PRACTICA KAFKA Y LABS

LAB 1

Instalar y arrancar Kafka

Descargamos Apache Kafka y lo descomprimimos en el directorio de home.

```
[umaster@ibmuamdocker ~]$ tar xvzf ~/Downloads/kafka_2.13-3.7.0.tgz
kafka_2.13-3.7.0/
kafka_2.13-3.7.0/LICENSE
kafka_2.13-3.7.0/NOTICE
kafka_2.13-3.7.0/bin/
kafka_2.13-3.7.0/bin/kafka-delete-records.sh
kafka_2.13-3.7.0/bin/trogdor.sh
```

En el directorio de binarios de Apache Kafka arrancamos Zookeeper.

```
[umaster@ibmuamdocker bin]$ ./zookeeper-server-start.sh ../config/zookeeper.properties
[2024-04-22 13:49:16,849] INFO Reading configuration from: ../config/zookeeper.properties (org.apache.z
ookeeper.server.quorum.QuorumPeerConfig)
[2024-04-22 13:49:16,850] WARN ../config/zookeeper.properties is relative. Prepend ./ to indicate that
you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-04-22 13:49:16,853] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.Qu
orumPeerConfig)
[2024-04-22 13:49:16,853] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPe
erConfig)
```

Abrimos otra terminal y, desde el mismo directorio de binarios de Kafka, arrancamos el broker.

```
umaster@ibmuamdocker:~/kafka_2.13-3.7.0/bin    umaster@ibmuamdocker:~/kafka_2.13-3.7.0/bin
[umaster@ibmuamdocker bin]$ ./kafka-server-start.sh ../config/server.properties
[2024-04-22 13:51:37,641] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jCont
rollerRegistration$)
[2024-04-22 13:51:37,856] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable cl
ient-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2024-04-22 13:51:37,923] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.u
tils.LoggingSignalHandler)
[2024-04-22 13:51:37,924] INFO starting (kafka.server.KafkaServer)
[2024-04-22 13:51:37,924] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2024-04-22 13:51:37,933] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2
181. (kafka.zookeeper.ZooKeeperClient)
```

Productor y Consumidor

Creamos un Topic con Productor y Consumidor y lo validamos.
(Ejecuto el comando con la ruta completa porque no funciona con bin/kafka-topics.sh.)

```
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/kafka-topics.sh --bootstrap-server local
host:9092 --replication-factor 1 --partitions 1 --create --topic test
Created topic test.
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/kafka-topics.sh --bootstrap-server local
host:9092 --list
test
[umaster@ibmuamdocker bin]$
```

Producimos y consumimos mensajes

```
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/kafka-console-producer.sh --brok
er-list localhost:9092 --topic test
>message00
>message01
>^[[A^[[Bmessage02
```

```
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/kafka-console-consumer.sh --boot
strap-server localhost:9092 --topic test --from-beginning
message00
message01
message02
^CProcessed a total of 3 messages
```

Almacenamiento de los mensajes

Revisamos donde están almacenados los mensajes:

```
[umaster@ibmuamdocker bin]$ tree /tmp/kafka-logs
/tmp/kafka-logs
├── cleaner-offset-checkpoint
├── __consumer_offsets-0
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── __consumer_offsets-1
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── __consumer_offsets-10
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── __consumer_offsets-11
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── __consumer_offsets-12
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
```

```
├── __consumer_offsets-8
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── __consumer_offsets-9
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── log-start-offset-checkpoint
├── meta.properties
├── recovery-point-offset-checkpoint
├── replication-offset-checkpoint
└── test-0
    ├── 00000000000000000000.index
    ├── 00000000000000000000.log
    ├── 00000000000000000000.timeindex
    ├── leader-epoch-checkpoint
    └── partition.metadata

51 directories, 260 files
[umaster@ibmuamdocker bin]$
```

```
[umaster@ibmuamdocker bin]$ cd /tmp/kafka-logs/test-0/
[umaster@ibmuamdocker test-0]$ od -bc 00000000000000000000.log | more
0000000 000 000 000 000 000 000 000 000 000 000 000 101 000 000 000 000
         \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0   A  \0  \0  \0  \0
0000020 002 044 125 245 112 000 000 000 000 000 000 000 000 001 217 005
        002   $   U 245   J  \0  \0  \0  \0  \0  \0  \0  \0 001 217 005
0000040 343 344 244 000 000 001 217 005 343 344 244 000 000 000 000 000
        343 344 244  \0  \0 001 217 005 343 344 244  \0  \0  \0  \0  \0
0000060 000 000 000 000 000 000 000 000 000 000 000 000 001 036 000 000
         \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0 001 036  \0  \0
0000100 000 001 022 155 145 163 163 141 147 145 060 060 000 000 000 000
         \0 001 022   m   e   s   s   a   g   e   0   0  \0  \0  \0  \0
0000120 000 000 000 000 001 000 000 000 101 000 000 000 000 002 070 176
         \0  \0  \0  \0 001  \0  \0  \0   A  \0  \0  \0  \0 002   8   ~
0000140 160 156 000 000 000 000 000 000 000 000 001 217 005 343 375 052
          p   n  \0  \0  \0  \0  \0  \0  \0  \0 001 217 005 343 375   *
0000160 000 000 001 217 005 343 375 052 000 000 000 000 000 000 000 000
         \0  \0 001 217 005 343 375   *  \0  \0  \0  \0  \0  \0  \0  \0
0000200 000 000 000 000 000 001 000 000 000 001 036 000 000 000 001 022
```

```
0000200 000 000 000 000 000 001 000 000 000 001 036 000 000 000 001 022
         \0  \0  \0  \0  \0 001  \0  \0  \0 001 036  \0  \0  \0 001 022
0000220 155 145 163 163 141 147 145 060 061 000 000 000 000 000 000 000
          m   e   s   s   a   g   e   0   1  \0  \0  \0  \0  \0  \0  \0
0000240 000 002 000 000 000 107 000 000 000 000 002 317 374 357 150 000
         \0 002  \0  \0  \0   G  \0  \0  \0  \0 002 317 374 357   h  \0
0000260 000 000 000 000 000 000 000 001 217 005 344 035 213 000 000 001
         \0  \0  \0  \0  \0  \0  \0 001 217 005 344 035 213  \0  \0 001
0000300 217 005 344 035 213 000 000 000 000 000 000 000 000 000 000 000
        217 005 344 035 213  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000320 000 000 002 000 000 000 001 052 000 000 000 001 036 033 133 101
         \0  \0 002  \0  \0  \0 001   *  \0  \0  \0 001 036 033   [   A
0000340 033 133 102 155 145 163 163 141 147 145 060 062 000
--More--
```

Borrar Topics

Borramos el topic y arrancamos el bróker y comprobamos los topics disponibles para validar.

```
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/kafka-topics.sh --bootstrap-server
localhost:9092 --delete --topic test
```

```
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/kafka-topics.sh --list --bootstrap-server localhost:9092
__consumer_offsets
```

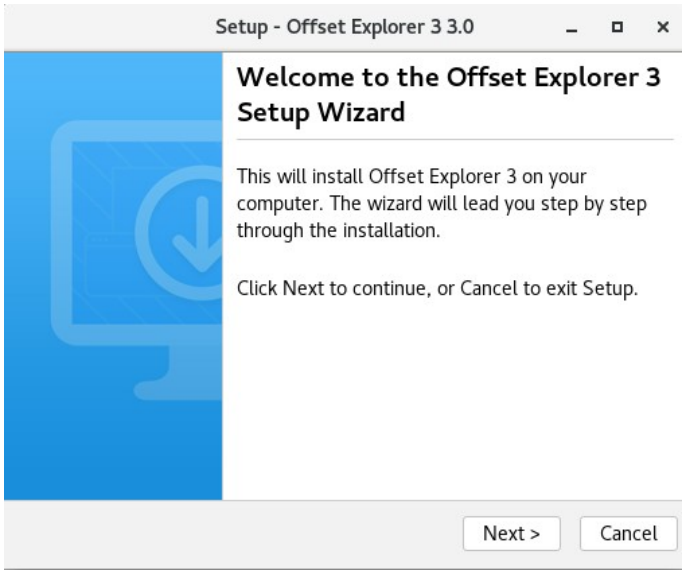
Observamos que no existe un topic llamado "test" disponible.

LAB 2

Instalar Offset Explorer

Descargamos Offset Explorer y lo ejecutamos:

```
[umaster@ibmuamdocker ~]$ sh ~/Downloads/offsetexplorer.sh
Starting Installer ...
```

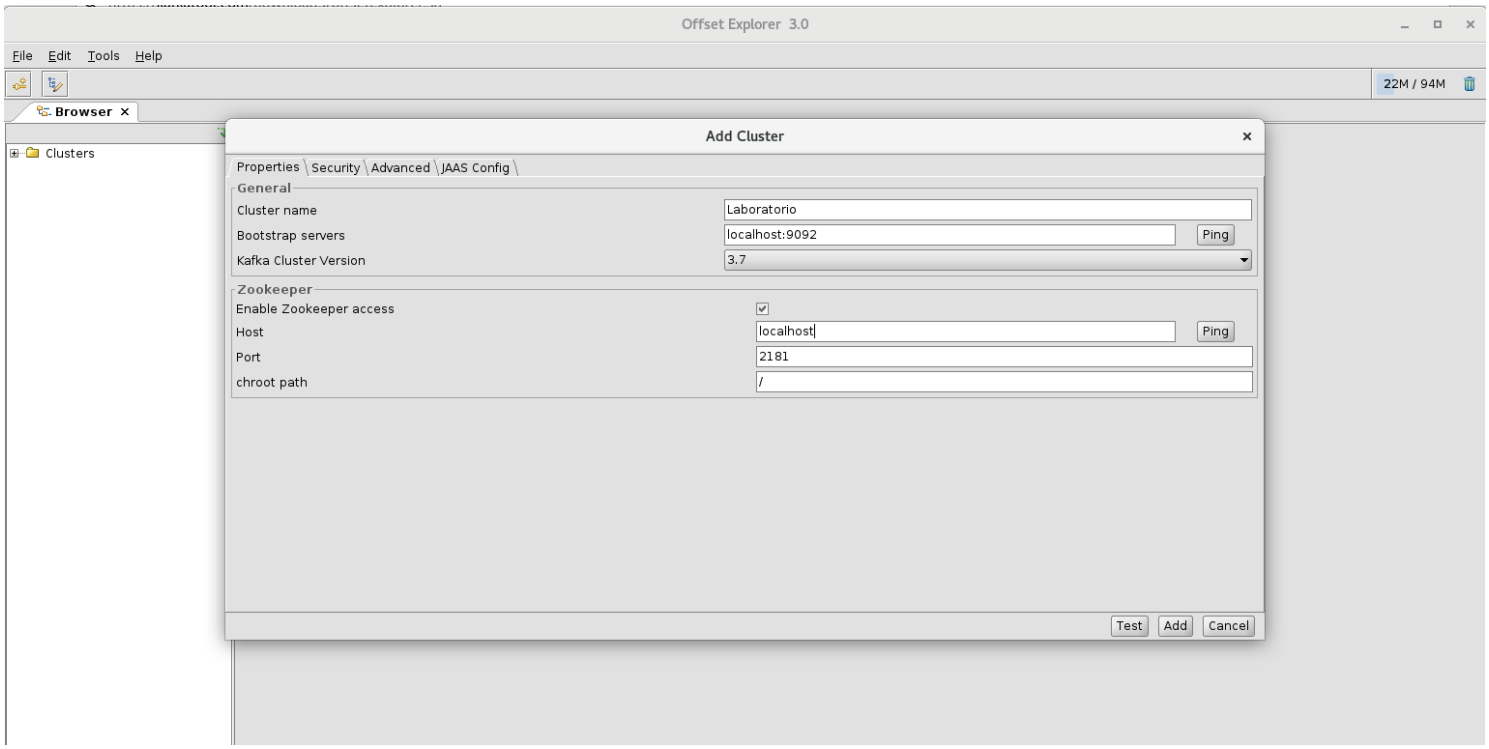


Hemos realizado la instalación, aceptando acuerdo de licencia, directorio por defecto de la instalación, directorio para la creación de los enlaces simbólicos.

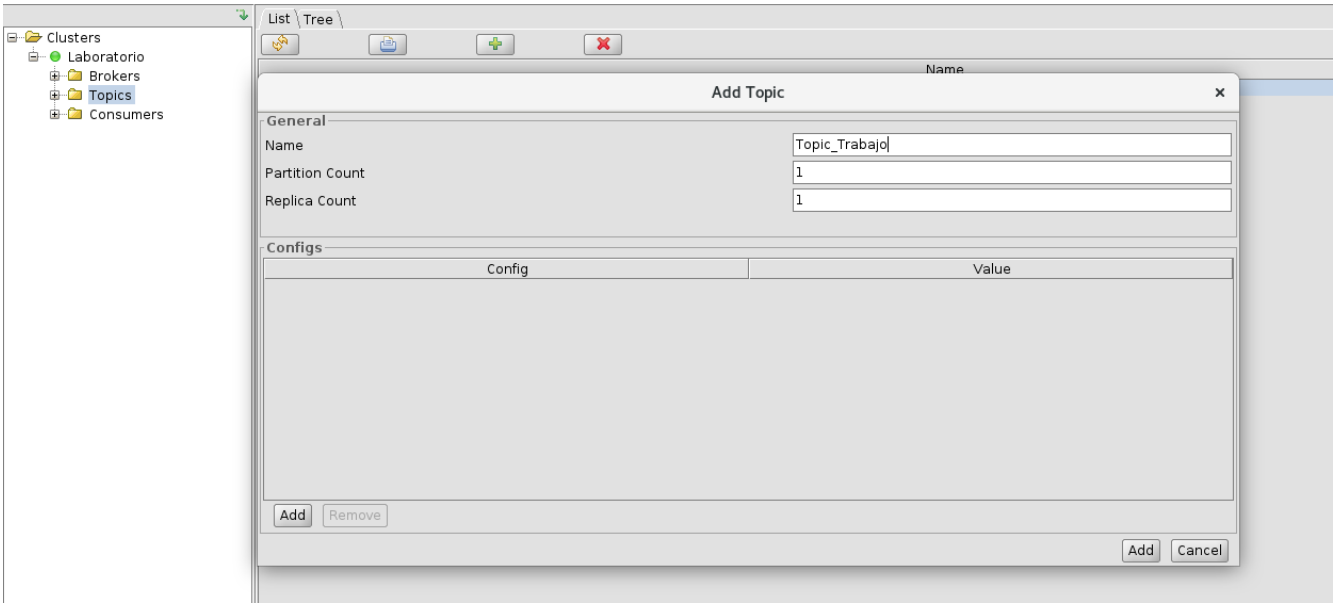
Configurar y comprobar

```
[umaster@ibmuamdocker ~]$ cd ~/offsetexplorer3
[umaster@ibmuamdocker offsetexplorer3]$ ./offsetexplorer
```

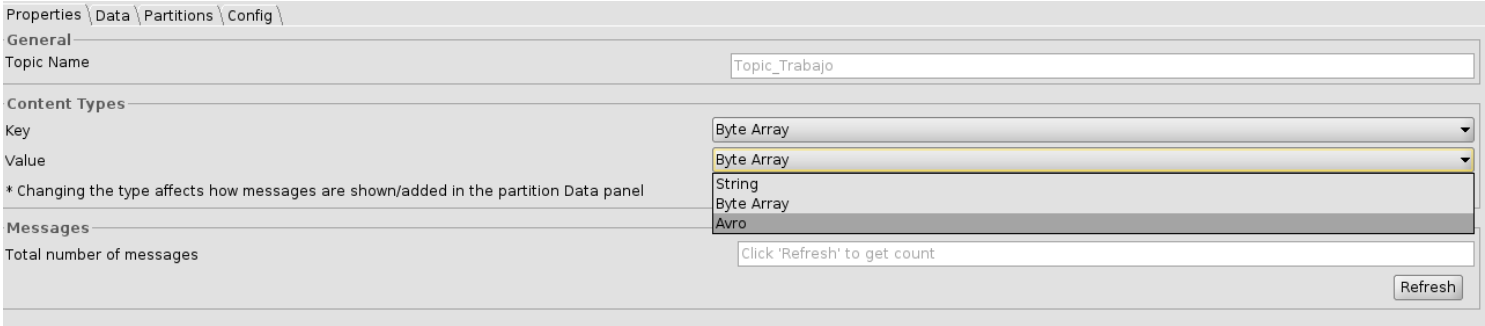
Creamos una conexión a nuestro cluster



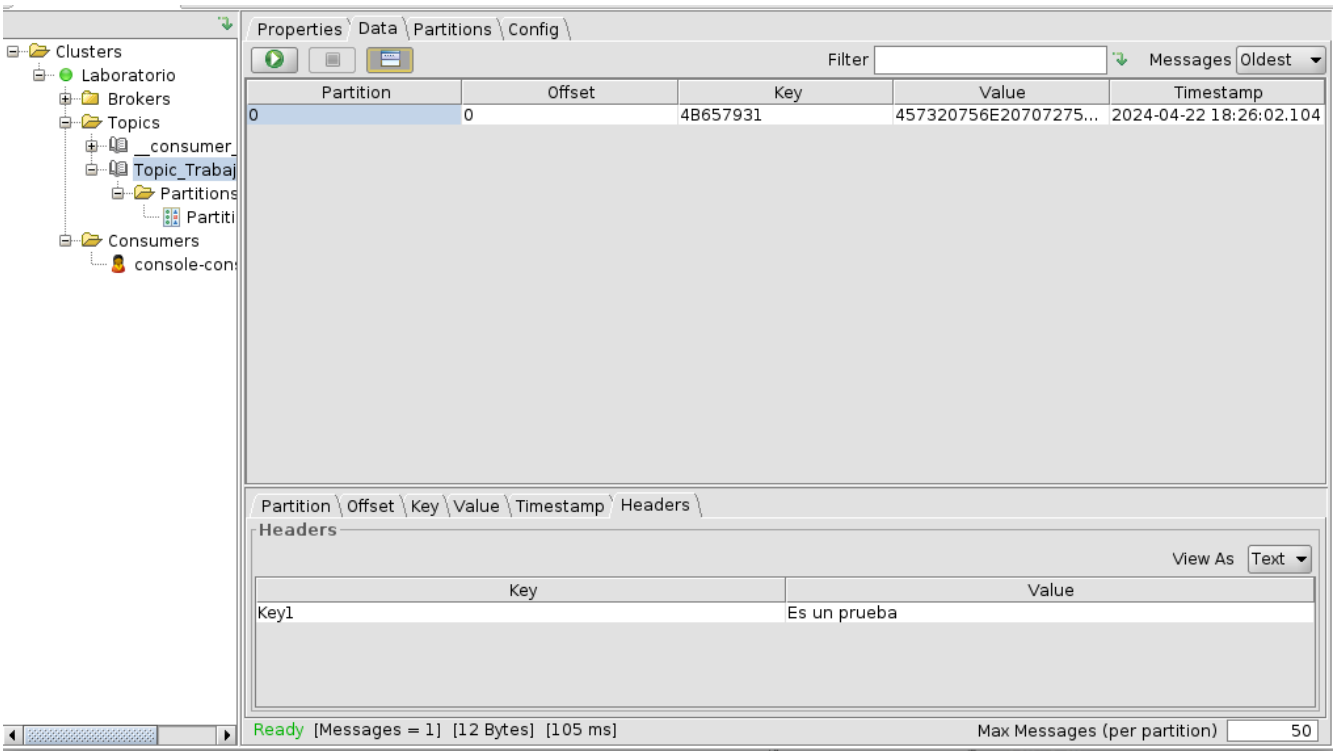
Añadimos un topic



Hemos establecido que la clave y el valor se representarán como un “Byte Array”.



Añadimos un mensaje.
Observamos los mensajes del topic:



LAB 3

Usando Kafka Connect

Editamos el fichero config/connect-standalone.properties

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ cd config/
[umaster@ibmuamdocker config]$ nano connect-standalone.properties
[umaster@ibmuamdocker config]$
```

```
  GNU nano 2.3.1      File: connect-standalone.properties        Modified

# (connectors, converters, transformations). The list should consist of to$
# any combination of:
# a) directories immediately containing jars with plugins and their depend$
# b) uber-jars with plugins and their dependencies
# c) directories immediately containing the package directory structure of$
# Note: symlinks will be followed to discover dependencies or plugins.
# Examples:
# plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/co$
#plugin.path=
plugin.path=/home/umaster/kafka_2.13-3.7.0/libs/connect-file-3.7.0.jar
```

Creamos el fichero test.txt e insertamos algunos valores

```
[umaster@ibmuamdocker config]$ cd ~/kafka_2.13-3.7.0/bin
[umaster@ibmuamdocker bin]$ touch test.txt
[umaster@ibmuamdocker bin]$ echo "Hola" >> test.txt
[umaster@ibmuamdocker bin]$ echo "Que pasa" >> test.txt
```

Observamos que el archivo test.txt existe y contiene las líneas: “Hola” y “Que pasa”

```
[umaster@ibmuamdocker bin]$ cat test.txt
Hola
Que pasa
```

Arrancamos los conectores

```
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/connect-standalone.sh /home/umaster/kafka_2.13-3.7.0/config/
connect-standalone.properties /home/umaster/kafka_2.13-3.7.0/config/connect-file-source.properties /home/umaster/kafka_2.13
-3.7.0/config/connect-file-sink.properties
```

Comprobamos el contenido del fichero test.sink.txt en el directorio bin

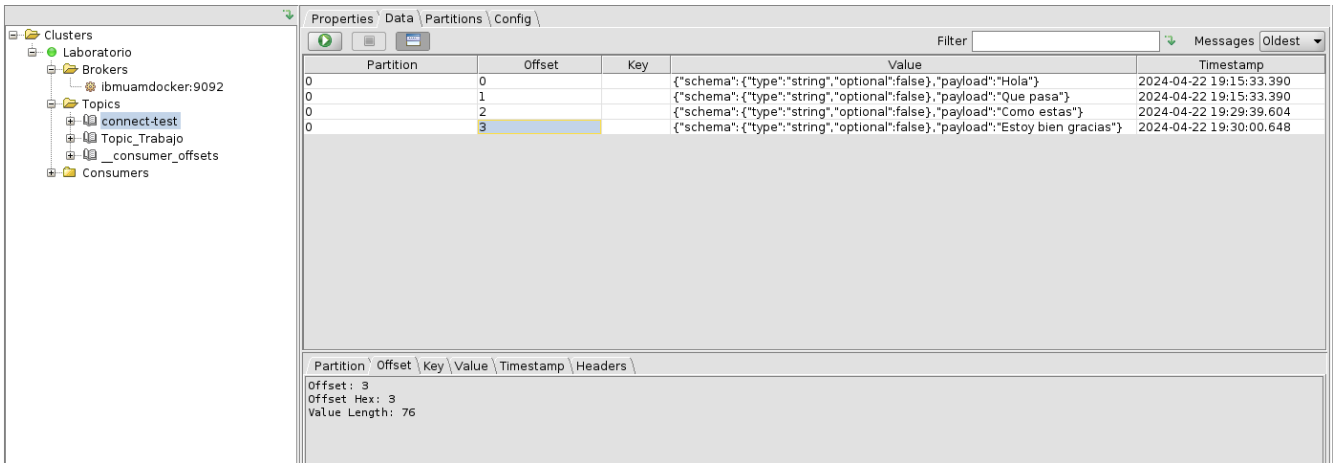
```
[umaster@ibmuamdocker bin]$ cat test.sink.txt
Hola
Que pasa
```

Comprobamos el contenido del topic que se ha creado

```
[umaster@ibmuamdocker bin]$ /home/umaster/kafka_2.13-3.7.0/bin/kafka-console-consu
mer.sh --bootstrap-server localhost:9092 --topic connect-test --from-beginning
{"schema":{"type":"string","optional":false},"payload":"Hola"}
{"schema":{"type":"string","optional":false},"payload":"Que pasa"}
```

Incluimos nuevas entradas en el fichero test.txt

```
[umaster@ibmuamdocker bin]$ touch test.txt
[umaster@ibmuamdocker bin]$ echo "Como estas" >> test.txt
[umaster@ibmuamdocker bin]$ echo "Estoy bien gracias" >> test.txt
[umaster@ibmuamdocker bin]$ cat test.txt
Hola
Que pasa
Como estas
Estoy bien gracias
[umaster@ibmuamdocker bin]$ cat test.sink.txt
Hola
Que pasa
Como estas
Estoy bien gracias
```



Hemos verificado que los mensajes se incluyen correctamente al topic “connect-test” y al fichero de destino test.sink.txt.

LAB 4

Desarrollo de clientes con python

Hemos actualizado pip e instalado los módulos necesarios

```
[umaster@ibmuamdocker ~]$ pip3 install --upgrade pip --user
Requirement already satisfied: pip in /usr/local/lib/python3.6/site-packages (21.3.1)
[umaster@ibmuamdocker ~]$ pip3 install kafka-Python --user
Requirement already satisfied: kafka-Python in ./.local/lib/python3.6/site-packages (2.0.2)
```

Adjunto los archivos de log llamados consumer.log y producer.log a esta práctica.

Plantilla de Productor
Hemos rellenado los codigos.

```
from kafka import KafkaProducer
from faker import Faker
from json import dumps, loads
import time
import logging
import sys
import random
```

```
#Logging
logging.basicConfig(format='%(asctime)s %(message)s',
                    datefmt='%d-%m-%Y %H:%M:%S',
                    filename='producer.log',
                    filemode='w')

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
#Faker
fake=Faker('es_ES')
```

```
#Configuración del entorno Kafka
topic_name = 'my-topic'
key = 'my-key'
bootstrap_servers = ['localhost:9092', 'localhost:9093']
logging.info('Arrancando...')
```

```
#Definiendo el conector
def connect_kafka_producer(bootstrap_servers):
    producer = None
    try:
        producer = KafkaProducer(bootstrap_servers=bootstrap_servers,
                                 key_serializer=str.encode(),
                                 value_serializer=lambda x: dumps(x).encode('utf-8'))
    except Exception as ex:
        logging.error('Exception while connecting producer')
        logging.error(ex)
    finally:
        return producer
```

```
#Definiendo la publicación del mensaje
def publish_message(producer, topic, key, value):
    try:
        producer.send(topic, key=key, value=value)
        producer.flush()
        logging.debug('Publicación del Mensaje OK')
    except Exception as ex:
        logging.error(value)
        logging.error('Excepción al publicar el mensaje')
        logging.error(ex)
    finally:
        return
```

```
producer = connect_kafka_producer(bootstrap_servers)
logging.info("Conexión a broker: "+str(producer))
for i in range(30):
    data={
       'user_id': fake.random_int(min=20000, max=100000),
       'user_name':fake.name(),
       'user_address':fake.street_address() + ' | ' + fake.city() + ' | ' + fake.country_code(),
       'platform': random.choice(['Mobile', 'Laptop', 'Tablet']),
       'signup_at': str(fake.date_time_this_month())
    }
    publish_message(producer, topic_name, key, data)
    logging.info(data)
    time.sleep(1)
```

Plantilla de Consumidor

```
from kafka import KafkaConsumer
from json import loads
import logging
```

```
#Logging
logging.basicConfig(format='%(asctime)s %(message)s',
                    datefmt='%d-%m-%Y %H:%M:%S',
                    filename='consumer.log',
                    filemode='w')

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
#Configuración del entorno Kafka
topic_name = 'my-topic'
key = 'my-key'
bootstrap_servers = ['localhost:9092']
logging.info('Arrancando...')
```

```
try:
    consumer = None
    consumer = KafkaConsumer(topic_name,
                             bootstrap_servers=bootstrap_servers,
                             auto_offset_reset='earliest',
                             enable_auto_commit=True,
                             group_id='my-processing-group',
                             value_deserializer=lambda x: loads(x.decode('utf-8')))
except Exception as ex:
    logging.error('Exception while connecting consumer')
    logging.error(ex)
```

```
for message in consumer:
    message = message.value
    print('Leido: {}'.format(message))
    logger = logging.info('Mensaje: {}'.format(message))
```

LAB 5

Configuración y arranque de nuevos brokers

Copiamos las propiedades en el directorio de config en el directorio Apache Kafka

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ cp config/server.properties config/server1.properties
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ cp config/server.properties config/server2.properties
```

En server1.properties establecemos los siguientes valores:

```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1

############################# Socket Server Settings #############################

# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9093
```

```
# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs-1
```

En server2.properties establecemos los siguientes valores:

```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=2

############################# Socket Server Settings #############################

# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9094
```

```
# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs-2
```

Arrancamos todos los brokers

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ nohup bin/kafka-server-start.sh config/server.properties > /dev/null 2>&1 &
[1] 14148
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ nohup bin/kafka-server-start.sh config/server1.properties > /dev/null 2>&1 &
[2] 14584
[1]   Exit 1                  nohup bin/kafka-server-start.sh config/server.properties > /dev/null 2>&1
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ nohup bin/kafka-server-start.sh config/server2.properties > /dev/null 2>&1 &
[3] 15010
[umaster@ibmuamdocker kafka_2.13-3.7.0]$
```

Creación de diferentes topics

Creamos topics con diferentes valores de factor de replicación y de particiones

Factor de replica 3 | Particiones 1 (nombre del topic my-rep3-part1-topic)

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092
--replication-factor 3 --partitions 1 --create --topic my-rep3-part1-topic
Created topic my-rep3-part1-topic.
```

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --des
cribe --topic my-rep3-part1-topic
Topic: my-rep3-part1-topic      TopicId: 6hZkG62jRrqO83jkHGrGrQ PartitionCount: 1       ReplicationF
actor: 3   Configs:
        Topic: my-rep3-part1-topic      Partition: 0    Leader: 1       Replicas: 1,0,2 Isr: 1,0,2
[umaster@ibmuamdocker kafka_2.13-3.7.0]$
```

Podemos observar nombre de topic, topic id, factor de replicación y particiones.

Repetimos para:
Factor de replica 3 | Particiones 3 (nombre del topic my-rep3-part3-topic)

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --repl
ication-factor 3 --partitions 3 --create --topic my-rep3-part3-topic
Created topic my-rep3-part3-topic.
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --desc
ribe --topic my-rep3-part3-topic
Topic: my-rep3-part3-topic      TopicId: 7GeTYE7fTJOX75_O290v4A PartitionCount: 3       ReplicationFa
ctor: 3        Configs:
        Topic: my-rep3-part3-topic      Partition: 0    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0
        Topic: my-rep3-part3-topic      Partition: 1    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2
        Topic: my-rep3-part3-topic      Partition: 2    Leader: 2       Replicas: 2,0,1 Isr: 2,0,1
```

Tenemos 3 particiones y 3 replicas.

Factor de replica 1 | Particiones 3 (nombre del topic my-rep1-part3-topic)

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --repl
ication-factor 1 --partitions 3 --create --topic my-rep1-part3-topic
Created topic my-rep1-part3-topic.
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --desc
ribe --topic my-rep1-part3-topic
Topic: my-rep1-part3-topic      TopicId: kf-zPl4JQdKrnKqx5dUMfg PartitionCount: 3       ReplicationFa
ctor: 1 Configs:
        Topic: my-rep1-part3-topic      Partition: 0    Leader: 1       Replicas: 1     Isr: 1
        Topic: my-rep1-part3-topic      Partition: 1    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-rep1-part3-topic      Partition: 2    Leader: 2       Replicas: 2     Isr: 2
```

Tenemos 3 particiones y 1 replicas.

Validamos “logs”
Adjunto los archivos de log llamados Kafka_Logs a esta práctica.
$ tree /tmp/kafka-logs

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ tree /tmp/kafka-logs
/tmp/kafka-logs
├── cleaner-offset-checkpoint
├── connect-test-0
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── __consumer_offsets-0
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
├── __consumer_offsets-1
│   ├── 00000000000000000000.index
│   ├── 00000000000000000000.log
│   ├── 00000000000000000000.timeindex
│   ├── leader-epoch-checkpoint
│   └── partition.metadata
```

```
├── recovery-point-offset-checkpoint
├── replication-offset-checkpoint
└── Topic_Trabajo-0
    ├── 00000000000000000000.index
    ├── 00000000000000000000.log
    ├── 00000000000000000000.timeindex
    ├── leader-epoch-checkpoint
    └── partition.metadata

57 directories, 290 files
```

$ tree /tmp/kafka-logs-1

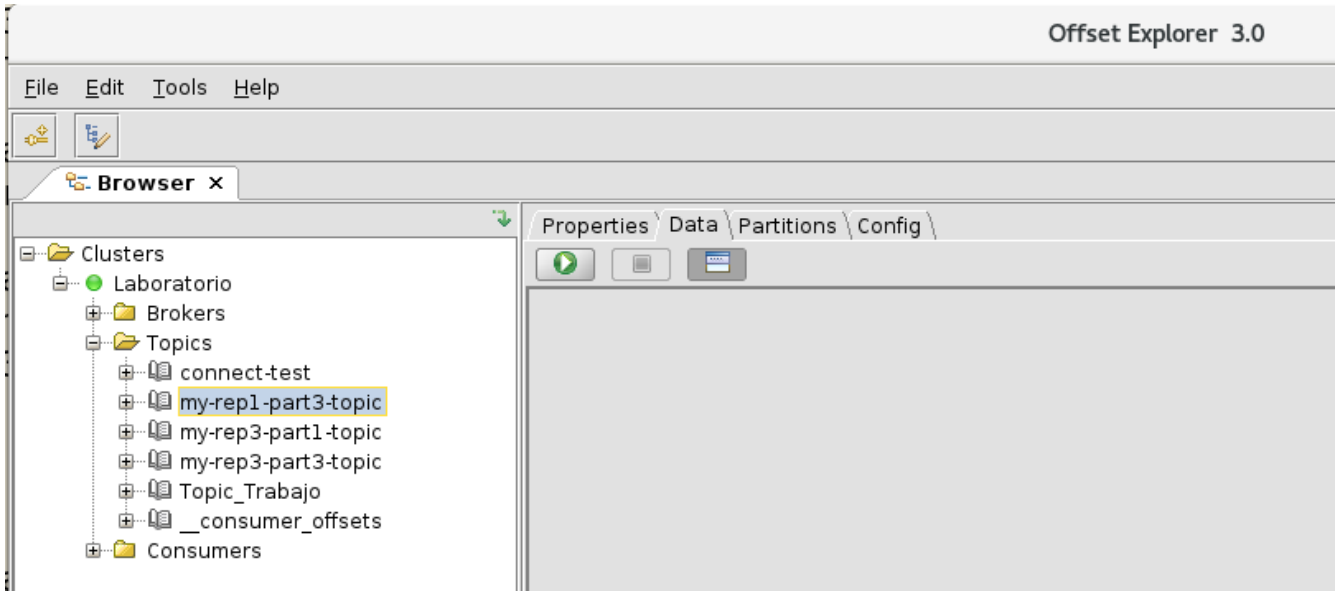
```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ tree /tmp/kafka-logs-1
/tmp/kafka-logs-1
├── cleaner-offset-checkpoint
├── log-start-offset-checkpoint
├── meta.properties
├── recovery-point-offset-checkpoint
└── replication-offset-checkpoint

0 directories, 5 files
```

$ tree /tmp/kafka-logs-2

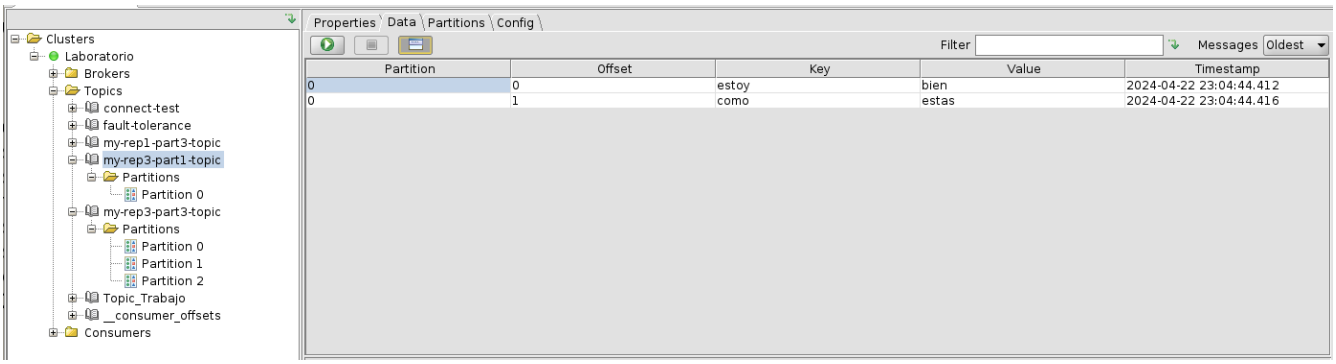
```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ tree /tmp/kafka-logs-2
/tmp/kafka-logs-2
├── cleaner-offset-checkpoint
├── log-start-offset-checkpoint
├── meta.properties
├── recovery-point-offset-checkpoint
└── replication-offset-checkpoint

0 directories, 5 files
[umaster@ibmuamdocker kafka_2.13-3.7.0]$
```

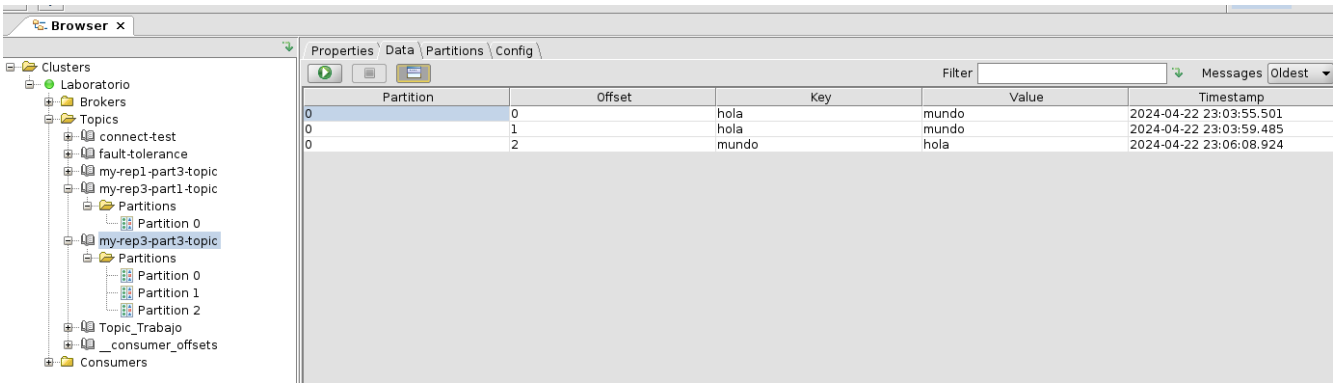
Observamos directorios de registro en el directorio kafka-logs, pero no existen directorios de registro en los directorios kafka-logs-1 y kafka-logs-2.

<u>Vemos que creamos los topics y podemos observar en Offset Explorer:</u>



Introducimos valores en los topics y comprobamos resultados con el Offset Explorer:





Al añadir nuevos valores, se pueden observar los pares clave-valor en los topics.

Multibroker. Comprobar tolerancia a fallos

Obtenemos el PID con el comando “ps”

$ ps -ef | grep server

PID: 2951, 3379, 3405, 4408, 4479, 9017, 9665, 14584, 15010

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ ps -ef | grep server
umaster   2951     1  0 12:38 ?        00:00:00 /usr/libexec/gnome-shell-calendar-server
umaster   3379  3235  0 12:38 ?        00:00:00 /usr/libexec/evolution-calendar-factory-subpro
ess --factory all --bus-name org.gnome.evolution.dataserver.Subprocess.Backend.Calendarx3235x2
--own-path /org/gnome/evolution/dataserver/Subprocess/Backend/Calendar/3235/2
umaster   3405  3389  0 12:38 ?        00:00:00 /usr/libexec/evolution-addressbook-factory-sub
rocess --factory all --bus-name org.gnome.evolution.dataserver.Subprocess.Backend.AddressBookx
389x2 --own-path /org/gnome/evolution/dataserver/Subprocess/Backend/AddressBook/3389/2
umaster   4408     1  0 12:42 ?        00:00:18 /usr/libexec/gnome-terminal-server
umaster   4479 30779  0 19:15 pts/3    00:00:26 java -Xms256M -Xmx2G -server -XX:+UseG1GC -XX:
```

```
umaster   9017  4415  0 13:49 pts/0    00:00:18 java -Xmx512M -Xms512M -server
```

```
umaster   9665  9575  0 13:51 pts/1    00:02:11 java -Xmx1G -Xms1G -server
```

```
umaster  14584     1  0 21:12 ?        00:00:12 java -Xmx1G -Xms1G -server
```

```
umaster  15010     1  0 21:12 ?        00:00:12 java -Xmx1G -Xms1G -server
```

$ ps -ef | grep server1

PID: 14584

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ ps -ef | grep server1
umaster  14584     1  0 21:12 ?        00:00:12 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:Max
GCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+ExplicitGCInvokesConcurrent -XX:Max
```

$ ps -ef | grep server2

PID: 15010

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ ps -ef | grep server2
umaster  15010     1  0 21:12 ?        00:00:12 java -Xmx1G -Xms1G -server -XX:+UseG1GC
```

Hemos matado el proceso del broker 2 y hemos verificado mediante el comando ps que se ha matado correctamente.

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ kill 15010
```

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ ps -ef | grep server2
umaster  24471 22110  0 22:40 pts/6    00:00:00 grep --color=auto server2
```

Creamos un nuevo topic y revisamos la configuración.

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092
--replication-factor 2 --partitions 2 --create --topic fault-tolerance
Created topic fault-tolerance.
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092
--describe --topic fault-tolerance
Topic: fault-tolerance  TopicId: Gi5q84ttRJ2U--h4ciDCdA PartitionCount: 2       ReplicationFact
or: 2   Configs:
        Topic: fault-tolerance  Partition: 0    Leader: 1       Replicas: 1,0   Isr: 1,0
        Topic: fault-tolerance  Partition: 1    Leader: 0       Replicas: 0,1   Isr: 0,1
[umaster@ibmuamdocker kafka_2.13-3.7.0]$
```

Observamos que hemos creado un topic y tiene 2 particiones y 2 réplicas con el nombre del topic.

Producimos y consumimos nuevos mensajes

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-console-producer.sh --broker-list localhost:
9092, localhost:9093 --topic fault-tolerance
>XX
>YY
```

```
>^C[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-console-consumer.sh --bootstrap-server lo
host:9092, localhost:9093 --from-beginning --topic fault-tolerance
XX
YY
^CProcessed a total of 2 messages
[umaster@ibmuamdocker kafka_2.13-3.7.0]$
```

Sabemos que PID del broker 1 es 14584

Hemos matado el proceso del broker 1 y hemos verificado mediante el comando ps que se ha matado correctamente.

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ kill 14584
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ ps -ef | grep server1
umaster  27015 22110  0 22:54 pts/6    00:00:00 grep --color=auto server1
[umaster@ibmuamdocker kafka_2.13-3.7.0]$
```

Revisamos la configuración del topic y validamos que seguimos disponiendo de los mensajes:

```
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-topics.sh --bootstrap-server localhost:9092
--describe --topic fault-tolerance
Topic: fault-tolerance  TopicId: Gi5q84ttRJ2U--h4ciDCdA PartitionCount: 2       ReplicationFact
or: 2   Configs:
        Topic: fault-tolerance  Partition: 0    Leader: 0       Replicas: 1,0   Isr: 0
        Topic: fault-tolerance  Partition: 1    Leader: 0       Replicas: 0,1   Isr: 0
[umaster@ibmuamdocker kafka_2.13-3.7.0]$ bin/kafka-console-consumer.sh --bootstrap-server local
host:9092, localhost:9093 --from-beginning --topic fault-tolerance
XX
YY
^CProcessed a total of 2 messages
[umaster@ibmuamdocker kafka_2.13-3.7.0]$
```

En Offset Explorer vemos que tenemos todos los topics que hemos creado