SALAH Follow
Apr 8 · 13 min read

My name is SALAH and i am a Big-Data engineer working at BIGAPPS

Big Apps is a specialist in performance management consulting and the integration of innovative technological solutions in Big Data.

> *We believe that IT is transforming our societies. We know that milestones are the result of sharing knowledge and the pleasure of working together. We are always looking for the best ways to go.*

READY TO BEGIN EXPERT ON BIG DATA ?

FOLLOW US!

Apache NiFi for DataFlow and Real-Time Streaming with Apache KAFKA

Apache NiFi as Flow based Programming platform.

Summery:

- Introduction to Apache Nifi and Apache Kafka
- What is Data Flow ?
- What is Apache NIFI ?
- What is Apache MINIFI ?
- What can be done with Apache Nifi ?
- Apache Nifi architecture
- How to install Apache Nifi on centos 7 ?
- Build a first processor and data processing
- Example 1

- **Apache NiFi and Apache Kafka together**
- **Apache Nifi as Producer and Consumer Kafka**
- **Example 2**
- **Apache Nifi in real-Time Event Streaming with Kafka**
- **Example 3**

Introduction

Apache Nifi

Today, we have many of ETL and data integration software, Some of these solutions are not free and more expansive, and others are maintained and operated by a community of developers looking to democratize the process.

with Dataflow Programming tools you can visually assemble programs from boxes and arrows, writing zero lines of code. Some of them are open source and some are suitable for ETL

> ***ETL** is short for extract, transform, load.*

Yes, you don't have to know any programming language. You just use ready-made "processors" represented with boxes, connect them with arrows, which represent exchange of data between "processors," and that's it.

> There are three main types of boxes: sources, processors, and sinks. Think Extract for sources, Transform for processors, and Load for sinks.'

Almost anything can be a source, for example, files on the disk or AWS, JDBC query, Hadoop, web service, MQTT, RabbitMQ, Kafka, Twitter, or UDP socket.

A processor can enhance, verify, filter, join, split, or adjust data. If ready-made processor boxes are not enough, you can code on Python, Shell, Groovy, or even Spark for data transformation.

Sinks are basically the same as sources, but they are designed for writing data.

Apache Kafka

Apache Kafka is an open source, distributed streaming platform used to storing, reading and analysing streaming data.

Kafka was originally created at LinkedIn, where it played a part in analysing the connections between their millions of professional users in order to build networks between people. It was given open source status and passed to the Apache Foundation — which coordinates and oversees development of open source software — in 2011.

Apache Kafka is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.

Apache NiFi can work as a Producer and a Consumer for Kafka. Both ways are suitable and depends upon requirements and scenarios.

For more details about Kafka you can follow this links :

Apache Kafka

Apache Kafka: A Distributed Streaming Platform.

kafka.apache.org

Introduction to Kafka — Confluent Platform

Apache Kafka® is a distributed streaming platform that: Publishes and subscribes to streams of records, similar to a...

docs.confluent.io

What does Dataflow mean?

Dataflow is the movement of data through a system comprised of software, hardware or a combination of both.

Dataflow is often defined using a model or diagram in which the entire process of data movement is mapped as it passes from one component to the next within a program or a system, taking into consideration how it changes form during the process.

What is Apache Nifi ?

Apache Nifi is an open source ETL tools and it was donated by the NSA to the Apache Foundation in 2014 and current development and support is

provided mostly by Hortonworks.

Apache Nifi is a data flow management systeme, that comes with a web UI built to provide an easy way to handle data flows in real-time, the most important aspect to understand for a quick start with Nifi is a flow-based programming

in plain terms you create a series of nods with a series of edges to create a graph that the data moves through

Example of web service that handles request to three different back-ends and return the result backs

in Nifi this nodes are processors and this edges are connectors, the data is stored within a packet of information known as a flow file, this flow file has things like content, attributes and edge, we will get in more specifics

Each individual processor comes with a variety of information readily available.

The status is whether is that processor stopped, started or incorrectly configured.

Time statistics gives you a brief window of the activity of that processor, useful in case more or less data is coming through than you thought

here is a very small sample of a few different processors available to us in Nifi, i personally like to regroup them like this : Inputs, Outputs, and the transformations and flow logic that goes in between

this input and outputs range from local files to cloud services to databases and everything in between, Apache Nifi is open-source and easily extendable, any processor not yet included can be created on the fly as per your own specifications, but for now here is the example provided by Nifi home page

What is Apache MINIFI ?

It is a sub-project of Apache NiFi, MINIFI can bring data from sources directly to a central NiFi instance and it is able to run most of NiFi's available processors.

MINIFI is used as an agent and we can applying primary features of NiFi at the earliest possible stage. and data can be collected from a variety of

protocols.

To learn more about MINIFI follow this link :

https://nifi.apache.org/minifi/index.html

What can be done with Apache Nifi processors ?

> *286 Pre-Built Processors*

- **Ingestion**: connectors to read/write data from/to several data sources
  - *Protocols*: HTTP (S), AMQP, MQTT, UDP, TCP, CEF,JMS, (S) FTP, etc.
  - *Brokers*: Kafka, JMS, AMQP, MQTT etc.
  - *Databases*: JDBC, MongoDB, HBase, Cassandra etc.
- **Extraction** (XML, JSON, Regex, Grok etc.)
- **Transformation :** - *Format conversion* (JSON to Avro, CSV to ORC etc.)
  - *Compression/decompression*, Merge, Split,encryption etc.
- **Data enrichment:** Attribute, content, rules etc.
- **Routing :** Priority, dynamic/static, based on content or metadata etc

Apache Nifi provide documentation for every processor

https://nifi.apache.org/docs.html

Apache Nifi architecture

NiFi is a Java program that runs within a Java virtual machine running on a server.The prominent components of Nifi are :

Web Server: Hosts NiFi's HTTP-based command and control API to enable its flow-based programming.

Flow Controller: A broker that manages schedule and provides threads for extensions. It keeps track of connections and writes to Repositories.

Processor: Does the work. It is the building block of data flow in NiFi. It is an interface mechanism to expose access to FlowFiles, their attributes, and content.

FlowFiles: Core abstraction key/value pair metadata attributes that help manage the data flow. It is a pointer to the actual content that is written in the repository.

FlowFile Repo: A write-ahead-log, a method where a log is written before action is taken. It is used to enable durability and atomicity. It keeps key/value pairs of metadata flowing in the system.

Content Repo: This is where content is written.

Provenance Repo: An indexed audit style information that tracks each piece of data that moves through the system.

Install Apache Nifi on Centos 7

First, we have to download the binairie file from this link :

```
wget https://www.apache.org/dyn/closer.lua?path=/nifi/1.11.4/nifi-
1.11.4-bin.tar.gz
```

Unzip the file :

```
>> tar -xvf nifi-1.11.4-bin.tar.gz
```

then we add the environment variables in the bash_profile by following this commands :

```
>> vim ~/.bash_profile

export NIFI=/home/salehsoft/dev/nifi-1.11.4
export PATH=$NIFI/bin:$PATH
```

we enter on conf file and make a copy for nifi.properties before make any changes just for best practice:

```
>> cd conf
>> cp nifi.properties nifi.properties.old
```

After that you can change what you want, for example the port that forward the nifi UI, the default port is 8080 and for me i use the port 9999.

To see current open ports, type:

```
>> firewall-cmd --list-ports
```

Type the following command to open TCP port 9999 :

```
>> firewall-cmd --permanent --add-port 9999/tcp
```

Type the following command to restart iptables and apply changes:

```
>> service iptables restart
```

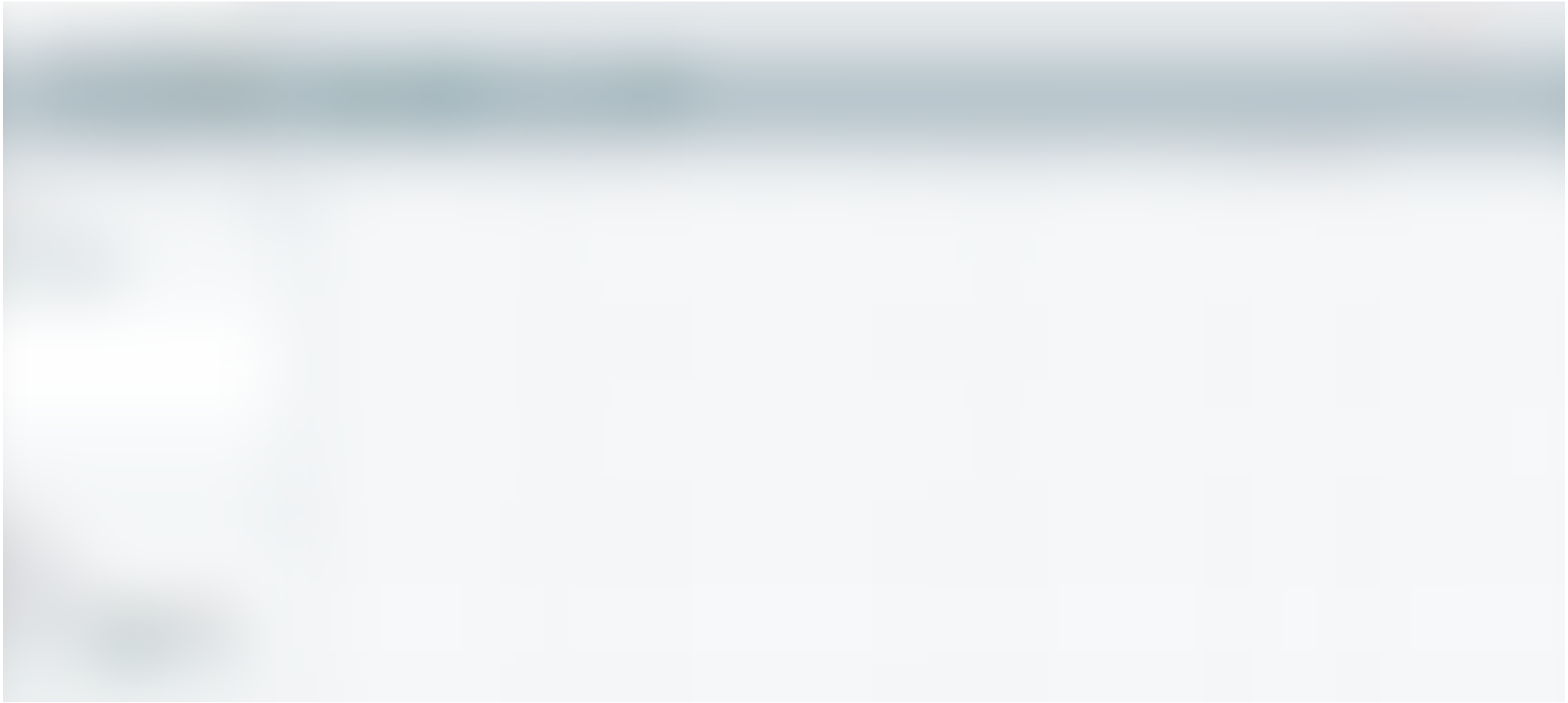
To start Nifi :

```
>> /bin/nifi.sh start
```

To see the logs follow this commands :

```
>> cd logs/
>> tail -f nifi-app.log
```

Now, we can go to the localhost:9999 and we got this empty canvas and you will get familiar with this items:



Build a first processor and data processing

Example 1

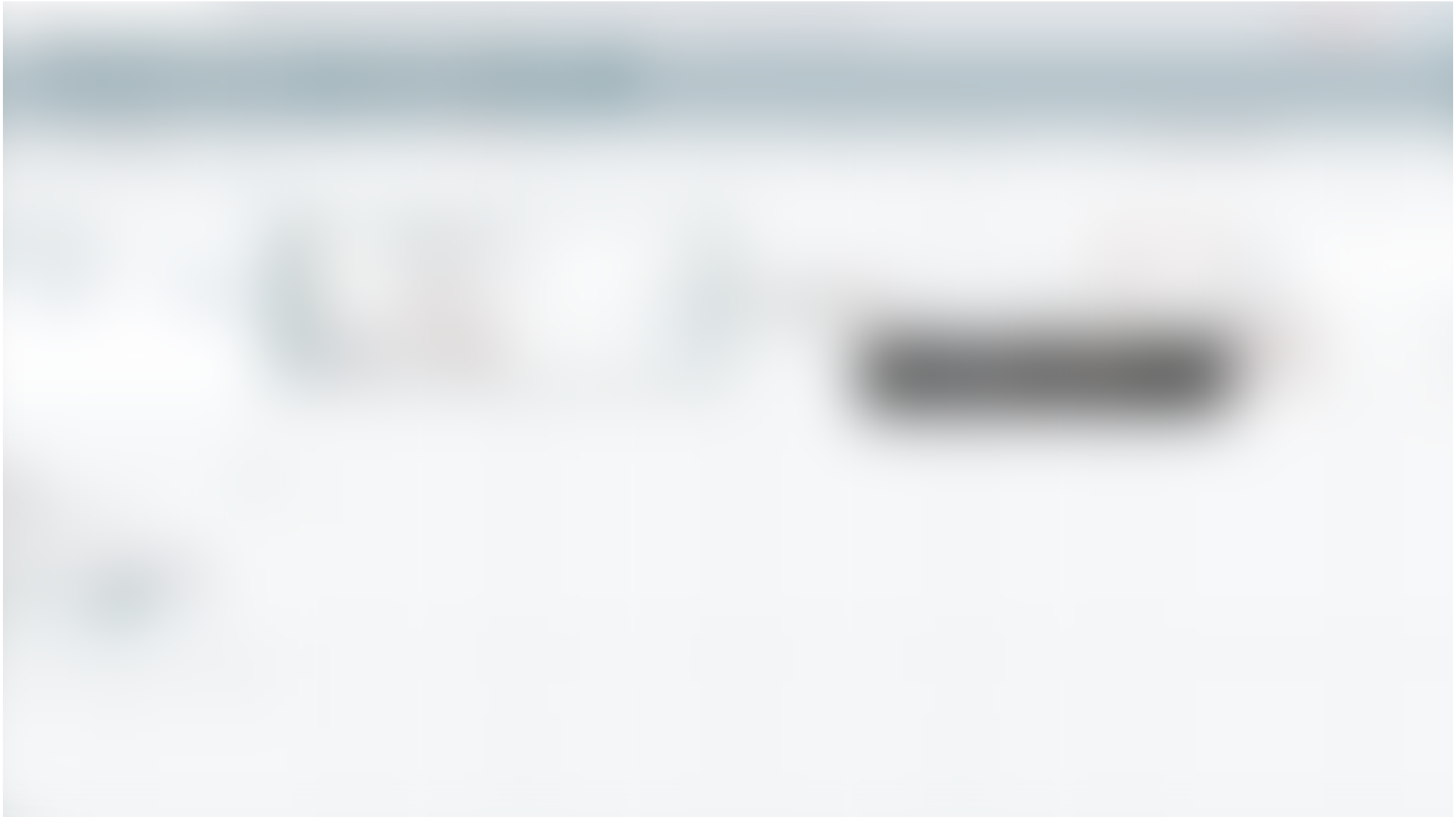
this example show you how to create a processor to generate flow file, generate work flow is the most basic processor in Nifi

we create 2 processors, (GenerateFlowFile, PutFile)

Configuration for the processor putFile, and i define the path of the output in my case is : /tmp/nifiFolder

Configuration for the processor putFile, and i define the path of the output in my case is : /tmp/nifiFolder

when we start the processor we have this :



A processor usually will have 3 outputs:

- Failure. If a FlowFile cannot be processed correctly, the original FlowFile will be routed to this output.
- Original. Once an incoming FlowFile has been processed, the original FlowFile is routed to this output.
- Success. FlowFiles that are successfully processed will be routed to this relationship.

> *If the PutFile processor becomes slow or freezes for some reason, FlowFiles generated by the GenerateFlowFile processor will be queued in the connection. After some time, back pressure will pause the GenerateFlowFile processor until the queue goes below the configured threshold.*

looking back to the /tmp/ and we see that the folder nifiFolder has been created and we can get the size of the folder :

```
>> du -sh nifiFolder // to get size of folder
>> cd nifiFolder & ls -l | wc -l // to get number of files in this
folder
```

We can also add other processor in the end like ***LogAttribute*** which i usually use to end my flows, and i can see what's happened, and for good practice.

So this is the first half of the flow example, in this example i am pulling data in from a bunsh of different places and i'm doing some kind of transformation and write it out to be used by other applications.

Apache NiFi and Apache Kafka together

Integration of Kafka and NiFi helps us to avoid writing lines of code to make it work. Easy to handle and understand the complete pipelines in one screen and easy to scale.

For Kafka, Apache NiFi has the capabilities to work both as a Producer and Consumer as well. Both ways are suitable and depends upon requirements and use cases.

We will ingest with NiFi and then filter, process, and segment it into Kafka topics. Kafka data will be in Apache Avro format with schemas specified in the Hortonworks Schema Registry.

This will be stored in Druid for real-time analytics and summaries. Hive, HDFS, and S3 will store the data for permanent storage.

Apache Nifi as a Producer

Apache Nifi can be used as a Kafka producer and will generate different type of data form many source as an input and forward it to the Kafka Broker.

Nifi will produce data and push on to the appropriate Kafka topic, by simply dragging and dropping a series of processors in Nifi (PublishKafka), and being able to visually monitor and control this pipeline.

Apache Nifi as Consumer

Apache NiFi can replace Kafka consumer and handle all of the logic. For instance, it can take the data from Kafka to move it forward. Here we avoid the Consumer code by just dragging and dropping the NiFi's ConsumeKafka processor. For example, you could deliver data from Kafka to HDFS without writing any code by using ConsumeKafka processor.

Example 2:

in this first simple example, i will create a producer with scala, and i create a Nifi consumer.

After have been installed Kafka, we have to start Zookeeper and Kafka, and create a topic.

For more details about Kafka and how to install :

Apache Kafka

Apache Kafka: A Distributed Streaming Platform.

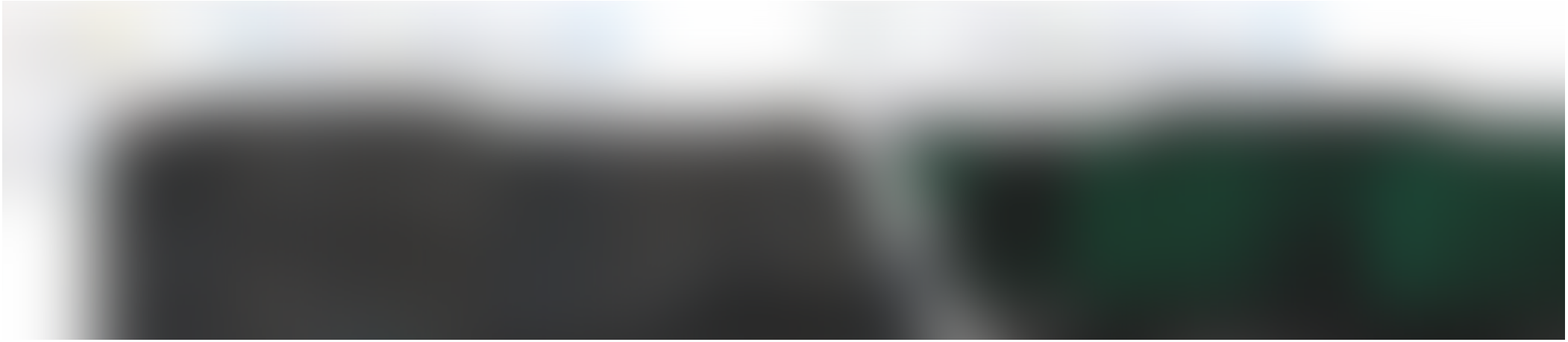
kafka.apache.org

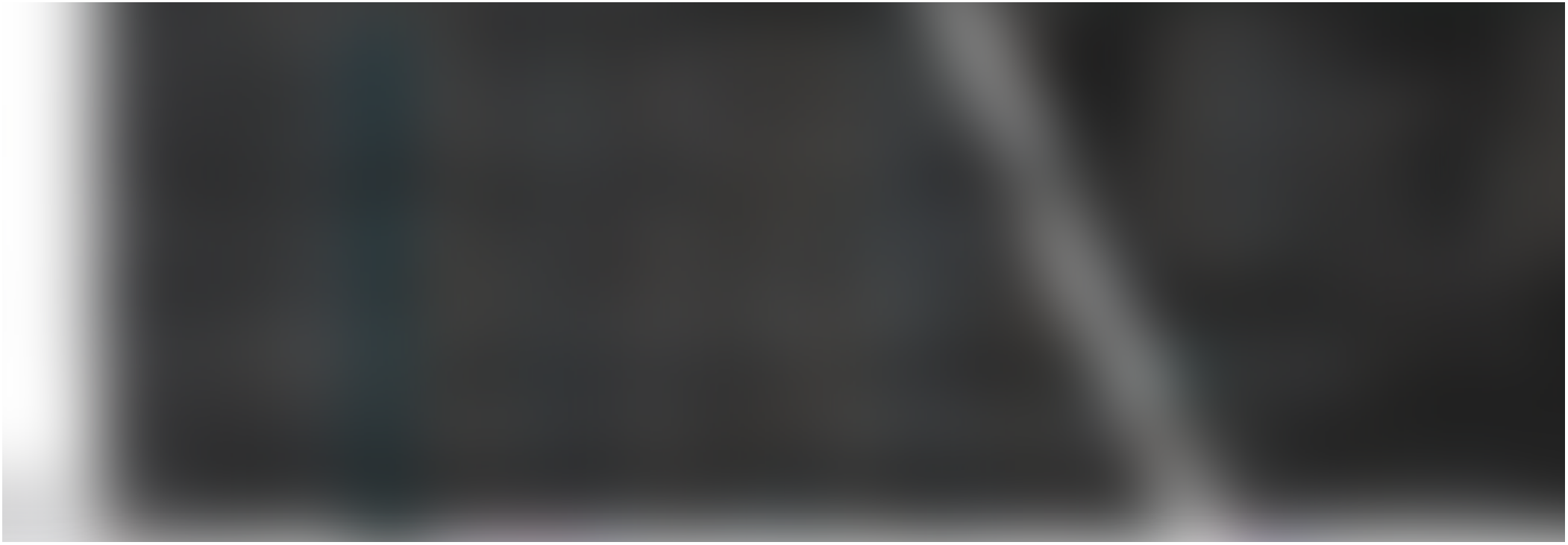
Type the following command to start Zookeeper :

```
>> sh zookeeper-server-start.sh ../config/zookeeper.properties
```

Type the following command to start Kafka :

```
>> sh kafka-server-start.sh ../config/server.properties
```





The topic that will contain our data is named Transaction

> Type the following command to create topic on kafka :

```
>> sh kafka-topics.sh --create --bootstrap-server localhost:9092 --
replication-factor 1 --partitions 1 --topic Transaction
```

Type the following command to see your list of topics :

```
>> sh kafka-topics.sh --list --bootstrap-server localhost:9092
```

I created a simple producer with scala to generate data and push on the kafka topic that named Transaction.

```
package kafka

import java.util.Properties

import org.apache.kafka.clients.producer.{KafkaProducer,
ProducerRecord}
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}

object Producer {
  def main(args: Array[String]): Unit = {
    val conf = new
SparkConf().setMaster("local[*]").setAppName("producer")
    val ssc = new StreamingContext(conf, Seconds(1))


    val props:Properties = new Properties()
    props.put("bootstrap.servers","127.0.0.1:9092")

props.put("key.serializer","org.apache.kafka.common.serialization.Lon
gSerializer")

props.put("value.serializer","org.apache.kafka.common.serialization.S
tringSerializer")
    props.put("acks","all")

    val producer = new KafkaProducer[Long, String](props)
```

```
    val topic = "Transaction"

    // create a timer that we will use to stop the processing after
60 seconds so we can print some results
    try {
      for (i <- 0 to 500) {
        val record = new ProducerRecord[Long, String](topic, i,
"bonjour je suis salah bigapps" + i)
        val metadata = producer.send(record)

println(record.topic(),record.key(),record.value(),metadata.get().tim
estamp(),metadata.get().partition())
        Thread.sleep(5000)
      }
    }catch{
      case e:Exception => e.printStackTrace()
    }finally {
      producer.close()
    }


  }

}
```

After executing the producer, now we will create the processor Nifi for consuming data from the topic Transaction in real time, and we can check the number of message produced and consumed. for this i create 2 processor in NIFI, (ConsumerKafka, PutFile)

In the following image Gift you will see steps by steps how to build consumerKafka and putFile:

We can see the generated file in the folder that we are created with the processor PutFile “/tmp/nifiKafka”

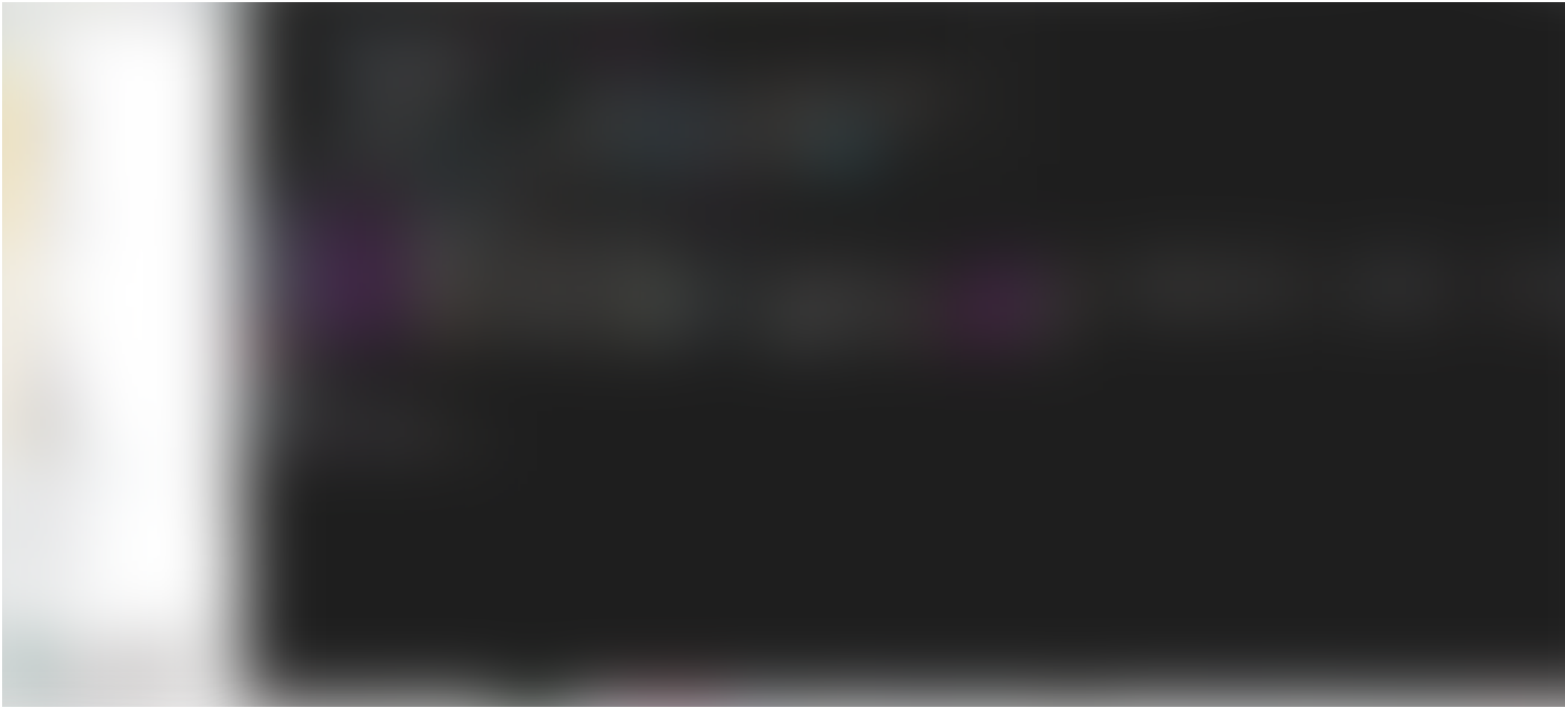
if we stop PutFile, the data will be in the queue, and we can consult data in the queue:

we can view or download the data from queue :

Example 3

> *In this example we will create producer and consumer only with NiFi, so we use* ***PublishKafka, ConsumerKafka, PutFile, TailFile, SplitText, RouteContent,***
>
> *The entry point of this example is thr processor* ***TailFaile c****onfigured with* ***nifi-app.log.***
>
> *The second processor is* ***SplitText, t****his processor will split the log received by 1 ligne and send this ligne to the third processor which is* ***RouteOnContent.*** *This processor will check if there are any of this word in this ligne of log* ***(INFO, Warn or ERROR)*** *and send the ligne of log to the appropriate* ***Topic kafka (we have 3 topic INFO, WARN, ERROR)*** *with the processor* ***PublishKafka.***
>
> *In the ends, we create a consumer with the processor* ***ConsumerKafka, and put the result to HDFS or in any repository that we want.***
>
> *In the end of this Example, the workflow will be like this :*

I have created 3 topics (INFO,WARN,ERROR) :



Our input data is nifi-app.log, we will split logs by line with SplitText Processor.

In this gift you can see all steps that i have made to create this workflow:

Conclusions

NiFi is a framework that we can find them as a part of HDF (Hortonworks Data Flow) it's used for managing complex data flows. Its flow-based programming makes it easy to use and learn. It is a cross platform application with easy installation, configuration, and runtime. It is highly scalable, agile, flexible, durable, and resilient.

In this article we have learn the basics of NiFi, kafka, and we have built our first workflow.

In the next article, we will made more complicated workflow, and manipulate different types of data (csv, Json, Avro, ...) from different resources, and explain other services like controller services, and how make the SchemaRegistry.

Thank you for reading

> *If you find this article useful, please feel free to share with others and if you have any questions, please let me now in comment.*

Apache Nifi Kafka Kafka Streams DevOps Dataops