Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [ ]:

```
NAME = "Vicent Ripoll Ramírez"
COLLABORATORS = ""
```

PEC5_1: Captura de información en streaming mediante Kafka

Kafka (https://kafka.apache.org) es una plataforma distribuida para gestionar eventos en streaming, que nos permite leer, escribir y procesar eventos (records o messages según Kafka) distribuidos en un clúster. En esta PEC vamos a trabajar esta tecnología.

Vamos a empezar creando un tópico denominado `PEC5<usuario>` en servicio Kafka de nuestro clúster (debéis substituir usuario por vuestro login). Un tópico es una colección ordenada de eventos que está almacenada de manera persistente, habitualmente en disco y de manera replicada. Kafka trata cada tópico en cada partición como un log (un conjunto ordenado de mensajes). Cada mensaje en una partición tiene asignado un único offset y estos mensajes tienen un periodo de retención por defecto de los mensajes en un tópico es de 7 días (604,800,000 ms), pero es posible cambiarlo en el momento de la creación del tópico.

Kafka trabaja en base al Zookeeper (https://zookeeper.apache.org) que se encarga de la gestión de clústeres con la finalidad de proporcionar un servicio de coordinación para aplicaciones distribuidas, se encuentra en el propio servidor al que conectáis, es decir en *localhost* y su puerto es el 2181. Los brokers de Kafka son *Cloudera02 y Cloudera03* y están accesibles en el puerto habitual 9092

IMPORTANTE: Por cuestiones de organización y de estabilidad del servidor, es importante que no creéis ningún tópico diferente al que se ha pedido, con vuestro login, cualquier otro tópico será borrado y en caso necesario el usuario será deshabilitado del sistema

IMPORTANTE: Para realizar esta práctica debes hacerlo mediante SSH desde terminal o VSCODE, y solamente adjuntar el código resultante en este mismo NOTEBOOK, siempre en un fichero notebook (ipynb)

> **Pregunta 1. (1 punto)** crea un tópico denominado `PEC5<usuario>` en Kafka en nuestro clúster, con un factor de replicación de 1 y una única partición, es decir vamos a utilizar un solo nodo para almacenar los mensajes que recibe Kafka. Además, vamos a especificar que los mensajes solo estén almacenados 2h en el tópico.

In [ ]:

```
vripollr@Cloudera01:~$ kafka-topics --create --zookeeper Cloudera01:2181/kafka --topic PAC5vripollr --partitions 1 --replication-
Created topic "PAC5vripollr".
```

> **Pregunta 2. (1 punto)** consulta el *tópico* que acabas de crear y muéstralo.

In [ ]:

```
vripollr@Cloudera01:~$ kafka-topics --list --zookeeper Cloudera01:2181/kafkaPAC5vripollr
PEC5alvarorp22
PEC5asolerib
PEC5blozanoba
PEC5fjalbarrang
PEC5gromerof1974
PEC5lgomeztorres
PEC5sgraul
PEC5svila0
PEC5xsellart1
PEC5xye
__consumer_offsets
fjalbarrang
```

> **Pregunta 3. (1 punto)**. Borra el tópico que acabas de crear.

In [ ]:

```
vripollr@Cloudera01:~$ kafka-topics --zookeeper Cloudera01:2181/kafka --delete --topic PAC5vripollr
Topic PAC5vripollr is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

Pregunta 4. (1 punto). Vuelve a crear el tópico y muestra la información detallada del tópico que hemos creado.

In [ ]:

```
vripollr@Cloudera01:~$ kafka-topics --create --zookeeper Cloudera01:2181/kafka --topic PAC5vripollr --partitions 1 --replication-
Created topic "PAC5vripollr".
vripollr@Cloudera01:~$ kafka-topics --zookeeper Cloudera01:2181/kafka --describe
Topic:PAC5vripollr      PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PAC5vripollr     Partition: 0    Leader: 97      Replicas: 97    Isr: 97
Topic:PEC5alvarorp22    PartitionCount:1        ReplicationFactor:1     Configs:
        Topic: PEC5alvarorp22   Partition: 0    Leader: 95      Replicas: 95    Isr: 95
Topic:PEC5asolerib      PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PEC5asolerib     Partition: 0    Leader: 97      Replicas: 97    Isr: 97
Topic:PEC5blozanoba     PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PEC5blozanoba    Partition: 0    Leader: 95      Replicas: 95    Isr: 95
Topic:PEC5fjalbarrang   PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PEC5fjalbarrang  Partition: 0    Leader: 97      Replicas: 97    Isr: 97
Topic:PEC5gromerof1974  PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PEC5gromerof1974 Partition: 0    Leader: 95      Replicas: 95    Isr: 95
Topic:PEC5lgomeztorres  PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PEC5lgomeztorres Partition: 0    Leader: 95      Replicas: 95    Isr: 95
Topic:PEC5sgraul        PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PEC5sgraul       Partition: 0    Leader: 95      Replicas: 95    Isr: 95
Topic:PEC5xsellart1     PartitionCount:1        ReplicationFactor:1     Configs:retention.ms=7200000
        Topic: PEC5xsellart1    Partition: 0    Leader: 95      Replicas: 95    Isr: 95
Topic:PEC5xye   PartitionCount:1        ReplicationFactor:1     Configs:delete.retention.ms=172800000
        Topic: PEC5xye  Partition: 0    Leader: 95      Replicas: 95    Isr: 95
Topic:__consumer_offsets        PartitionCount:50       ReplicationFactor:1     Configs:segment.bytes=104857600,cleanup.policy=co
        Topic: __consumer_offsets       Partition: 0    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 1    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 2    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 3    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 4    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 5    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 6    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 7    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 8    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 9    Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 10   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 11   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 12   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 13   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 14   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 15   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 16   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 17   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 18   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 19   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 20   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 21   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 22   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 23   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 24   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 25   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 26   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 27   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 28   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 29   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 30   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 31   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 32   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 33   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 34   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 35   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 36   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 37   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 38   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 39   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 40   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 41   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 42   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 43   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 44   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 45   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 46   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 47   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 48   Leader: 95      Replicas: 95    Isr: 95
        Topic: __consumer_offsets       Partition: 49   Leader: 95      Replicas: 95    Isr: 95
Topic:fjalbarrang       PartitionCount:1        ReplicationFactor:1     Configs:
        Topic: fjalbarrang      Partition: 0    Leader: 95      Replicas: 95    Isr: 95
```

Observa detalladamente la información que muestra y coméntala.

In [ ]:

```
#Se muestran los topics registrados y en cada uno la siguiente información:
#-Topic: nombre del topic
#-PartitionCount: se ha especificado al crear el topic, se trata del número de divisiones que se realizan sobre el topic para
#repartirlo entre los nodos. En este caso su valor es 1.
#-ReplicationFactor: también se ha especificado al crear el topic, se trata del número de réplicas por partición. En este caso
#su valor es 1. Al fijar las particiones y las réplicas a 1 estamos imponiendo que los topics se guarden en un único nodo.
#-Configs:retention.ms. Se trata del número de milisegundos que kafka guardará los mensajes antes de borrarlos. Su valor es
#7200000ms = 2h.
#-Partition: número identificativo asociado a la partición. Como solo se ha creado una única partición, aparece únicamente
#la partición 0. Si hubiéramos creado n particiones aparecería Partition: 0,..., Partition: n-1.
#-Leader: 95/97. Los brokers con id 95 y 97 del clúster son los responables de la escritura y lectura de la partición asociada,
#Partition: 0, en este caso.
#-Replicas: se muestra el id de los brokers encargados de replicar los datos de cada partición. En algunos casos podría aparecer
#Replicas: 2, 1 (por ejemplo) lo que significaría que Kafka estaría dando prioridad al broker 2 como leader de la partición.
#-Isr: son las siglas de In-Sync Replicas. Se trata de las replicas que están en sincronía entre ellas en el cluster. Esto
#incluye las replicas leader y follower. Lo ideal es que Isr == Replicas, lo que significaría que todas las replicas estarían
#sincronizadas. Esto es precisamente lo que hemos obtenido ya que Replicas = 97 y Isr = 97.
```

Pregunta 5. (1 punto) Vamos a crear un evento en el tópico, recuerda que como el resto de la PEC, esta sentencia la debemos ejecutar desde terminal para poder interactuar. Recuerda de realizar `CTRL+c` cuando hayas finalizado de enviar mensajes.

In [ ]:

```
vripollr@Cloudera01:~$ kafka-console-producer --broker-list Cloudera02:9092 --topic PAC5vripollr
>hola
>mundo
>hello
>world
```

Pregunta 6. (1 punto) Finalmente se pide consultar por el terminal mediante el programa consumidor de *tópicos* que incorpora Kafka. Ejecutamos un consumidor conectando a los diversos brokers existentes e indicando el tópico y la partición a la que se han enviado. Puedes abrir dos terminales y verificar que los mensajes que se envían con el productor al broker, los podemos consultar con el consumidor de consola que incorpora Kafka.

In [ ]:

```
#Terminal 1
vripollr@Cloudera01:~$ kafka-console-producer -broker-list Cloudera02:9092 --topic PAC5vripollr
>hola
>que
>tal
#Terminal 2
vripollr@Cloudera01:~$ kafka-console-consumer --bootstrap-server Cloudera02:9092 --topic PAC5vripollr
hola
que
tal
```

Pregunta 7. (1 punto)

Para automatizar la generación y consumo de datos es habitual trabajar mediante algún lenguaje de programación, como por ejemplo Python, en vez de directamente sobre Bash. A continuación, vamos trabajar el funcionamiento de Kafka mediante el uso de Python y con la librería por defecto que **NO DEBEIS INSTALARLA DADO QUE YA SE ENCUENTRA DISPONIBLE LA VERSION CORRECTA**. Podéis encontrar toda la documentación asociada a la API proporcionada en Kafka (https://kafka-python.readthedocs.io/en/master/). Vamos a empezar por los conceptos más básicos y que ya hemos trabajado: escribir en el tópico que hemos creado. Para ello vamos a configurar un Kafka producer (https://kafka-python.readthedocs.io/en/master/apidoc/KafkaProducer.html) que va a insertar valores numéricos en un tópico de Kafka cada 3 segundos. Mientras está el producer escribiendo vamos a proceder a la lectura de los mensajes en el ejercicio 8.

Se pide escribir una secuencia numérica de 300 números (1 a 300) en el tópico `PEC5<usuario>` de Kafka que acabamos de crear. Cada uno de los mensajes que escribimos en el tópico debe contener información sobre el tópico dónde escribir, una key y el valor binario del valor a escribir (p.ej. value=b'287'). Es fundamental revisar la API asociada al Kafka producer (https://kafka-python.readthedocs.io/en/master/apidoc/KafkaProducer.html) para la realización de los ejercicios.

Insistimos en el uso de la librería por defecto de Python para acceder a Kafka (https://kafka-python.readthedocs.io/en/master/), **NO DEBEIS INSTALARLA DADO QUE YA SE ENCUENTRA DISPONIBLE LA VERSION CORRECTA**.

Esquema

```
from kafka import KafkaProducer
import numpy as np
<FILL_IN>
for i in range(1,300):
    <FILL_IN>
producer.flush()
```

In [4]:

```
from kafka import KafkaProducer
import numpy as np
producer = KafkaProducer(bootstrap_servers='Cloudera02:9092')
for i in range (1,300):
    producer.send('PEC5vripollr',key=bytes('prueba1','utf-8'),value=bytes(str(i),'utf-8'))
producer.flush()
```

Pregunta 8. (1 punto)

Mediante la biblioteca de Python para Kafka (https://kafka-python.readthedocs.io/en/master/), **NO DEBEIS INSTALARLA DADO QUE YA SE ENCUENTRA DISPONIBLE LA VERSION CORRECTA** , leer los mensajes enviado en el ejercicio 7 mostrando exclusivamente los valores, no el resto de propiedades del mensaje. Es importante revisar el uso de Kafka mediante Python (https://kafka-python.readthedocs.io/en/master/usage.html) y los parámetros del Kafka consumer (https://kafka-python.readthedocs.io/en/master/apidoc/KafkaConsumer.html).

Esquema

```
from kafka import KafkaConsumer
<FILL_IN>
for message in consumer:
    <FILL_IN>
```

In [5]:

```
from kafka import KafkaConsumer
consumer = KafkaConsumer('PEC5vripollr',bootstrap_servers='Cloudera02:9092',auto_offset_reset='smallest',consumer_timeout_ms=1000
for message in consumer:
    print(message.value)
```

```
b'b\'{"callsign": "SWR233D ", "velocity": 250.96, "longitude": -1.9263, "country": "Switzerland", "latitude": 42.
3013, "vertical_rate": 0}\''
b'b\'{"callsign": "TAP1902 ", "velocity": 160.82, "longitude": -8.6887, "country": "Portugal", "latitude": 37.876
9, "vertical_rate": 0}\''
b'b\'{"callsign": "SCR168  ", "velocity": 185.64, "longitude": -7.686, "country": "Germany", "latitude": 42.8601,
"vertical_rate": 0.33}\''
b'b\'{"callsign": "TAP1364 ", "velocity": 234.05, "longitude": -8.3562, "country": "Portugal", "latitude": 42.300
8, "vertical_rate": 0}\''
b'b\'{"callsign": "TAP789  ", "velocity": 209.28, "longitude": -6.5309, "country": "Portugal", "latitude": 41.850
5, "vertical_rate": 0}\''
b'b\'{"callsign": "TAP023  ", "velocity": 195.51, "longitude": -9.6333, "country": "Portugal", "latitude": 37.99
6, "vertical_rate": 7.48}\''
b'b\'{"callsign": "EJU63ME ", "velocity": 237.91, "longitude": -10.0565, "country": "Austria", "latitude": 38.224
3, "vertical_rate": 0}\''
b'b\'{"callsign": "IBS3803 ", "velocity": 203.69, "longitude": -0.8112, "country": "Spain", "latitude": 39.6536,
"vertical_rate": 0}\''
b'b\'{"callsign": "IBE31RG ", "velocity": 221.4, "longitude": -8.0522, "country": "Spain", "latitude": 39.2181,
"vertical_rate": 14.31}\''
b'b\'{"callsign": "SWT960P ", "velocity": 132.08, "longitude": -5.1409, "country": "Spain", "latitude": 40.2299,
```

In [ ]: