# 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

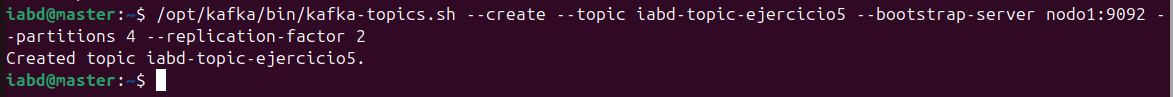
Realiza el caso de uso 2

### EJERCICIO 5

Se pide:

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

/opt/kafka/bin/kafka-topics.sh --create --topic iabd-topic-ejercicio5 --bootstrap-server nodo1:9092 --partitions 4 --replication-factor 2

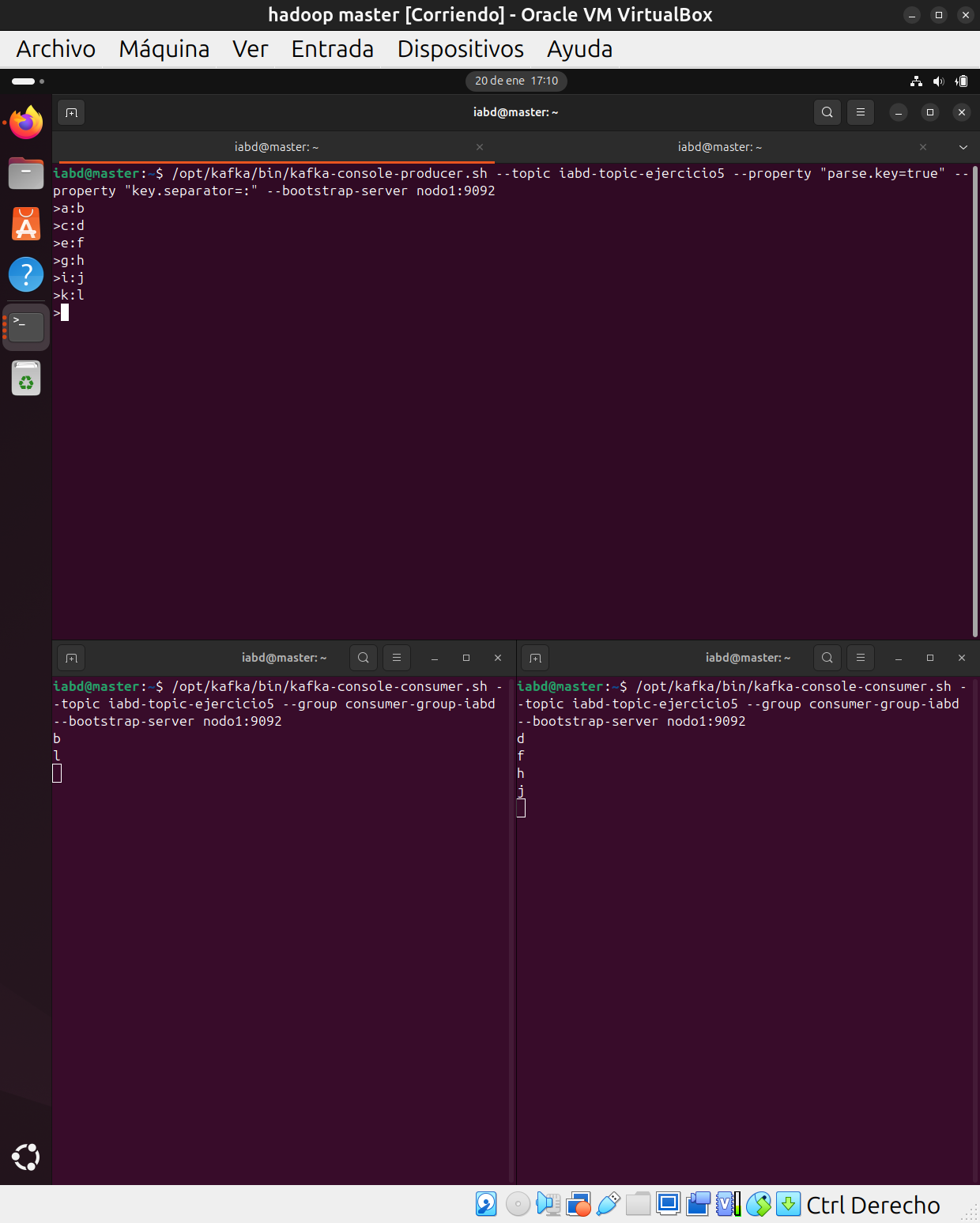


1. A continuación, lanza dos consumidores que pertenezcan al grupo de consumidores consumer-group-iabd.

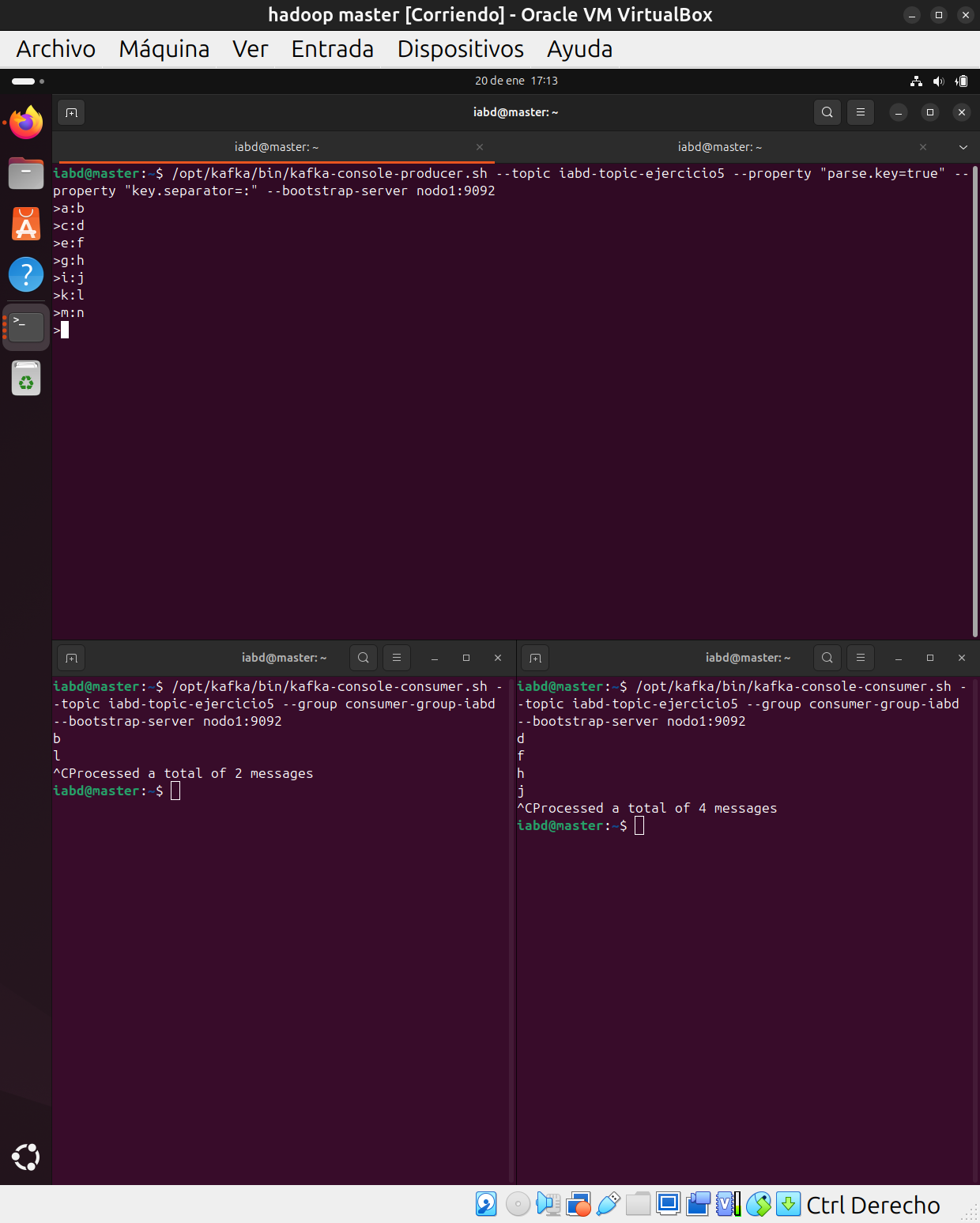
/opt/kafka/bin/kafka-console-consumer.sh --topic iabd-topic-ejercicio5 --group consumer-group-iabd --bootstrap-server nodo1:9092

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

/opt/kafka/bin/kafka-console-producer.sh --topic iabd-topic-ejercicio5 --property "parse.key=true" --property "key.separator=:" --bootstrap-server nodo1:9092

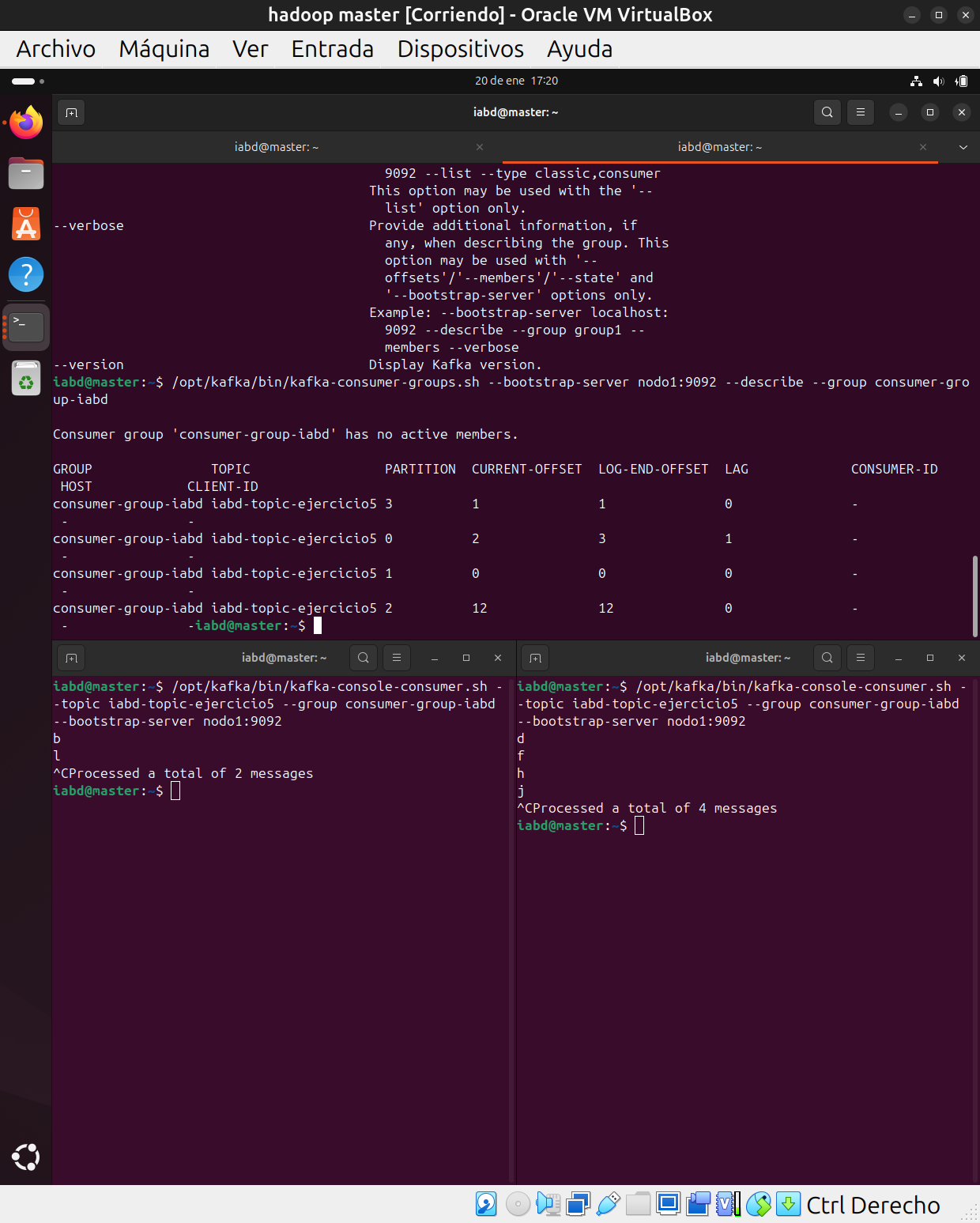


1. Detén ambos consumidores.
2. Envía un nuevo mensaje.



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

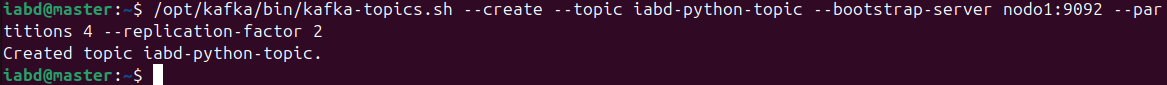
/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server nodo1:9092 --describe --group consumer-group-iabd



Podemos ver que en uno de los grupos, el lag es de 1 lo que significa que queda algún mensaje pendiente.

### 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:

personas.py

from kafka import KafkaProducer

from faker import Faker

import json

import time

# Crear un objeto Faker para generar datos ficticios

fake = Faker()

# Crear un productor Kafka

producer = KafkaProducer(

bootstrap\_servers='nodo1:9092', # Cambia esta URL si es necesario

value\_serializer=lambda v: json.dumps(v).encode('utf-8') # Serialización de los datos a JSON

)

# Función para crear un diccionario con los datos de una persona

def generar\_persona():

return {

'nombre': fake.first\_name(),

'apellido': fake.last\_name(),

'email': fake.email(),

'direccion': fake.address(),

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

'fecha\_nacimiento': fake.date\_of\_birth().isoformat()

}

# Enviar datos al topic cada 10 segundos

try:

while True:

persona = generar\_persona() # Generar una persona

print(f"Enviando datos: {persona}")

producer.send('iabd-python-topic', persona) # Enviar los datos al topic

time.sleep(10) # Esperar 10 segundos antes de enviar la siguiente persona

except KeyboardInterrupt:

print("Interrumpido por el usuario.")

finally:

producer.close()

1. un consumidor que reciba las personas y, mediante pymongo, las inserte en mongodb en una colección llamada kafka\_personas.

Consumidor.py

from kafka import KafkaConsumer

import pymongo

import json

# Conectar a MongoDB

client = pymongo.MongoClient('mongodb://localhost:27017/') # Cambia la URL si es necesario

db = client['personas\_db'] # Base de datos llamada personas\_db

collection = db['kafka\_personas'] # Colección llamada kafka\_personas

# Crear un consumidor Kafka

consumer = KafkaConsumer(

'iabd-python-topic', # Nombre del topic a consumir

bootstrap\_servers='nodo1:9092', # Dirección del broker Kafka

group\_id='personas-consumer-group', # Grupo de consumidores

value\_deserializer=lambda x: json.loads(x.decode('utf-8')) # Deserializar los mensajes de JSON

)

# Función para insertar persona en MongoDB

def insertar\_en\_mongodb(persona):

try:

collection.insert\_one(persona) # Insertar un documento en la colección

print(f"Persona insertada: {persona}")

except Exception as e:

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

# Consumir mensajes del topic y procesarlos

try:

for mensaje in consumer:

persona = mensaje.value # Extraer el mensaje

insertar\_en\_mongodb(persona) # Insertar la persona en MongoDB

except KeyboardInterrupt:

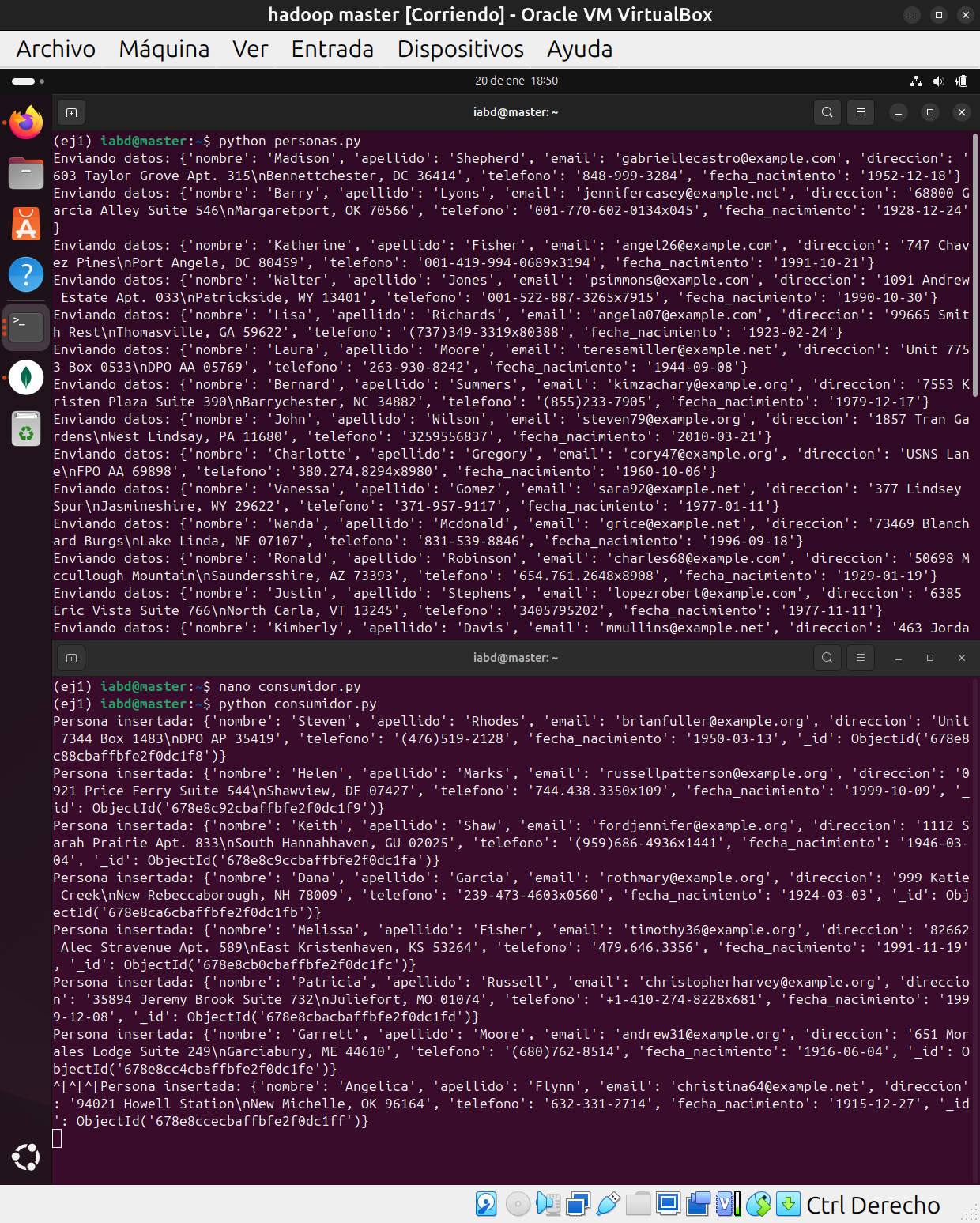
print("Consumo interrumpido por el usuario.")

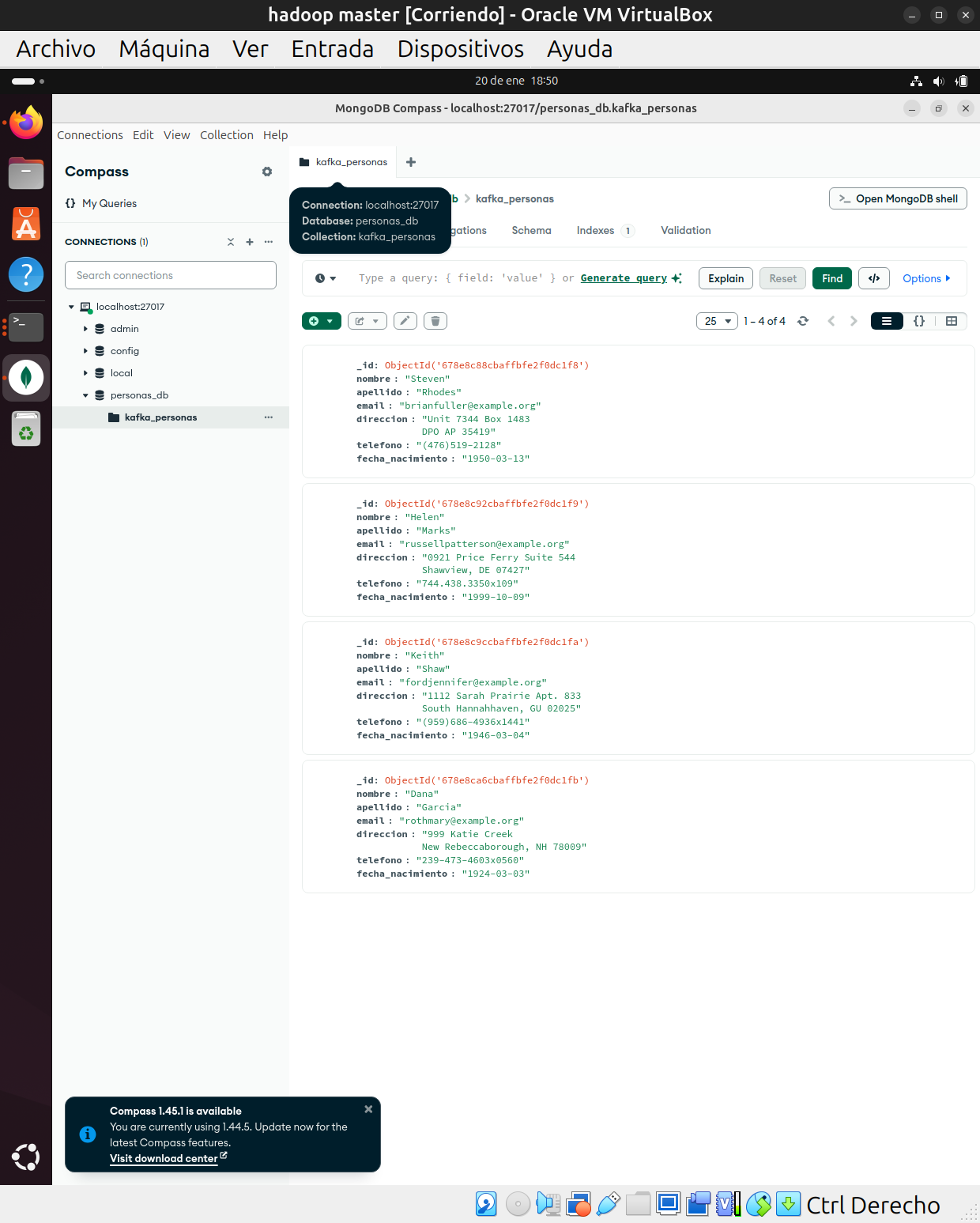
finally:

consumer.close()

client.close()

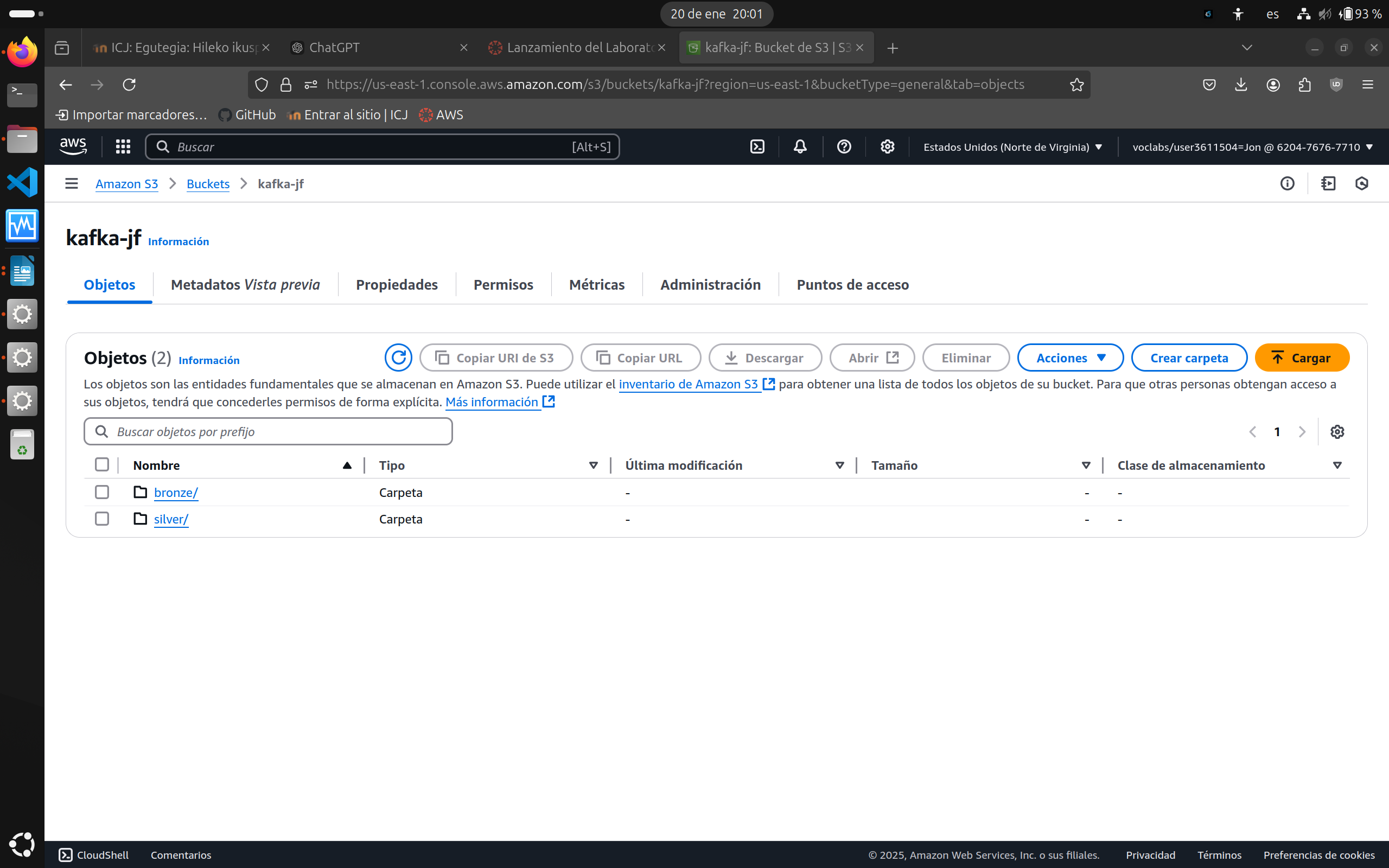
Comprueba que funciona correctamente.

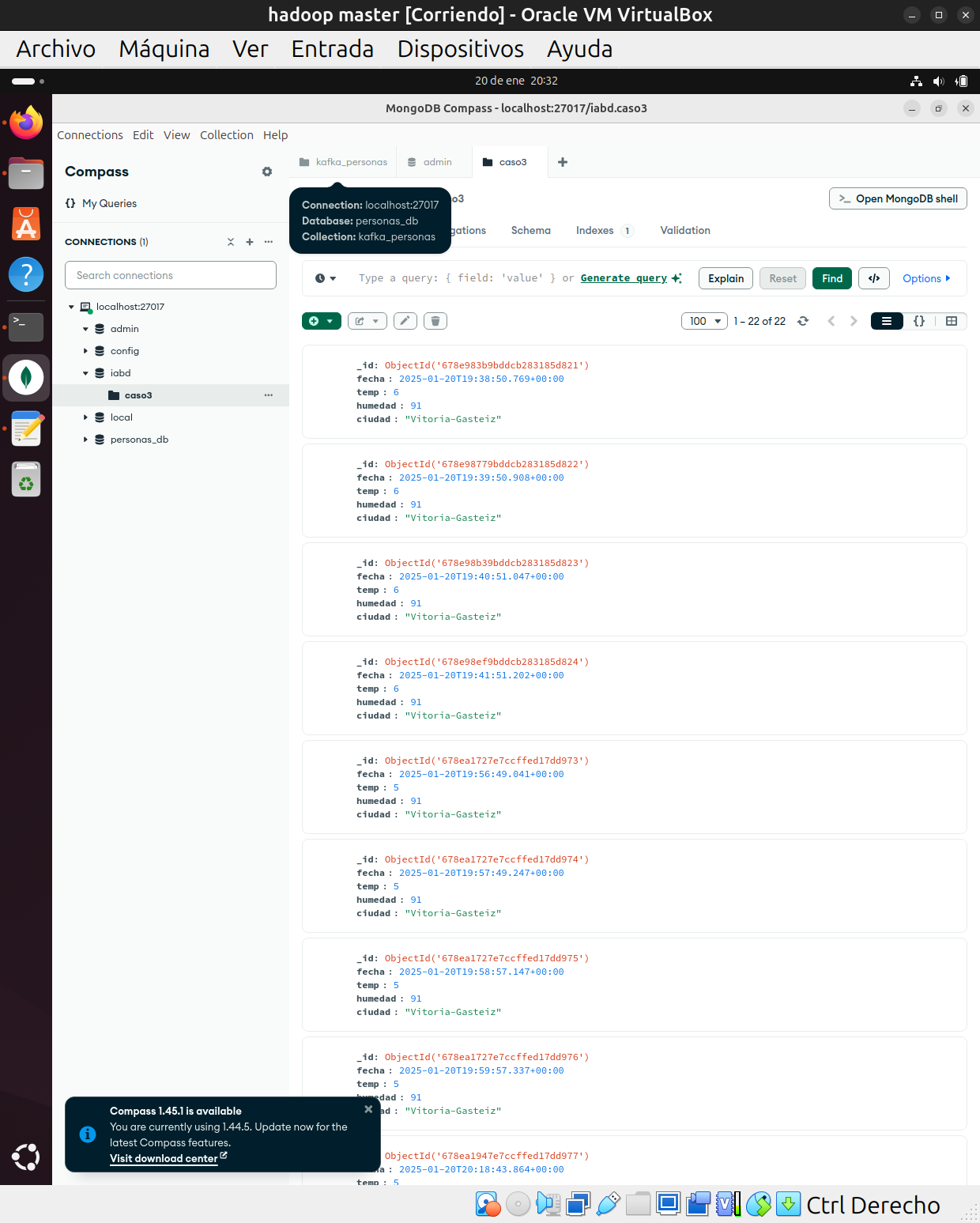


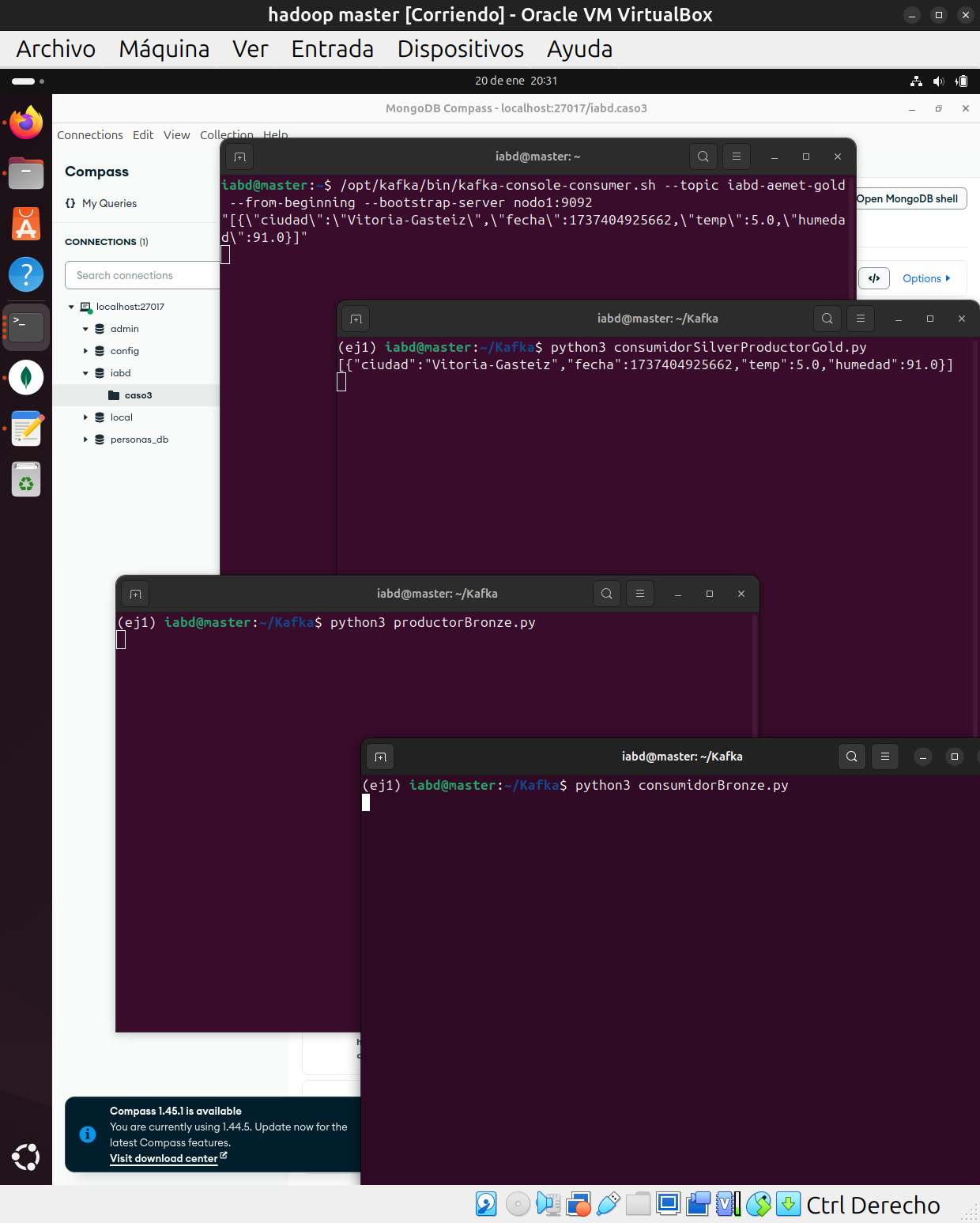


### EJERCICIO 7

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).

### 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:

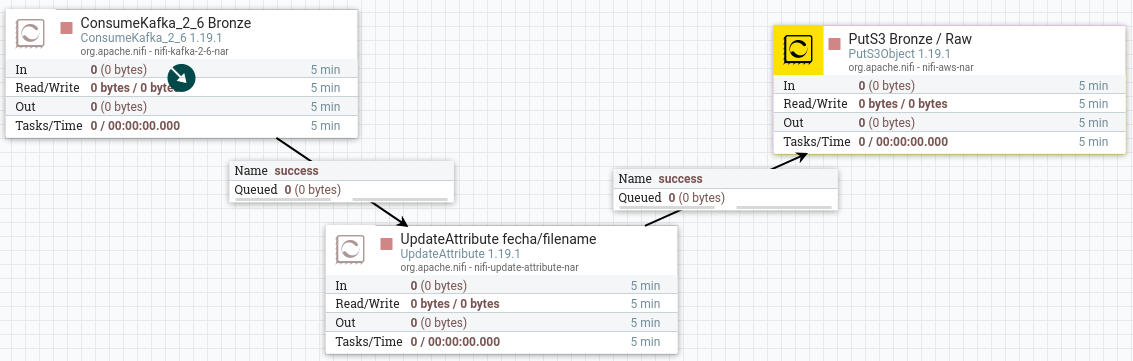
Diagrama

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.