Atelier 6: Prise en main KsqlDB.

Etape 1 : Récupérer docker-compose.yml

Etape 2 : Lancer la stack

Exécutez la commande

docker-compose up -d

L'option **-d** spécifie le mode détaché - cela suppose que les conteneurs s'exécutent en arrière-plan.

Etape 3 : Vérifier le statut des process

docker-compose ps

Etape 4 : Lancer KsqlDB CLI

docker exec -it ksqldb-cli ksql http://primary-ksqldb-server:8088

```
sh-4.4$ ksql http://ksqldb-server:8088
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0

                  ===========================================
                  =       _              _ ____  ____       =
                  =      | | _____  __ _| |  _ \| __ )      =
                  =      | |/ / __|/ _` | | | | |  _ \      =
                  =      |   <\__ \ (_| | | |_| | |_) |     =
                  =      |_|\_\___/\__, |_|____/|____/      =
                  =                   |_|                   =
                  =        The Database purpose-built       =
                  =        for stream processing apps       =
                  ===========================================

Copyright 2017-2022 Confluent Inc.

CLI v0.27.2, Server v0.27.2 located at http://ksqldb-server:8088
Server Status: RUNNING

Having trouble? Type 'help' (case-insensitive) for a rundown of how things work!

ksql>
```

Etape 5 : Création d'un Stream

kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic ORDERS

ksql> **CREATE STREAM ORDERS (order_id VARCHAR, product_name VARCHAR, number INT, total_price DOUBLE) WITH (VALUE_FORMAT='JSON ', PARTITIONS=1, KAFKA_TOPIC='ORDERS') ;**

ksql> list streams;

ou

ksql> **SHOW STREAMS**;

SET 'auto.offset.reset' = 'earliest';

ksql> **SELECT * FROM ORDERS EMIT CHANGES;**

- **Ouvrir une nouvelle instance de ksqlDB CLI :**

docker exec -it ksqldb-cli ksql http://primary-ksqldb-server:8088

```
ksql> INSERT INTO ORDERS (order_id,product_name,number,total_price)
VALUES ('1','book',3,57.66);
ksql> INSERT INTO ORDERS (order_id,product_name,number,total_price)
VALUES ('2','book',1,13.50);
ksql> INSERT INTO ORDERS (order_id,product_name,number,total_price)
VALUES ('3','laptop',3,1000.99);
ksql>
```

Lancer un consumer Kafka sur le topic ORDERS :

docker exec -it broker*(le nom de votre broker kafka)* **bash** (Pour lancer un terminal)

kafka-console-consumer --topic ORDERS –bootstrap-servers broker:9092 –from-beginning

Agrégation & stockage sur une Table

Création d'une Table

CREATE TABLE PRODUCT_TOTAL_PRICE AS SELECT product_name, SUM(total_price) AS sum_total_price FROM ORDERS GROUP BY product_name;

Insertion des données

```
ksql> INSERT INTO ORDERS (order_id,product_name,number,total_price)
VALUES ('1','book',3,57.66);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('2','book',2,13.50);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('3','laptop',1,1000.99);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('4','laptop',2,2000.00);
```

```
>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('5','book',6,110.99);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('6','headset',1,99.99);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('7','laptop',1,500.99);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('8','book',2,20.17);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('9','headset',5,400.14);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('10','book',1,5.99);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('11','book',5,55.00);

>INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('12','book',2,15.00);
ksql>
```

Création d'un consumer pour le topic PRODUCT_TOTAL_PRICE

```
kafka-console-consumer --topic PRODUCT_TOTAL_PRICE --bootstrap-server
localhost:9092 --from-beginning --property print.key=true --property
key.separator="-"
laptop-{"SUM_TOTAL_PRICE":3501.9799999999996}
headset-{"SUM_TOTAL_PRICE":500.13}
book-{"SUM_TOTAL_PRICE":278.31}
```

Interrogation des données

```
ksql> select * from PRODUCT_TOTAL_PRICE emit changes;
```

Jointure de streams

Création des streams PAYMENTS et ORDERS_AND_PAYMENTS.

```
CREATE STREAM PAYMENTS (order_id VARCHAR, payment_type VARCHAR)
WITH (KAFKA_TOPIC='PAYMENTS', VALUE_FORMAT='json', partitions=1);

CREATE STREAM ORDERS_AND_PAYMENTS
WITH (KAFKA_TOPIC='ORDERS_AND_PAYMENTS', VALUE_FORMAT='json',
partitions=1)
AS SELECT * FROM ORDERS INNER JOIN PAYMENTS
>WITHIN 7 DAYS
>ON ORDERS.order_id = PAYMENTS.order_id;

ksql> show topics;
```

Insertion des données.

```
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('1','book',3,57.66);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('2','book',2,13.50);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('3','laptop',1,1000.99);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('4','laptop',2,2000.00);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('5','book',6,110.99);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('6','headset',1,99.99);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('7','laptop',1,500.99);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('8','book',2,20.17);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('9','headset',5,400.14);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('10','book',1,5.99);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('11','book',5,55.00);
INSERT INTO ORDERS (order_id,product_name,number,total_price) VALUES
('12','book',2,15.00);

INSERT INTO PAYMENTS (order_id, payment_type) VALUES ('1', 'Paypal');
INSERT INTO PAYMENTS (order_id, payment_type) VALUES ('3',
'Mastercard');
INSERT INTO PAYMENTS (order_id, payment_type) VALUES ('2',
'Mastercard');
```

Lancer ensuite un consumer kafka

```
kafka-console-consumer --topic ORDERS_AND_PAYMENTS --bootstrap-server
localhost:9092 --from-beginning --property print.key=true --property
key.separator="-"
```