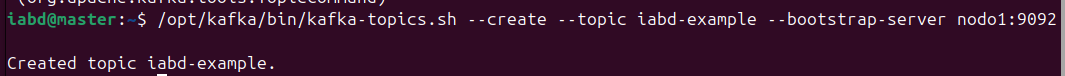
# EJERCICIOS

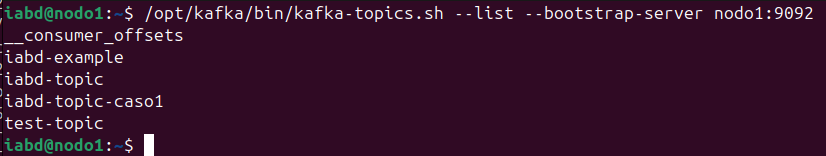
### EJERCICIO 1

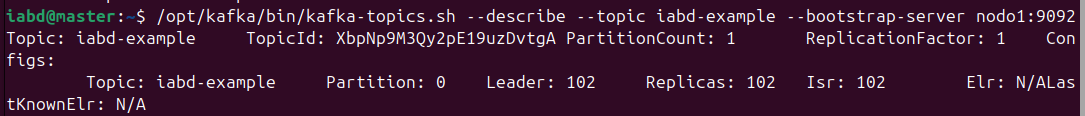
Crea un clúster Kafka con 3 nodos y 3 brokers con sistemas Ubuntu (para hacerlo sigue el documento 4\_2Instlación Kafka).

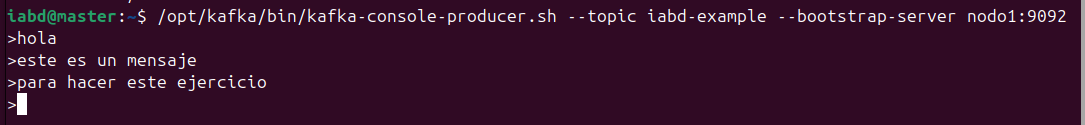
### EJERCICIO 2

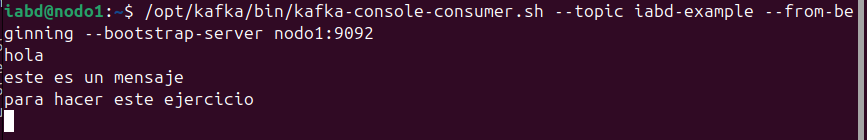
Realiza el caso de uso 0





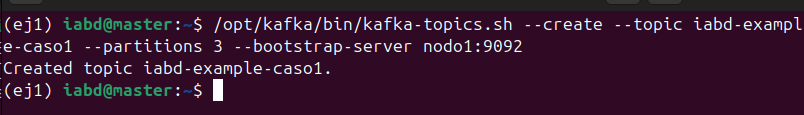


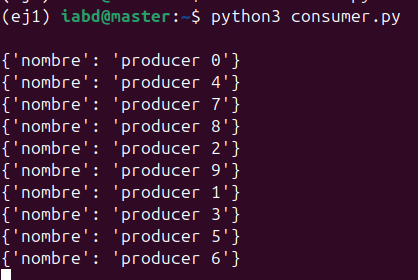




### EJERCICIO 3

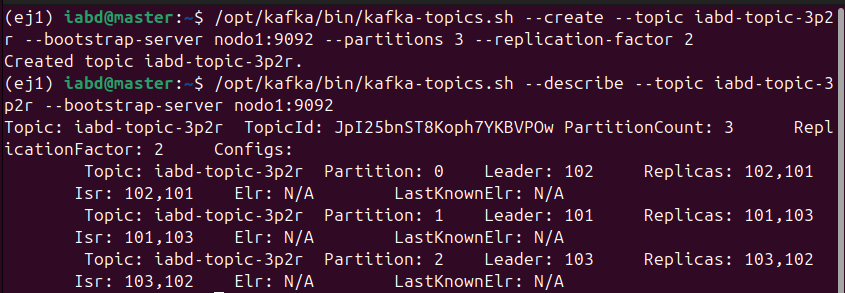
Realiza el caso de uso 1.

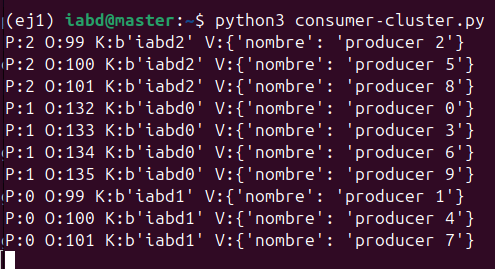




### EJERCICIO 4

Realiza el caso de uso 2

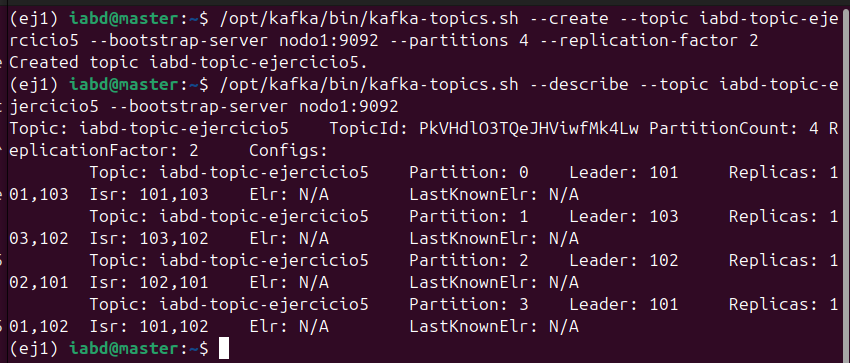




### EJERCICIO 5

Se pide:

1. Crea un topic llamado iabd-topic-ejercicio5 con 4 particiones y un factor de replicación 2.

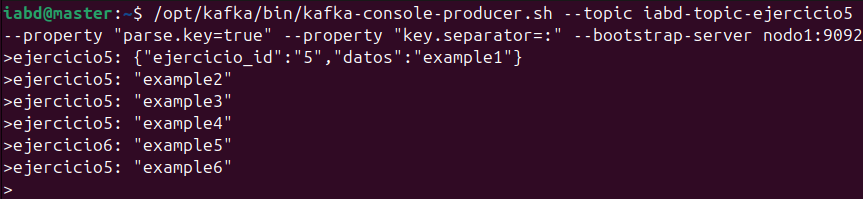


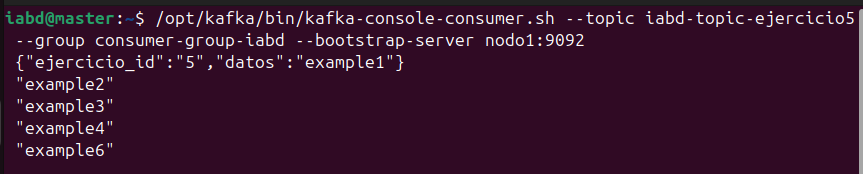
1. A continuación, lanza dos consumidores que pertenezcan al grupo de consumidores consumer-group-iabd.

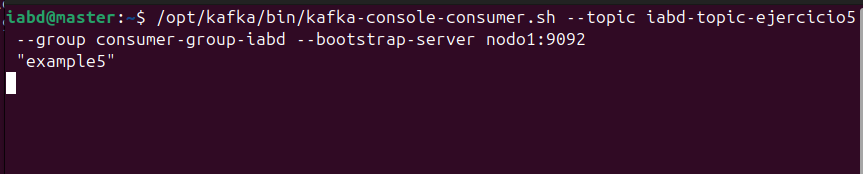




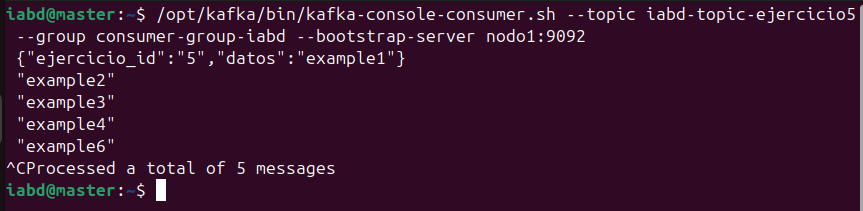
1. Lanza un productor y envía varios mensajes compuestos de clave:valor y comprueba cómo aparecen en los consumidores.

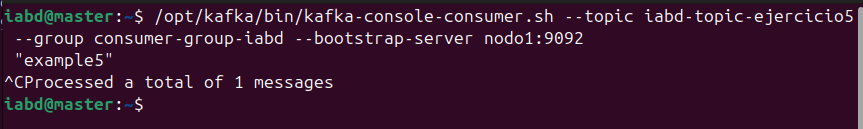




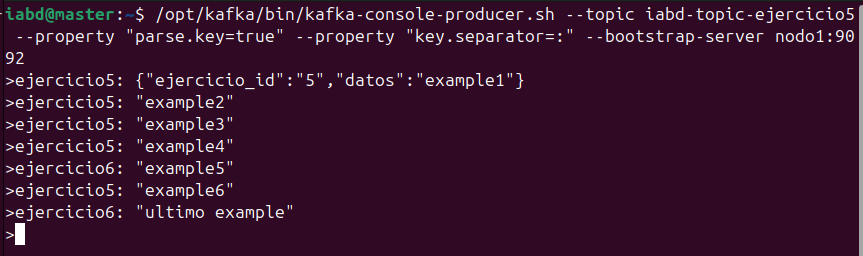


1. Detén ambos consumidores.

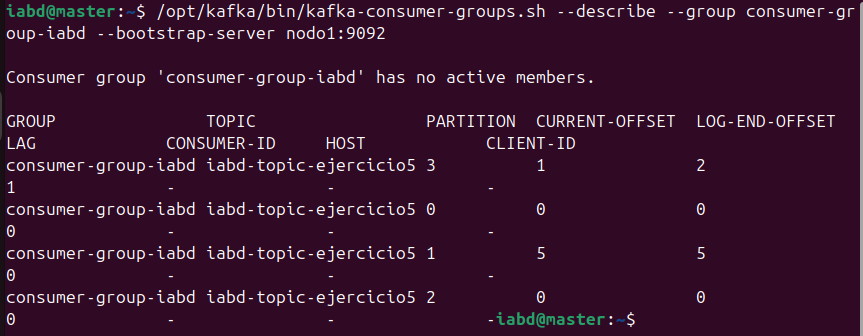




1. Envía un nuevo mensaje.



1. Obtén información sobre el estado del grupo de consumidores consumer-group-iabd y explica sus valores.



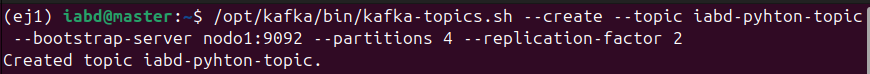
* El grupo de consumidores consumer-group-iabd no tiene miembros activos en este momento, como indica la primera línea ("Consumer group 'consumer-group-iabd' has no active members").
* Los offsets actuales y los offsets finales en las particiones 0, 1 y 2 son iguales o tienen una pequeña diferencia, lo que indica que los mensajes se han consumido hasta el punto final disponible en esas particiones.
* En la partición 3, el LAG es 1, lo que sugiere que aún queda 1 mensaje pendiente de consumo.
* En la partición 0, 1 y 2 no hay mensajes pendientes de consumo, ya que el LAG es 0.
* La ausencia de valores en las columnas **CONSUMER-ID**, **HOST** y **CLIENT-ID** confirma que no hay consumidores activos para el grupo consumer-group-iabd en este momento.

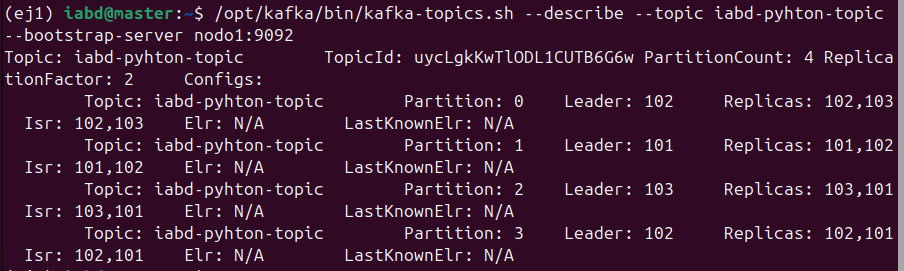
### EJERCICIO 6

Crea un topic denominado iabd-python-topic (con 4 particiones) y utilizando python crea:

1. un productor que envíe datos de una persona cada 10 segundos al topic. Para ello, crea el script personas.py y utiliza la librería Faker para crear los datos ficticios de personas:
2. un consumidor que reciba las personas y, mediante pymongo, las inserte en mongodb en una colección llamada kafka\_personas.

Comprueba que funciona correctamente.





*Script personas.py:*

from kafka import KafkaProducer

import json

import time

from faker import Faker

# Inicializar Faker

fake = Faker()

# Configuración del productor Kafka

producer = KafkaProducer(

bootstrap\_servers=['nodo1:9092', 'nodo2:9092', 'nodo3:9092'], # Lista de nodos

value\_serializer=lambda x: json.dumps(x).encode('utf-8')

)

# Función para generar una persona

def generar\_persona():

return {

"nombre": fake.name(),

"direccion": fake.address(),

"email": fake.email(),

"telefono": fake.phone\_number(),

"fecha\_nacimiento": fake.date\_of\_birth().strftime("%Y-%m-%d")

}

# Enviar personas cada 10 segundos

while True:

persona = generar\_persona()

print(f"Enviando: {persona}")

producer.send('iabd-python-topic', value=persona)

time.sleep(10)

*Script Consumidor:*

from kafka import KafkaConsumer

import json

from pymongo import MongoClient

from pymongo.errors import ConnectionError, ServerSelectionTimeoutError

# Intentar conectar a MongoDB

try:

client = MongoClient('mongodb://localhost:27017/') # Cambia la URL si MongoDB está en otro nodo

# Verificar si la conexión es exitosa

client.admin.command('ping') # Esto verifica si MongoDB está disponible

db = client['kafka']

collection = db['kafka\_personas']

print("Conexión a MongoDB exitosa.")

except (ConnectionError, ServerSelectionTimeoutError) as e:

print(f"Error de conexión a MongoDB: {e}")

exit(1) # Salir del script si no se puede conectar

# Configuración del consumidor Kafka

consumer = KafkaConsumer(

'iabd-python-topic',

bootstrap\_servers=['nodo1:9092', 'nodo2:9092', 'nodo3:9092'], # Lista de nodos

group\_id='grupo\_personas',

value\_deserializer=lambda x: json.loads(x.decode('utf-8'))

)

# Consumir los mensajes y guardarlos en MongoDB

for mensaje in consumer:

try:

persona = mensaje.value

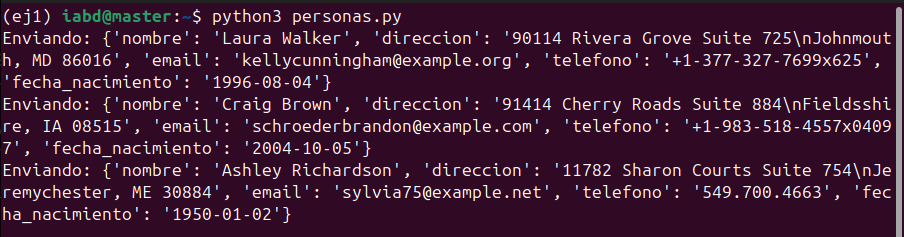
print(f"Recibido: {persona}")

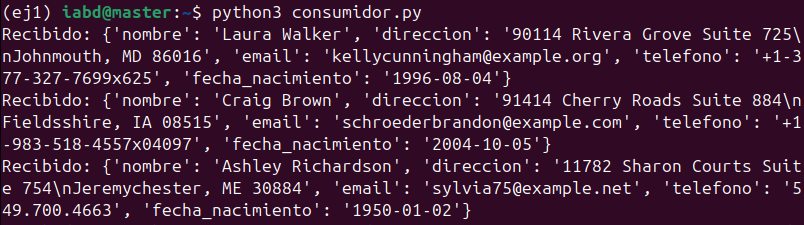
# Insertar la persona en la colección de MongoDB

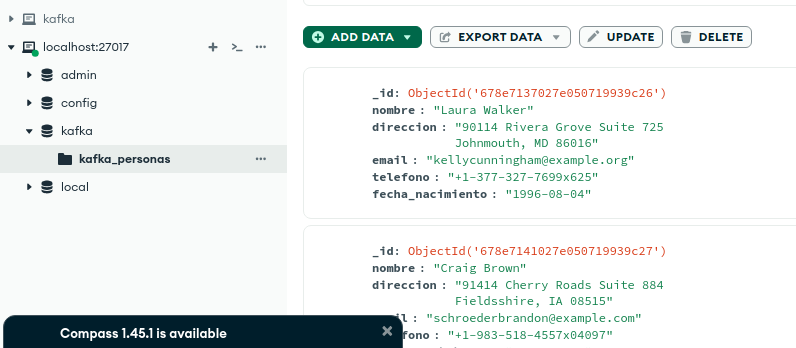
collection.insert\_one(persona)

except Exception as e:

print(f"Error al insertar en MongoDB: {e}")

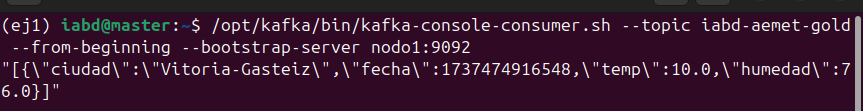


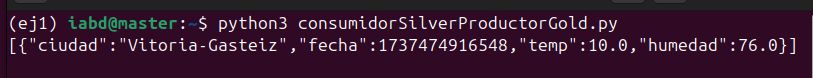


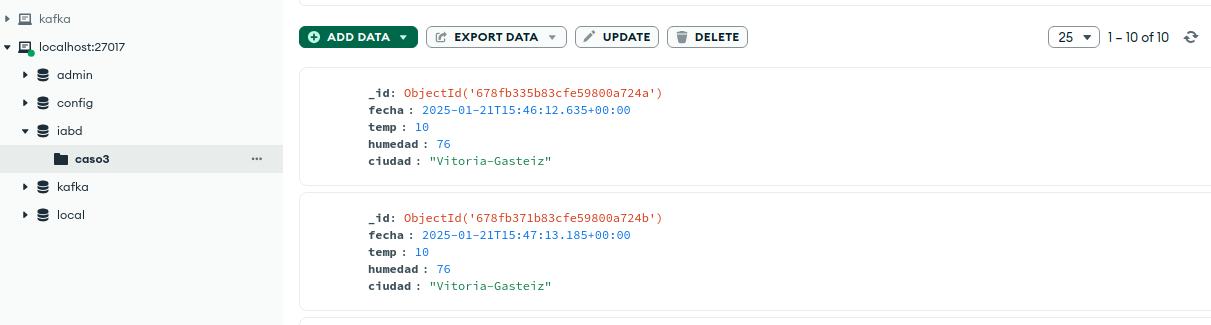


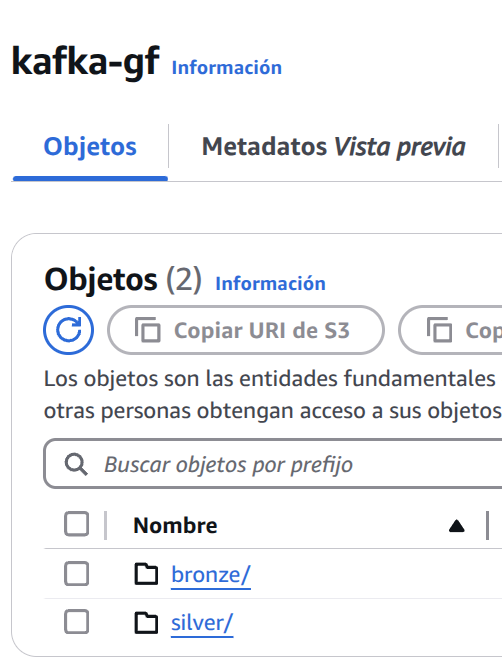
### EJERCICIO 7

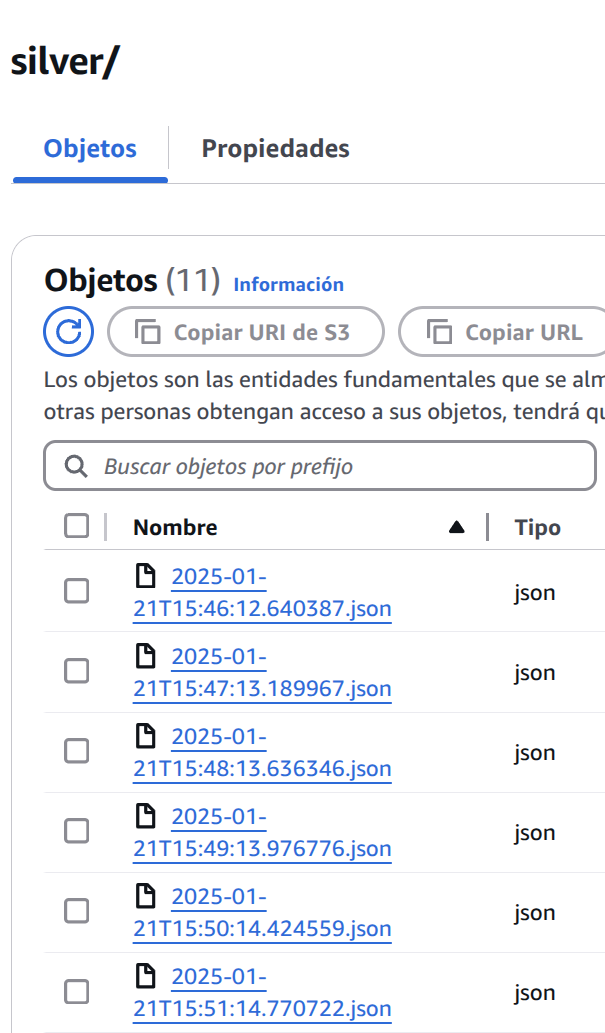
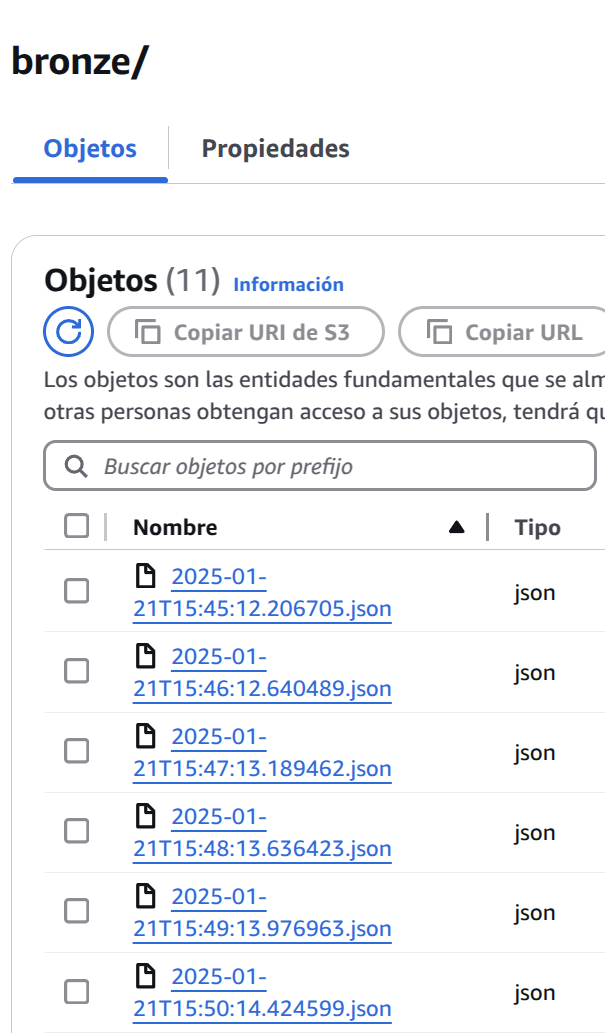
Realiza el caso de uso 3











### EJERCICIO 8

Realiza el caso de uso 3, pero separando el consumidorSilverProductorGold en tres consumidores diferentes, uno que consuma y guarde en S3 (consumidorSilverS3.py), otro que consuma y guarde en MongoDB (consumidorSilverMongoDB.py) y el tercero que consuma, agrupe mensajes y produzca el mensaje al topic gold (consumidorSilverGroupProducerGold.py).

*Script ConsumidorSilverS3.py:*

from kafka import KafkaConsumer

from datetime import datetime

import boto3

from json import loads

# Conexión con S3

s3r = boto3.resource('s3', region\_name='us-east-1')

bucket = s3r.Bucket('kafka-gf')

# Función para procesar el mensaje

def esquema\_json(dct):

result = {}

if ('fecha' in dct):

result["fecha"] = datetime.fromisoformat(dct["fecha"])

if 'temp' in dct:

result["temp"] = float(dct["temp"])

if 'humedad' in dct:

result["humedad"] = float(dct["humedad"])

if 'ciudad' in dct:

result["ciudad"] = dct["ciudad"]

return result

# Paso 1 - Consumir Silver

consumer = KafkaConsumer(

'iabd-aemet-silver',

enable\_auto\_commit=True,

group\_id='iabd-ejercicio8-1',

bootstrap\_servers=['nodo1:9092'])

for m in consumer:

resp\_json = m.value

doc\_json = loads(resp\_json, object\_hook=esquema\_json)

# Paso 2 - Metemos en S3 cada mensaje

nom\_fichero = "silver/" + datetime.now().isoformat() + ".json"

bucket.put\_object(Key=nom\_fichero, Body=resp\_json)

*Script ConsumidorSilverMongoDB.py :*

from kafka import KafkaConsumer

from datetime import datetime

from pymongo import MongoClient

from json import loads

# Conexión con MongoDB

clienteMongo = MongoClient('mongodb://localhost:27017')

# Función para procesar el mensaje

def esquema\_json(dct):

result = {}

if ('fecha' in dct):

result["fecha"] = datetime.fromisoformat(dct["fecha"])

if 'temp' in dct:

result["temp"] = float(dct["temp"])

if 'humedad' in dct:

result["humedad"] = float(dct["humedad"])

if 'ciudad' in dct:

result["ciudad"] = dct["ciudad"]

return result

# Paso 1 - Consumir Silver

consumer = KafkaConsumer(

'iabd-aemet-silver',

enable\_auto\_commit=True,

group\_id='iabd-ejercicio8-2',

bootstrap\_servers=['nodo1:9092'])

for m in consumer:

resp\_json = m.value

doc\_json = loads(resp\_json, object\_hook=esquema\_json)

# Paso 2 - Lo insertamos en MongoDB

colcaso3 = clienteMongo.iabd.caso3

colcaso3.insert\_one(doc\_json)

*Script ConsumidorSilverGroupProducerGold.py:*

from kafka import KafkaConsumer, KafkaProducer

import pandas as pd

from datetime import datetime

from json import loads, dumps

# Función para procesar el mensaje

def esquema\_json(dct):

result = {}

if ('fecha' in dct):

result["fecha"] = datetime.fromisoformat(dct["fecha"])

if 'temp' in dct:

result["temp"] = float(dct["temp"])

if 'humedad' in dct:

result["humedad"] = float(dct["humedad"])

if 'ciudad' in dct:

result["ciudad"] = dct["ciudad"]

return result

# Paso 1 - Consumir Silver

consumer = KafkaConsumer(

'iabd-aemet-silver',

enable\_auto\_commit=True,

group\_id='iabd-ejercicio8-3',

bootstrap\_servers=['nodo1:9092'])

# Paso 5 - Producir Gold

producer = KafkaProducer(

bootstrap\_servers=['nodo1:9092'])

cantidad = 1

mensajes = []

for m in consumer:

resp\_json = m.value

doc\_json = loads(resp\_json, object\_hook=esquema\_json)

mensajes.append(doc\_json)

# Guardamos 10 mensajes

cantidad += 1

if cantidad == 11:

# Paso 4 - realizamos el cálculo

# Usamos pandas

pd\_mensajes = pd.DataFrame(mensajes)

pd\_mensajes.temp = pd\_mensajes['temp'].astype('float')

pd\_mensajes.humedad = pd\_mensajes['humedad'].astype('float')

pd\_agg = pd\_mensajes.groupby("ciudad").agg({"fecha":"max", "temp":"mean", "humedad":"mean"})

pd\_agg.reset\_index(inplace=True)

json\_gold = pd\_agg.to\_json(orient="records")

print(json\_gold)

# Paso 5 - producimos el mensaje a Kafka

producer.send("iabd-aemet-gold", value=dumps(json\_gold).encode('utf-8'))

# vaciamos la lista e inicializamos el contador

mensajes = []

cantidad = 1

*python3 consumidorBronze.py*

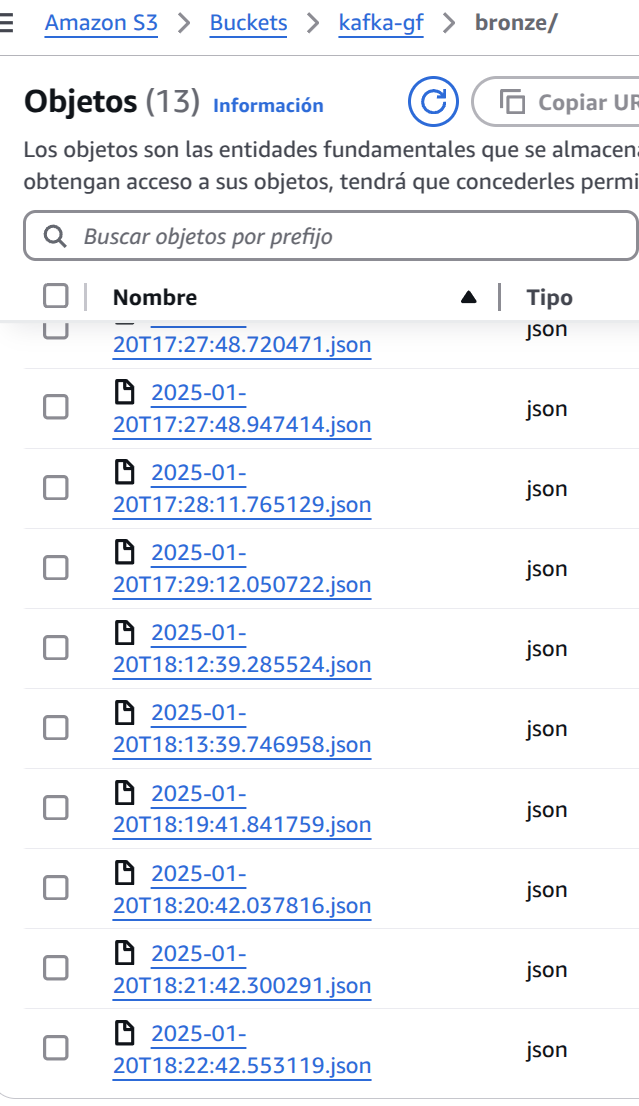
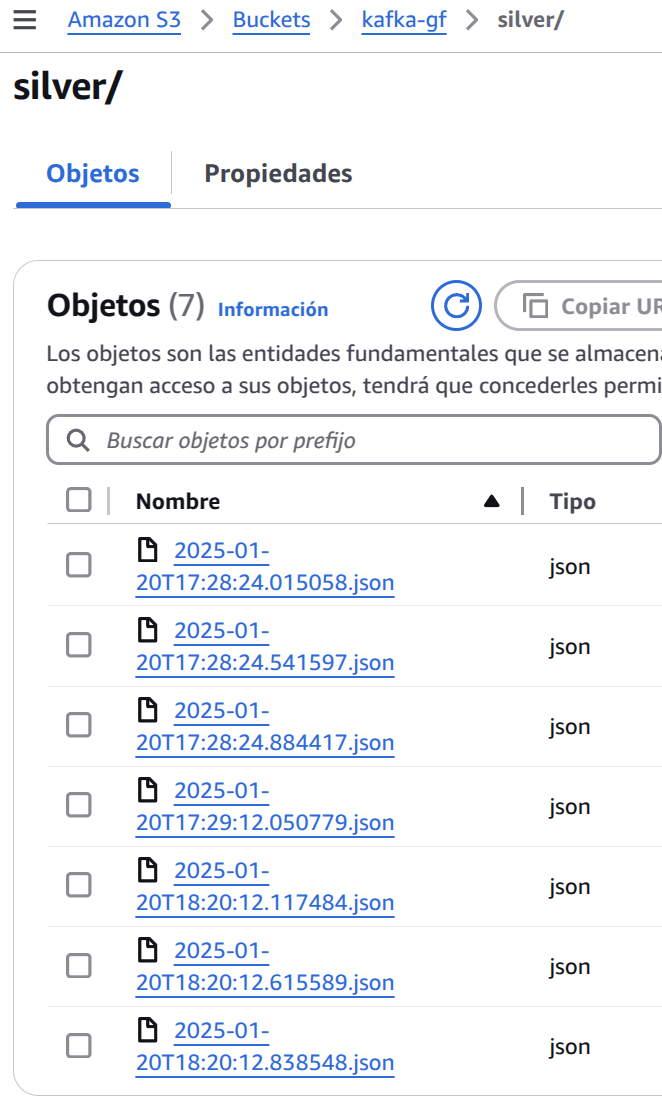
*python3 productorBronze.py*

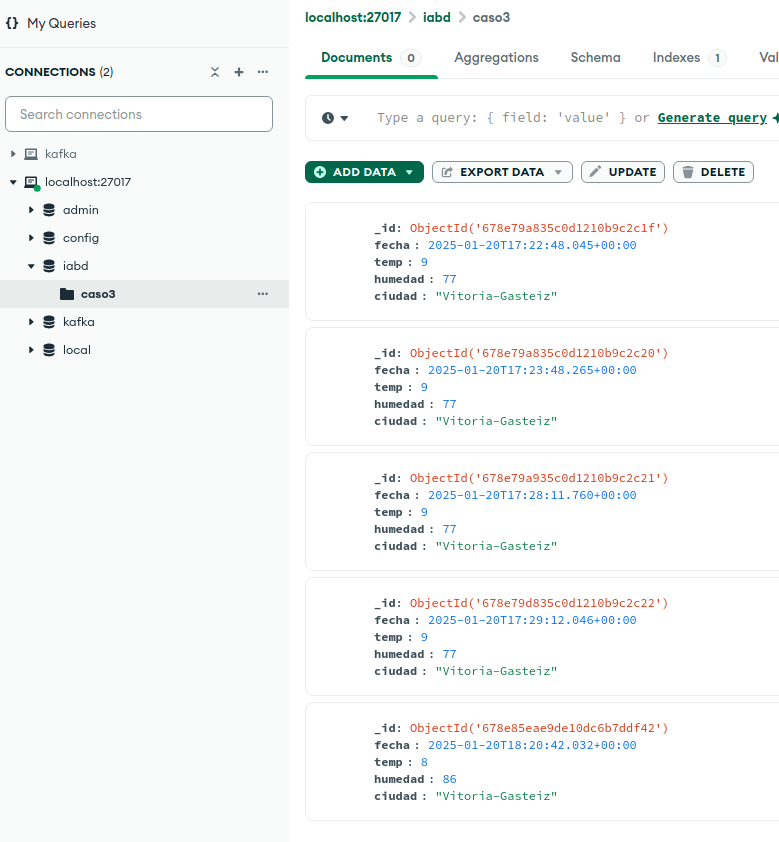
*python3 consumidorSilverS3.py*

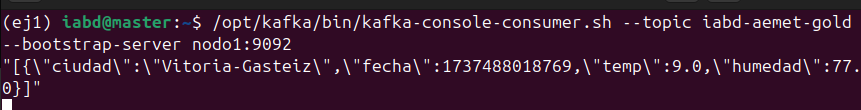
*python3 consumidorSilverMongoDB.py*

*python3 consumidorSilverGroupProducerGold.py*









Es el ultimo hora: 18:20 y se ve en las captura del S3 que se crean los archivos desde las 18:12 hasta 18.22

### EJERCICIO 9

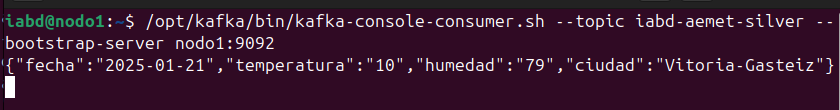
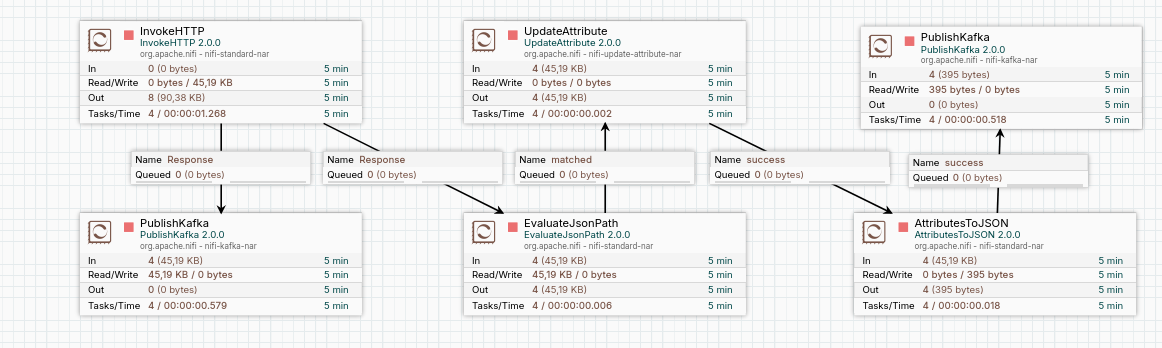
Antes de hacer el ejercicio tienes que leer el documento BGD\_2\_7\_Nifi.docx (Big Data Aplicado UD2) y hacer los ejercicios BGD\_2\_8\_Ejercicios.docx.

Haz con Nifi del Caso 3, la parte de crear el productor Bronze y el productor Silver, mediante un grupo de procesadores que llamaremos NifiKafka que contendrá un flujo similar a:

Diagrama

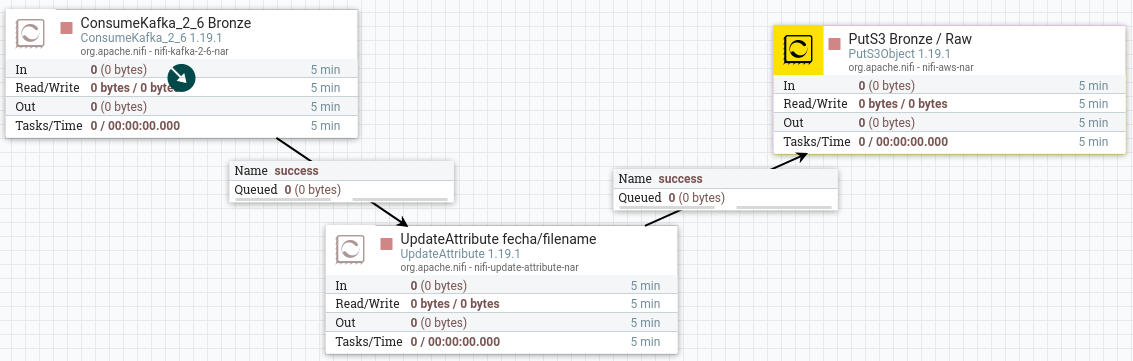
Descripción generada automáticamente con confianza media

Comprueba que funciona correctamente y después de acabar crea un grupo y descarga el flujo en formato json.

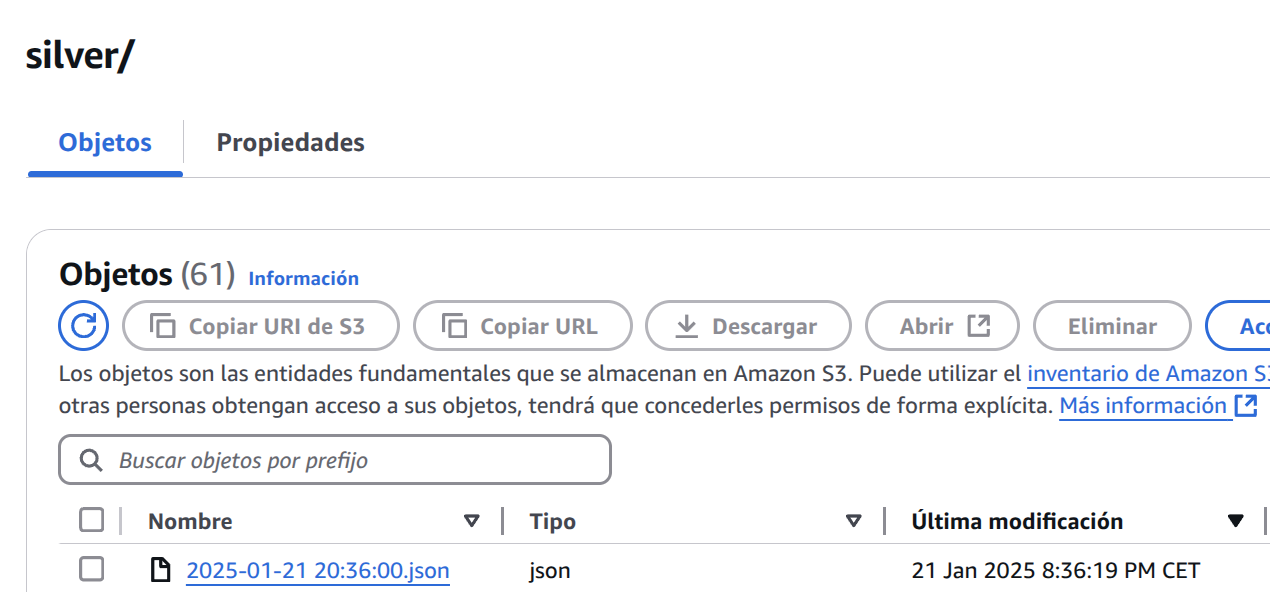
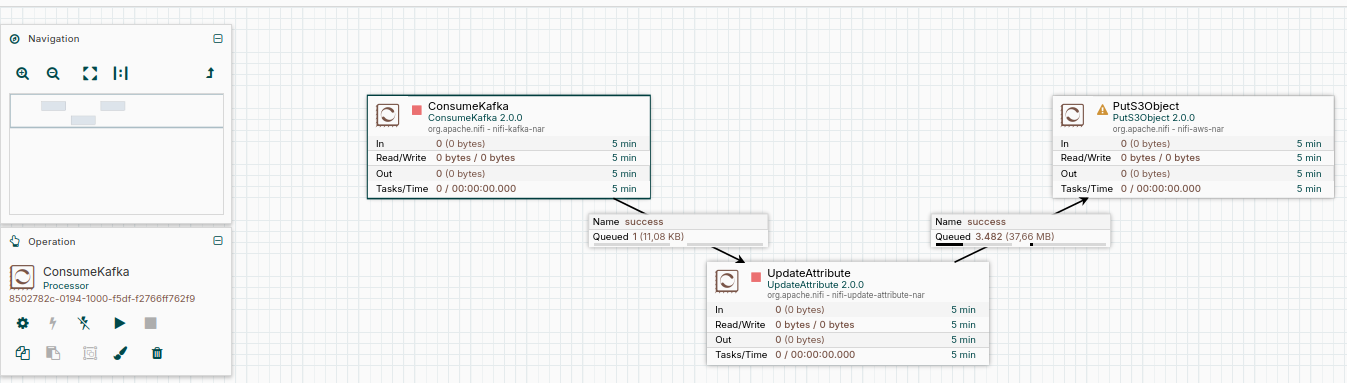


### EJERCICIO 10

Siguiendo con el ejercicio anterior ahora con Nifi hay que recuperar los mensajes del topic iabd-aemet-silver y colocarlos en S3, mediante un grupo de procesadores que llamaremos kafka\_consumer\_bronze que contendrá un flujo similar a:



Comprueba que funciona correctamente y después de acabar crea un grupo y descarga el flujo en formato json (a la hora de guardar el S3 nos va a dar un error de credenciales porque PutS3Object no soporta las sesiones temporales que da AWS Academy).



### EJERCICIO 11

Crea un conector (Kafka Connect) para que consuma los mensajes del topic iabd-aemet-bronze y los inserte en MongoDB de forma automática.

Para ello, puedes utilizar el [MongoDB Connector (Source and Sink)](https://www.confluent.io/hub/mongodb/kafka-connect-mongodb) y consultar su [documentación](https://www.mongodb.com/docs/kafka-connector/current/) y un ejemplo de [fichero de configuración](https://github.com/mongodb/mongo-kafka/blob/master/config/MongoSinkConnector.properties).

Ten en cuenta que para lanzar los dos conectores a la vez deberás lanzar Kafka Connect y pasarle todos los conectores como parámetros:

### EJERCICIO 12 (OPCIONAL)

Crear un clúster de Kafka con 4 brokers usando docker.