



28/04/2024

Apache NIFI

Workflow para inserción de datos en diferentes bases de datos.

Alumno: PEDRO GARCÍA
(Pedro G - pedrokkdownload@gmail.com)

Módulo: NIFI

Docente: ULISES OJEDA

Estudios: Máster experto en Ciencia y Arquitectura de Datos
Ampliación de ARQUITECTURA DE DATOS

Curso: Edición 59

1. INTRODUCCIÓN	1
2. ENTORNO DEL PROYECTO	2
3. CREACIONES AUXILIARES.....	3
4. CASO DE USO.....	5
5. COMPONENTES DEL DESARROLLO.....	6
Grupos creados	6
Servicios de Controlador creados	7
6. DETALLE DE LAS CONFIGURACIONES	8
GR01_Descargar_CSV_y_Convertir_a_JSON	8
GR02_JSON_a_Postgres_y_a_Mongo_y_a_MySQL.....	8
7. COMPROBACIONES DE EJECUCIÓN	10
8. CONCLUSIONES	12

1. Introducción

Este proyecto corresponde al módulo Apache NIFI del máster de BIGDATA de la empresa Datahack en su edición 59.

Descripción de la situación planteada

El objetivo de la práctica es consolidar los conocimientos mediante un proyecto de NiFi que integre varios de los operadores y componentes estudiados durante las clases.

Se desarrollará la práctica con los siguientes requisitos obligatorios:

- Implementar un workflow con una finalidad abierta.
- Utilizar al menos 3 procesadores.
- Procesar algún tipo de dato externo: ficheros, eventos, resultado de queries, HTTP, etc.

Se valorará positivamente la interacción con bases de datos, el uso de Expression Language, las buenas prácticas al definir el flujo y el uso de grupos.

Entregable del proyecto

El entregable consta de un archivo zip con los siguientes archivos:

- Memoria explicativa en formato pdf: memoria_pedrog_practica_nifi.pdf
- Archivo docker-compose.yml para levantar el entorno.
- Archivo json resultante de exportar el flow.

Software utilizado

Plataforma utilizada:

- Intel con arquitectura de 64 bits

Aplicaciones utilizadas en el desarrollo del proyecto:

- Windows 11 v. 23H2
- Firefox Browser v.123.0
- Docker Desktop v.4.27
- Visual Studio Code v. 1.86
- PowerShell v.7.4.2

2. Entorno del proyecto

docker-compose.yml

Vamos a utilizar el fichero docker-compose.yml, con el mismo versionado de las imágenes que las utilizadas en las clases de NiFi. Los contenedores creados del docker-compose.yml son:

- ☐ Servicios de Bases de datos: db (base de datos postgres), mongo, mysql, redis
- ☐ Resto de Servicios del entorno: bastion, , mykafka, nifi, rabbit, registry, zookeeper

Chequeos desde consola: Comprobamos contenedores y salida de mensajes



























```

PS C:\Users\Pedro\Documents\master99\nifi_practica> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
2dc3f05358c    bitnami/kafka:2.6.0-debian-10-r3   "/opt/bitnami/script-   26 seconds ago Up 22 seconds 0.0.0.0:9092->9092/tcp
d63ff35597a4    rabbitmq:3.13-management           "docker-entrypoint.s-   26 seconds ago Up 22 seconds 4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp,
0.0.0.0:15672->15672/tcp
9a1aebc0133     nifi_practica-nifi-1               "nifi/scripts/start.sh" 26 seconds ago Up 23 seconds 0.0.0.0:5555->5555/tcp, 8080/tcp, 0.0.0.0:8443->8443/tcp, 10080/tcp
b577a34d1685    mysql:8.3.0                         "docker-entrypoint.s-   26 seconds ago Up 23 seconds 0.0.0.0:3306->3306/tcp, 33060/tcp
9e273be95bd6    apache/nifi-registry:2.0.0-M2      "nifi/scripts/start.sh" 26 seconds ago Up 23 seconds 0.0.0.0:18080->18080/tcp, 18443/tcp
45d6e14c14d1    redis:latest                       "docker-entrypoint.s-   26 seconds ago Up 23 seconds (healthy) 6379/tcp
f959dc5b886     postgres:alpine3.19              "docker-entrypoint.s-   26 seconds ago Up 23 seconds 0.0.0.0:5432->5432/tcp
7386c28d37a7    bitnami/zookeeper:latest          "/opt/bitnami/script-   26 seconds ago Up 24 seconds 2181/tcp, 2888/tcp, 3888/tcp, 8080/tcp
836d3f41c811    ubuntu:22.04                      "sleep 10000"           26 seconds ago Up 24 seconds
fa8db28f3d37    nifi_practica-mongo-1             "/bin/bash /tmp/init-   26 seconds ago Up 22 seconds 0.0.0.0:27017->27017/tcp
nifi_practica-mongo-1

nifi-1 | 2024-04-25 11:03:16,554 INFO [Checkpoint FlowFile Repository] o.a.n.c.r.WriteAheadFlowFileRepository Successfully checkpointed FlowFile Repository
registry-1 | 2024-04-25 11:03:24,694 INFO [NiFi Registry Web Server-45] o.a.n.r.w.m.IllegalStateExceptionMapper java.lang.IllegalStateException: Access tokens
flict response.
registry-1 | 2024-04-25 11:03:24,713 INFO [NiFi Registry Web Server-47] o.a.n.r.w.m.IllegalStateExceptionMapper java.lang.IllegalStateException: User authenti
n running over HTTPS.. Returning Conflict response.
nifi-1 | 2024-04-25 11:03:36,553 INFO [Checkpoint FlowFile Repository] o.a.n.c.r.WriteAheadFlowFileRepository Initiating checkpoint of FlowFile Repository
nifi-1 | 2024-04-25 11:03:36,553 INFO [Checkpoint FlowFile Repository] o.a.n.c.r.WriteAheadFlowFileRepository Successfully checkpointed FlowFile Repositor
mongo-1 | {"t":{"$date":"2024-04-25T11:03:42.740+00:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"Checkpointner", "msg":"WiredTiger message", "attr":{"m
d7080}], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 35, snapshot max: 35 snapshot count: 0, oldest timestamp: (0, 0) ,
te gen: 7"}
nifi-1 | 2024-04-25 11:03:56,553 INFO [Checkpoint FlowFile Repository] o.a.n.c.r.WriteAheadFlowFileRepository Initiating checkpoint of FlowFile Repository
nifi-1 | 2024-04-25 11:03:56,553 INFO [Checkpoint FlowFile Repository] o.a.n.c.r.WriteAheadFlowFileRepository Successfully checkpointed FlowFile Repositor

```

Chequeos desde docker desktop: Contenedores y Volúmenes

Name ↑	Image	Status	CPU (%)	Port(s)	Last started
 nifi_practica		Running (10/1)	2.79%		41 minutes ago
 bastion-1 d6ead34fe0b1 	ubuntu:22.04	Running	0%		41 minutes ago
 db-1 b9bace247296 	postgres:alpine3.19	Running	0%	5432:5432 	41 minutes ago
 mongo-1 f4cd991a9adc 	nifi_practica-mongo	Running	0.34%	27017:27017 	41 minutes ago
 mykafka-1 d9af0563d58e 	bitnami/kafka:2.6.0-debian-10-r3	Running	0.48%	9092:9092 	41 minutes ago
 mysql-1 e6adcad8e639 	mysql:8.3.0	Running	0.53%	3306:3306 	41 minutes ago
 nifi-1 b09b6ca16b01 	nifi_practica-nifi	Running	0.89%	5555:5555  Show all ports (2)	41 minutes ago
 rabbit-1 6cbd86092a51 	rabbitmq:3.13-management	Running	0.18%	15672:15672  Show all ports (2)	41 minutes ago
 redis-1 668a59e499b2 	redis:latest	Running	0.1%		41 minutes ago
 registry-1 c40e0dab9e20 	apache/nifi-registry:2.0.0-M2	Running	0.18%	18080:18080 	41 minutes ago
Name ↑		Status	Created		
nifi_practica_nifi		in use	42 minutes ago		
nifi_practica_registry		in use	42 minutes ago		

3. Creaciones auxiliares

Damos de alta un nuevo bucket local para ir importando las diferentes versiones que vayamos implementado del flow.

New Bucket ✕

Bucket Name

Description

☐ Make publicly visible ?

☐ Keep this dialog open after creating bucket

Crearemos un nuevo registro local para poder utilizar las diferentes versiones del flow importado en el bucket.

Add Registry Client

Name

Type

Description

En las propiedades le indico la url en la que se encuentra el registro

Edit Registry Client

Required field +

Property	Value
URL	<small>?</small> http://registry:18080/nifi-registry/
SSL Context Service	<small>?</small> No value set

Cada vez que realicemos cambios en el flow vamos a ir guardando versiones que luego importaremos

Import New Flow

Flow Name

pedrog_practica_nifi

v.1

Flow Description

Bucket

Bucket_pedrog_practica_nifi

Flow Definition

pedrog_practica_nifi

Looks good!

SELECT FILE

CANCEL

IMPORT

A lo largo de la práctica vamos importando también otros json que nos pueden servir de ayuda para la configuración de los diferentes procesadores

Sort by: Name (a - z) ▼	
Create_Insert_SQL_Table - Bucket_pedrog_practica_nifi Flow	VERSIONS 1
NiFi_Flow_ALL_EXAMPLES - Bucket_pedrog_practica_nifi Flow	VERSIONS 1
pedrog_practica_nifi - Bucket_pedrog_practica_nifi Flow	VERSIONS 4

4. Caso de uso

Vamos a desarrollar un workflow que nos permitirá manejar diversos componentes en NiFi para cumplir con los requisitos de la práctica.

Detallamos a continuación la jerarquía de los grupos creados para nuestro caso de uso, junto con las acciones principales de cada grupo.

NiFi flow

➤ **pedrog_practica_nifi**

- **GR01_Descargar_CSV_y_Convertir_a_JSON**

- *(1). Descargamos un archivo CSV desde una url
- *(2). Almacenamos el archivo CSV en un directorio local
- *(3). Convertimos el archivo CSV a JSON
- *(4). Almacenamos el archivo JSON en un directorio local

- **GR01_Descargar_CSV_y_Convertir_a_JSON**

- *(1). Cogemos el archivo JSON desde un directorio local
- *(2). Creamos la tabla tweets, si no existe, en Postgres
- *(3). Creamos la tabla tweets, si no existe, en MySQL
- *(4). Spliteamos el archivo JSON
- *(5). Cargamos los datos en la tabla tweets en Postgres
- *(6). Cargamos los datos en la tabla tweets en Mongo
- *(7). Cargamos los datos en la tabla tweets en MySQL

En la url https://github.com/pedrokkdownload/nifi_practica hemos creado un repositorio público para el desarrollo de esta práctica. Se ha colocado el fichero tweets_data.csv, que tiene información relacionada con diversas líneas aéreas. La ruta de dicho fichero la hemos utilizado como url para la descarga inicial del flow:

https://raw.githubusercontent.com/pedrokkdownload/nifi_practica/main/tweets_data.csv

5. Componentes del desarrollo

Grupos creados

✓ pedrog_practica_nifi				
🎯 0	🔍 0	▶ 0	■ 17	⚠ 0 ✂ 0
Queued	0 (0 bytes)			
In	0 (0 bytes) → 0			5 min
Read/Write	0 bytes / 0 bytes			5 min
Out	0 → 0 (0 bytes)			5 min
✓ 0	✖ 0	⬆ 0	⚠ 0	? 0





GR01_Descargar_CSV_y_Convertir_a_JSON				
🎯 0	🔍 0	▶ 0	■ 8	⚠ 0 ✂ 0
Queued	0 (0 bytes)			
In	0 (0 bytes) → 0			5 min
Read/Write	0 bytes / 0 bytes			5 min
Out	0 → 0 (0 bytes)			5 min
✓ 0	✖ 0	⬆ 0	⚠ 0	? 0

GR02_JSON_a_Postgres_y_a_Mongo_y_a_MySQL				
🎯 0	🔍 0	▶ 0	■ 9	⚠ 0 ✂ 0
Queued	0 (0 bytes)			
In	0 (0 bytes) → 0			5 min
Read/Write	0 bytes / 0 bytes			5 min
Out	0 → 0 (0 bytes)			5 min
✓ 0	✖ 0	⬆ 0	⚠ 0	? 0

Servicios de Controlador creados








GR01_Descargar_CSV_y_Convertir_a_JSON Configuration

GENERAL	CONTROLLER SERVICES
---------	---------------------

	Name ▲	Type
	AvroReader_GR01	AvroReader 2.0.0-M2
	AvroSchemaRegistry_GR01	AvroSchemaRegistry 2.0.0-M2
	CSVReader_GR01	CSVReader 2.0.0-M2
	JsonRecordSetWriter_GR01	JsonRecordSetWriter 2.0.0-M2

GR02_JSON_a_Postgres_y_a_Mongo_y_a_MySQL Configuration

GENERAL	CONTROLLER SERVICES
---------	---------------------

	Name ▲	Type
	CSVReader_GR02	CSVReader 2.0.0-M2
	CSVRecordSetWriter_GR02	CSVRecordSetWriter 2.0.0-M2
	DBCPConnectionPool_GR02_para_MySQL	DBCPConnectionPool 2.0.0-M2
	DBCPConnectionPool_GR02_para_Postgres	DBCPConnectionPool 2.0.0-M2
	JsonRecordSetWriter_GR02	JsonRecordSetWriter 2.0.0-M2
	JsonTreeReader_GR02	JsonTreeReader 2.0.0-M2
	MongoDBControllerService_GR02	MongoDBControllerService 2.0.0-M2

6. Detalle de las configuraciones

Como resumen de la configuración realizada en los diferentes procesadores, vamos a indicar únicamente los detalles más destacables de la configuración de algunos de los componentes y las acciones obligatorias previas al arranque de los grupos.

GR01 Descargar CSV y Convertir a JSON

Acciones obligatorias previas:

- No es necesario realizar ninguna acción previa

Detalles destacables de configuración

- InvokeHTTP -> SCHEDULING
 - Run Schedule: **10 sec**
- UpdateAttribute (Poner el nombre al archivo JSON) -> PROPERTIES
 - filename: **`${filename:replaceAll('(.*)\\.csv', '$1.json')}`**

GR02 JSON a Postgres y a Mongo y a MySQL

Acciones obligatorias previas (la primera vez que importamos el flow):

- Servicio DBCPConnectionPool_GR02_para_Postgres ->Propiedades
 - Introducir la contraseña en la propiedad Password y aplicar
- Servicio DBCPConnectionPool_GR02_para_MySQL ->Propiedades
 - Introducir la contraseña en la propiedad Password y aplicar

Detalles destacables de configuración

- Conexiones (del procesador Split en adelante)
 - Back Pressure Object Threshold: **20000 (*)**
- PutDatabaseRecord (Insertar datos en Postgres) ->SCHEDULING
 - Concurrent Tasks: **16 (**)**
- PutDatabaseRecord (Insertar datos en Postgres) ->PROPERTIES
 - Schema Name: **public**
 - Table Name: **tweets**

- PutMongoRecord ->SCHEDULING
 - Concurrent Tasks: **16 (**)**
- PutMongoRecord ->PROPERTIES
 - Mongo Database Name: **practica**
 - Mongo Collection Name: **tweets**
- PutDatabaseRecord (Insertar datos en MySQL) ->SCHEDULING
 - Concurrent Tasks: **16 (**)**
- PutDatabaseRecord (Insertar datos en MySQL) ->PROPERTIES
 - Schema Name: **mysql**
 - Table Name: **tweets**

(*) El umbral de la conexión se ha puesto en función del número máx. de registros (14640) existente en el fichero csv con el que se inicia la práctica

(**) El número de tareas concurrentes se ha puesto en función del número max. de hilos (16) de la CPU presente en la plataforma Intel utilizada para el desarrollo de la práctica.

Nota explicativa del diseño

En el flujo se introducen los procesadores *UpdateDatabaseTable*, que realizan la creación de la tabla tweets tanto en Postgres como en MySQL, antes de que se lleve a cabo el *SplitRecord*, para que dicha operación se realice únicamente una vez.

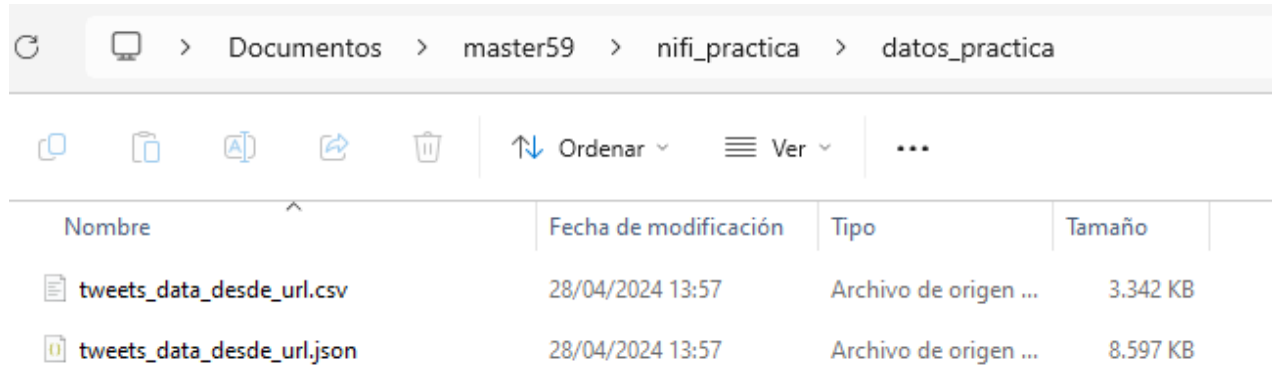
Inicialmente estos procesadores se habían incluido entre el *SplitRecord* y el respectivo *PutDatabaseRecord*. Esto hacía que el tiempo final de inserción de todos los registros en la tabla correspondiente fuese hasta 5 veces más del tiempo que se obtiene en el diseño final. En dicho diseño inicial, las colas existentes entre los *UpdateDatabaseTable* y los *PutDatabaseRecord* recogían registros más lentamente que lo que tardaba el *PutDatabaseRecord* en insertarlos.

El inconveniente de esto, la primera vez que importamos el flow en una nueva plataforma, es tener que cumplir con los dos requisitos obligatorios de introducir las contraseñas respectivas en ambos Servicios antes de que iniciemos el flow. Si no lo hacemos ninguna de las inserciones en las bases de datos funcionará, ya que no llega a procesarse el *SplitRecord*.

7. Comprobaciones de ejecución

GR01 Descargar CSV y Convertir a JSON

Datos en el directorio local



Nombre	Fecha de modificación	Tipo	Tamaño
tweets_data_desde_url.csv	28/04/2024 13:57	Archivo de origen ...	3.342 KB
tweets_data_desde_url.json	28/04/2024 13:57	Archivo de origen ...	8.597 KB

GR02 JSON a Postgres y a Mongo y a MySQL

Datos en Postgres

```
PS C:\Users\Pedro\Documents\master59\nifi_practica> docker exec -it nifi_practica-db-1 bash
32f1da5ae2f6:/# psql -U postgres
psql (16.2)
Type "help" for help.

postgres=# select count(*) from tweets;
 count
-----
 14640
(1 row)

postgres=# |
```

Datos en Mongo

```
PS C:\Users\Pedro\Documents\master59\nifi_practica> docker exec -it nifi_practica-mongo-1 bash
mongodb@22fe69e9c110:/# mongosh --username admin --password admin
Current Mongosh Log ID: 662e42b2846fd107f52202d7
Connecting to: mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.5
Using MongoDB: 5.0.15
Using Mongosh: 2.2.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2024-04-28T08:25:07.997+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/xdisk
2024-04-28T08:25:09.112+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
-----

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test> show dbs
admin      100.00 KiB
config     72.00 KiB
local      72.00 KiB
practica   4.23 MiB
test       16.00 KiB
test> use practica
switched to db practica
practica> db.tweets.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
14640
practica> |
```

Datos en MySQL

```
PS C:\Users\Pedro\Documents\master59\nifi_practica> docker exec -it nifi_practica-mysql-1 bash
bash-4.4# mysql -u root -pmysql
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select count(*) from tweets;
+-----+
| count(*) |
+-----+
|      14640 |
+-----+
1 row in set (0.00 sec)

mysql> |
```

8. Conclusiones

No hemos tenido problemas de versiones ya que se han utilizado las imágenes del docker-compose.yml proporcionado por el docente.

El punto complicado de la practica ha sido haber introducido tres procesadores de diferentes de bases de datos (Postgres, Mongo y MySQL), además haberlos introducido en el mismo grupo.

El camino recorrido en la realización de la práctica ha hecho que el aprendizaje de NiFi haya resultado muy satisfactorio.