
Streaming De Datos IoT Con Kafka

Iot Data Streaming With Kafka

**Omar Castillo-Alarcón¹, Edwin Fredy Chambi-Mamani², Gludher Quispe-Cotacallapa³,
Erwin Cruz-Mamani⁴**

Computación en la Nube y Big Data

Grupo 03

Universidad Nacional de San Agustín

Arequipa, Perú

Resumen

Se implementó un cluster kafka en el proveedor Confluent Cloud, en el cual se desarrollaron cuatro tópicos de IoT de flota de camiones, IoT sensor de Luz, Compras, Tarjetas de Crédito. Así mismo a la salida de los pipelines se configuraron salidas en un espacio FTP y en un aplicación de consola en Python. En otras palabras se implementaron cuatro Productores y cuatro Consumidores.

I. INTRODUCCIÓN:

Confluent Cloud es un proveedor cloud de apache kafka. Lo proporciona Confluent, la empresa fundada por los creadores de Kafka. Confluent Cloud, permite el uso de créditos gratis para el aprendizaje de uso de su plataforma.

Provee una serie de conectores kafka de producción y consumo de configuración no tan compleja, así mismo tiene mecanismos para la manipulación de datos al momento del stream, desde lo básico hasta modelos de Machine Learning.

IoT usa como protocolo MQTT, nuestra solución no usa MQTT sino una aproximación propia, convirtiendo un microcontrolador con sensores y puerto serie en un sensor con capacidad de internet (un IoT), los datos del sensor son guardados en una DDBB MySQL en un Droplet (Servidor virtual) en Digital Ocean, enmarcando esta descripción como un

¹ Omar Castillo-Alarcón, Universidad Nacional de San Agustín, **UNSA**, ocastillo@unsa.edu.pe.

² Edwin Fredy Chambi-Mamani, Universidad Nacional de San Agustín, **UNSA**, echambimam@unsa.edu.pe.

³ Gludher Quispe-Cotacallapa, Universidad Nacional de San Agustín, **UNSA**, gquispeco@unsa.edu.pe.

⁴ Erwin Cruz-Mamani, Universidad Nacional de San Agustín, **UNSA**, ecruzmama@unsa.edu.pe.

ejercicio académico.

II. MATERIALES Y MÉTODOS:

El sensor está compuesto de la siguiente manera, un LDR (Resistencia variable con la luz) en una entrada analógica A3 de un microcontrolador Arduino. Este microcontrolador envía los datos de medida en A3 a 9600 bps por una conexión serial a una Laptop.

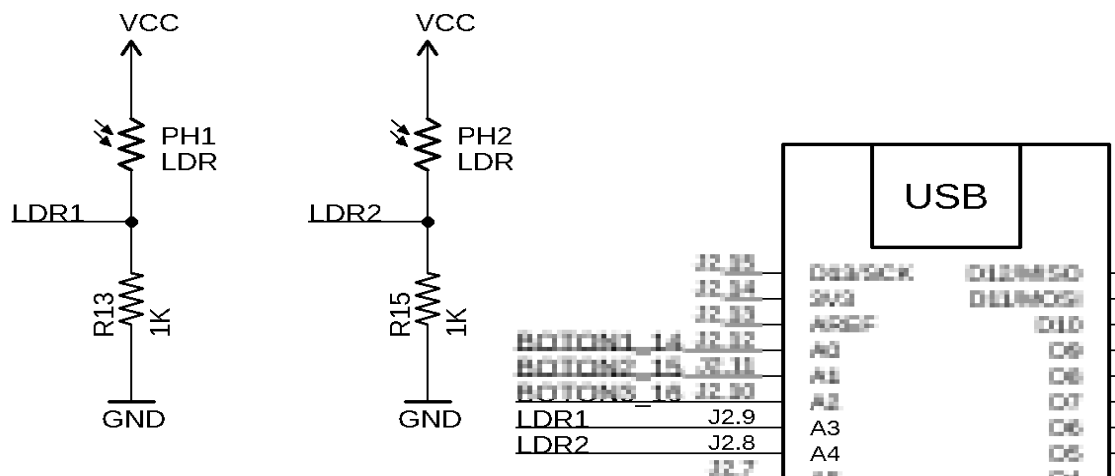


Figura 1, Esquema electrónico del Sensor de Luz y Arduino (LDR 1).

Para el envío del microcontrolador se usó:

```
/*
Ejemplo de Lectura Analogica de Luz
*/
const int analogInPin = A3; // Analog input
int sensorValue = 0;        // value

void setup() {
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(analogInPin);
  Serial.println(sensorValue);
  delay(5000);
}
```

Para la recepción y envío de datos de luz se escribió el siguiente programa, el cual recibe los datos seriales del Arduino y lo envía a una DDBB remota en Digital Ocean.

```
#!/usr/bin/python3
# UNSA MCC

import serial #importa libreria serial
import time   #importa time para hacer delays
import datetime #importa libreria dia de hoy fechas
```

```

import pymysql #importa libreria base de datos en mysql

server = '128.199.6.xxx' #iot database
user = 'newuser'
password = 'you_password_here'
ddbb='unsa'
charset = 'utf8mb4'

try: #intenta conectarte y haz
    conexion = pymysql.connect(host=server, user=user, password=password,
                               db=ddbb, charset=charset,
                               cursorclass=pymysql.cursors.DictCursor)
    print("Conexión correcta")
except (pymysql.err.OperationalError, pymysql.err.InternalError) as e: #si n
    print("Ocurrió un error al conectar: ", e) #e es el tipo de error de mysql

dispositivo = serial.Serial('COM4',9600)
dispositivo.flushInput()

time.sleep(1) # retardo estabiliza puerto osea delay

estado = 1

while estado==1:

    try:
        conexion = pymysql.connect(host=server, user=user, password=password,
                                    db=ddbb, charset=charset,
                                    cursorclass=pymysql.cursors.DictCursor)

        estado = True
        print("Conexión correcta 2")
        #time.sleep(2)
        luz = dispositivo.readline().decode('utf-8').rstrip() #l guarda en 'luz'
        fecha = datetime.datetime.now()
        fecha = fecha.strftime("%Y/%m/%d %H:%M:%S")
        fecha = str(fecha) #'fecha' es fecha actual de datetime
        luz = int(luz)
        print (luz)

        try:
            with conexion.cursor() as cursor: #metodo de escritura
                # Crear un record
                sql = "INSERT INTO `datos` (`fecha`,`luz`) VALUES (%s, %s)"
                cursor.execute(sql, (fecha, luz)) #inserta variables en l
                conexion.commit() ##si no se hace commit no se guarda datos
                print ('Exito commit, esperando 10 segundos')
                dispositivo.flushInput()
                time.sleep(0.1)

        finally:
            conexion.close() #cerrar coneccion
            cursor.close()
            print('Conexion y cursor cerrados')
            time.sleep(10)

    except (pymysql.err.OperationalError, pymysql.err.InternalError) as e:

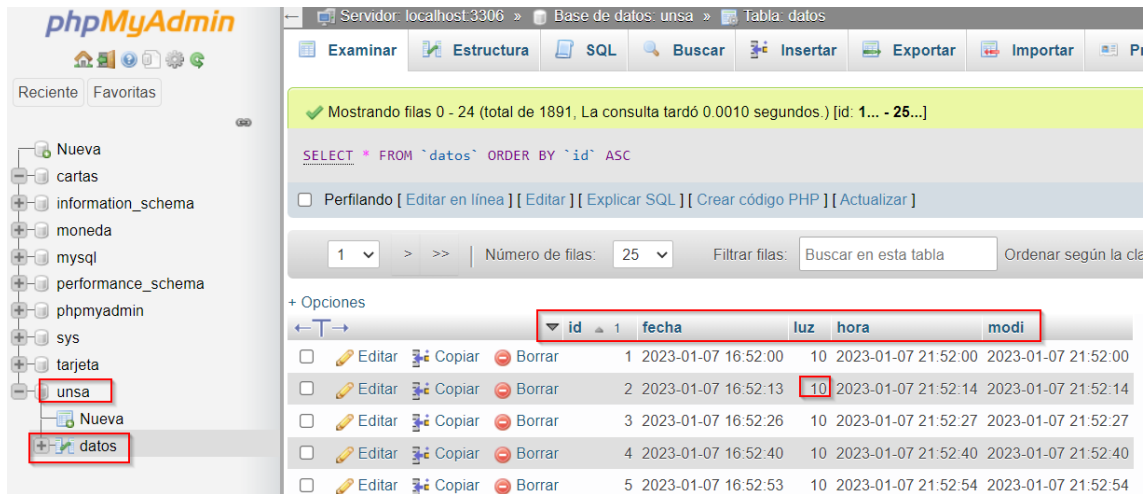
```

```

print("Ocurrió un error al conectar: ", e)
estado= False
break

```

La base de datos en Digital Ocean tiene el siguiente esquema:



	id	fecha	luz	hora	modi
<input type="checkbox"/>	1	2023-01-07 16:52:00	10	2023-01-07 21:52:00	2023-01-07 21:52:00
<input type="checkbox"/>	2	2023-01-07 16:52:13	10	2023-01-07 21:52:14	2023-01-07 21:52:14
<input type="checkbox"/>	3	2023-01-07 16:52:26	10	2023-01-07 21:52:27	2023-01-07 21:52:27
<input type="checkbox"/>	4	2023-01-07 16:52:40	10	2023-01-07 21:52:40	2023-01-07 21:52:40
<input type="checkbox"/>	5	2023-01-07 16:52:53	10	2023-01-07 21:52:54	2023-01-07 21:52:54

Figura 2, Base de Datos en Digital Ocean, contiene los registros de la medida del sensor de luz.

Para el Sink del stream configuramos un SFTP en nuestro droplet para que reciba la salida de los conectores del cluster, en este caso usamos un servidor SFTP con las credenciales de usuario “sammy”.

```

root@ubuntu-s-1vcpu-1gb-sfo3-01:/home/sammy/topics# ftp sammy@localhost
Trying [::1]:21 ...
Connected to localhost.
220 (vsFTPd 3.0.5)
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 

```

Cuando se ejecuta el programa de envío de datos de luz a la DDBB se tiene la siguiente salida:

```

Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec 6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython -- An enhanced Interactive Python.

In [1]: runfile('H:/My Drive/UNSA/kafka/sensordeluz.py', wdir='H:/My Drive/UNSA/kafka')
Conexión correcta
Conexión correcta 2
25
Exito commit, esperando 10 segundos
Conexion y cursor cerrados
Conexión correcta 2
41
Exito commit, esperando 10 segundos
Conexion y cursor cerrados

```

Figura 3, Salida de terminal del sensor de luz.

Una vez comprobado el funcionamiento de nuestro sensor IoT. Procedemos a configurar nuestro cluster Kafka en Confluent Cloud. Para esto primeramente nos haremos de una cuenta de uso gratuito para capacitación (al momento de la redacción de este informe este crédito gratuito era de 400\$).

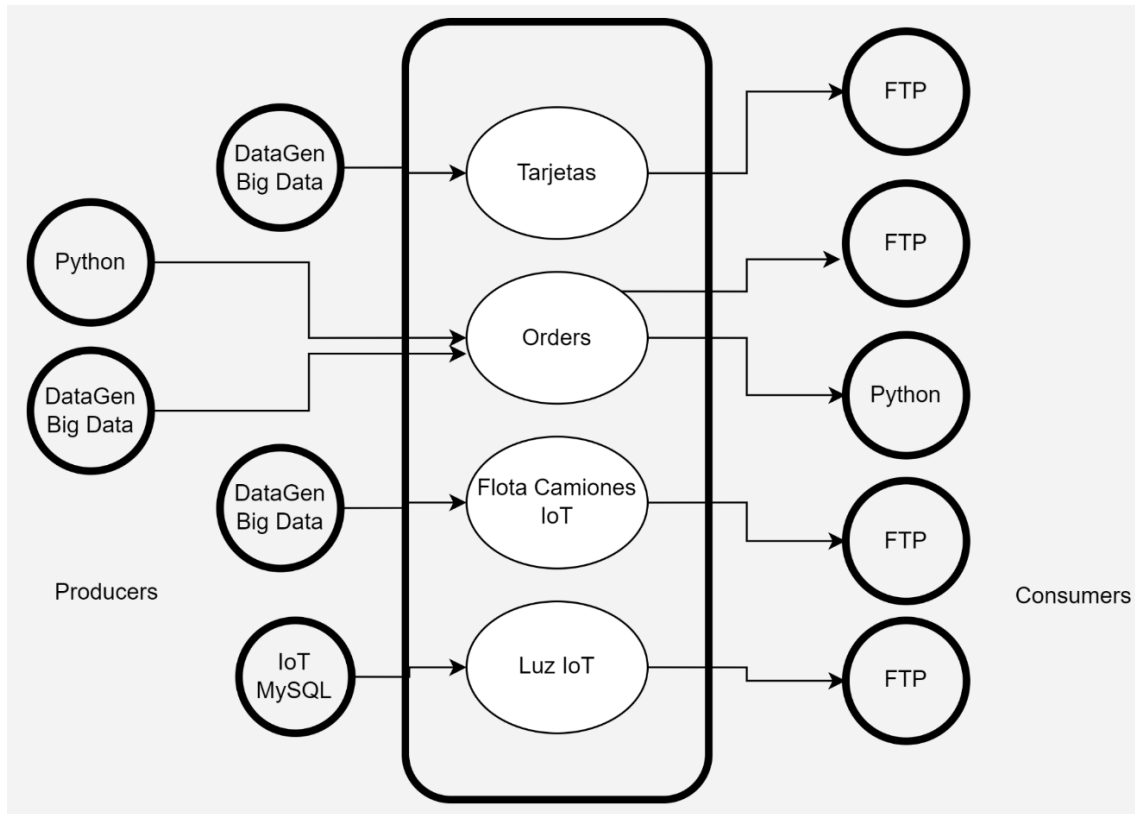
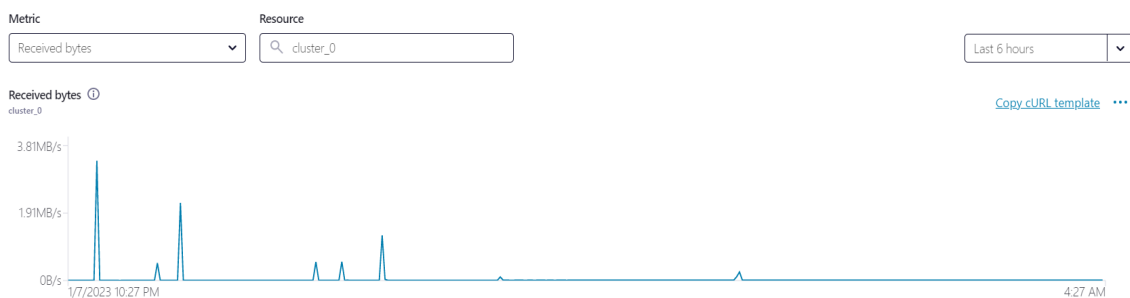


Figura 4, Escenario implementado en Confluent Cloud.

A continuación se muestran algunas métricas del cluster kafka.



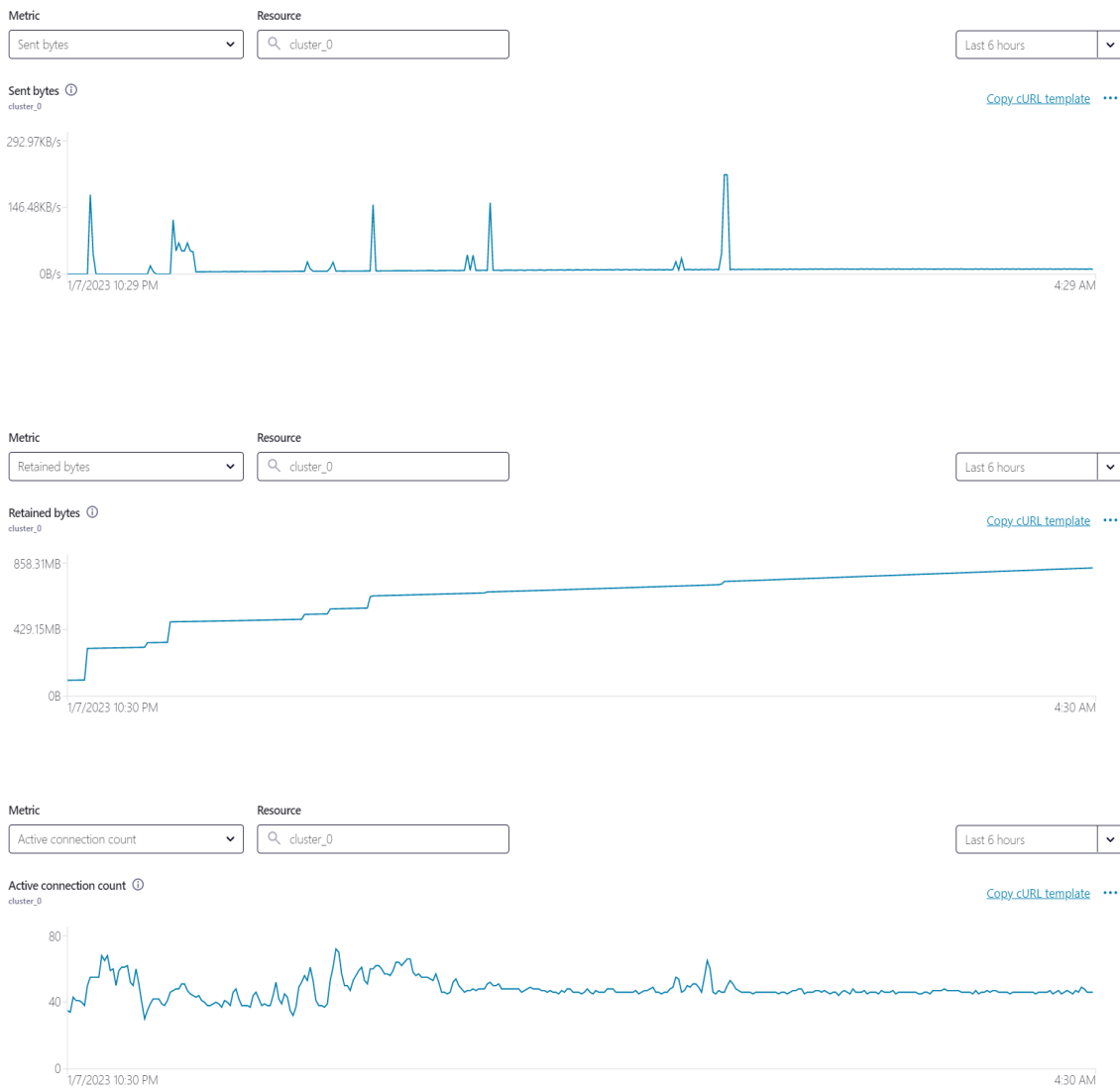
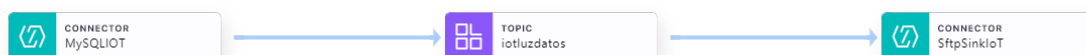


Figura 5, Métricas del cluster, fijarse en los bytes retenidos (configurado como retención infinita).

Para los productores de Big Data se usaron generadores configurados para escenarios diversos, así como nuestros datos de IoT de luces, en nuestro caso:

- Generador de Stream de Tarjetas de Crédito en formato AVRO.
- Generador de Stream de Compras en formato AVRO.
- Generador de Stream IoT para flota de camiones en formato AVRO.
- Datos de MySQL del sensor de luz.

Los Streams tienen la siguiente forma:



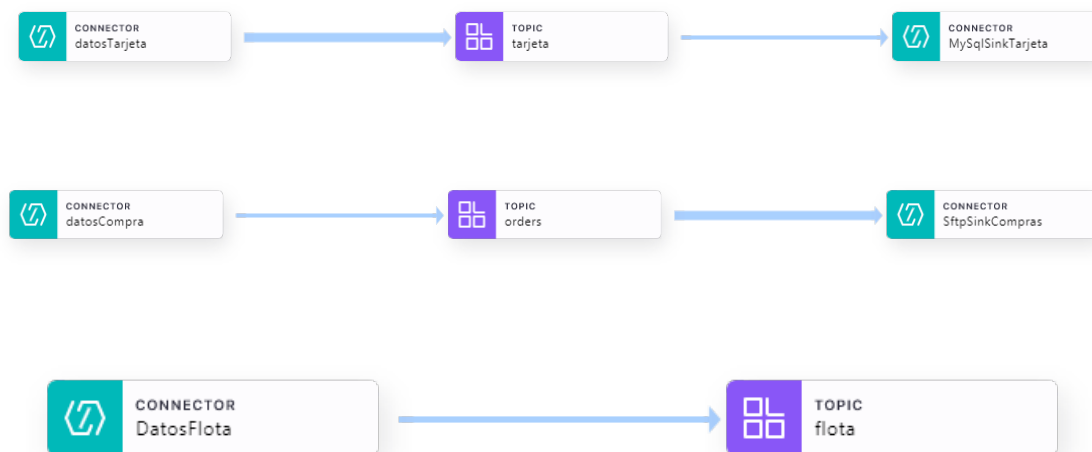


Figura 6, Esquema de los streams, no se muestran todos los conectores de salida, porque hacen polling cada hora en el caso de FTP y a veces se tiene timeout en caso de MySQL.

III. RESULTADOS:

```

root@ubuntu-s-1vcpu-1gb-sfo3-01:/home/sammy/topics# ls
flota  iotluzdatos  orders  tarjeta
root@ubuntu-s-1vcpu-1gb-sfo3-01:/home/sammy/topics#

```

Se descargo los JSON procesados en el stream y se colgaron en el repositorio, poseen la siguiente forma.

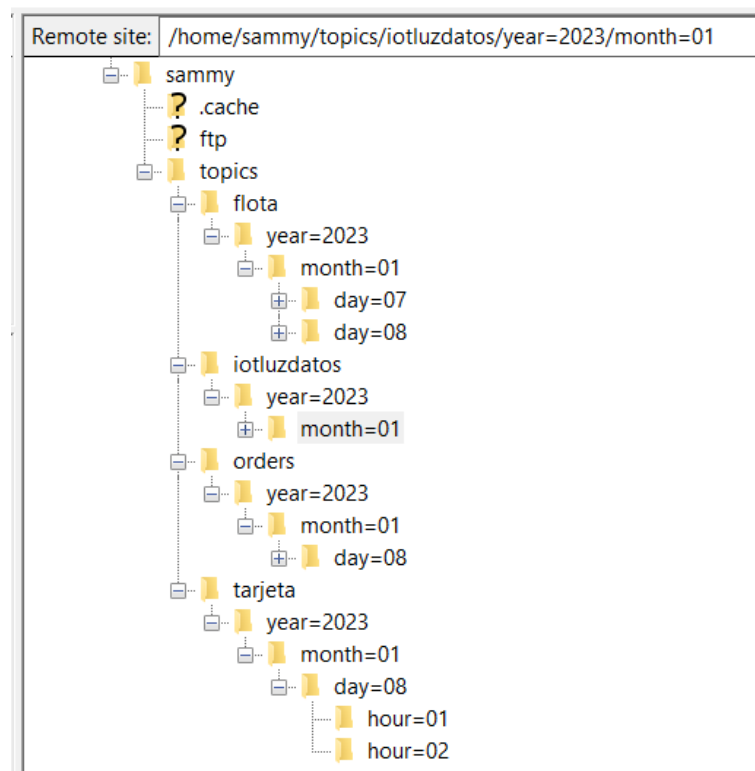
omart > Downloads > file > topics > iotluzdatos > year=2023 > month=01 > day=08 > hour=02

Name	Date modified	Type	Size
iotluzdatos+0+0000046347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000047347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000048347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000049347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000050347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000051347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000052347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000053347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000054347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000055347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000056347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000057347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000058347	1/7/2023 11:06 PM	JSON Source File	82 KB
iotluzdatos+0+0000059347	1/7/2023 11:06 PM	JSON Source File	82 KB

Y abriendo un JSON se tiene:

```
C: > Users > omar > Downloads > file > topics > iotluzdatos > year=2023 > month=01 > day=08 > hour=02 > {} iotluzd
1  {"id":382,"fecha":1673124590000,"luz":40,"hora":1673142590000,"modi":1673142590000}
2  {"id":383,"fecha":1673124601000,"luz":41,"hora":1673142601000,"modi":1673142601000}
3  {"id":384,"fecha":1673124612000,"luz":41,"hora":1673142612000,"modi":1673142612000}
4  {"id":385,"fecha":1673124623000,"luz":40,"hora":1673142624000,"modi":1673142624000}
5  {"id":386,"fecha":1673124635000,"luz":41,"hora":1673142635000,"modi":1673142635000}
6  {"id":387,"fecha":1673124646000,"luz":41,"hora":1673142646000,"modi":1673142646000}
7  {"id":388,"fecha":1673124657000,"luz":41,"hora":1673142657000,"modi":1673142657000}
8  {"id":389,"fecha":1673124668000,"luz":41,"hora":1673142669000,"modi":1673142669000}
9  {"id":390,"fecha":1673124680000,"luz":41,"hora":1673142680000,"modi":1673142680000}
10 {"id":391,"fecha":1673124691000,"luz":41,"hora":1673142691000,"modi":1673142691000}
11 {"id":392,"fecha":1673124702000,"luz":41,"hora":1673142702000,"modi":1673142702000}
12 {"id":393,"fecha":1673124713000,"luz":41,"hora":1673142713000,"modi":1673142713000}
13 {"id":394,"fecha":1673124724000,"luz":23,"hora":1673142724000,"modi":1673142724000}
14 {"id":395,"fecha":1673124735000,"luz":23,"hora":1673142735000,"modi":1673142735000}
15 {"id":396,"fecha":1673124746000,"luz":25,"hora":1673142747000,"modi":1673142747000}
16 {"id":397,"fecha":1673124758000,"luz":24,"hora":1673142758000,"modi":1673142758000}
```

Y en el Consumidor Sink FTP se tiene los tópicos como se muestra.



IV. OBSERVACIONES

- A. Cómo trabaja la propuesta, Usa generadores y una DDBB de luz en MySQL proveniente de un sensor de luz, los cuales alimentan un cluster Kafka con 4 tópicos ya descritos, a la salida se tiene Sinks de FTP y DDBB como

consumidores, así como consola python.

- B. Problemas y desafíos en la implementación, Recursos, no es barato montar clusters y el crédito de entrenamiento puede acabarse en pocos días. Así mismo la integración con la mayoría de servicios externos es de paga, lo qué imposibilita el uso de ese servicio (Por ejemplo Grafana, el Pipeline pasa por hasta 3 servicios externos).
- C. Interacción con herramientas externas a Kafka, se probó con MySQL, SFTP, productores y Consumidores de las clases de Confluent en Python y se probó S3 pero sin éxito, la mayoría de conectores son de pago por hora, pero al otro lado del pipeline los precios no se pueden afrontar para un ejercicio académico.

V. CONCLUSIONES

- A. Para hacer Big Data no solo se requiere tener acceso a grandes volúmenes de información sino también, acceso a recursos financieros, ya qué los clusters no son económicos y dependiendo de la carga, particiones y tareas, el costo puede llegar a ser prohibitivo.
- B. El acceso a Datos Big Data no está al alcance de todos, lo qué si esta es Generadores de stream de Big Data qué cumplen la función de Productor, pero permitiendo dimensionar capacidades operativas de acuerdo a la configuración de el Generador así como pruebas de rendimiento.

VI. REFERENCIAS

- A. White Paper Confluent Cloud. Y documentación del Proveedor, así como videos de su canal de Youtube.