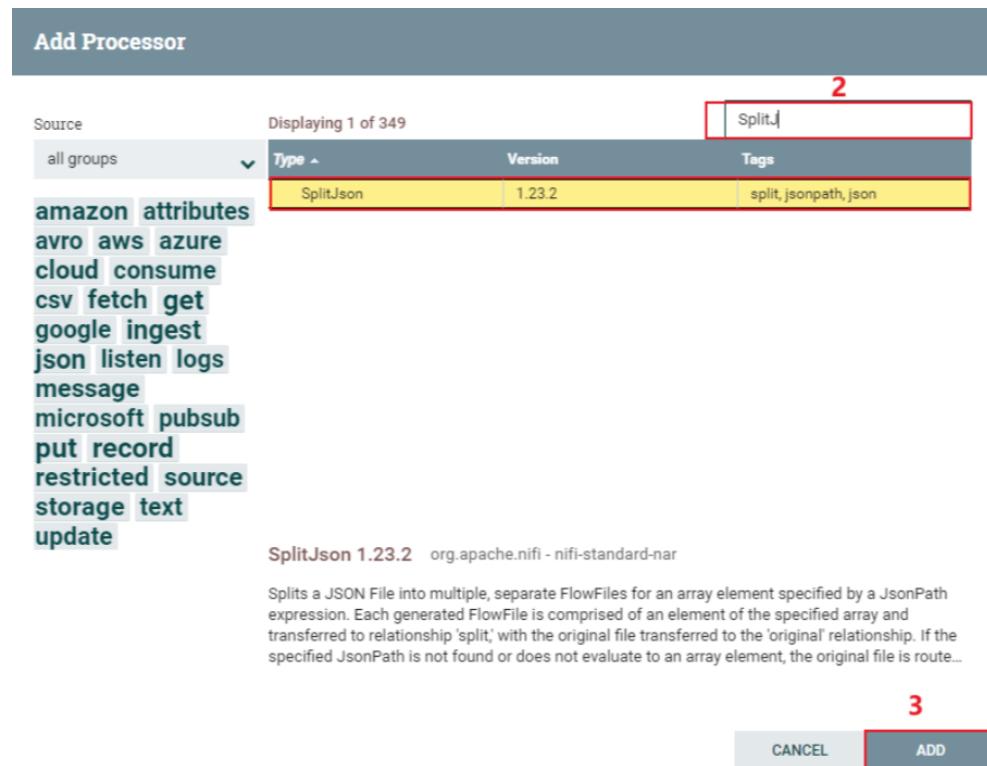


图 19: 添加处理器 SplitJson

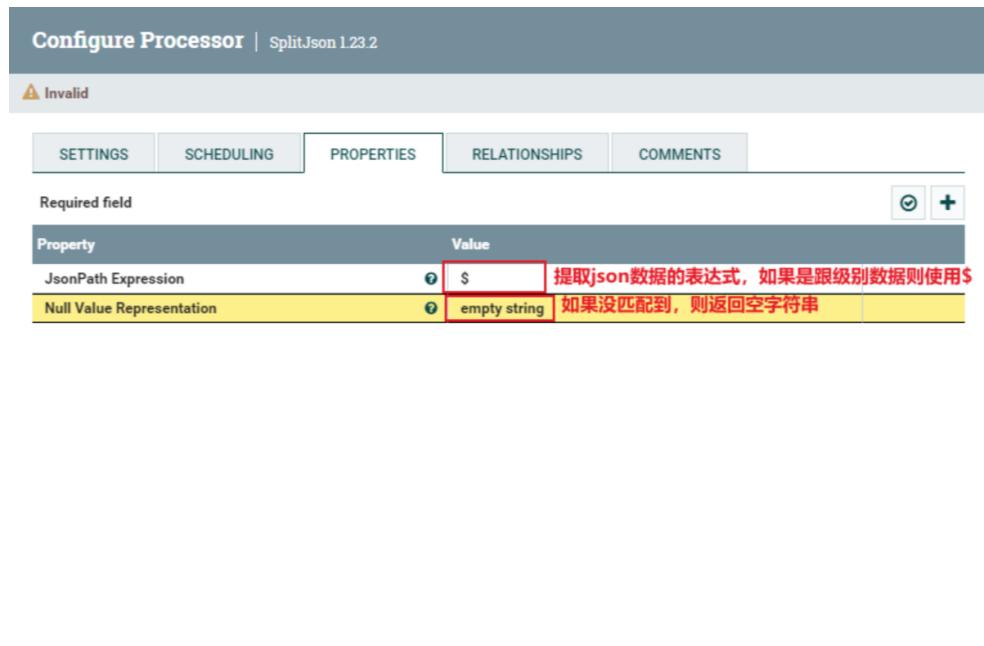


图 20: 配置处理器 SplitJson

### 连接处理器

将 GetFile 处理器和 SplitJson 处理器连接起来，勾选 For Relationships，然后选择

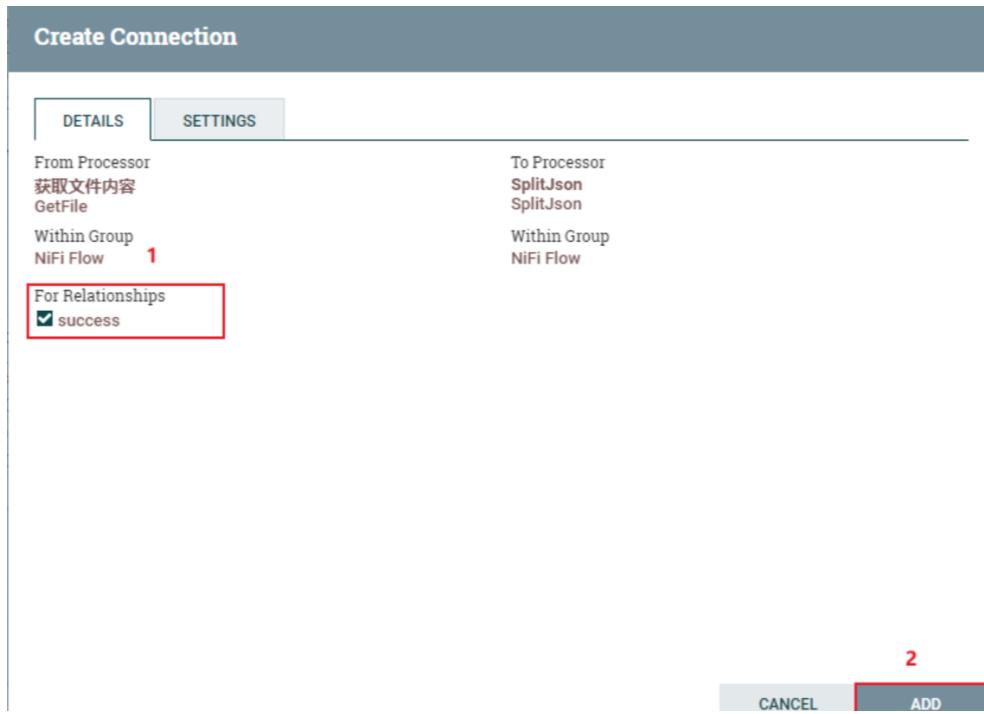


图 21: 将 GetFile 处理器和 SplitJson 处理器连接

## Json 转为SQL

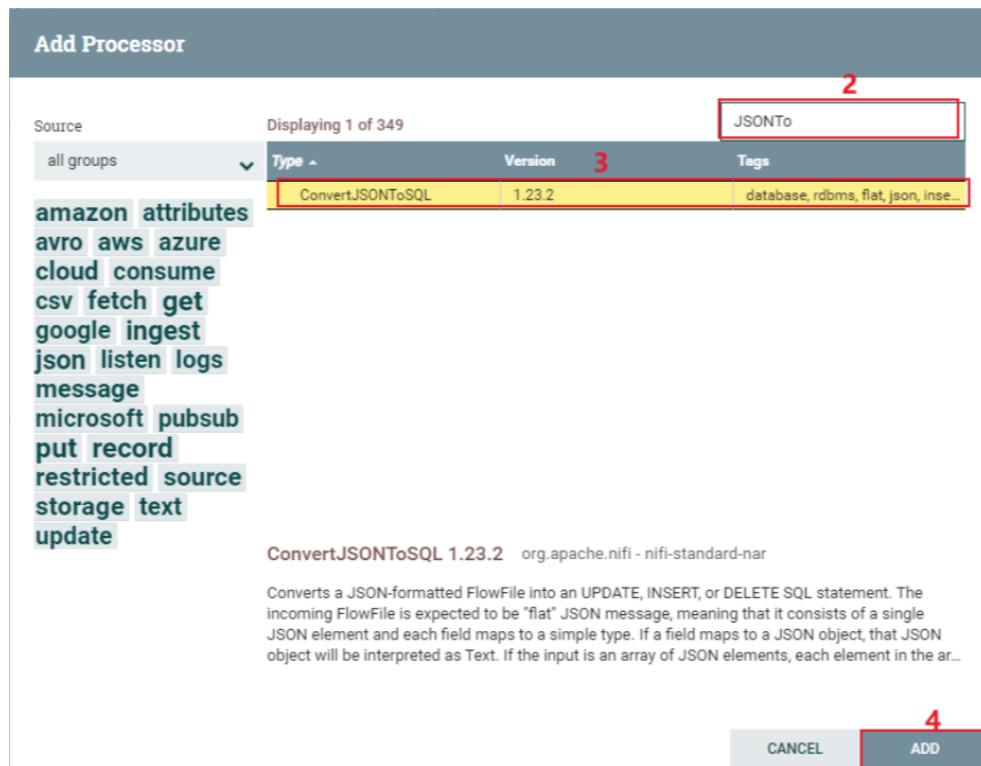


图 22: 添加处理器 ConvertJSONToSQL

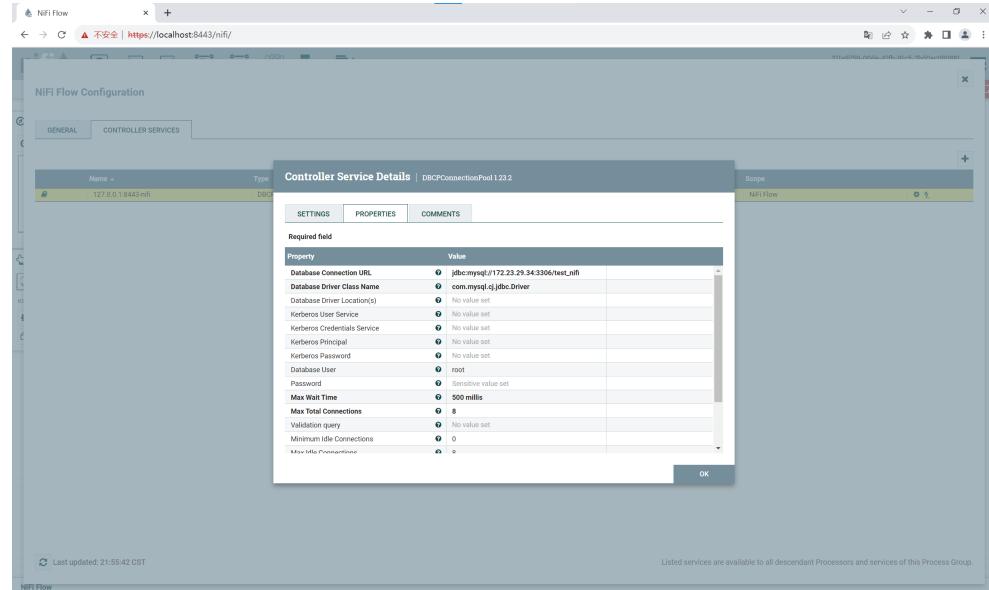


图 23: 配置处理器 ConvertJSONToSQL

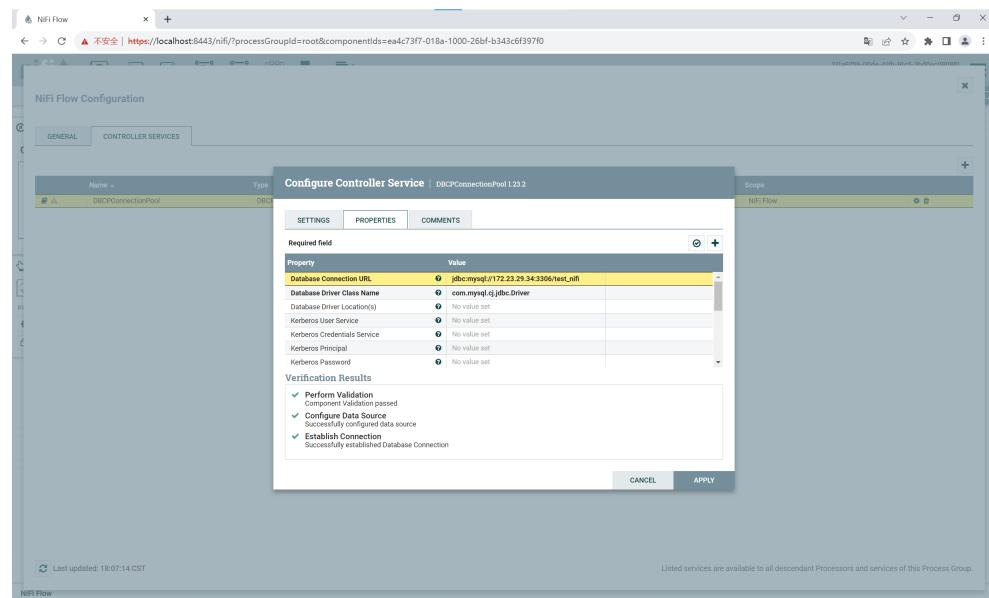


图 24: 测试 jdbc 连接

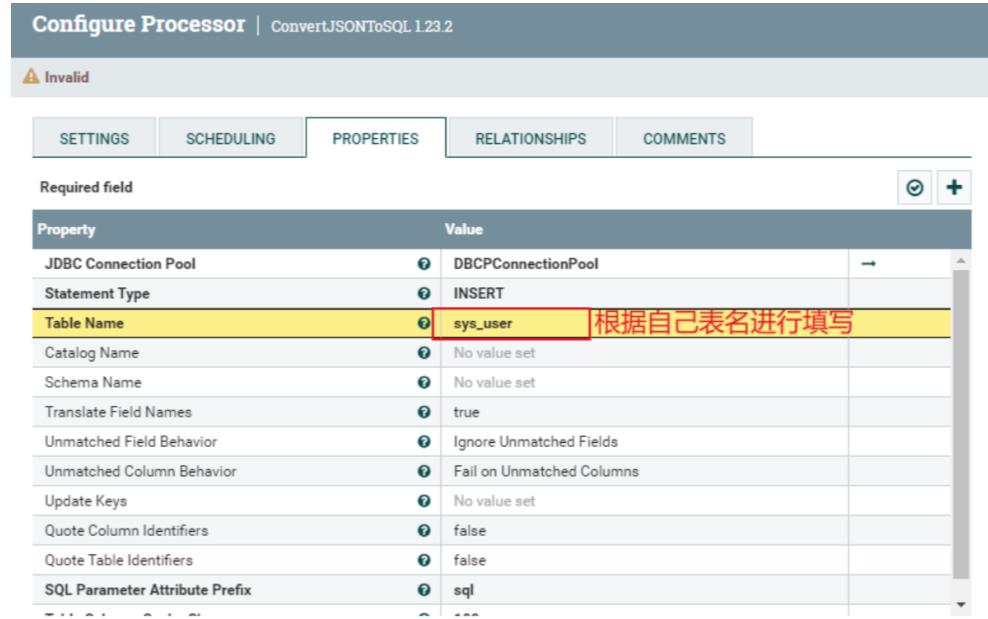


图 25: 配置 Table Name 和 Statement Type

### 连接处理器

将 SplitJson 处理器和 ConvertJSONToSQL 处理器进行连接， Relationships 选择 split



图 26: Relationships 选择 split

执行生成的SQL  
添加处理器：PutSQL

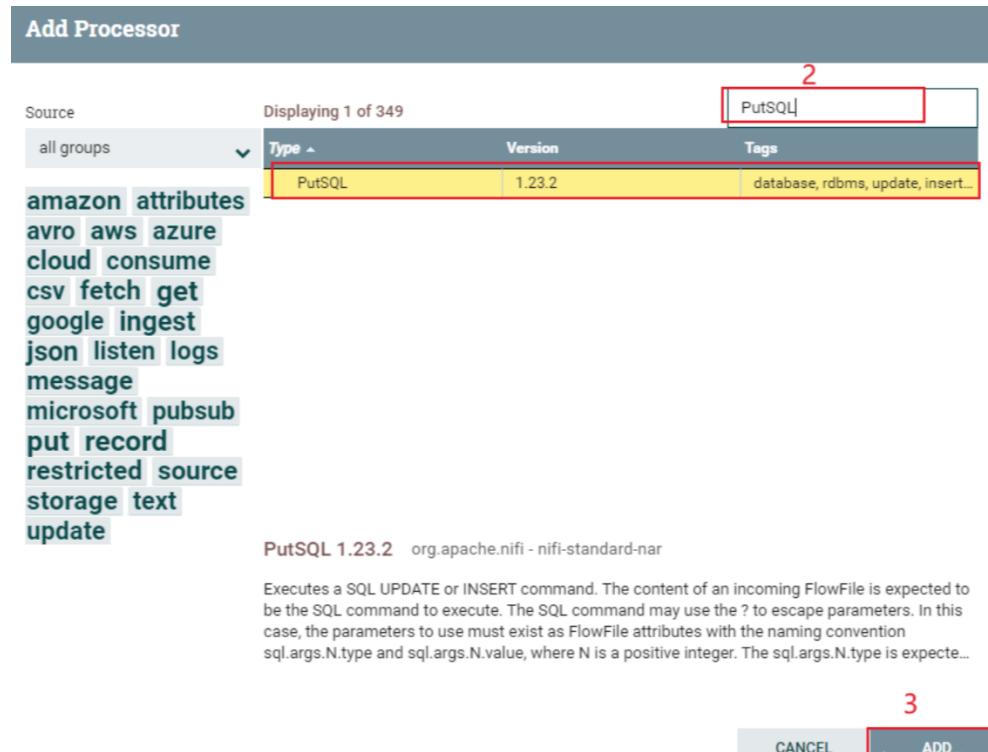


图 27: 添加处理器: PutSQL

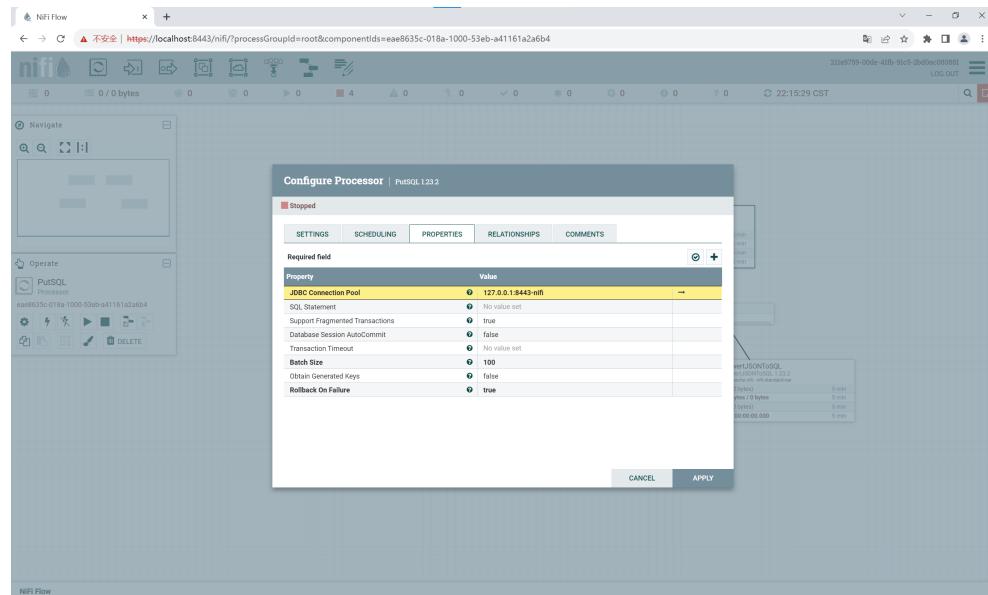


图 28: 配置 putsql

### 连接处理器

将ConvertJSONToSQL 处理器和PutSQL 处理器进行连接， Relationships 选择 sql



图 29: Relationships 选择 sql

## 5. 实验结果

可以很清晰的看到当运行这个数据流时，数据库的记录每 10 秒添加一次记录，这些添加的内容恰好就是 `data.json` 的内容。

```
mysql> select * from sys_user;
+----+-----+-----+-----+-----+-----+
| id | name | age | gender | create_time | is_deleted |
+----+-----+-----+-----+-----+-----+
| 1 | 张三 | 23 | 1 | 2023-10-01 19:26:44 | 0 |
| 2 | 李四 | 24 | 1 | 2023-10-01 19:26:44 | 0 |
| 3 | 小红 | 18 | 0 | 2023-10-01 19:26:44 | 0 |
| 4 | 张三 | 23 | 1 | 2023-10-01 19:26:54 | 0 |
| 5 | 李四 | 24 | 1 | 2023-10-01 19:26:54 | 0 |
| 6 | 小红 | 18 | 0 | 2023-10-01 19:26:54 | 0 |
| 7 | 张三 | 23 | 1 | 2023-10-01 19:27:04 | 0 |
| 8 | 李四 | 24 | 1 | 2023-10-01 19:27:04 | 0 |
| 9 | 小红 | 18 | 0 | 2023-10-01 19:27:04 | 0 |
| 10 | 张三 | 23 | 1 | 2023-10-01 19:27:14 | 0 |
| 11 | 李四 | 24 | 1 | 2023-10-01 19:27:14 | 0 |
| 12 | 小红 | 18 | 0 | 2023-10-01 19:27:14 | 0 |
| 13 | 张三 | 23 | 1 | 2023-10-01 19:27:24 | 0 |
| 14 | 李四 | 24 | 1 | 2023-10-01 19:27:24 | 0 |
| 15 | 小红 | 18 | 0 | 2023-10-01 19:27:24 | 0 |
| 16 | 张三 | 23 | 1 | 2023-10-01 19:27:34 | 0 |
| 17 | 李四 | 24 | 1 | 2023-10-01 19:27:34 | 0 |
| 18 | 小红 | 18 | 0 | 2023-10-01 19:27:34 | 0 |
| 19 | 张三 | 23 | 1 | 2023-10-01 19:27:44 | 0 |
| 20 | 李四 | 24 | 1 | 2023-10-01 19:27:44 | 0 |
| 21 | 小红 | 18 | 0 | 2023-10-01 19:27:44 | 0 |
| 22 | 张三 | 23 | 1 | 2023-10-01 19:27:54 | 0 |
| 23 | 李四 | 24 | 1 | 2023-10-01 19:27:54 | 0 |
| 24 | 小红 | 18 | 0 | 2023-10-01 19:27:54 | 0 |
| 25 | 张三 | 23 | 1 | 2023-10-01 19:28:04 | 0 |
| 26 | 李四 | 24 | 1 | 2023-10-01 19:28:04 | 0 |
| 27 | 小红 | 18 | 0 | 2023-10-01 19:28:04 | 0 |
| 28 | 张三 | 23 | 1 | 2023-10-01 19:28:14 | 0 |
| 29 | 李四 | 24 | 1 | 2023-10-01 19:28:14 | 0 |
| 30 | 小红 | 18 | 0 | 2023-10-01 19:28:14 | 0 |
| 31 | 张三 | 23 | 1 | 2023-10-01 19:28:24 | 0 |
| 32 | 李四 | 24 | 1 | 2023-10-01 19:28:24 | 0 |
| 33 | 小红 | 18 | 0 | 2023-10-01 19:28:24 | 0 |
| 34 | 张三 | 23 | 1 | 2023-10-01 19:28:34 | 0 |
| 35 | 李四 | 24 | 1 | 2023-10-01 19:28:34 | 0 |
| 36 | 小红 | 18 | 0 | 2023-10-01 19:28:34 | 0 |
+----+-----+-----+-----+-----+-----+
36 rows in set (0.00 sec)

mysql>
```

图 30: sql 数据库中的 sys-user

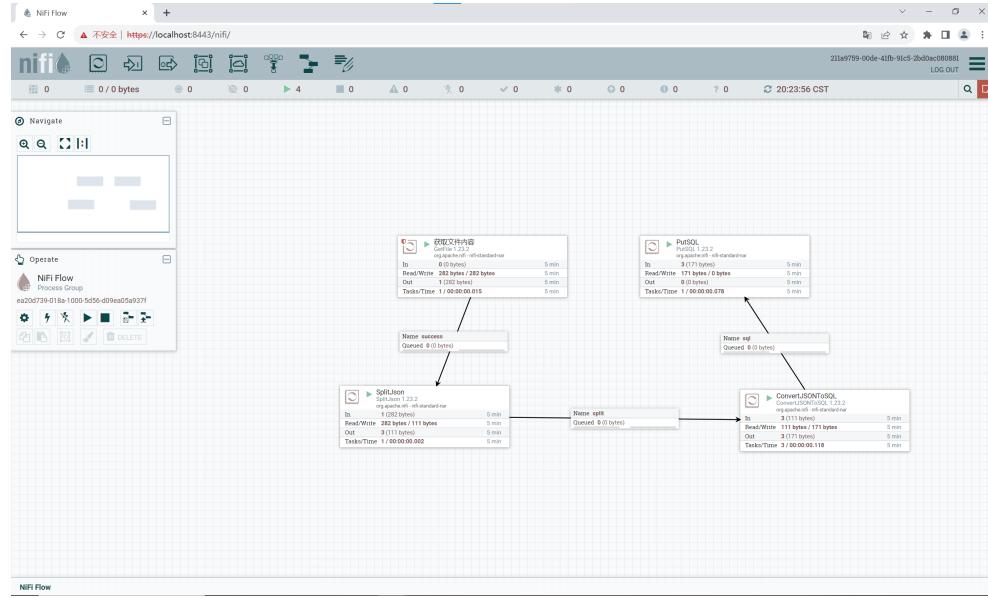


图 31: 最终配置结果

### 3 任务四使用 Flume 收集日志

#### 1. 安装和配置 flume

##### 1. 使用 wget 命令 从清华源下载安装包

```
wget https://mirrors.tuna.tsinghua.edu.cn/apache/flume/1.11.0/apache-flume-
```

##### 2. 解压安装包到 /opt 目录下

```
tar -zxf apache-flume-1.11.0-bin.tar.gz -C /opt/
```

##### 3. 重命名 flume

```
mv apache-flume-1.11.0-bin flume
```

##### 4. 进入 /opt/flume/conf 目录

```
cd /opt/flume/conf
```

##### 5 先复制一份 flume-env.sh.template 文件

```
cp flume-env.sh.template flume-env.sh
```

```
6 修改 flume-env.sh 的 Java_home
```

```
vim flume-env.sh
```

```
export JAVA_HOME=/usr/java8
```

```
[root@izf8z1ijlr1dl2avdfihpzZ ~]# flume-ng version
Flume 1.11.0
Source code repository: https://git.apache.org/repos/asf/flume.git
Revision: 1a15927e594fd0d05a59d804b90a9c31ec93f5e1
Compiled by rgoers on Sun Oct 16 14:44:15 MST 2022
From source with checksum bbbca682177262aac3a89defde369a37
[root@izf8z1ijlr1dl2avdfihpzZ ~]# █
```

图 32: 验证 flume 安装成功

## 2. 配置 Flume 以收集日志并将其存储在 HDFS 中

flume 提供了 Taildir Source，可实时监控一批文件，并记录每个文件最新消费位  
Flume 核心配置

tailDirSource 配置

```
a1.sources = r1
a1.sources.r1.type = TAILDIR
a1.sources.r1.channels = c1
a1.sources.r1.positionFile = /export/servers/flume/taildir_position.json
a1.sources.r1.filegroups = f1 f2
a1.sources.r1.filegroups.f1 = /export/data/test1/example.log
a1.sources.r1.filegroups.f2 = /export/data/test2/*log.*
```

##配置说明

```
filegroups
```

指定 filegroups，可以有多个，以空格分隔；（TailSource 可以同时监控 tail 多个 positionFile

配置检查点文件的路径，检查点文件会以 json 格式保存已经 tail 文件的位置，解决

```
filegroups.
```

配置每个 filegroup 的文件绝对路径，文件名可以用正则表达式匹配  
sink 配置

本次将日志采集到HDFS中，需要使用 HDFSSink 文件。HDFSSink 需要配置滚动属性。

基于 hdfs 文件副本数

配置项： hdfs . minBlockReplicas

默认值： 和 hdfs 的副本数一致

说明

hdfs . minBlockReplicas 是为了让 flume 感知不到 hdfs 的块复制，这样滚动方式配置示例说明：

假如 hdfs 的副本为 3，配置的滚动时间为 10 秒，那么在第二秒的时候，flume 检测到

---

完整版配置文件

```
vim /opt/flume/conf/log2hdfs.conf

# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = TAILDIR
a1.sources.r1.positionFile = /opt/flume/taildir_position.json
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /export/data/test1/example.log

# Describe the sink
#指定 hdfs sink
a1.sinks.k1.type = hdfs
#hdfs 目录，带有时间信息
a1.sinks.k1.hdfs.path = /flume/tailout/%Y-%m-%d/
```

```

#生成的 hdfs 文件名的前缀
a1.sinks.k1.hdfs.filePrefix = events-
#指定滚动时间， 默认是30秒， 设置为0表示禁用该策略
a1.sinks.k1.hdfs.rollInterval = 0
#指定滚动大小， 设置为0表示禁用该策略
a1.sinks.k1.hdfs.rollSize = 200000000
#指定滚动条数
a1.sinks.k1.hdfs.rollCount = 0
a1.sinks.k1.hdfs.batchSize = 100
a1.sinks.k1.hdfs.useLocalTimeStamp = true
#副本策略
a1.sinks.k1.hdfs.minBlockReplicas=1
#生成的文件类型， 默认是 Sequencefile， 可用 DataStream，则为普通文本
a1.sinks.k1.hdfs.fileType = DataStream

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1

```

启动 flume agent 采集数据

创建目录

```
mkdir -p /opt/data/test1/
```

进入 test1 中创建 log 文件

```
touch example.log
```

flume 启动命令

```
flume-ng agent --conf-file conf/log2hdfs.conf --name a1 -Dflume.root.logg
```

```
>_ 2. root@iZf8z1ijl1dl2avdfihpzZ:/opt/flume      >_ 3. root@iZf8z1ijl1dl2avdfihpzZ:~          >_ 4. root@iZf8z1ijl1dl2avdfihpzZ:/opt/flume/conf ×

# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = TAILDIR
a1.sources.r1.positionFile = /opt/flume/taildir_position.json
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /opt/flume/data/example.log

# Describe the sink
#指定hdfs sink
a1.sinks.k1.type = hdfs
#hdfs目录，带有时间信息
a1.sinks.k1.hdfs.path = /flume/tailout/%Y-%m-%d/
#生成的hdfs文件名的前缀
a1.sinks.k1.hdfs.filePrefix = events-
#指定滚动时间，默认是30秒，设置为0表示禁用该策略
a1.sinks.k1.hdfs.rollInterval = 0
#指定滚动大小，设置为0表示禁用该策略
a1.sinks.k1.hdfs.rollSize = 200000000
#指定滚动条数
a1.sinks.k1.hdfs.rollCount = 0
a1.sinks.k1.hdfs.batchSize = 100
a1.sinks.k1.hdfs.useLocalTimeStamp = true
#副本策略
a1.sinks.k1.hdfs.minBlockReplicas=1
#生成的文件类型，默认是Sequencefile，可用DataStream，则为普通文本
a1.sinks.k1.hdfs fileType = DataStream

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1

~
~
~
~
~
~
~
~

"log2hdfs.conf" 39L, 1223C
```

图 33: log2hdfs.conf 的内容

```
[root@iZj61ljlrd2wdfhpz hadoop]# jps
92096 SecondaryNameNode
93175 ResourceManager
93320 NodeManager
93485 DataNode
93948 TaskTracker
93422 NodeManager
[root@iZj61ljlrd2wdfhpz hadoop]# cd ..;/flume/
[root@iZj61ljlrd2wdfhpz flume]# flume-ng agent --conf-file conf/log2hdfs.conf --name a -Dflume.root.logger=INFO,console
Warning: No configuration directory set use --conf-dir> to override.
Info: Including hadoop libraries found via $HADOOP_HOME/bin/hadoop for HDFS access
Info: Using configuration from conf/log2hdfs.conf
[...]
e exec /usr/java/jdk1.8.0_201/bin/java -Xmx200m -Djava.util.logging.level=INFO -cp "/opt/flume/lib/*:/opt/hadoop/etc/hadoop:/opt/hadoop/share/hadoop/common/lib/*:/opt/hadoop/share/hadoop/common/*:/opt/hadoop/share/hadoop/hdfs/*:/opt/hadoop/share/hadoop/hdfs/lib/*:/opt/hadoop/share/hadoop/mapreduce/*:/opt/hadoop/share/hadoop/mapreduce/lib/*:/opt/hadoop/share/hadoop/mapreduce/*:/opt/hadoop/contrib/lib/capacity-scheduler.jar" org.apache.flume.node.Application --conf-file conf/log2hdfs.conf --name a1
SLF4J: Found binding in [jar:file:/opt/flume/lib/log4j-1.2.17.jar!/org.slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/flume/lib/hadoop-common-2.1.0.jar!/org.slf4j/impl/StaticLoggerBinder.class]
SLF4J: Actual binding is of type [org.apache.logging.log4j.Log4jLoggerFactory]
```

图 34: flume flume-*ng* agent 命令

## 1. 新建一个 master 节点

打开一个新的命令行终端

```
cd /opt/flume/data/
```

重复输入以下命令

”hello BIG DATA” >> example.log

```
"hello world" >> example.log
```

Welcome to Alibaba Cloud Elastic Compute Service !

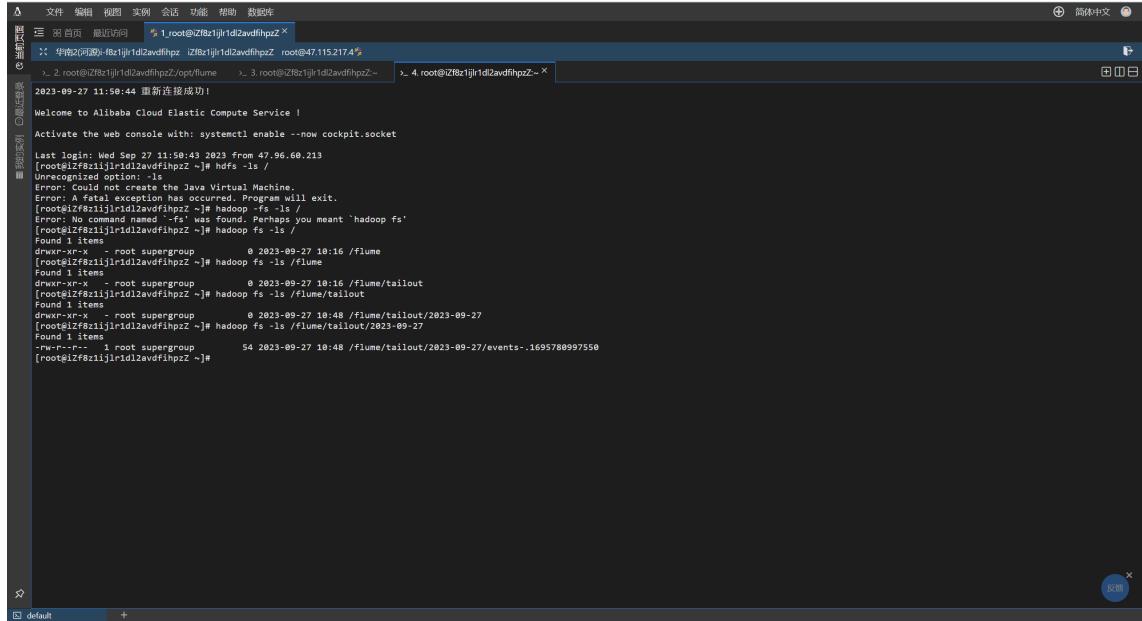
Activate the web console with: systemctl enable --now cockpit.socket

```
Last login: Wed Sep 27 18:15:14 2023 from 118.31.243.222
[root@iZfr8z1j1r1d12avdfhpz ~]# cd /opt/flume/
[root@iZfr8z1j1r1d12avdfhpz ~]# ls
bin  Checkpoint  conf  doc  lib  LICENSE  NOTICE  README.md  RELEASE-NOTES  taldir_position.json  tools
[root@iZfr8z1j1r1d12avdfhpz ~]# cat data/
[root@iZfr8z1j1r1d12avdfhpz ~]# ls
example.log
[root@iZfr8z1j1r1d12avdfhpz ~]# echo "Hello world!!!" > example.log
[root@iZfr8z1j1r1d12avdfhpz ~]# cat example.log
[root@iZfr8z1j1r1d12avdfhpz ~]# cat example.log
Hello world
Hello world
Hello world!
Hello world!
Hello world!
[root@iZfr8z1j1r1d12avdfhpz ~]# cat example.log
Hello world
Hello world
Hello world
Hello world!
Hello world!
[root@iZfr8z1j1r1d12avdfhpz ~]# echo "Hello BIG DATA" > example.log
[root@iZfr8z1j1r1d12avdfhpz ~]# cat example.log
Hello world!
Hello world!
Hello BIG DATA
[root@iZfr8z1j1r1d12avdfhpz ~]# echo "Hello world" > example.log
[root@iZfr8z1j1r1d12avdfhpz ~]# echo "Hello world" > example.log
[root@iZfr8z1j1r1d12avdfhpz ~]# cat example.log
Hello world
[root@iZfr8z1j1r1d12avdfhpz ~]#
```

图 35: flume 向 example 中输入

## 查看HDFS中保存的日志文件

```
hadoop fs -ls /flume/tailout/2023-09-27
```



```
2023-09-27 11:59:44 重新连接成功!
Welcome to Alibaba Cloud Elastic Compute Service !
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Sep 27 11:50:43 2023 from 47.96.60.213
[root@iZf8z1i1j1r1d2avdfihpz ~]# hadoop -ls
Unrecognized option: 'ls'
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred.  See the log file for details.
[root@iZf8z1i1j1r1d2avdfihpz ~]# hadoop -ls /
Error: No command named `fs` was found. Perhaps you meant `hadoop fs`?
[root@iZf8z1i1j1r1d2avdfihpz ~]# hadoop fs -ls /
Found 1 items
drwxr-xr-x - root supergroup          0 2023-09-27 10:16 /flume
[root@iZf8z1i1j1r1d2avdfihpz ~]# hadoop fs -ls /flume/tailout
Found 1 items
drwxr-xr-x - root supergroup          0 2023-09-27 10:16 /flume/tailout
[root@iZf8z1i1j1r1d2avdfihpz ~]# hadoop fs -ls /flume/tailout/2023-09-27
Found 1 items
drwxr-xr-x - root supergroup          0 2023-09-27 10:48 /flume/tailout/2023-09-27
[root@iZf8z1i1j1r1d2avdfihpz ~]# hadoop fs -ls /flume/tailout/2023-09-27
Found 1 items
drwxr-xr-x - root supergroup          54 2023-09-27 10:48 /flume/tailout/2023-09-27/events--1695780997550
```

图 36: flume 查看 hdfs 中的日志文件

### 3. 验证：确认 HDFS 中存储的日志

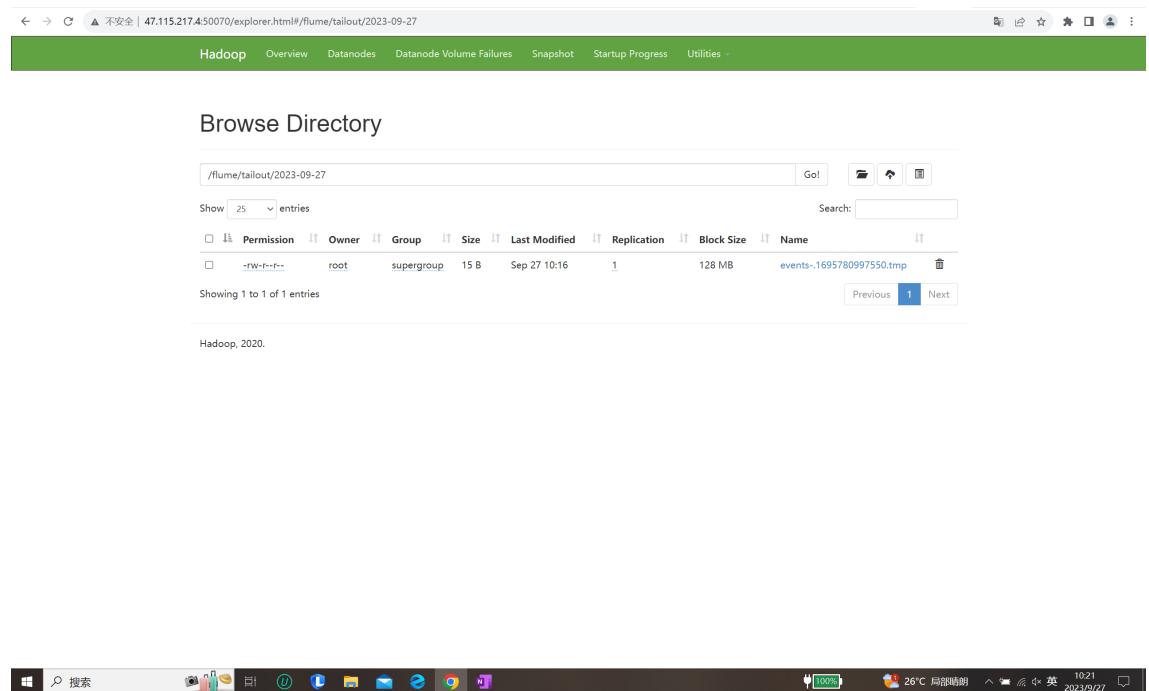


图 37: flume 查看 hdfs 中的日志文件

## 4 过程中发现的困难及如何克服

启动 hadoop 的时候: hadoop namenode -format 命令又启动了一次, 结果 namenode 和 datanode 不能同时开启, 上网查了很多资料, 最后删除了 hadoop 目录下的 temp, 重新运行 hadoop namenode -format 就好了.