

1. Nifi

On commence le travail sur Nifi.

- Editer le dockerfile pour avoir les bons chemins pour le device “share_data”

```
> docker-compose.yml x | hadoop-hive.env x
... docker-compose.yml
236     mysql_data:
237         driver: local # Define the driver and options under the volume name
238         driver_opts:
239             type: none
240             device: /home/rofelson/Documents/01.Projects/Master 2 DIT AI _ Data
241                   Engineering/Big Data Technology/volume_mysql/
242             o: bind
243     share_data:
244         driver: local # Define the driver and options under the volume name
245         driver_opts:
246             type: none
247             device: /home/rofelson/Documents/01.Projects/Master 2 DIT AI _ Data
248                   Engineering/Big Data Technology/shared/
249             o: bind
```

- Copier les fichiers “*-site.xml” vers <shared_data_path>/xml et le connecteur vers <shared_data_path>/jar sur l’hôte
- démarrer les containers avec “podman compose up”
- on se connecte à nifi avec les identifiants qui sont dans le dockerfile
- on va travailler dans le groupe qu’on aura créé “nifi-mysql-connector”
- On rajoute et on paramètre un processeur “QueryDatabaseTableRecord”

Configure Processor | QueryDatabaseTableRecord 1.24.0

Stopped

SETTINGS | SCHEDULING | PROPERTIES | RELATIONSHIPS | COMMENTS

Required field

Property	Value	
Database Connection Pooling Service	QuincaillerieConnectionPool	→
Database Type	MySQL	
Table Name	ventes	
Columns to Return	No value set	
Additional WHERE clause	No value set	
Custom Query	No value set	
Record Writer	CSVRecordSetWriter	→
Maximum-value Columns	id	
Initial Load Strategy	Start at Beginning	
Max Wait Time	0 seconds	
Fetch Size	0	
Max Rows Per Flow File	0	
Output Batch Size	0	
Maximum Number of Fragments	0	
Normalize Table/Column Names	false	
Use Avro Logical Types	false	

CANCEL | APPLY

- ON rajoute et on configure la connexion à la base de donnée qu'on a créée au préalable avec Dbeaver

Controller Service Details
DBCPConnectionPool 1.24.0

SETTINGS
PROPERTIES
COMMENTS

Required field

Property	Value
Database Connection URL	jdbc:mysql://nanp-mysql:3306/quincaillerie
Database Driver Class Name	com.mysql.cj.jdbc.Driver
Database Driver Location(s)	/partage/jar/mysql-connector-j-8.3.0.jar
Kerberos User Service	No value set
Kerberos Credentials Service	No value set
Kerberos Principal	No value set
Kerberos Password	No value set
Database User	root
Password	Sensitive value set
Max Wait Time	500 millis
Max Total Connections	8
Validation query	No value set
Minimum Idle Connections	0
Max Idle Connections	8
Max Connection Lifetime	-1
Time Between Eviction Runs	-1
Minimum Evictable Idle Time	30 mins
Soft Minimum Evictable Idle Time	1

OK

- On rajoute un updateAttribute pour rajouter un nom de fichier et demander à créer le dossier de destination s'il n'existe pas

Configure Processor
UpdateAttribute 1.24.0

Stopped

SETTINGS
SCHEDULING
PROPERTIES
RELATIONSHIPS
COMMENTS

Required field

Property	Value
Delete Attributes Expression	No value set
Store State	Do not store state
Stateful Variables Initial Value	No value set
Cache Value Lookup Cache Size	100
filename	quincaillerie_\${now() :format("yyyyMMddHHmmss")}.csv
target.dir.created	true

- On rajoute un processeur “putHDFS” pour mettre les infos sur le serveur

Stopped

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field



Property		Value
Hadoop Configuration Resources		/partage/xml/core-site.xml,/partage/xml/hdfs-site.xml
Kerberos Credentials Service		No value set
Kerberos User Service		No value set
Kerberos Principal		No value set
Kerberos Keytab		No value set
Kerberos Password		No value set
Kerberos Relogin Period		4 hours
Additional Classpath Resources		No value set
Directory		/tmp/quincaillerie
Conflict Resolution Strategy		fail
Writing Strategy		Simple write
Block Size		10MB
IO Buffer Size		No value set
Replication		1
Permissions umask		No value set
Remote Owner		No value set

CANCEL

APPLY

- On vérifie leur présence sur le namenode pour confirmer le succès de l'opération

Browse Directory

/tmp/quincaillerie

Go!



Show 25 entries

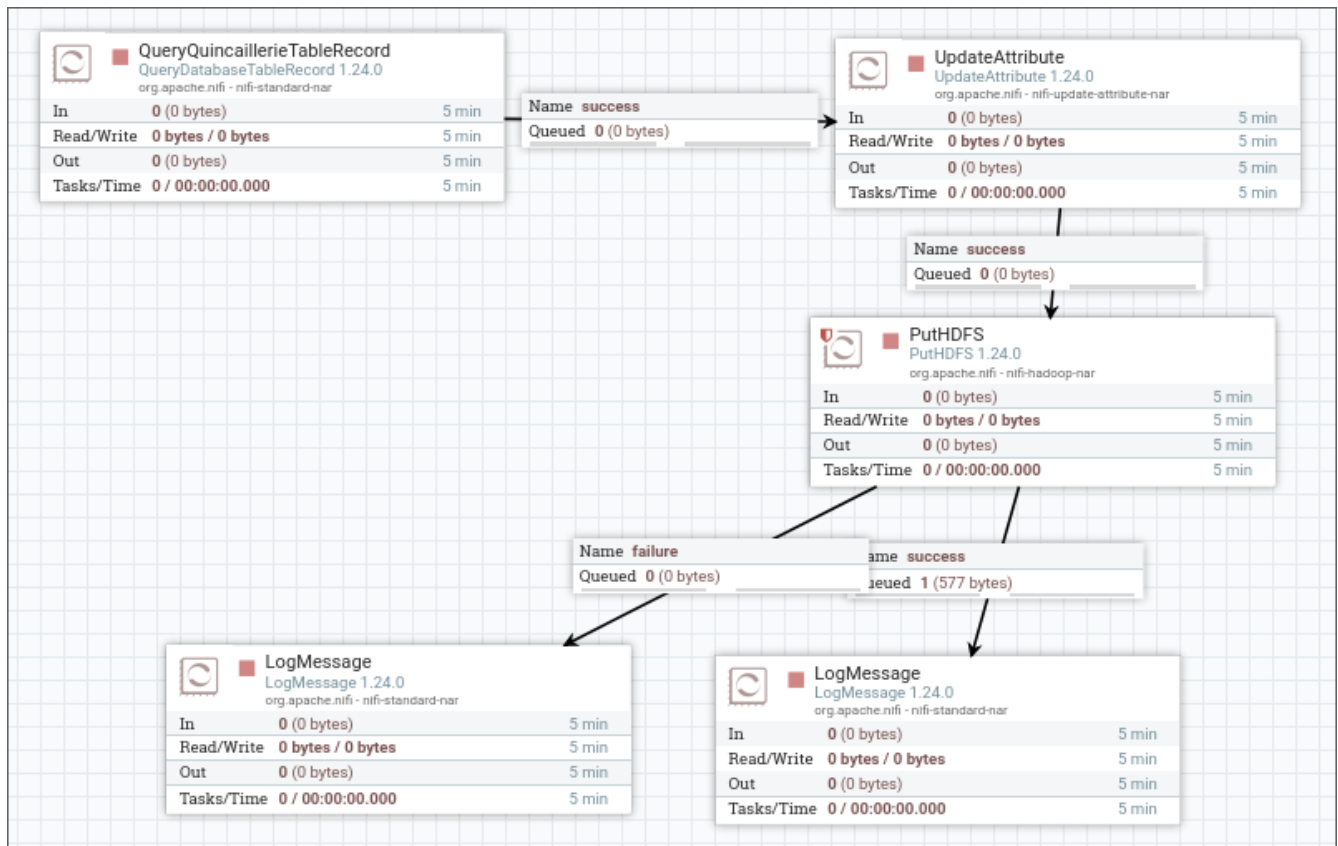
Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	nifi	supergroup	577 B	Mar 30 15:06	1	10 MB	quincaillerie_20240330150605.csv	
<input type="checkbox"/>	-rw-r--r--	nifi	supergroup	577 B	Mar 30 18:52	1	10 MB	quincaillerie_20240330185241.csv	

Showing 1 to 2 of 2 entries

Previous 1 Next

- Le template enregistré sera joint au mail



2. Hive

On va se connecter au container hive:

“podman exec -it hive-server bash”

- On se connecte à hive avec “jdbc:hive2://hive-server:10000/”
- On crée la base de donnée et la table (la capture provient de mes notes)

- Create appropriate db

```
1 CREATE DATABASE IF NOT EXISTS quincaillerie;
2 USE quincaillerie;
```

sql

- Create external table quincaillerie

```
1 DROP TABLE quincaillerie;
2
3 CREATE EXTERNAL TABLE IF NOT EXISTS quincaillerie(
4   id int,
5   code_client string,
6   code_produit string,
7   qte int,
8   date_ops bigint,
9   pu int,
10  montant int
11 )
12 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
13 COLLECTION ITEMS TERMINATED BY '\n'
14 LOCATION '/tmp/quincaillerie'
15 tblproperties ("skip.header.line.count"="1");
```

sql

- On peut observer les données chargées select * from quincaillerie;

0: jdbc:hive2://localhost:10000/quincaillerie> select * from quincaillerie;

quincaillerie.id	quincaillerie.code_client	quincaillerie.code_produit	quincaillerie.qte	quincaillerie.date_ops	quincaillerie.pu	quincaillerie.montant
1	CLI1	PROD1	3	1710806400000	40	120
2	CLI2	PROD1	5	1710806400000	25	125
3	CLI1	PROD2	2	1710806400000	500	1000
4	CLI3	PROD3	6	1710806400000	100	600
5	CLI3	PROD3	7	1710806400000	100	700
6	CLI4	PROD1	1	1710806400000	120	120
7	CLI1	PROD2	3	1710806400000	500	1500
8	CLI3	PROD1	2	1710806400000	300	600
9	CLI5	PROD2	3	1710806400000	400	1200
10	CLI3	PROD1	7	1710633600000	1000	7000
11	CLI3	PROD3	2	1710806400000	400	800
12	CLI4	PROD2	10	1710806400000	20	200
13	CLI3	PROD2	3	1710806400000	200	600
14	CLI4	PROD4	6	1711238400000	20	300

- On crée une vue pour avoir le revenu total de chaque produit par ordre décroissant

- product income view

```
1 CREATE OR REPLACE VIEW product_income AS
2 SELECT code_produit, sum(qte) as qte_vendue, sum(montant) as montant_vendu
3 FROM quincaillerie
4 GROUP BY code_produit
5 ORDER BY montant_vendu DESC;
```

sql

```
0: jdbc:hive2://localhost:10000/quincaillerie> select * from product_income;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+-----+-----+-----+
| product_income.code_produit | product_income.qte_vendue | product_income.montant_vendu |
+-----+-----+-----+
| PROD1 | 36 | 15930 |
| PROD2 | 42 | 9000 |
| PROD3 | 30 | 4200 |
| PROD4 | 12 | 600 |
+-----+-----+-----+
```

- On peut rajouter une vue pour où on remet la date dans un format lisible par l'humain

• datetime of date_ops

```
1 CREATE OR REPLACE VIEW original_vendu AS
2 select id,
3   code_client,
4   code_produit,
5   qte,
6   to_date(from_utc_timestamp(date_ops, 'UTC')) as date_ops,
7   pu,
8   montant
9 FROM quincaillerie;
```

sql

```
0: jdbc:hive2://localhost:10000/quincaillerie> select * from original_vendu
.....> ;
+-----+-----+-----+-----+-----+-----+-----+
| original_vendu.id | original_vendu.code_client | original_vendu.code_produit | original_vendu.qte | original_vendu.date_ops | original_vendu.pu | original_vendu.montant |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | CLI1 | PROD1 | 3 | 2024-03-19 | 40 | 120 |
| 2 | CLI2 | PROD1 | 5 | 2024-03-19 | 25 | 125 |
| 3 | CLI1 | PROD2 | 2 | 2024-03-19 | 500 | 1000 |
| 4 | CLI3 | PROD3 | 6 | 2024-03-19 | 100 | 600 |
| 5 | CLI3 | PROD3 | 7 | 2024-03-19 | 100 | 700 |
| 6 | CLI4 | PROD1 | 1 | 2024-03-19 | 120 | 120 |
| 7 | CLI1 | PROD2 | 3 | 2024-03-19 | 500 | 1500 |
| 8 | CLI3 | PROD1 | 2 | 2024-03-19 | 300 | 600 |
| 9 | CLI5 | PROD2 | 3 | 2024-03-19 | 400 | 1200 |
| 10 | CLI3 | PROD1 | 7 | 2024-03-17 | 1000 | 7000 |
| 11 | CLI3 | PROD3 | 2 | 2024-03-19 | 400 | 800 |
| 12 | CLI4 | PROD2 | 10 | 2024-03-19 | 20 | 200 |
| 13 | CLI3 | PROD2 | 3 | 2024-03-19 | 200 | 600 |
| 14 | CLI4 | PROD4 | 6 | 2024-03-24 | 20 | 300 |
+-----+-----+-----+-----+-----+-----+-----+
```

- ON peut rajouter une vue pour voir le montant dépensé par chaque client

• best spending client view

```
1 CREATE OR REPLACE VIEW client_view AS
2 SELECT code_client, count(code_client) AS nombre_achats, sum(montant) AS montant_vendu
3 FROM quincaillerie
4 GROUP BY code_client
5 ORDER BY montant_vendu DESC;
```

sql

```
0: jdbc:hive2://localhost:10000/quincaillerie> select * from client_view;  
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider  
using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
```

client_view.code_client	client_view.nombre_achats	client_view.montant_vendu
CLI3	12	20600
CLI1	6	5240
CLI5	2	2400
CLI4	6	1240
CLI2	2	250