



&lt;&gt; Code

Issues

Pull requests

Actions

Projects

Wiki

Security



main

ud7-practica-1-kafka-Dansarasix-DML / Readme.md



Dansarasix-DML Update Readme.md

79ccfb1 · 2 months ago



375 lines (293 loc) · 13.6 KB

Preview

Code

Blame



Raw



# Big Data Aplicado

## UD 7 - Apache Kafka

### Práctica 1 Kafka

1. Configura un cluster de Kafka junto a un cluster de Kafka Connect como el explicado en clase. Tienes todas las instrucciones en la [documentación del curso](#)

Lo primero que debemos hacer es descargar Kafka desde su página oficial.



GET STARTED

DOCS

POWERED BY

COMMUNITY

APACHE

DOWNLOAD KAFKA

## DOWNLOAD

The project goal is to have 3 releases a year, which means a release every 4 months. Bugfix releases are made as needed for supported releases only. It is possible to verify every download by following these [procedures](#) and using these [KEYS](#).

### SUPPORTED RELEASES

#### 4.0.0

- Released March 18, 2025
- [Release Notes](#)
- Docker image: [apache/kafka:4.0.0](#)
- Docker Native image: [apache/kafka-native:4.0.0](#)
- Source download: [kafka-4.0.0-src.tgz](#) ([asc](#), [sha512](#))
- Binary download: [kafka\\_2.13-4.0.0.tgz](#) ([asc](#), [sha512](#))

Kafka 4.0.0 includes a significant number of new features and fixes. For more information, please read our [blog post](#), the detailed [Upgrade Notes](#)

Lo hacemos con este comando:

```
wget https://d1cdn.apache.org/kafka/3.9.0/kafka_2.13-3.9.0.tgz
```



```
hadoop@master:~$ wget https://dlcdn.apache.org/kafka/3.9.0/kafka_2.13-3.9.0.tgz
--2025-04-04 10:15:51-- https://dlcdn.apache.org/kafka/3.9.0/kafka_2.13-3.9.0.tgz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 122037770 (116M) [application/x-gzip]
Saving to: 'kafka_2.13-3.9.0.tgz'

kafka_2.13-3.9.0.tgz      100%[=====>] 116,38M  13,2MB/s   in 6,4s
2025-04-04 10:16:00 (18,1 MB/s) - 'kafka_2.13-3.9.0.tgz' saved [122037770/122037770]
```


Luego descomprimos los archivos y los movemos a /opt .

```
tar -xzf kafka_2.13-3.9.0.tgz
cd /opt/kafka_2.13-3.9.0
```



```
hadoop@master:~$ tar -xzf kafka_2.13-3.9.0.tgz
hadoop@master:~$ sudo mv kafka_2.13-3.9.0 /opt/kafka_2.13-3.9.0
[sudo] password for hadoop:
hadoop@master:~$ cd /opt/kafka_2.13-3.9.0
hadoop@master:/opt/kafka_2.13-3.9.0$ ls
bin  config  libs  LICENSE  licenses  NOTICE  site-docs
```

Para Kafka-Connect lo mismo, vamos a la página de los plugins de Kafka-Connect y los descargamos. Debemos descargarlos manualmente ya que no tenemos Confluent.



Version 10.8.2

**Plugin type:** ①  
Sink, Source

**Enterprise support:** ①  
Confluent supported

**Verification:** ①  
Confluent built

**License:** ①

### JDBC Connector (Source and Sink)

The JDBC source and sink connectors allow you to exchange data between relational databases and Kafka. The JDBC source connector allows you to import data from any relational database with a JDBC driver into Kafka topics.

Show more ▾

#### Choose how to deploy your JDBC Connector (Source and Sink)

Available	Self-managed	Custom connect
<b>Self-Hosted</b> <p>The JDBC Connector (Source and Sink) is available as a self-hosted connector.</p>	<b>Confluent Platform</b> <ul style="list-style-type: none"> <li>✓ Everything in Self-Hosted</li> <li>+ Access to Confluent proprietary connectors</li> <li>+ Self-managed Schema Registry</li> <li>+ Confluent Platform support</li> </ul>	<b>Confluent Cloud</b> <ul style="list-style-type: none"> <li>+ Manage connectors (UI, TF, CLI, REST)</li> <li>+ Deploy custom connectors</li> <li>+ Enterprise support for Kafka Connect infrastructure</li> <li>+ Pay-as-you-go</li> </ul>

Lo hacemos con el siguiente comando, descargamos y descomprimos los archivos en la carpeta /opt/kafka/plugins que no es el directorio de Kafka sino el directorio de ejemplos de Kafka.

```
wget https://hub-downloads.confluent.io/api/plugins/confluentinc/kafka-connect-jdbc/versions/10.8.2/confluentinc-kafka-connect-jdbc-10.8.2.zip
unzip confluentinc-kafka-connect-jdbc-10.8.2.zip
```



En la página oficial de Confluent se menciona que hay un conector específico para MySQL. Por ello, lo descargamos también.

The screenshot shows the Confluent Documentation page for the MySQL Server connector. The left sidebar contains a navigation menu with links to JDBC Connector, JDBC Source, JDBC Sink, JDBC Drivers (highlighted), Changelog, Third Party Libraries, and Glossary. Below the menu, it indicates the version is 10.8 (current). The main content area is titled 'MySQL Server' and describes how to connect to MySQL using the Connect/J JDBC driver. It provides instructions on downloading the 'Compressed TAR Archive' and extracting the contents to a temporary directory. The right sidebar, titled 'On this page:', lists links to General guidelines, Microsoft SQL Server, Kerberos Authentication, PostgreSQL Database, Oracle Database, Kerberos authentication, IBM DB2, and MySQL Server (highlighted).

```
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-j-9.2.0.tar.gz
tar -xzf mysql-connector-j-9.2.0.tar.gz
```

Como vamos a trabajar con HDFS, necesitamos el correspondiente *conector sink* para guardar los datos en HDFS.

The screenshot displays the 'HDFS 3 Sink Connector' page. It features a large icon of the connector and a 'Version 1.2.5' badge. Below the icon, it lists the plugin type as 'Sink', enterprise support as 'Confluent supported', verification as 'Confluent built', and license as 'Confluent Software Evaluation License'. The main section, 'Choose how to deploy your HDFS 3 Sink Connector', offers three deployment options: 'Self-Hosted' (marked 'Not available'), 'Confluent Platform' (marked 'Self-managed'), and 'Confluent Cloud' (marked 'Custom connect'). Each option lists its features, such as 'Everything in Self-Hosted', 'Access to Confluent proprietary connectors', and 'Manage connectors (UI, TF, CLI, REST)' for the Cloud option.

```
wget https://hub-downloads.confluent.io/api/plugins/confluentinc/kafka-connect-hdfs3/versions/1.2.3/confluentinc-kafka-connect-hdfs3-1.2.3.zip
unzip confluentinc-kafka-connect-hdfs3-1.2.3.zip
```

El resultado final debe ser el siguiente:

```
hadoop@master:/opt/kafka/plugins$ ls
confluentinc-kafka-connect-hdfs3-1.2.3
confluentinc-kafka-connect-hdfs3-1.2.3.zip
confluentinc-kafka-connect-jdbc-10.8.2
confluentinc-kafka-connect-jdbc-10.8.2.zip
mysql-connector-j-9.2.0
mysql-connector-j-9.2.0.tar.gz
```

2. Cambia el directorio `ejemplo4` por práctica 1 más tus iniciales. Por ejemplo, en mi caso `practica1_JRR`

Creamos la carpeta `practica1_DML`.

```
hadoop@master:/opt/kafka$ mkdir practica1_DML
```

En esta carpeta tendremos la siguiente estructura:

```
practica1_DML
|
|----- config # Archivos de configuración
|
|----- logs   # Logs de Kafka
|
|----- libs    # Librerías/Plugins que se vayan a utilizar
```



3. Cambia el nombre del archivo de configuración de los nodos del cluster para que acaben por tus iniciales. Por ejemplo, en mi caso:

`controller1.properties_JRR`, `broker1.properties_JRR`, `broker2.properties_JRR`

Copiamos los archivos y los movemos al directorio `config`.

```
cp config/kraft/controller.properties
/opt/kafka/practica1_DML/config/controller1.properties_DML
cp config/kraft/broker.properties
/opt/kafka/practica1_DML/config/broker1.properties_DML
cp config/kraft/broker.properties
/opt/kafka/practica1_DML/config/broker2.properties_DML
```



```
hadoop@master:/opt/kafka/practica1_DML/config$ ls
broker1.properties_DML controller1.properties_DML
broker2.properties_DML
```

También debemos copiar los plugins que vayamos a necesitar a la carpeta `libs`:

```
cp -r confluentinc-kafka-connect-jdbc-10.8.2/
/opt/kafka/practica1_DML/libs/
```



```
cp mysql-connector-j-9.2.0/mysql-connector-j-9.2.0.jar
/opt/kafka/practica1_DML/libs/confluentinc-kafka-connect-jdbc-10.8.2/lib
cp -r confluentinc-kafka-connect-hdfs3-1.2.3/
/opt/kafka/practica1_DML/libs/
```

La configuración para los archivos es la siguiente:

```
# Para controller1.properties_DML
# Server Basics
process.roles=controller
node.id=1
controller.quorum.voters=1@localhost:9093
# Socket Server Settings
listeners=CONTROLLER://localhost:9093
controller.listener.names=CONTROLLER
# Log Basics
log.dirs=/opt/kafka/practica1_DML/logs/controller1
```



```
# Para broker1.properties_DML
# Server Basics
process.roles=broker
node.id=2
controller.quorum.voters=1@localhost:9093
# Socket Server Settings
listeners=PLAINTEXT://localhost:9094
advertised.listeners=PLAINTEXT://localhost:9094
# Log Basics
log.dirs=/opt/kafka/practica1_DML/logs/broker1
```



```
# Para broker2.properties_DML
# Server Basics
process.roles=broker
node.id=3
controller.quorum.voters=1@localhost:9093
# Socket Server Settings
listeners=PLAINTEXT://localhost:9095
advertised.listeners=PLAINTEXT://localhost:9095
# Log Basics
log.dirs=/opt/kafka/practica1_DML/logs/broker2
```



Podemos probar que todo funciona:

Primero debemos crear un ID para nuestro cluster de Kafka y luego formatear los directorios logs .

```
#Genera un cluster UUID
KAFKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
echo $KAFKA_CLUSTER_ID

#Formateamos los directorios de log
bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c
/opt/kafka/practica1_DML/config/controller1.properties_DML
bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c
/opt/kafka/practica1_DML/config/broker1.properties_DML
bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c
/opt/kafka/practica1_DML/config/broker2.properties_DML
```



```
hadoop@master:/opt/kafka_2.13-3.9.0$ KAFKA_CLUSTER_ID="$(bin/kafka-
storage.sh random-uuid)"
echo $KAFKA_CLUSTER_ID
imcPjSEeSSWtgrduRKNLkg
```

```
hadoop@master:/opt/kafka_2.13-3.9.0$ bin/kafka-storage.sh format -t
$KAFKA_CLUSTER_ID -c /opt/kafka/practica1_DML/config/controller1.p
roperties_DML
bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c /opt/kafka/prac
tica1_DML/config/broker1.properties_DML
bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c /opt/kafka/prac
tica1_DML/config/broker2.properties_DML
Formatting metadata directory /opt/kafka/practica1_DML/logs/control
ler1 with metadata.version 3.9-IV0.
Formatting metadata directory /opt/kafka/practica1_DML/logs/broker1
with metadata.version 3.9-IV0.
Formatting metadata directory /opt/kafka/practica1_DML/logs/broker2
with metadata.version 3.9-IV0.
```

Para ejecutar el controller y los brokers haremos los siguientes comandos, uno en una terminal diferente:

```
#Ejecuta el servidor Kafka
bin/kafka-server-start.sh
/opt/kafka/practica1_DML/config/controller1.properties_DML
bin/kafka-server-start.sh
/opt/kafka/practica1_DML/config/broker1.properties_DML
bin/kafka-server-start.sh
/opt/kafka/practica1_DML/config/broker2.properties_DML
```





<pre>[2025-04-06 10:22:51,930] INFO Kafka version: 3.9.0 (org.apache.kaf ka.common.utils.AppInfoParser) [2025-04-06 10:22:51,932] INFO Kafka commitId: a60e31147e6b01ee (or g.apache.kafka.common.utils.AppInfoParser) [2025-04-06 10:22:51,936] INFO Kafka startTimeMs: 1743934971882 (or g.apache.kafka.common.utils.AppInfoParser) [2025-04-06 10:22:51,944] INFO [KafkaRaftServer nodeId=1] Kafka Ser ver started (kafka.server.KafkaRaftServer) [2025-04-06 10:22:52,595] INFO [ControllerRegistrationManager id=1 incarnation=wOKFbGxpTkGmUMDIII-ugw] Our registration has been persi sted to the metadata log. (kafka.server.ControllerRegistrationManag er) [2025-04-06 10:22:52,632] INFO [ControllerRegistrationManager id=1 incarnation=wOKFbGxpTkGmUMDIII-ugw] RegistrationResponseHandler: co ntroller acknowledged ControllerRegistrationRequest. (kafka.server. ControllerRegistrationManager)</pre> <p><b>Controller 1</b></p>	<pre>[2025-04-06 10:23:22,299] INFO [BrokerServer id=2] Waiting for all of the SocketServer Acceptors to be started (kafka.server.BrokerSer ver) [2025-04-06 10:23:22,303] INFO [BrokerServer id=2] Finished waiting for all of the SocketServer Acceptors to be started (kafka.server. BrokerServer) [2025-04-06 10:23:22,312] INFO [BrokerServer id=2] Transition from STARTING to STARTED (kafka.server.BrokerServer) [2025-04-06 10:23:22,317] INFO Kafka version: 3.9.0 (org.apache.kaf ka.common.utils.AppInfoParser) [2025-04-06 10:23:22,319] INFO Kafka commitId: a60e31147e6b01ee (or g.apache.kafka.common.utils.AppInfoParser) [2025-04-06 10:23:22,322] INFO Kafka startTimeMs: 1743935002314 (or g.apache.kafka.common.utils.AppInfoParser) [2025-04-06 10:23:22,337] INFO [KafkaRaftServer nodeId=2] Kafka Ser ver started (kafka.server.KafkaRaftServer)</pre> <p><b>BROKER 1</b></p>
<pre>[2025-04-06 10:23:33,988] INFO [BrokerServer id=3] Waiting for all of the SocketServer Acceptors to be started (kafka.server.BrokerSer ver) [2025-04-06 10:23:33,990] INFO [BrokerServer id=3] Finished waiting for all of the SocketServer Acceptors to be started (kafka.server. BrokerServer) [2025-04-06 10:23:33,992] INFO [BrokerServer id=3] Transition from STARTING to STARTED (kafka.server.BrokerServer) [2025-04-06 10:23:33,994] INFO Kafka version: 3.9.0 (org.apache.kaf ka.common.utils.AppInfoParser) [2025-04-06 10:23:33,994] INFO Kafka commitId: a60e31147e6b01ee (or g.apache.kafka.common.utils.AppInfoParser) [2025-04-06 10:23:33,994] INFO Kafka startTimeMs: 1743935013992 (or g.apache.kafka.common.utils.AppInfoParser) [2025-04-06 10:23:34,028] INFO [KafkaRaftServer nodeId=3] Kafka Ser ver started (kafka.server.KafkaRaftServer)</pre> <p><b>BROKER 2</b></p>	<p>72 de estas son actualizaciones de seguridad estándares. Para ver estas actualizaciones adicionales, ejecute: <code>apt list --upgradable</code></p> <p>Active ESM Apps para recibir futuras actualizaciones de se guridad adicionales. Vea <a href="https://ubuntu.com/esm">https://ubuntu.com/esm</a> o ejecute «<code>sudo pro status</code>»</p> <p>New release '24.04.2 LTS' available. Run 'do-release-upgrade' to upgrade to it.</p> <p>Last login: Sun Apr 6 10:01:47 2025 from 192.168.165.1 <b>hadoop@master:~\$</b></p>

Vemos que en todos sale la frase "Kafka Server started", lo cual indica que todo ha ido bien. Para parar los procesos simplemente hacemos `ctrl+c`. Primero paramos los brokers ya que el controller no se parará hasta que no haya ningún broker activo.

Además, podemos crear un topic:

```
bin/kafka-topics.sh --create --topic empleados-employees --bootstrap-
server localhost:9094 --replication-factor 2 --partitions 2
bin/kafka-topics.sh --describe --topic empleados-employees --bootstrap-
server localhost:9094
```

```
hadoop@master:/opt/kafka_2.13-3.9.0$ bin/kafka-topics.sh --create --
--topic empleados-employees --bootstrap-server localhost:9094 --repl
ication-factor 2 --partitions 2
Created topic empleados-employees.
hadoop@master:/opt/kafka_2.13-3.9.0$ bin/kafka-topics.sh --describe
--topic empleados-employees --bootstrap-server localhost:9094
Topic: empleados-employees      TopicId: EebwBITiSbSw34fsf406lg Par
titionCount: 2  ReplicationFactor: 2  Configs: segment.bytes=1073
741824
      Topic: empleados-employees  Partition: 0    Leader: 2 R
eplicas: 2,3    Isr: 2,3    Elr:    LastKnownElr:
      Topic: empleados-employees  Partition: 1    Leader: 3 R
eplicas: 3,2    Isr: 3,2    Elr:    LastKnownElr:
```

Y listar los topics activos:

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9094
```

```
hadoop@master:/opt/kafka_2.13-3.9.0$ bin/kafka-topics.sh --list --b
ootstrap-server localhost:9094
empleados-employees
```

#### 4. Haz lo mismo con los archivos de configuración de los workers del cluster Kafka Connect

Como los workers no se pueden copiar (o al menos no se menciona), los creamos directamente en la carpeta `config`. La configuración para cada uno es la siguiente:

```
# Configuración del Worker 1
bootstrap.servers=localhost:9094,localhost:9095
group.id=connect-cluster
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
key.converter.schemas.enable=true
value.converter.schemas.enable=true
offset.storage.topic=connect-offsets
offset.storage.replication.factor=2
config.storage.topic=connect-configs
config.storage.replication.factor=2
status.storage.topic=connect-status
status.storage.replication.factor=2
plugin.path=/opt/kafka/practica1_DML/libs
listeners=http://localhost:8083
```



```
# Configuración del Worker 2
bootstrap.servers=localhost:9094,localhost:9095
group.id=connect-cluster
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
key.converter.schemas.enable=true
value.converter.schemas.enable=true
offset.storage.topic=connect-offsets
offset.storage.replication.factor=2
config.storage.topic=connect-configs
config.storage.replication.factor=2
status.storage.topic=connect-status
status.storage.replication.factor=2
plugin.path=/opt/kafka/practica1_DML/libs
listeners=http://localhost:8084
```



```
# Configuración del Worker 3
bootstrap.servers=localhost:9094,localhost:9095
group.id=connect-cluster
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
key.converter.schemas.enable=true
value.converter.schemas.enable=true
offset.storage.topic=connect-offsets
offset.storage.replication.factor=2
config.storage.topic=connect-configs
config.storage.replication.factor=2
```





```
status.storage.topic=connect-status
status.storage.replication.factor=2
plugin.path=/opt/kafka/practica1_DML/libs
listeners=http://localhost:8085
```

```
hadoop@master:/opt/kafka/practica1_DML/config$ ls
broker1.properties_DML      worker1.properties_DML
broker2.properties_DML      worker2.properties_DML
controller1.properties_DML  worker3.properties_DML
```

Para lanzar los workers haremos los siguientes comandos (cada uno en una terminal):

```
bin/connect-distributed.sh
/opt/kafka/practica1_DML/config/worker1.properties_DML
bin/connect-distributed.sh
/opt/kafka/practica1_DML/config/worker2.properties_DML
bin/connect-distributed.sh
/opt/kafka/practica1_DML/config/worker3.properties_DML
```



<pre>hadoop@master:/opt/kafka/practical_DML/config\$ curl http://localhost:8083/connectors [] hadoop@master:/opt/kafka/practical_DML/config\$ curl http://localhost:8084/connectors [] hadoop@master:/opt/kafka/practical_DML/config\$ curl http://localhost:8085/connectors []</pre>	<pre>[2025-04-06 11:10:49,331] INFO [Worker clientId=connect-localhost:8083, groupId=connect-cluster] Joined group at generation 6 with protocol version 2 and got assignment: Assignment{error=0, leader='connect-localhost:8083-d9ba3af6-d5e6-4c69-9731-e920563ffddb', leaderUrl='http://localhost:8083/', offset=1, connectorIds=[], taskIds=[], revokedConnectorIds=[], revokedTaskIds=[], delay=0} with rebalance delay: 0 (org.apache.kafka.connect.runtime.distributed.DistributedHerder:2648) [2025-04-06 11:10:49,332] INFO [Worker clientId=connect-localhost:8083, groupId=connect-cluster] Starting connectors and tasks using config offset 1 (org.apache.kafka.connect.runtime.distributed.DistributedHerder:1979) [2025-04-06 11:10:49,332] INFO [Worker clientId=connect-localhost:8083, groupId=connect-cluster] Finished starting connectors and tasks (org.apache.kafka.connect.runtime.distributed.DistributedHerder:2008)</pre> <p><b>WORKER 1</b></p>
<pre>fka.connect.runtime.distributed.DistributedHerder:1884) [2025-04-06 11:10:49,347] INFO [Worker clientId=connect-localhost:8084, groupId=connect-cluster] Finished reading to end of log and updated config snapshot, new config log offset: 1 (org.apache.kafka.connect.runtime.distributed.DistributedHerder:1911) [2025-04-06 11:10:49,347] INFO [Worker clientId=connect-localhost:8084, groupId=connect-cluster] Starting connectors and tasks using config offset 1 (org.apache.kafka.connect.runtime.distributed.DistributedHerder:1979) [2025-04-06 11:10:49,348] INFO [Worker clientId=connect-localhost:8084, groupId=connect-cluster] Finished starting connectors and tasks (org.apache.kafka.connect.runtime.distributed.DistributedHerder:2008) [2025-04-06 11:21:50,639] INFO 127.0.0.1 -- [06/abr/2025:11:21:50+0000] "GET /connectors HTTP/1.1" 200 2 "-" "curl/7.81.0" 504 (org.apache.kafka.connect.runtime.rest.RestServer:62)</pre> <p><b>WORKER 2</b></p>	<pre>nect-localhost:8083-d9ba3af6-d5e6-4c69-9731-e920563ffddb', leaderUrl='http://localhost:8083/', offset=1, connectorIds=[], taskIds=[], revokedConnectorIds=[], revokedTaskIds=[], delay=0} with rebalance delay: 0 (org.apache.kafka.connect.runtime.distributed.DistributedHerder:2648) [2025-04-06 11:10:49,334] INFO [Worker clientId=connect-localhost:8085, groupId=connect-cluster] Starting connectors and tasks using config offset 1 (org.apache.kafka.connect.runtime.distributed.DistributedHerder:1979) [2025-04-06 11:10:49,335] INFO [Worker clientId=connect-localhost:8085, groupId=connect-cluster] Finished starting connectors and tasks (org.apache.kafka.connect.runtime.distributed.DistributedHerder:2008) [2025-04-06 11:21:56,808] INFO 127.0.0.1 -- [06/abr/2025:11:21:56+0000] "GET /connectors HTTP/1.1" 200 2 "-" "curl/7.81.0" 206 (org.apache.kafka.connect.runtime.rest.RestServer:62)</pre> <p><b>WORKER 3</b></p>

Vemos que los workers han iniciado, si realizamos el siguiente comando:

```
curl http://localhost:8083/connectors
curl http://localhost:8084/connectors
curl http://localhost:8085/connectors
```



Veremos que devolverá un array vacío indicando que no hay conectores cargados. Los paramos con `ctrl+c`.

5. Configura los mismos conectores source y sink del ejemplo de clase. El nombre de los conectores deberá acabar por tus iniciales, como los puntos anteriores.

Los conectores que usaremos también los guardaremos en la carpeta `config`.

```
// Configuración para mysql-employees-source-connector_DML.json
{
  "name": "mysql-employees-source-connector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
    "tasks.max": "3",
    "connection.url": "jdbc:mysql://localhost:3306/employees",
    "connection.user": "root",
    "connection.password": "1234",
    "table.whitelist": "employees",
    "mode": "incrementing",
    "incrementing.column.name": "emp_no",
    "topic.prefix": "empleados-",
    "logs.dir": "/opt/kafka/practica1_DML/logs/JdbcSource",
    "poll.interval.ms": "5000"
  }
}
```



**¿Por qué usamos localhost?** Esta práctica se puede realizar con un contenedor Docker donde está MySQL, pero como Docker Desktop presenta conflictos con Virtualbox y MySQL se instaló previamente para Hive decidiré usar el MySQL dentro de la máquina virtual. Si se deseara usar un contenedor Docker cambia la `connection.url` por `"jdbc:mysql://IP_EXTERNA:3306/employees"` donde `IP_EXTERNA` es la IP del puerto anfitrión.

Además, `root` es preciso que tenga una contraseña o usar otro usuario, sino no funcionará

```
// Configuración para hdfs3-employees-sink-connector_DML.json
{
  "name": "hdfs3-employees-sink-connector",
  "config": {
    "connector.class": "io.confluent.connect.hdfs3.Hdfs3SinkConnector",
    "tasks.max": "3",
    "confluent.topic.bootstrap.servers": "localhost:9094",
    "topics": "empleados-employees",
    "store.url": "hdfs://cluster-bda:9000/bda/kafka/practica1_DML",
    "logs.dir": "logs/hdfs3sink",
    "format.class": "io.confluent.connect.hdfs3.avro.AvroFormat",
    "path.format": "'year'=YYYY/'month'=MM/'day'=dd",
    "flush.size": "1000",
    "hadoop.conf.dir": "/opt/hadoop-3.4.1/etc/hadoop/",
    "hadoop.home": "/opt/hadoop-3.4.1/"
  }
}
```



Lo siguiente será preparar los datos que usaremos. Para ello, descargaremos los datos con este comando:

```
git clone https://github.com/datacharmer/test_db.git
cd test_db
```



```
hadoop@master:~$ git clone https://github.com/datacharmer/test_db.git
Cloning into 'test_db'...
remote: Enumerating objects: 121, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 121 (delta 44), reused 44 (delta 44), pack-reused 68 (from 1)
Receiving objects: 100% (121/121), 73.43 MiB | 16.21 MiB/s, done.
Resolving deltas: 100% (62/62), done.
```

Luego agregaremos el archivo `employees.sql` a MySQL.

```
mysql -u root -p < employees.sql
```



```
hadoop@master:~$ cd test_db/
hadoop@master:~/test_db$ sudo mysql -u root -p < employees.sql
[sudo] password for hadoop:
Enter password:
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
data_load_time_diff
00:01:42
```

Comprobamos que los datos se han importado en condiciones.

```
mysql> show databases;
+-----+
| Database |
+-----+
| employees |
| information_schema |
| metastore |
| mysql |
| performance_schema |
| sys |
+-----+
6 rows in set (0,32 sec)
```

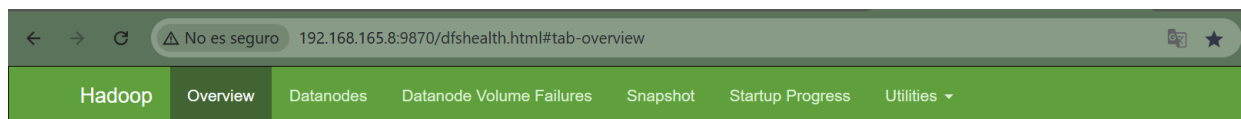
499974	1956-09-10	Shuichi	Piazza	F	1989-09-16
499975	1952-11-09	Masali	Chorvat	M	1992-01-23
499976	1963-08-20	Guozhong	Felder	M	1988-12-26
499977	1956-06-05	Martial	Weisert	F	1996-09-17
499978	1960-03-29	Chiranjit	Kuzuoka	M	1990-05-24
499979	1962-10-29	Prasadram	Waleschkowski	M	1994-01-04
499980	1959-06-28	Gino	Usery	M	1991-02-11
499981	1955-01-02	Yunming	Mitina	F	1991-03-07
499982	1954-08-25	Mohammed	Pleszkun	M	1986-02-21
499983	1955-08-29	Uri	Juneja	F	1989-08-28
499984	1959-08-31	Kaijung	Rodham	M	1985-09-11
499985	1964-12-26	Gila	Lukaszewicz	M	1997-02-11
499986	1952-07-22	Nathan	Ranta	F	1985-08-11
499987	1961-09-05	Rimli	Dusink	F	1998-09-20
499988	1962-09-28	Bangqing	Kleiser	F	1986-06-06
499989	1954-05-26	Keiichiro	Lindqvist	M	1993-10-28
499990	1963-11-03	Khaled	Kohling	M	1985-10-10
499991	1962-02-26	Pohua	Sichman	F	1989-01-12
499992	1960-10-12	Siamak	Salverda	F	1987-05-10
499993	1963-06-04	DeForest	Mullainathan	M	1997-04-07
499994	1952-02-26	Navin	Argence	F	1990-04-24
499995	1958-09-24	Dekang	Lichtner	F	1993-01-12
499996	1953-03-07	Zito	Baaz	M	1990-09-27
499997	1961-08-03	Berhard	Lenart	M	1986-04-21
499998	1956-09-05	Patricia	Breugel	M	1993-10-13
499999	1958-05-01	Sachin	Tsukuda	M	1997-11-30

```
300024 rows in set (0,29 sec)
```

Lo siguiente será tener levantado Hadoop (no es necesario YARN en este caso). Y creamos el fichero correspondiente.

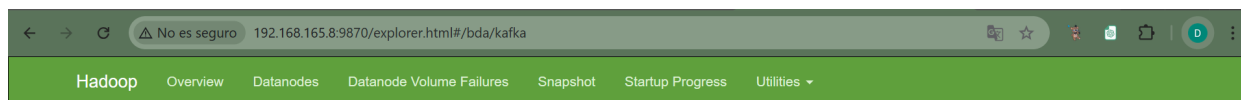
```
hdfs dfs -mkdir -p /bda/kafka/practica1_DML
```





## Overview 'cluster-bda:9000' (✓active)

Started:	Sun Apr 06 12:08:39 +0200 2025
Version:	3.4.1, r4d7825309348956336b8f06a08322b78422849b1
Compiled:	Wed Oct 09 16:57:00 +0200 2024 by mthakur from branch-3.4.1
Cluster ID:	CID-1ef0a261-95dc-43e9-b377-20c6192e6124
Block Pool ID:	BP-1639053288-127.0.1.1-1734535190628



## Browse Directory

/bda/kafka

Go!

Show

25

entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	hadoop	supergroup	0 B	Apr 06 13:51	0	0 B	practica1_DML	

Showing 1 to 1 of 1 entries

Previous

1

Next

Hadoop, 2024.

Una vez todo está preparado, es momento de lanzar los conectores. Hacemos una última comprobación:

```
curl http://localhost:8083/connectors
```



Los pasos a seguir son los siguientes:

1. Usamos la Consumer API para consumir datos:

```
bin/kafka-console-consumer.sh --topic empleados-employees --from-beginning --bootstrap-server localhost:9094
```



2. Iniciamos el primer conector que accede a la base de datos.

```
curl -X POST -H "Content-Type: application/json" --data @/opt/kafka/practica1_DML/config/mysql-employees-source-connector_DML.json http://localhost:8083/connectors
```



3. Iniciamos el segundo conector que accede a HDFS para guardar los datos transformados en formato Avro.

```
curl -X POST -H "Content-Type: application/json" --data
@/opt/kafka/practica1_DML/config/hdfs3-employees-sink-connector_DML.json
http://localhost:8083/connectors
```



### [Ver el vídeo](#)

#### 4. Accedemos a HDFS para ver los resultados

Como ha habido problemas a la hora de acceder, intenté acceder parando los procesos ya que se realentizaba el HDFS. Técnicamente debe salir algo similar a esto:

#### 6. Investiga y usa otros conectores diferentes al del ejemplo de clase (**Opcional**)

#### 7. Requisitos de la entrega. Deben estar en el repositorio

- Todos los archivos de configuración del cluster Kafka
- Todos los archivos de configuración del cluster Kafka Connect
- Todos los archivos multimedia necesarios (imágenes, gifs, videos,...) que demuestren que tienes todo correctamente configurado y funcionando.