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
```

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
```

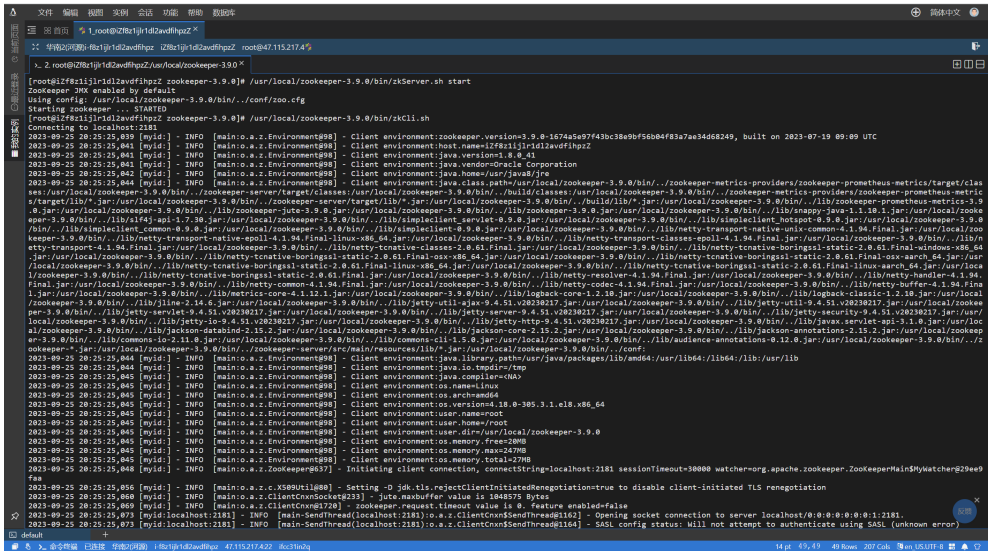

图 1: zookeeper 启动

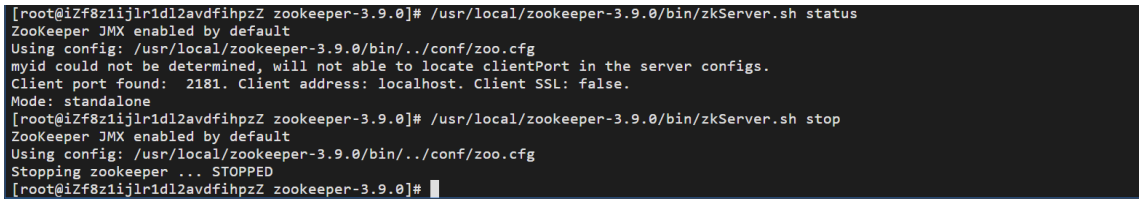

图 2: zookeeper status 命令

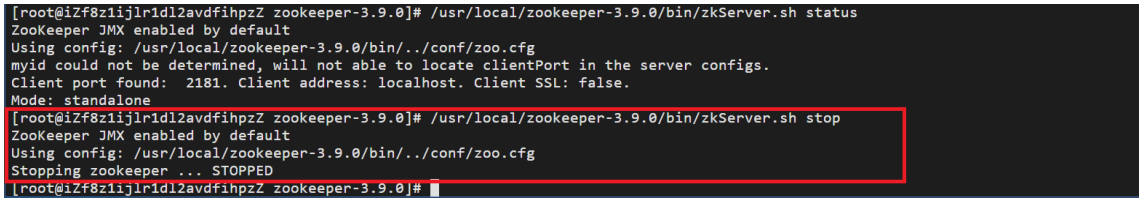

图 3: zookeeper stop 命令

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.pro
```

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@iZf8z1ijlr1dl2avdfihpzZ ~]# /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@iZf8z1ijlr1dl2avdfihpzZ ~]# /usr/local/kafka2.12/bin/kafka-server-start.sh -daemon config/server.properties
[root@iZf8z1ijlr1dl2avdfihpzZ ~]#
```

图 4: kafka version 命令


```
[root@iZf8z1ijlr1dl2avdfihpzZ ~]# /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@iZf8z1ijlr1dl2avdfihpzZ ~]# /usr/local/kafka2.12/bin/kafka-server-start.sh -daemon config/server.properties
[root@iZf8z1ijlr1dl2avdfihpzZ ~]#
```


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-fa
```

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 testT
>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
>hhh
>hello
>actually you are
```

```
[root@iZf8z1ijlr1dl2avdfihpzZ kafka2.12]# bin/kafka-server-start.sh -daemon config/server.properties
[root@iZf8z1ijlr1dl2avdfihpzZ kafka2.12]# bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic testTopic
Created topic testTopic.
[root@iZf8z1ijlr1dl2avdfihpzZ kafka2.12]# bin/kafka-topics.sh --list --zookeeper localhost:2181
testTopic
[root@iZf8z1ijlr1dl2avdfihpzZ kafka2.12]# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic testTopic
>hhh you are a big shaniao
>what?
>actually you are!
>^C[root@iZf8z1ijlr1dl2avdfihpzZ 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

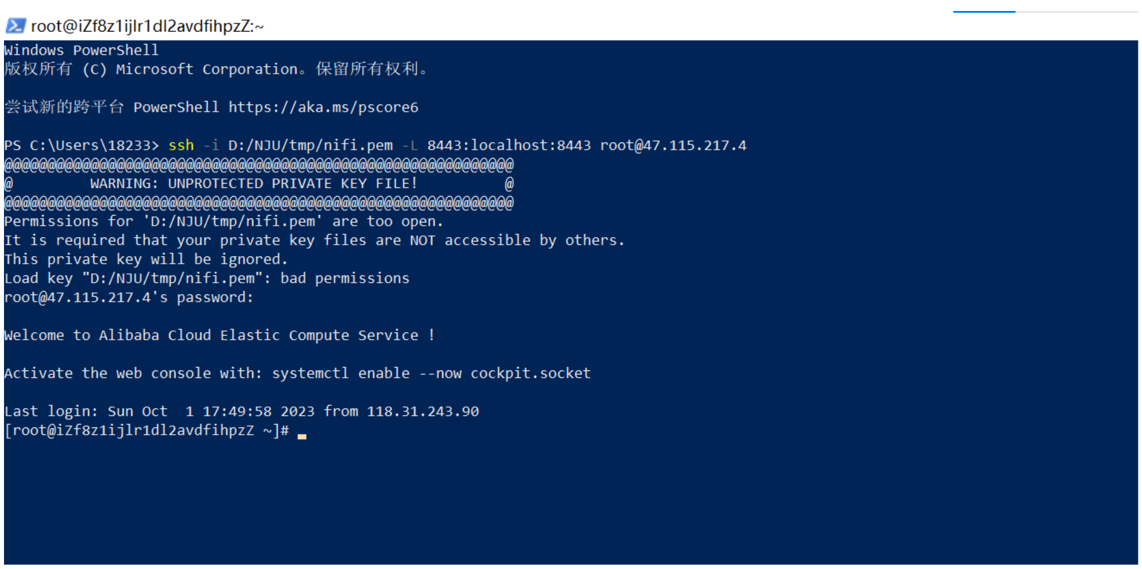

图 8: ssh 端口映射

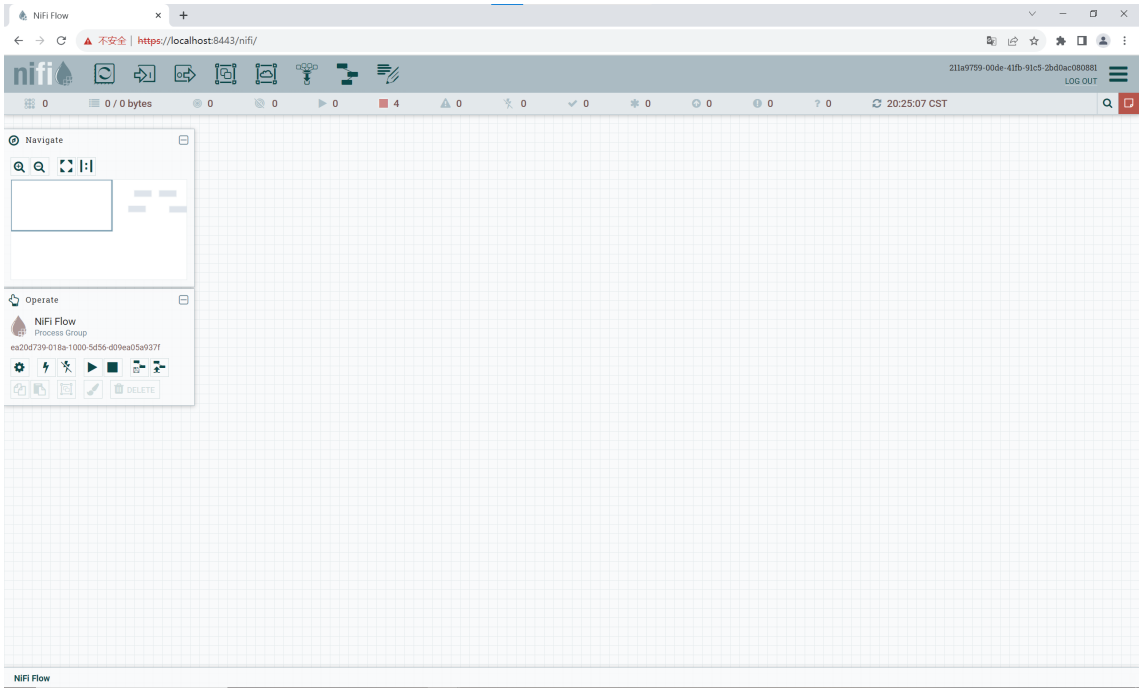

图 9: nifi 的 web 界面

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 ext]# 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@iZf8z1ijlr1dl2avdfihpzZ ~]# cd /usr/java8/
[root@iZf8z1ijlr1dl2avdfihpzZ java8]# ls
ADDITIONAL_LICENSE_INFO  ASSEMBLY_EXCEPTION  bin  demo  include  jre  lib  LICENSE  man  release  sample  src.zip  THIRD_PARTY_README
[root@iZf8z1ijlr1dl2avdfihpzZ java8]# cd jre/
[root@iZf8z1ijlr1dl2avdfihpzZ jre]# ls
ADDITIONAL_LICENSE_INFO  ASSEMBLY_EXCEPTION  bin  lib  LICENSE  THIRD_PARTY_README
[root@iZf8z1ijlr1dl2avdfihpzZ jre]# cd lib/
[root@iZf8z1ijlr1dl2avdfihpzZ lib]# ls
amd64              charsets.jar  content-types.properties  flavormap.properties              jce.jar    jsse.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@iZf8z1ijlr1dl2avdfihpzZ lib]# cd ext/
[root@iZf8z1ijlr1dl2avdfihpzZ ext]# ls
cldrdata.jar  dnsns.jar  localedata.jar  meta-index  mysql-connector-java.jar  nashorn.jar  sunec.jar  sunjce_provider.jar  sunpkcs11.jar  zipfs.jar
[root@iZf8z1ijlr1dl2avdfihpzZ ext]# pwd
/usr/java8/jre/lib/ext
[root@iZf8z1ijlr1dl2avdfihpzZ 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
    }
]
```

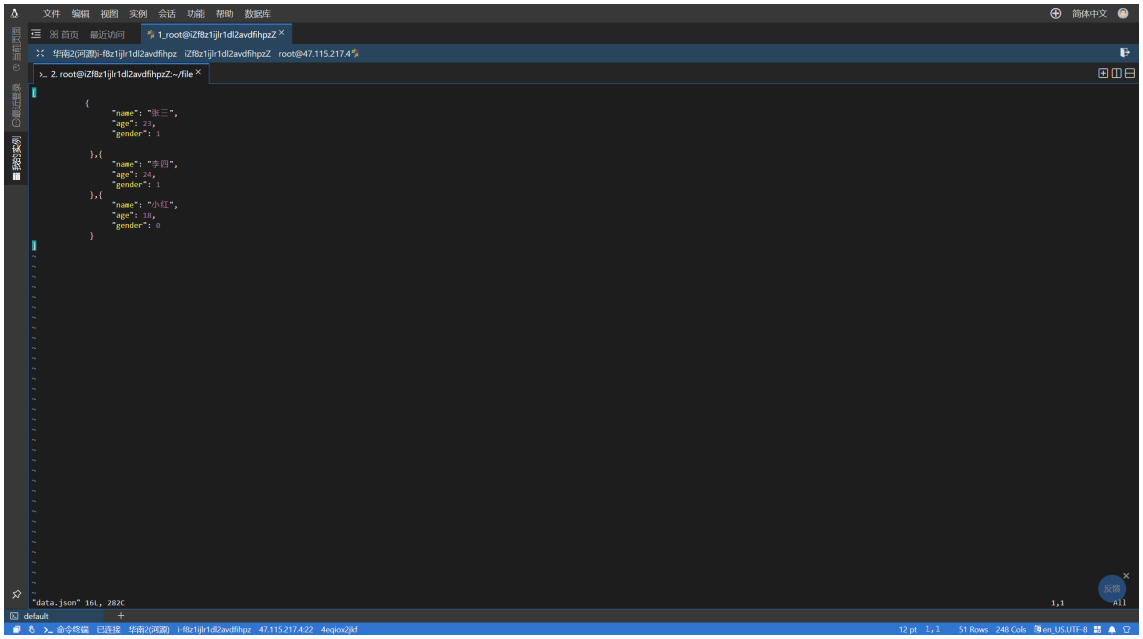

图 13: data.json

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
  `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

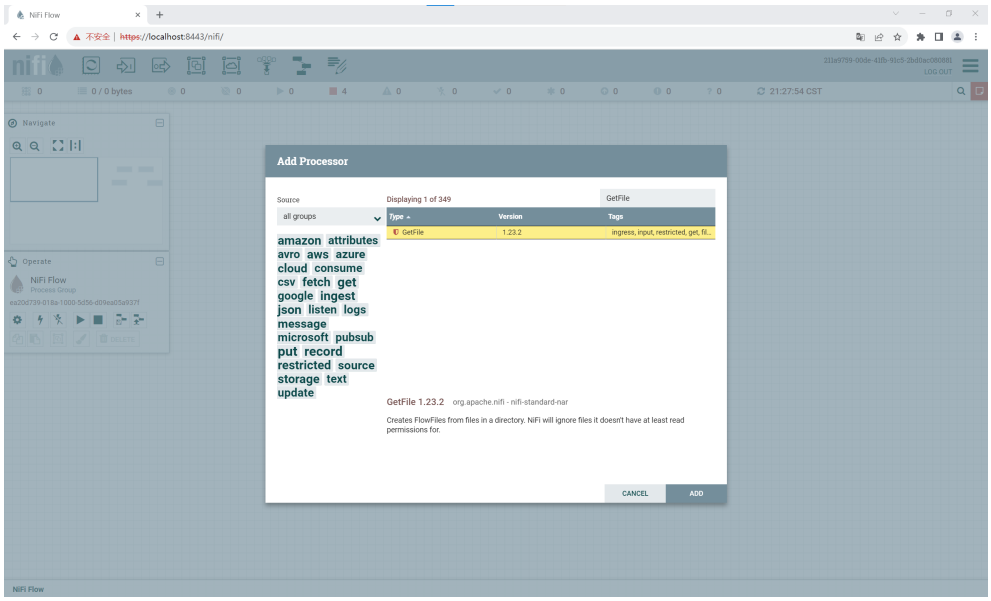

图 14: 筛选 GetFile, 点击 ADD 添加到画布中

配置 GetFile 处理器
双击添加的处理器，弹出对应的配置界面

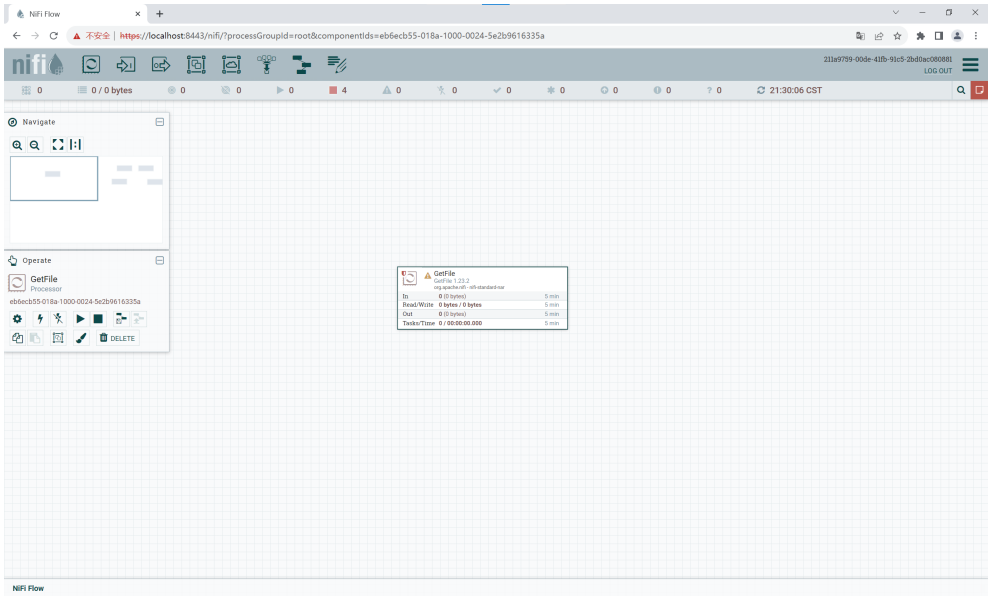

图 15: GetFile

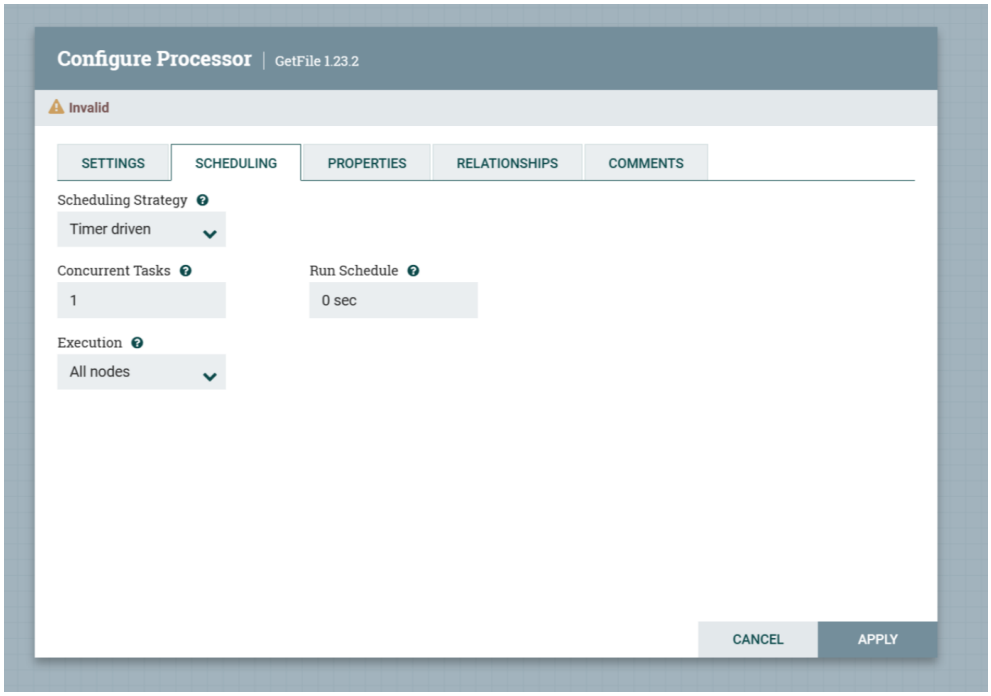

图 16: 配置界面

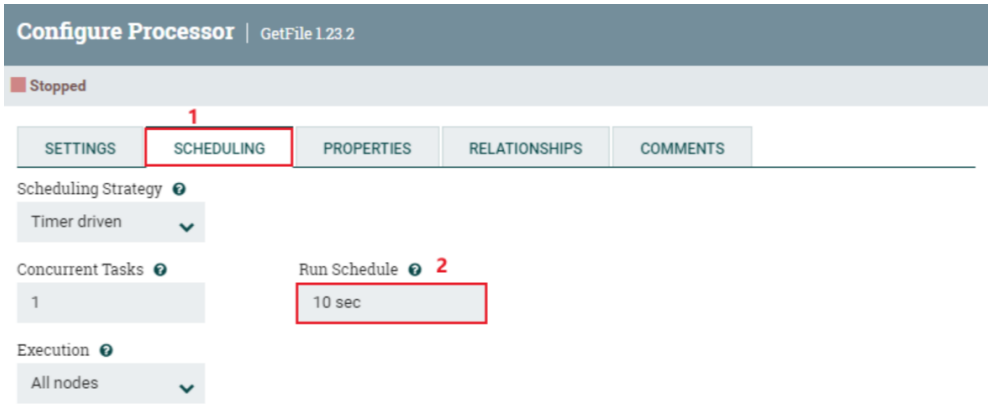

图 17: 设置 10s 定时器

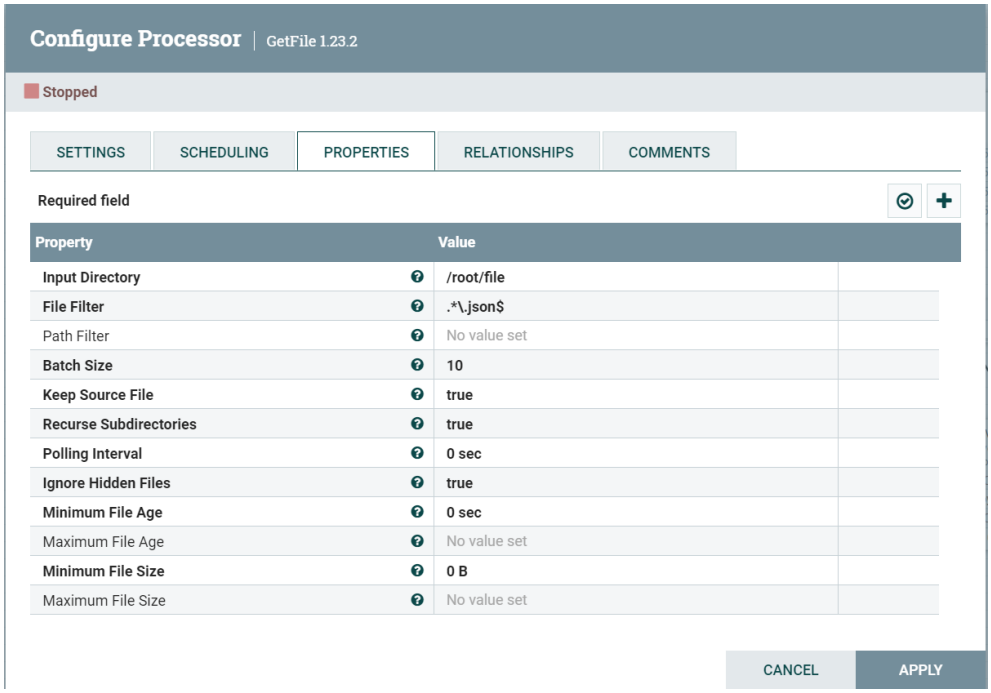

图 18: 设置文件路径

添加和配置处理器： SplitJson

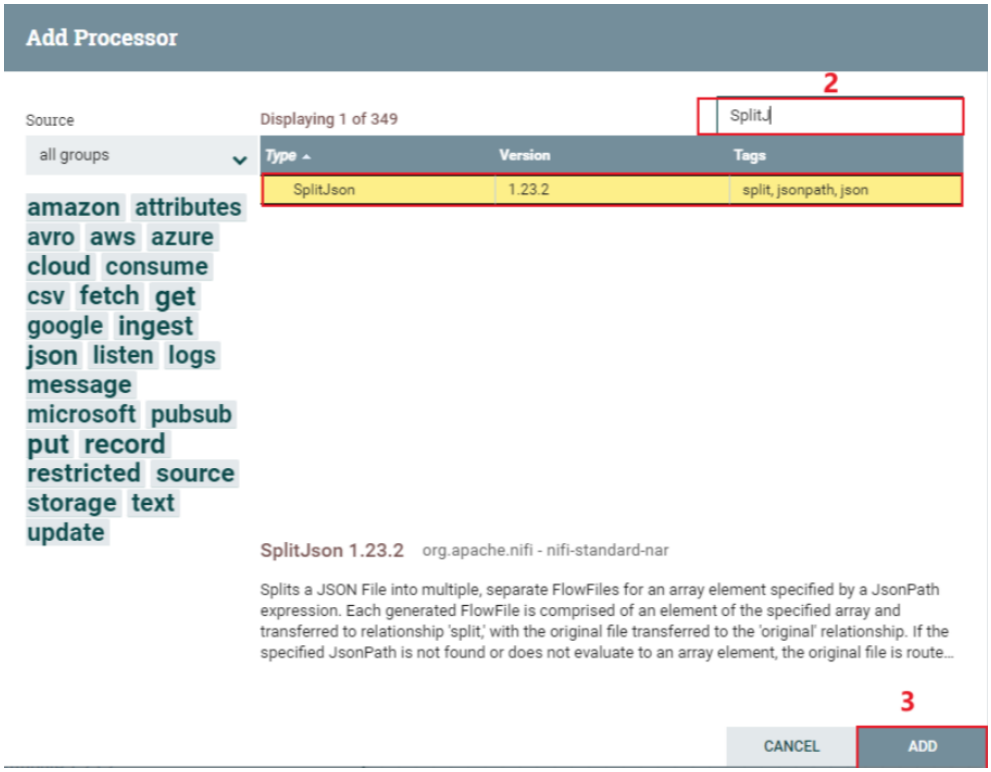

图 19: 添加处理器 SplitJson

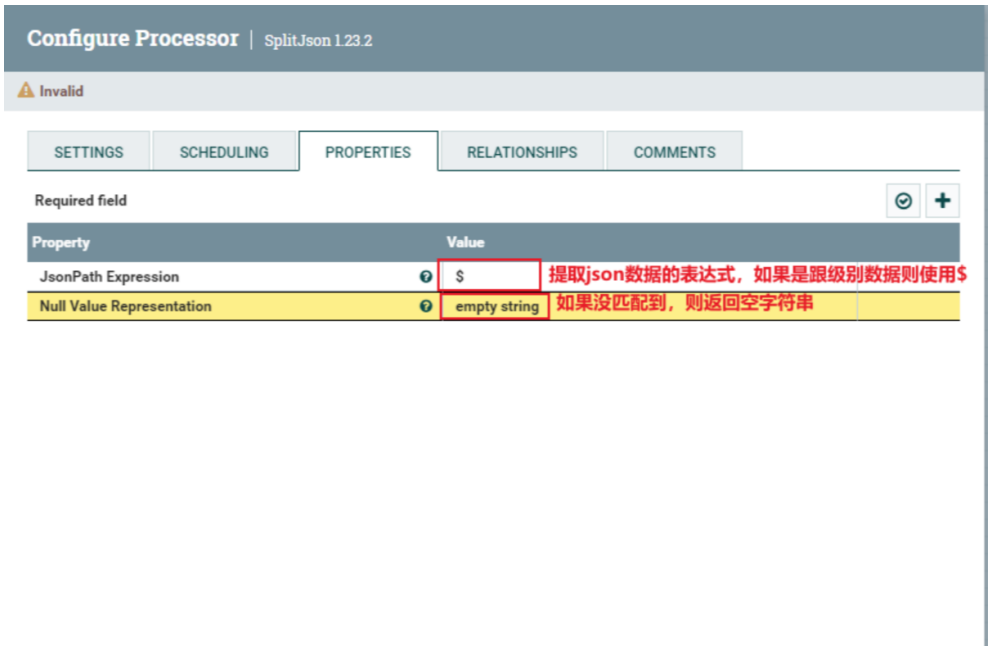

图 20: 配置处理器 SplitJson

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

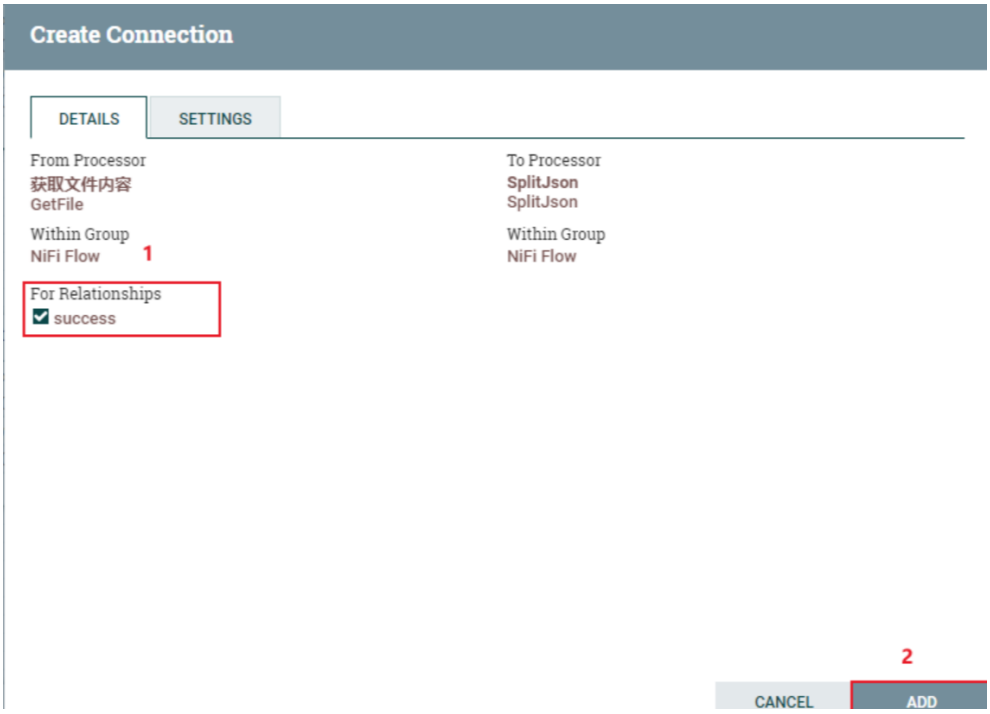

图 21: 将 GetFile 处理器和 SplitJson 处理器连接

Json 转为SQL

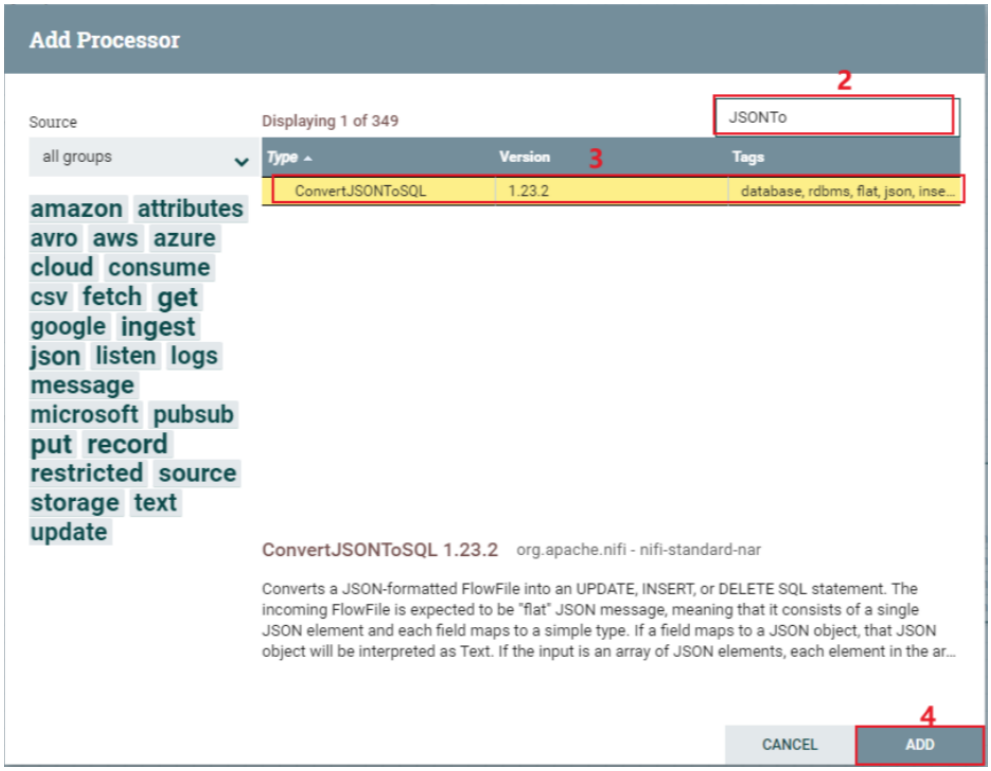

图 22: 添加处理器 ConvertJSONToSQL

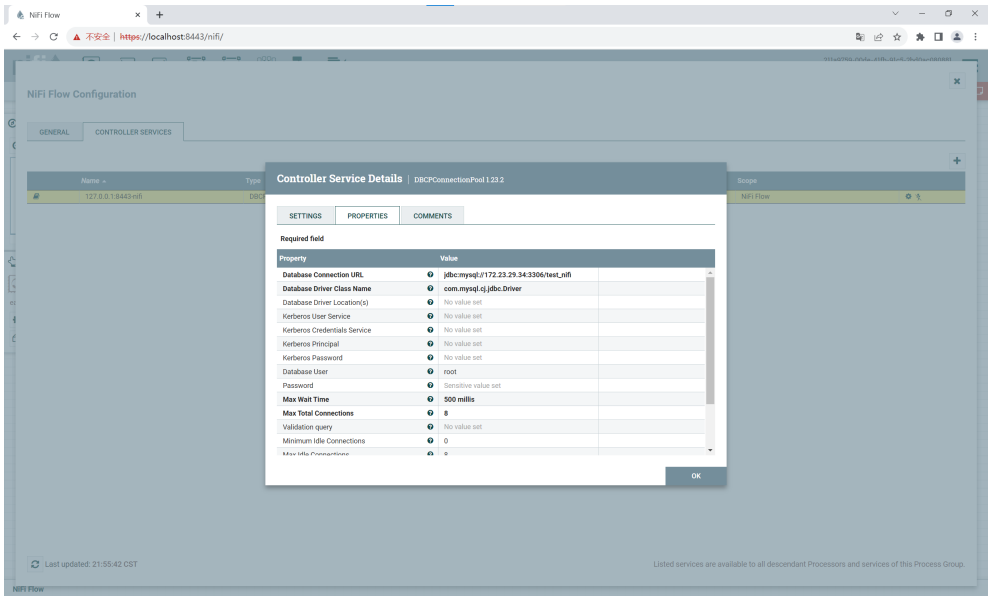

图 23: 配置处理器 ConvertJSONToSQL

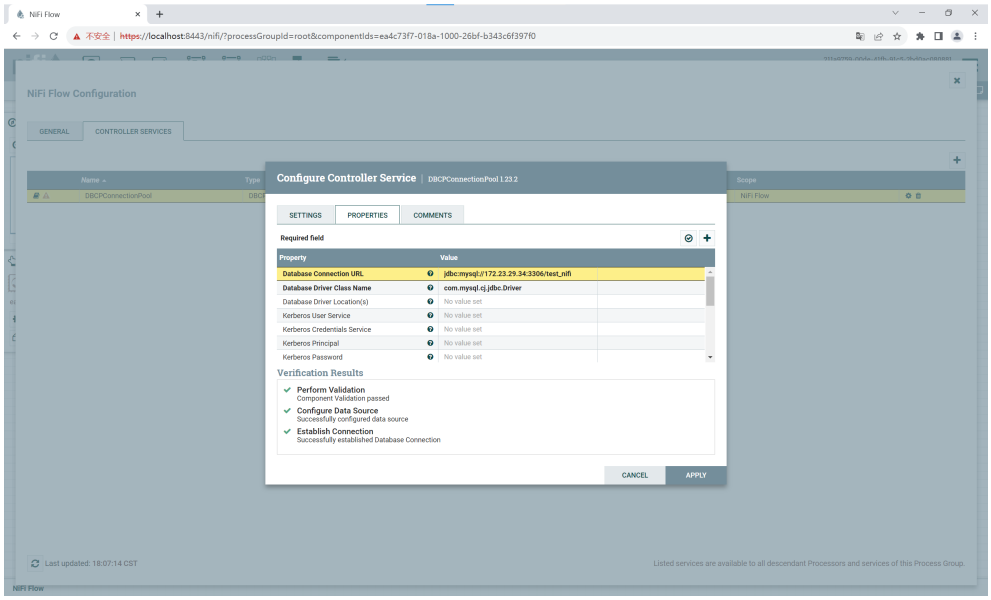

图 24: 测试 jdbc 连接

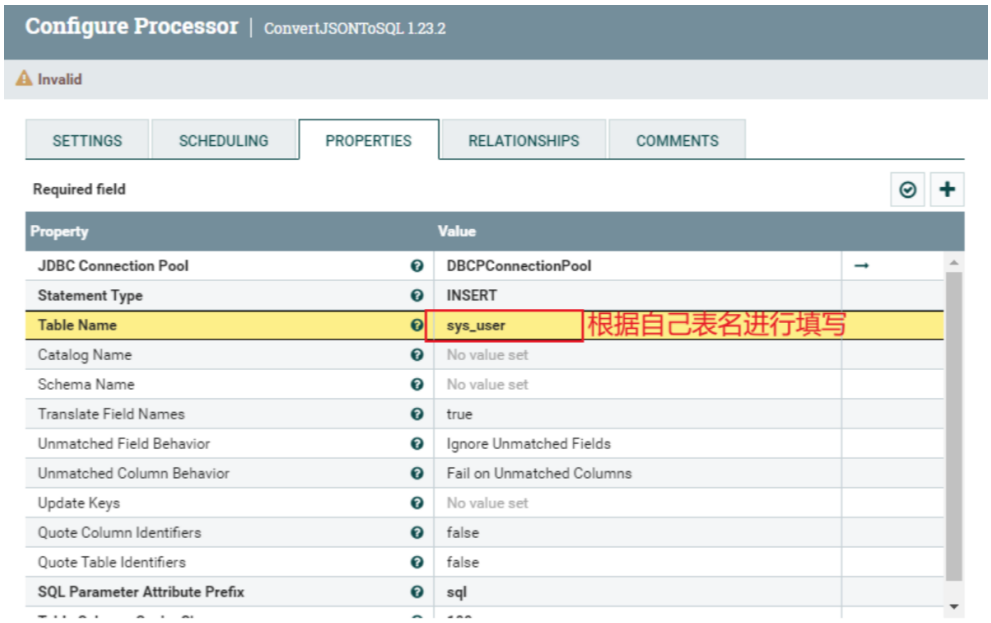

图 25: 配置 Table Name 和 Statement Type

连接处理器

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

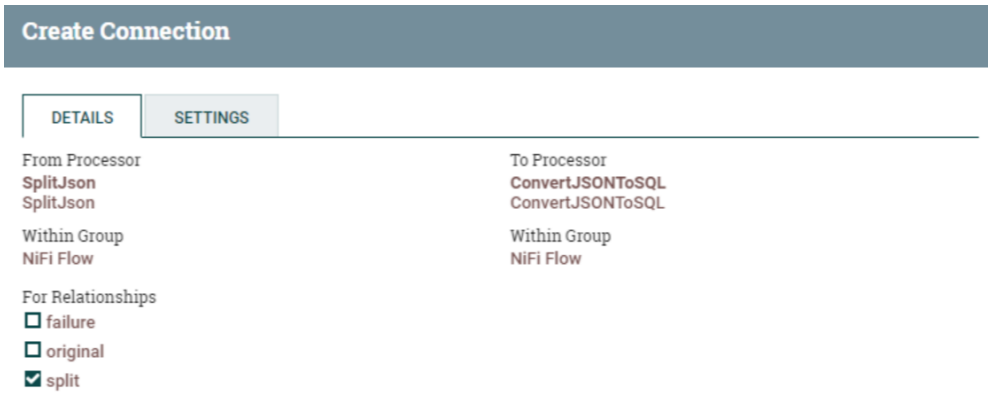

图 26: Relationships 选择 split

执行生成的SQL

添加处理器： PutSQL

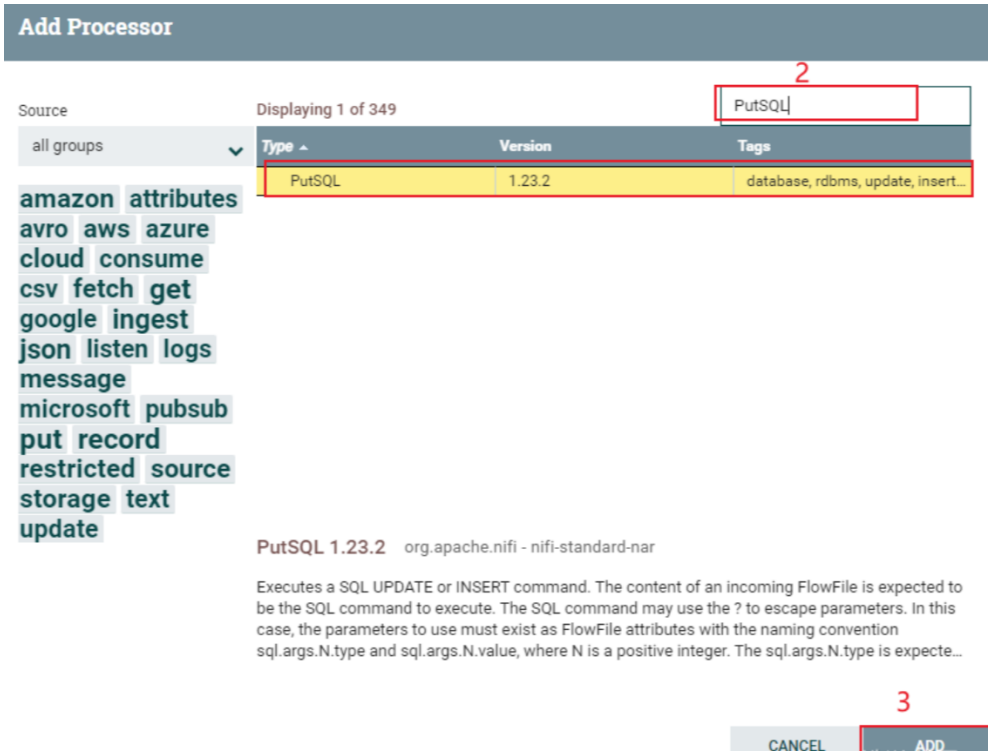

图 27: 添加处理器：PutSQL

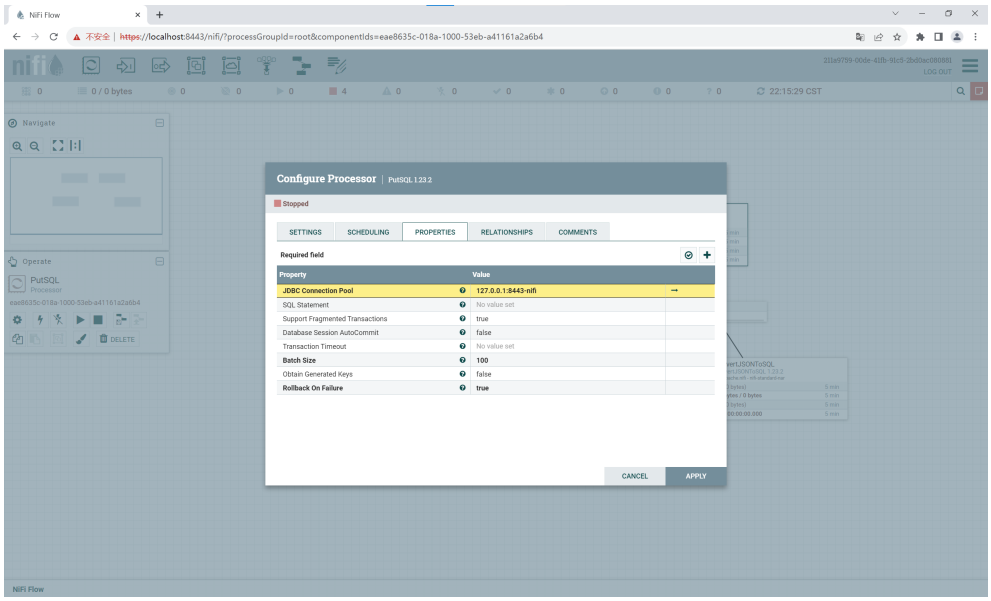

图 28: 配置 putsql

连接处理器

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

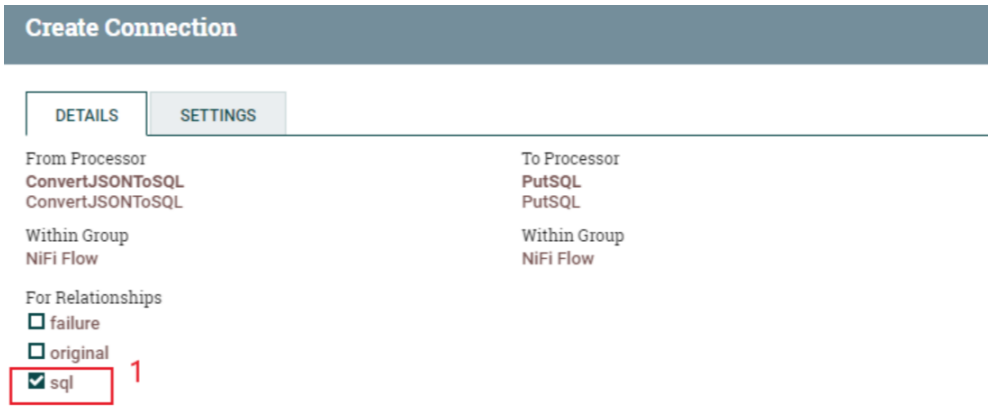

图 29: Relationships 选择 sql

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

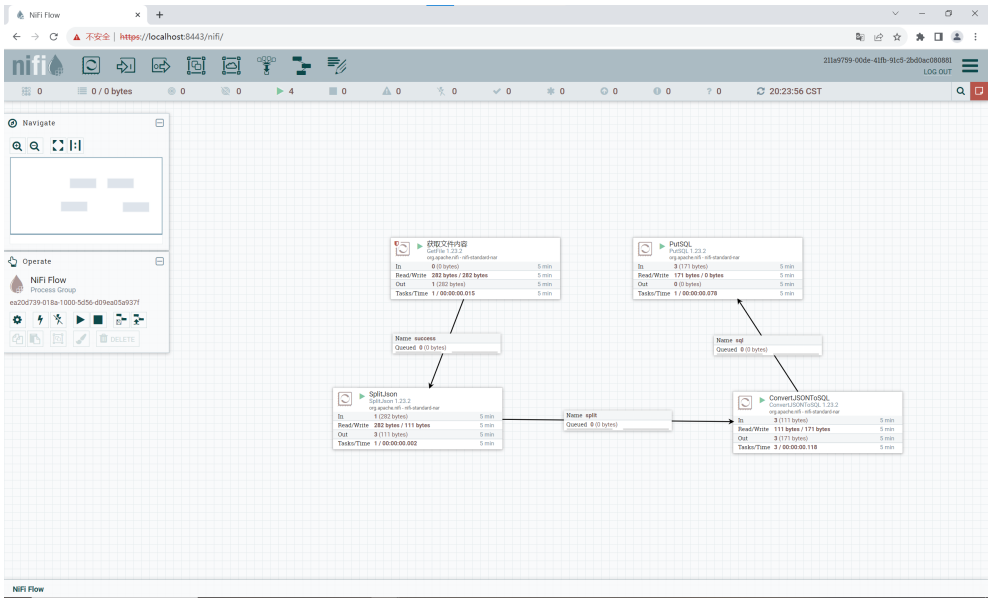

图 31: 最终配置结果

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-flu

   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

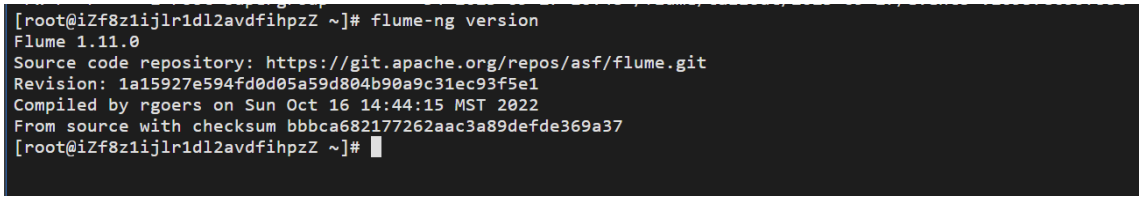

图 32: 验证 flume 安装成功

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
```

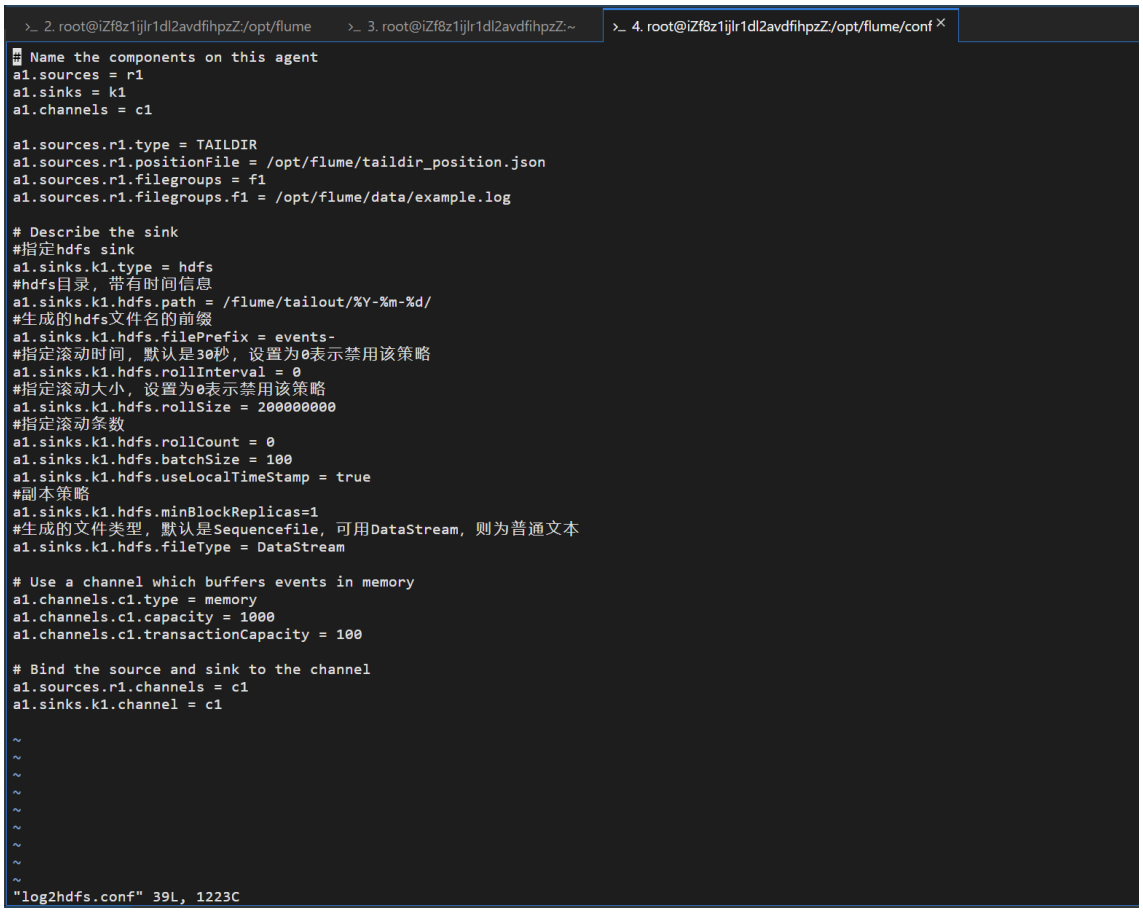

图 33: log2hdfs.conf 的内容

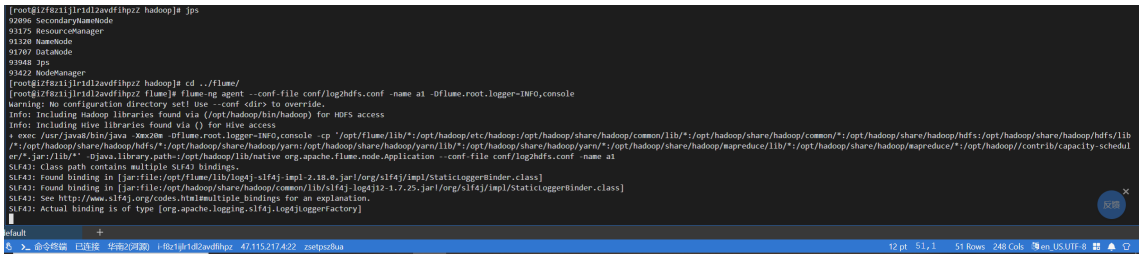

图 34: flume flume-ng agent 命令

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

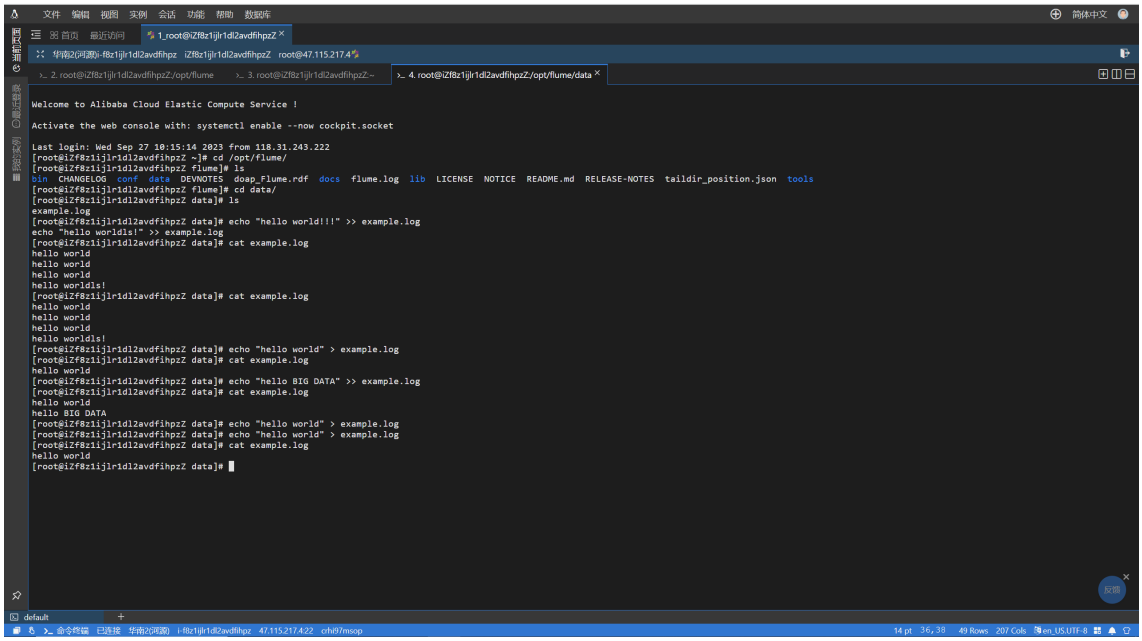

图 35: flume 向 example 中输入

查看HDFS中保存的日志文件

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

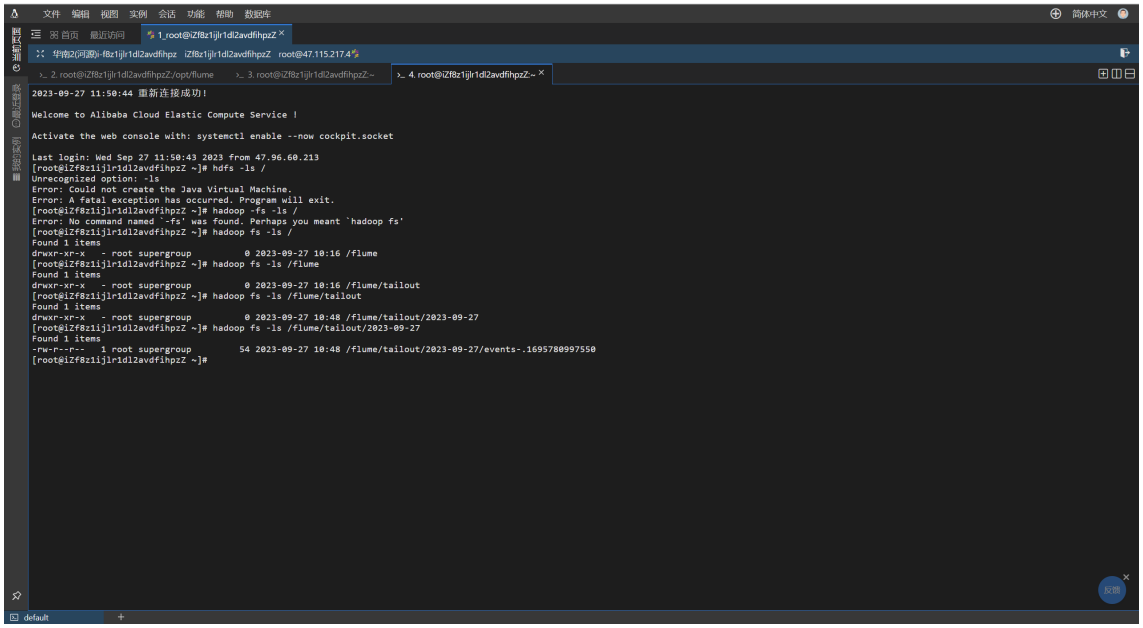

图 36: flume 查看 hdfs 中的日志文件

3. 验证：确认 HDFS 中存储的日志

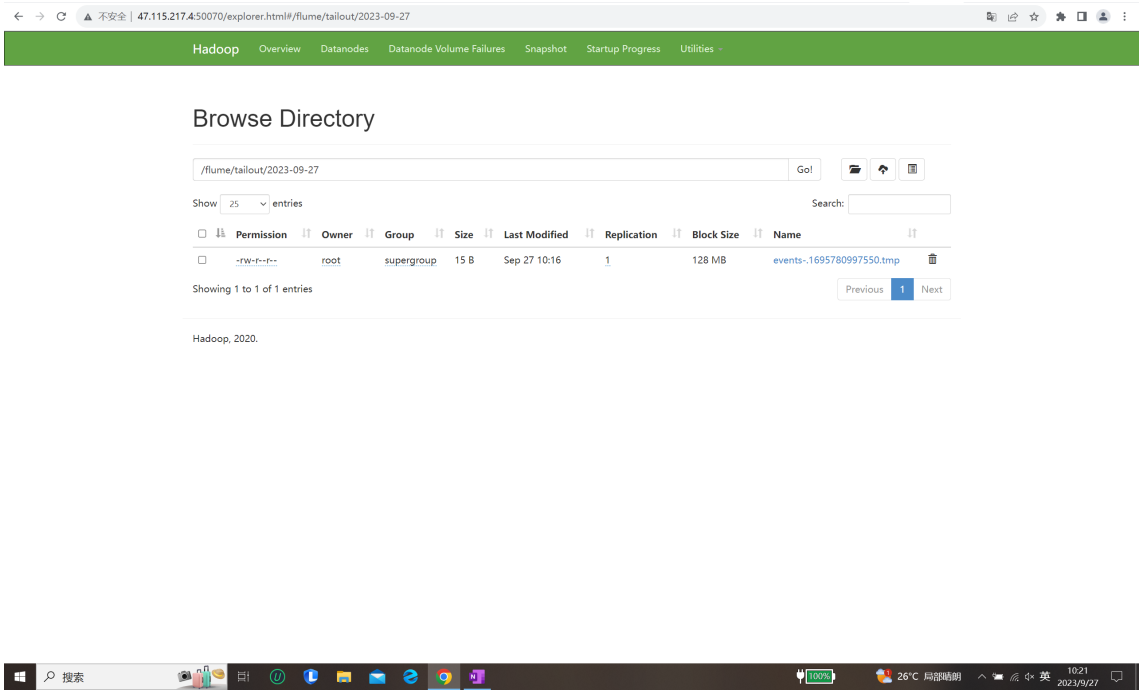

图 37: flume 查看 hdfs 中的日志文件

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.