

# Tarea evaluable

## CE\_5075 6.1

Big data aplicado



# Índice

<b>NOTA A TENER EN CUENTA</b>	<b>2</b>
<b>APARTADO 1</b>	<b>3</b>
<b>APARTADO 2</b>	<b>7</b>

## **NOTA A TENER EN CUENTA:**

He tenido un problema con la máquina virtual de Fedora, donde tenemos configurado el sistema de Hadoop desde el Tema 1. No puedo acceder a mi usuario porque no reconoce la contraseña, y tras dedicarle bastante tiempo en el tema anterior, no logré solucionarlo. Este es un problema del que ya hablé por email con Toni.

Dado que en este tema también es necesario acceder a Fedora, intenté realizar el ejercicio en la máquina virtual de Cloudera, ya que también cuenta con HDFS. Sin embargo, no he conseguido levantar los servicios ni con “start-all.sh” ni con “hadoop-daemon.sh start namenode” o “hadoop-daemon.sh start namenode/datanode”

Por ello, en esta entrega incluyo explicaciones y capturas de pantalla de todo lo que esté en mi mano. En lo referente a Hadoop, no podré aportar capturas, pero proporcionaré una explicación detallada para completar la práctica lo mejor posible.

Disculpa las molestias y gracias.

David Ramírez

## APARTADO 1

Tenemos un archivo CSV en el que simularemos que un sensor va escribiendo lecturas de diversas variables ambientales, separadas por comas (,), con el siguiente formato:

- **fecha:** cadena de texto con la fecha de la lectura, en formato dd/mm/yyyy. Por ejemplo: 23/02/2025.
- **hora:** cadena de texto con la hora de la lectura, en formato hh:mm:ss. Por ejemplo: 16:07:30.
- **temperatura:** número real (float), expresado en grados centígrados. Por ejemplo: 17.7.
- **humedad:** número real (float), expresado en porcentaje. Por ejemplo: 75.1.
- **presión:** número real (entero), expresado en milibares. Por ejemplo: 1012.

Debes configurar un conector *source* (en modo distribuido) para leer estos datos y guardarlos en un *topic* de Kafka.

Luego, utilizando Python y la librería *hdfs*, deberás volcar los eventos a un archivo en HDFS de tu clúster Hadoop. Si prefieres, puedes usar una configuración con solo un único nodo.

El formato del archivo será el siguiente, con los valores separados por punto y coma (;):

- **datahora:** número entero, con la fecha y hora en formato timestamp de Unix.
- **temperatura**
- **presión**
- **humedad**

Para probarlo, simula las lecturas del sensor utilizando *echo*, como hemos hecho en las notas. Por ejemplo:

```
echo "23/02/2025,16:07:30,17.7,75.1,1012" >> fitxer.csv
```

Empezamos el ejercicio posicionándonos en el directorio donde están los archivos de kafka, en mi caso: `/home/david/Desktop/kafka_2.13-3.8.1`

Una vez hecho esto pasamos a poner en marcha el servidor de kafka arrancando Zookeeper y en otra terminal arrancamos el broker de kafka.

```
# Arrancar ZooKeeper
bin/zookeeper-server-start.sh config/zookeeper.properties

# Arrancar el broker de Kafka
bin/kafka-server-start.sh config/server.properties
```

Ahora procedemos a crear un nuevo *topic*, en mi caso lo he llamado “apartado-1”:

```
# Crear un topic
bin/kafka-topics.sh --create --topic apartado-1 --bootstrap-server localhost:9092
```

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (1.095s)
bin/kafka-topics.sh --create --topic apartado-1 --bootstrap-server localhost:9092
Created topic apartado-1.
```

El siguiente paso es configurar el conector, en este caso, uno de tipo source para leer los datos desde un archivo. Dado que utilizaremos el modo distribuido, trabajaremos con un archivo JSON.

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (2m 24.41s)
nano config/connect-file-source.json

david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.036s)
cat config/connect-file-source.json
{
  "name": "file-source-distributed",
  "config": {
    "connector.class": "FileStreamSource",
    "tasks.max": "1",
    "file": "/home/david/Desktop/kafka_2.13-3.8.1/apartado1.csv",
    "topic": "apartado-1"
  }
}

david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1
```

Una vez hecho esto podemos poner en marcha el worker que actuará como una API REST:

```
# Poner en marcha el worker
bin/connect-distributed.sh config/connect-distributed.properties
```

Cambiamos de terminal y para comprobar que todo funciona correctamente, hacemos un POST del archivo de configuración del conector source enviando una request a la API.

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.132s)
curl -X POST -H "Content-Type: application/json" --data @config/connect-file-source.json http://localhost:8083/connectors
{"name":"file-source-distributed","config":{"connector.class":"FileStreamSource","tasks.max":"1","file":"/home/david/Desktop/kafka_2.13-3.8.1/apartado1.csv","topic":"apartado-1","name":"file-source-distributed"},"tasks":[],"type":"source"}

david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.048s)
curl http://localhost:8083/connectors
["file-source-distributed"]
```

Llegados a este punto tenemos la configuración completada. Y para comprobar que está bien configurado podemos hacer una prueba de cómo se guardan los datos en el *topic* que hemos creado.

Usando el siguiente comando podemos generar un nuevo evento, es decir, una escritura en el archivo:

```
echo "23/02/2025,16:07:30,17.7,75.1,1012" >> apartado1.csv
```

Y con este comando podremos ver los eventos en el *topic*:

```
bin/kafka-console-consumer.sh --bootstrap-server
localhost:9092 --topic apartado-1 --from-beginning
```

Al ejecutarlos obtenemos los datos enviados dentro del “payload”

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.03s)
echo "23/02/2025,16:07:30,17.7,75.1,1012" >> apartado1.csv

david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic apartado-1 --from-beginning
{"schema":{"type":"string","optional":false},"payload":"23/02/2025,16:07:30,17.7,75.1,1012"}
```

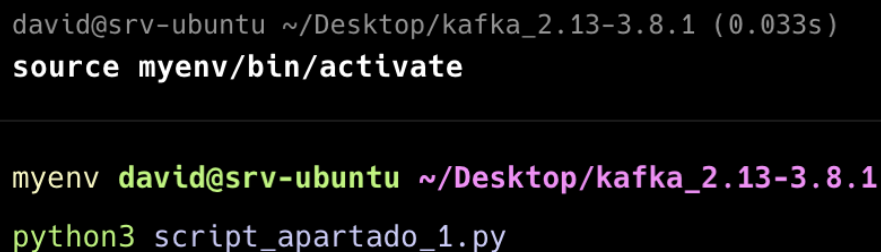
Ahora es el momento de crear el script *consumer* del *topic* usando Python, el archivo con el código del script se encuentra en esta misma carpeta bajo el nombre “script\_apartado\_1.py”.

Es importante que antes de ejecutar el script toda la parte de Hadoop esté bien configurada como se detalla en los apuntes. Activando el WebHDFS en el NameNode, creando el directorio `/user/hadoop/kafka`, el archivo `.csv` y dándole permisos.

Después de esto, vamos a otra terminal, entramos en el modo de entorno virtual donde tenemos instaladas las librerías y podemos ejecutar el script:

```
source myenv/bin/activate
```

```
python3 script_apartado_1.py
```



A terminal window with a black background and white text. The prompt is `david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.033s)`. The user enters `source myenv/bin/activate`. The prompt changes to `myenv david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1`. The user then enters `python3 script_apartado_1.py`.

Es a partir de este punto en el que no puedo añadir más capturas del proceso en la parte de hadoop ya que no tengo acceso. Sin embargo conozco el proceso y por eso he querido explicar al máximo cómo realizar este apartado.

Para comprobar en hadoop si el resultado ha sido exitoso podemos ejecutar este comando:

```
hdfs dfs -cat user/hadoop/kafka/apartado1.csv
```

Deberíamos esperar ver por la terminal los datos añadidos anteriormente.

## APARTADO 2

Queremos guardar los mensajes de Bluesky a partir de una (o varias) palabras de búsqueda (elige la que prefieras, pero es mejor una que genere mensajes con frecuencia) en una base de datos MySQL para poder procesarlos posteriormente.

Debes configurar un conector **source** (en modo distribuido) para leer mensajes de Bluesky según tu búsqueda y almacenarlos como eventos en un **topic de Kafka**.

Luego, utilizando el conector **JDBC sink** (también en modo distribuido), debes guardar esos eventos en una tabla de MySQL. La tabla debe contener todos los datos del objeto **payload** del evento, excepto el campo **langs**. Estos son los campos requeridos:

- **uri**
- **cid**
- **text**
- **createdAt**
- **handle**
- **displayName**
- **avatar**

Al igual que en apartado 1, empezamos posicionándonos en el directorio donde están los archivos de kafka, en mi caso: */home/david/Desktop/kafka\_2.13-3.8.1*

Pasamos a poner en marcha el servidor de kafka arrancando Zookeeper y en otra terminal arrancamos el broker de kafka.

```
# Arrancar ZooKeeper
bin/zookeeper-server-start.sh config/zookeeper.properties

# Arrancar el broker de Kafka
bin/kafka-server-start.sh config/server.properties
```

Ahora procedemos a crear un nuevo *topic*, en este segundo apartado lo he llamado "apartado-2":

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.975s)
bin/kafka-topics.sh --create --topic apartado-2 --bootstrap-server localhost:9092
Created topic apartado-2.
```



Configuramos el archivo *source* para Bluesky:

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (3m 3.06s)
nano config/bluesky-source.json




david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.034s)
cat config/bluesky-source.json
{
  "name": "bluesky-generativeAI",
  "config": {
    "connector.class": "uk.co.dalelane.kafkaconnect.bluesky.source.BlueskySourceConnector",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "bluesky.identity": "d-ramirez.bsky-social",
    "bluesky.password": "yshh-p3mg-wxpg-h5i6",
    "bluesky.searchterm": "GenerativeAI",
    "bluesky.topic": "apartado-2",
    "tasks.max": 1
  }
}
```

Una vez más como en el apartado anterior, cambiamos de terminal y hacemos un POST del archivo de configuración del conector *source* enviando una request a la API.

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.034s)
curl -X POST -H "Content-Type: application/json" --data @config/bluesky-source.json http://localhost:8083/connectors

{"name":"bluesky-generativeAI","config":{"connector.class":"uk.co.dalelane.kafkaconnect.bluesky.source.BlueskySourceConnector","key.converter":"org.apache.kafka.connect.storage.StringConverter","value.converter":"org.apache.kafka.connect.json.JsonConverter","bluesky.identity":"d-ramirez.bsky.social","bluesky.password":"yshh-p3mg-wxpg-h5i6","bluesky.searchterm":"GenerativeAI","bluesky.topic":"apartado-2","tasks.max":"1","name":"bluesky-generativeAI"},"tasks":[],"type":"source"}
```

Y podemos ver que al escuchar los posts publicados en Bluesky con la palabra “GenerativeAI” obtenemos los resultados.

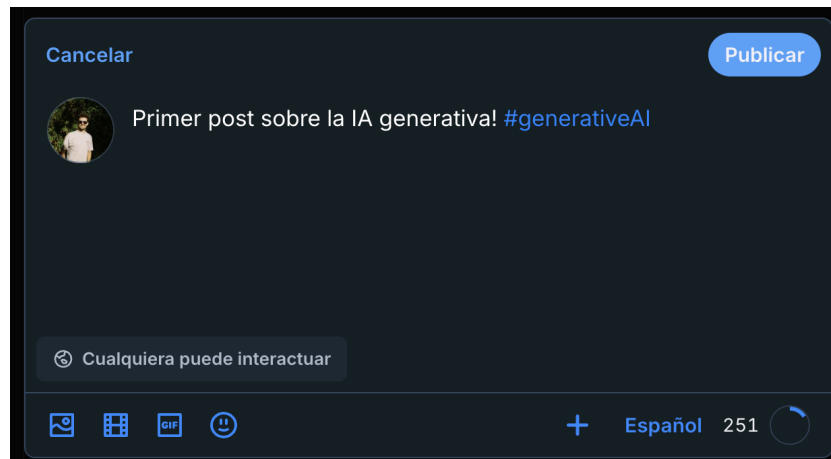
```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (4.927s)   
```

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic apartado-2 --from-beginning
```

```
{ "schema": { "type": "struct", "fields": [ { "type": "struct", "fields": [ { "type": "string", "optional": false, "field": "uri" }, { "type": "string", "optional": false, "field": "cid" } ], "optional": false, "field": "id" }, { "type": "string", "optional": false, "field": "text" }, { "type": "array", "items": { "type": "string", "optional": false }, "optional": false, "field": "langs" }, { "type": "int64", "optional": true, "name": "org.apache.kafka.connect.data.Timestamp", "version": 1, "field": "createdAt" }, { "type": "struct", "fields": [ { "type": "string", "optional": false, "field": "handle" }, { "type": "string", "optional": true, "field": "displayName" }, { "type": "string", "optional": true, "field": "avatar" } ], "optional": false, "field": "author" }, { "optional": false, "name": "status", "payload": { "id": { "uri": "at://did:plc:zvtvzfpxdst7bztvoagrzsgs/app.bsky.feed.post/3lkt5747imi26", "cid": "buffreibtajqjd24qfjybvbl2edpbqkvft5kxhtt46epovmahqvbf2uqjkape"}, "text": "CapAcuity Connects buff.ly/SDR00R9 via @NoelleRussell of @noellerussell on @Thinkers360 #AgenticAI #Agile #GenerativeAI", "langs": [], "createdAt": "1742488800487", "author": { "handle": "thinkers360.bsky.social", "displayName": "Thinkers360 - Expert Network & Influencer Marketing Agency", "avatar": "https://cdn.bsky.app/img/avatar/plain/did:plc:zvtvzfpxdst7bztvoagrzsgs/bafkreiadi3oz7jo6lccknmlmam6dwwza7v5hfrwlxagzhgsswusguypni.jpeg"} } } }
```

```
{ "schema": { "type": "struct", "fields": [ { "type": "struct", "fields": [ { "type": "string", "optional": false, "field": "uri" }, { "type": "string", "optional": false, "field": "cid" } ], "optional": false, "field": "id" }, { "type": "string", "optional": false, "field": "text" }, { "type": "array", "items": { "type": "string", "optional": false }, "optional": false, "field": "langs" }, { "type": "int64", "optional": true, "name": "org.apache.kafka.connect.data.Timestamp", "version": 1, "field": "createdAt" }, { "type": "struct", "fields": [ { "type": "string", "optional": false, "field": "handle" }, { "type": "string", "optional": true, "field": "displayName" }, { "type": "string", "optional": true, "field": "avatar" } ], "optional": false, "field": "author" }, { "optional": false, "name": "status", "payload": { "id": { "ur
```

Incluso al subir un post lo vemos entre los datos que recibimos



```
d post effectively. (n)ready to elevate your AI game. Start fine tuning today. (ab03b(ab200,
n\n#AI #Llama33 #OracleCloud #GenerativeAI #FineTuning #MetaAI", "langs": [], "createdAt": 1742586
172765, "author": {"handle": "talk-nerdyto-me.bsky.social", "displayName": "Talk Nerdy to Me!", "ava
tar": "https://cdn.bsky.app/img/avatar/plain/did:plc:gi5bi7qpa55m47comxxrni3z/bafkreiddiwhxmp2y
kfx6dc4byevaonpvqzks3hda726ucgvthlycfzuove@jpeg"}}
{"schema": {"type": "struct", "fields": [{"type": "struct", "fields": [{"type": "string", "optional": fa
lse, "field": "uri"}, {"type": "string", "optional": false, "field": "cid"}], "optional": false, "field":
"id"}, {"type": "string", "optional": false, "field": "text"}, {"type": "array", "items": {"type": "strin
g", "optional": false}, "optional": false, "field": "langs"}, {"type": "int64", "optional": true, "name":
"org.apache.kafka.connect.data.Timestamp", "version": 1, "field": "createdAt"}, {"type": "struct", "f
ields": [{"type": "string", "optional": false, "field": "handle"}, {"type": "string", "optional": true, "fi
eld": "displayName"}, {"type": "string", "optional": true, "field": "avatar"}], "optional": false, "fi
eld": "author"}], "optional": false, "name": "status", "payload": {"id": {"uri": "at://did:plc:qxdxlfpy
tidriazlgzoa5wfv/app.bsky.feed.post/3lkvxtu24u2c", "cid": "bafyreif57curdy32rsigwnwenszjj5y3p
xfzfupbvzhhssovsb2pqox7dm"}, "text": "Primer post sobre la IA generativa! #GenerativeAI", "langs":
["es"], "createdAt": 1742586269991, "author": {"handle": "d-ramirez.bsky.social", "displayName": null
, "avatar": "https://cdn.bsky.app/img/avatar/plain/did:plc:qxdxlfpytidriazlgzoa5wfv/bafkreid35m5
tfhxuiv5bddglrq2cdyfejxzjqxsiesdus5v72mmwsvmxq@jpeg"}}}
{"schema": {"type": "struct", "fields": [{"type": "struct", "fields": [{"type": "string", "optional": fa
lse, "field": "uri"}, {"type": "string", "optional": false, "field": "cid"}], "optional": false, "field":
"id"}, {"type": "string", "optional": false, "field": "text"}, {"type": "array", "items": {"type": "strin
g", "optional": false}, "optional": false, "field": "langs"}, {"type": "int64", "optional": true, "name":
"org.apache.kafka.connect.data.Timestamp", "version": 1, "field": "createdAt"}], "optional": false,
"field": "id"

```

```
{
  "schema": {
    "type": "struct",
    "fields": [
      {
        "type": "struct",
        "fields": [
          { "type": "string", "optional": false, "field": "uri" },
          { "type": "string", "optional": false, "field": "cid" }
        ],
        "optional": false,
        "field": "id"
      },
      { "type": "string", "optional": false, "field": "text" },
      {
        "type": "array",
        "items": { "type": "string", "optional": false },
        "optional": false,
        "field": "langs"
      },
      {
        "type": "int64",
        "optional": true,
        "name": "org.apache.kafka.connect.data.Timestamp",
        "version": 1,
        "field": "createdAt"
      }
    ],
    "type": "struct",
    "fields": [

```

```
    { "type": "string", "optional": false, "field": "handle" },
    { "type": "string", "optional": true, "field": "display
      name" },
    { "type": "string", "optional": true, "field": "avatar"
    },
    {
      "optional": false,
      "field": "author"
    }
  ],
  "optional": false,
  "name": "status"
},
"payload": {
  "id": {
    "uri": "at://did:plc:qxdxlfpytidriazlgzoa5wfv/app.bsky.feed.p
    st/3lkvxtu24u2c",
    "cid": "bafyreif57curdy32rsigwnwenszjj5y3pxfzfupbvzhhssovsb2p
    x7dm"
  },
  "text": "Primer post sobre la IA generativa! #GenerativeAI",
  "langs": ["es"],
  "createdAt": 1742586269991,
  "author": {
    "handle": "d-ramirez.bsky.social",
    "displayName": null,
    "avatar": "https://cdn.bsky.app/img/avatar/plain/did:plc:qxdx
    lfpytidriazlgzoa5wfv/bafkreid35m5tfhxuiv5bddglrq2cdyfejxzjqxsiesdus5v72mmwsvmxq@jpeg"
  }
}
```

He dejado adjunto el json con esta estructura entre los archivos de la entrega bajo el nombre de "topic\_no\_formateado.json"

Ahora, como bien decía el enunciado, debemos realizar transformaciones en los datos para hacer que algunas de las estructuras estén soportadas por MySQL. Eliminamos el conector y detenemos el *worker* para realizar este paso

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1
curl -X DELETE http://localhost:8083/connectors/bluesky-generativeAI
```

Y a través de *CTRL+C* (en *Mac*) detenemos el *worker* activo.

Una vez hecho esto configuramos el nuevo archivo de *source*

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (1m 38.20s)
nano config/bluesky-source.json

david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.031s)
cat config/bluesky-source.json
{
  "name": "bluesky-apartado-2-generative-ai",
  "config": {
    "connector.class": "uk.co.dalelane.kafkaconnect.bluesky.source.BlueskySourceConnector",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "bluesky.identity": "d-ramirez.bsky.social",
    "bluesky.password": "yssh-p3mg-wxpg-h5i6",
    "bluesky.searchterm": "GenerativeAI",
    "bluesky.topic": "bluesky_apartado_2",
    "transforms": "dropLangs,flatten,renameUri,renameCid,renameHandle,renameDisplayName,renameAvatar",
    "transforms.dropLangs.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.dropLangs.exclude": "langs",
    "transforms.flatten.type": "org.apache.kafka.connect.transforms.Flatten$Value",
    "transforms.renameCid.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.renameCid.renames": "id.cid:cid",
    "transforms.renameUri.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.renameUri.renames": "id.uri:uri",
    "transforms.renameHandle.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.renameHandle.renames": "author.handle:handle",
    "transforms.renameDisplayName.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.renameDisplayName.renames": "author.displayName:displayName",
    "transforms.renameAvatar.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.renameAvatar.renames": "author.avatar:avatar",
    "tasks.max": "1"
  }
}
```

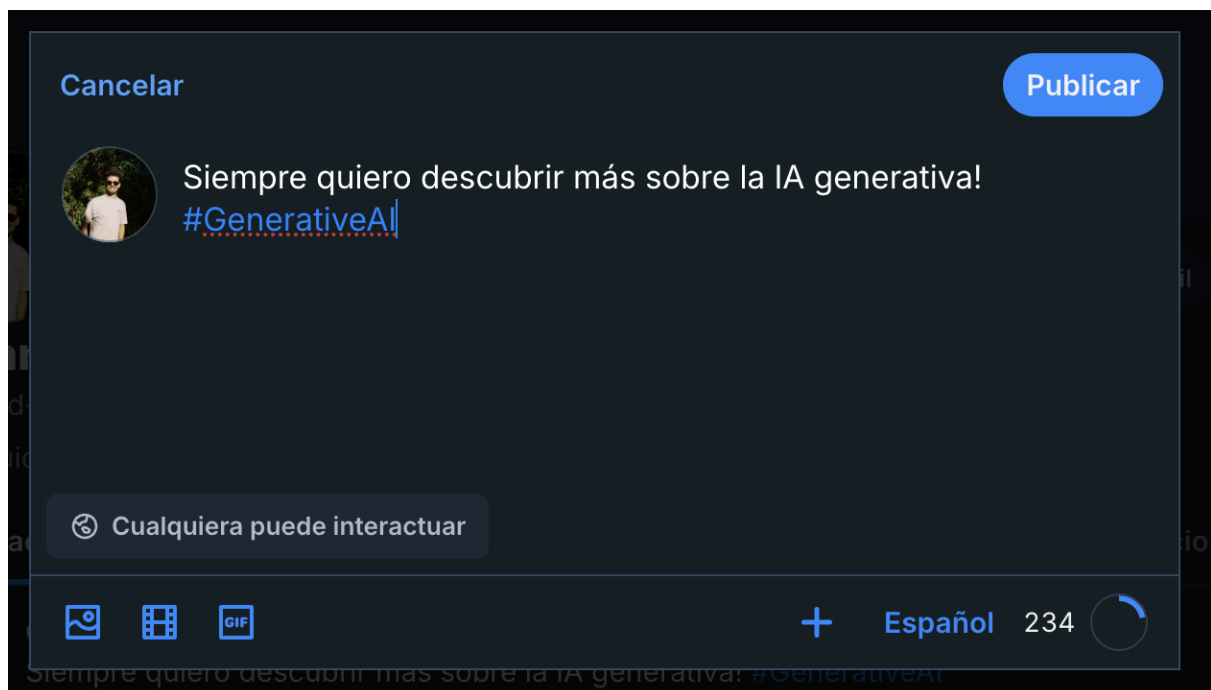
Volvemos a hacer POST del source y lanzar el consumidor de kafka como hemos hecho en el punto anterior. Y ahora para generar un nuevo evento, vamos a crear un post en Bluesky con la palabra correspondiente:

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.134s)
curl -X POST -H "Content-Type: application/json" --data @config/bluesky-source.json http://localhost:8083/connectors

{"name":"bluesky-apartado-2-generative-ai","config":{"connector.class":"uk.co.dalelane.kafkacnect.bluesky.source.BlueskySourceConnector","key.converter":"org.apache.kafka.connect.storage.StringConverter","value.converter":"org.apache.kafka.connect.json.JsonConverter","bluesky.identity":"d-ramirez.bsky.social","bluesky.password":"yssh-p3mg-wxpg-h5i6","bluesky.searchterm":"GenerativeAI","bluesky.topic":"bluesky_apartado_2","transforms":"dropLangs,flatten,renameUri,renameCid,renameHandle,renameDisplayName,renameAvatar","transforms.dropLangs.type":"org.apache.kafka.connect.transforms.ReplaceField$Value","transforms.dropLangs.exclude":"langs","transforms.flatten.type":"org.apache.kafka.connect.transforms.Flatten$Value","transforms.renameCid.type":"org.apache.kafka.connect.transforms.ReplaceField$Value","transforms.renameCid.renames":"id.cid:cid","transforms.renameUri.type":"org.apache.kafka.connect.transforms.ReplaceField$Value","transforms.renameUri.renames":"id.uri:uri","transforms.renameHandle.type":"org.apache.kafka.connect.transforms.ReplaceField$Value","transforms.renameHandle.renames":"author.handle:handle","transforms.renameDisplayName.type":"org.apache.kafka.connect.transforms.ReplaceField$Value","transforms.renameDisplayName.renames":"author.displayName:displayName","transforms.renameAvatar.type":"org.apache.kafka.connect.transforms.ReplaceField$Value","transforms.renameAvatar.renames":"author.avatar:avatar","tasks.max":"1","name":"bluesky-apartado-2-generative-ai"},"tasks":[],"type":"source"}

david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.018s)
curl http://localhost:8083/connectors/bluesky-apartado-2-generative-ai/status

{"name":"bluesky-apartado-2-generative-ai","connector":{"state":"RUNNING","worker_id":"127.0.1.1:8083"},"tasks":[{"id":0,"state":"RUNNING","worker_id":"127.0.1.1:8083"},"type":"source"}
```



```

displayName": "SEY00", "avatar": "https://cdn.bsky.app/img/avatar/plain/did:plc:qdwyrx3hujlydqlp
dm3si7qa/bafkreid6w6jfy44iu773nq7pf6f7mlaox75pnds5xuopypyjntirth7jsq@jpeg"}}
{"schema": {"type": "struct", "fields": [{"type": "string", "optional": false, "field": "uri"}, {"type":
"string", "optional": false, "field": "cid"}, {"type": "string", "optional": false, "field": "text"}, {"t
ype": "int64", "optional": true, "name": "org.apache.kafka.connect.data.Timestamp", "version": 1, "fie
ld": "createdAt"}, {"type": "string", "optional": false, "field": "handle"}, {"type": "string", "optiona
l": true, "field": "displayName"}, {"type": "string", "optional": true, "field": "avatar"}], "optional":
false, "name": "status"}, {"payload": {"uri": "at://did:plc:qxdxlfpytidriazlgzoa5wfv/app.bsky.feed.p
ost/3lkxop7zpgm2c", "cid": "bafyreicvjyvmx3cinmrgzfd6dgemnmpjhffudu7zuyrw74iq6kermhbe7m", "text":
"Siempre quiero descubrir más sobre la IA generativa! #GenerativeAI", "createdAt": 1742645036571
, "handle": "d-ramirez.bsky.social", "displayName": null, "avatar": "https://cdn.bsky.app/img/avatar
/plain/did:plc:qxdxlfpytidriazlgzoa5wfv/bafkreid35m5tfhxuiv5bddglrq2cdyfejxzjqxsiesdus5v72mmw
svmxq@jpeg"}}}

```

Y aquí vemos el evento formateado. Una vez más, lo he adjuntado entre los archivos, bajo el nombre de "topic\_formateado.json"

```

{
  "schema": {
    "type": "struct",
    "fields": [
      { "type": "string", "optional": false, "field": "uri" },
      { "type": "string", "optional": false, "field": "cid" },
      { "type": "string", "optional": false, "field": "text" },
      {
        "type": "int64",
        "optional": true,
        "name": "org.apache.kafka.connect.data.Timestamp",
        "version": 1,
        "field": "createdAt"
      },
      { "type": "string", "optional": false, "field": "handle" },
      { "type": "string", "optional": true, "field": "displayName" },
      { "type": "string", "optional": true, "field": "avatar" }
    ],
    "optional": false,
    "name": "status"
  },
  "payload": {
    "uri": "at://did:plc:qxdxlfpytidriazlgzoa5wfv/app.bsky.feed.post/3lkxop7zpgm2c",
    "cid": "bafyreicvjyvmx3cinmrgzfd6dgemnmpjhffudu7zuyrw74iq6kermhbe7m",
    "text": "Siempre quiero descubrir más sobre la IA generativa! #GenerativeAI",
    "createdAt": 1742645036571,
    "handle": "d-ramirez.bsky.social",
    "displayName": null,
    "avatar": "https://cdn.bsky.app/img/avatar/plain/did:plc:qxdxlfpytidriazlgzoa5wfv"
  }
}

```

El siguiente proceso va a ser preparar la base de datos en MySQL, configurar el *sink* para almacenar los datos y lanzar el POST del *sink*.

```
CREATE DATABASE bluesky_db;

USE bluesky_db;

-- El código que crea la tabla no lo he ejecutado ya que
-- lo ejecuta el archivo del sink automáticamente.

-- Lo dejo aquí anotado simplemente para mostrar la estructura de la tabla

CREATE TABLE generative_ai_posts (
  uri VARCHAR(255),
  cid VARCHAR(255),
  text VARCHAR(255),
  createdAt DATETIME,
  handle VARCHAR(255),
  displayName VARCHAR(255) DEFAULT NULL,
  avatar VARCHAR(500) DEFAULT NULL
);
```

david@srv-ubuntu ~/Desktop/kafka\_2.13-3.8.1 (5m 45.72s)

**nano config/mysql-sink.json**

david@srv-ubuntu:~/Desktop/kafka\_2.13-3.8.1 (0.008s)

**cat config/mysql-sink.json**

```
{
  "name": "mysql-sink-bluesky",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
    "tasks.max": 1,
    "topics": "bluesky_apartado_2",

    "connection.url": "jdbc:mysql://localhost/bluesky_db",
    "connection.user": "root",
    "connection.password": "root1234",

    "auto.create": "true",
    "auto.evolve": "true",
    "insert.mode": "insert",

    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter.schemas.enable": "true",

    "transforms": "ReplaceFieldNames",
    "transforms.ReplaceFieldNames.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.ReplaceFieldNames.whitelist": "uri,cid,text,createdAt,handle,displayName,avatar"
  }
}
```

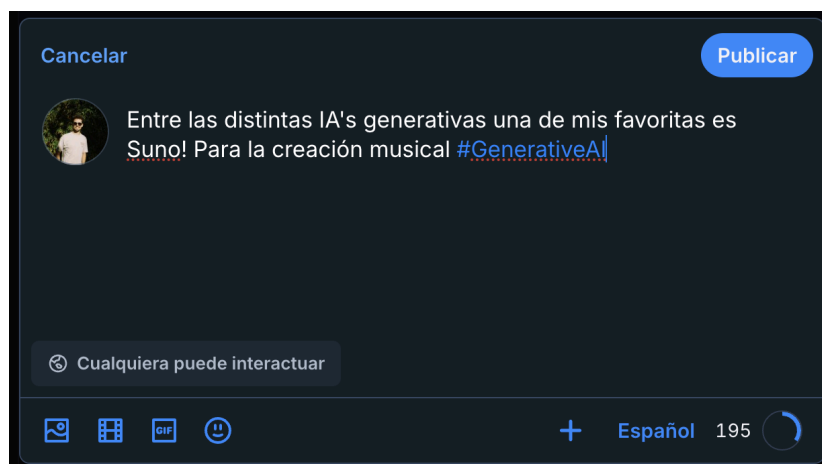
```
curl -X POST -H "Content-Type: application/json" --data @config/mysql-sink.json http://localhost:8083/connectors
```

```
{
  "name": "mysql-sink-bluesky",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
    "tasks.max": "1",
    "topics": "bluesky_apartado_2",
    "connection.url": "jdbc:mysql://localhost/bluesky_db",
    "connection.user": "root",
    "connection.password": "root1234",
    "auto.create": "true",
    "auto.evolve": "true",
    "insert.mode": "insert",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter.schemas.enable": "true",
    "transforms": "ReplaceFieldNames",
    "transforms.ReplaceFieldNames.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
    "transforms.ReplaceFieldNames.whitelist": "uri,cid,text,createdAt,handle,displayName,avatar",
    "name": "mysql-sink-bluesky"
  },
  "tasks": [],
  "type": "sink"
}
```

```
david@srv-ubuntu ~/Desktop/kafka_2.13-3.8.1 (0.013s)
curl http://localhost:8083/connectors
["bluesky-apartado-2-generative-ai", "mysql-sink-bluesky"]
```

Como última prueba creamos un post que será guardado en la tabla de la base de datos en MySQL.

**Nota:** En este punto he tenido que crear el post dos veces ya que tenía problemas con el archivo del sink, esa es la razón por la que está duplicado dentro de la base de datos. Además, borré el primer post de todos los que creé por motivos de pruebas durante el ejercicio, por lo que tampoco aparece en la tabla de la base de datos.



```
mysql> SELECT handle, text, createdAt FROM bluesky_apartado_2 WHERE handle="d-ramirez.bsky.social"
-> ;
```

handle	text	createdAt
d-ramirez.bsky.social	Siempre quiero descubrir más sobre la IA generativa! #GenerativeAI	2025-03-22 12:03:56.571
d-ramirez.bsky.social	Entre las distintas IA's generativas una de mis favoritas es Suno! Para la creación musical #GenerativeAI	2025-03-22 14:54:46.597
d-ramirez.bsky.social	Entre las distintas IA's generativas una de mis favoritas es Suno! Para la creación musical #GenerativeAI	2025-03-22 16:30:29.330

```
3 rows in set (0.01 sec)
```