SOMMAIRE

I - Contexte

Nous mettons en place une base données ci-dessus qui contiendra des tables, liées entre elles avec une clé étrangère, notre base de données doit contenir une procédure stockée, un trigger et des utilisateurs avec des droits restreint. Enfin nous allons mettre en place une réplication bidirectionnelle avec gestion en continuité des services.

II - Création de la base de donnée

Nous créons une databases nommé "Amazon" avec les tables suivantes qui contient des **clés primaires** et des *#clés étrangères* pour les liés entre eux en SQL:

```
MariaDB [(none)]> CREATE DATABASE Amazon ;
Query OK, 1 row affected (0,000 sec)

MariaDB [(none)]> USE Amazon;
E Client (
idCLI varchar(8) NOT NULL PRIMARY KEY,
Nom varchar(12) NOT NULL,
Adresse varchar(20) Database changed
MariaDB [Amazon]> CREATE TABLE Client (
    -> idCLI varchar(8) NOT NULL PRIMARY KEY,
    -> Nom varchar(12) NOT NULL,
    -> Adresse varchar(20) NOT NULL
    -> );
L,
    FOREIGN KEY (idArt) REFERENCES Article(idArt),
    FOREIGN KEY (idCLI) REFERENCES Client(idCLI)
);
Query OK, 0 rows affected (0,003 sec)

MariaDB [Amazon]>
MariaDB [Amazon]>
MariaDB [Amazon]> CREATE TABLE Article (
    ->   idArt VARCHAR(8) PRIMARY KEY,
    ->   Quantite INT NOT NULL,
    ->   PriUnit INT NOT NULL
    -> );
Query OK, 0 rows affected (0,002 sec)

MariaDB [Amazon]>
MariaDB [Amazon]>
MariaDB [Amazon]> CREATE TABLE Commande (
    ->     NumCom INT AUTO_INCREMENT PRIMARY KEY,
    ->     Date DATE NOT NULL,
    ->     Prix float NOT NULL,
    ->     idArt VARCHAR(8),
    ->     idCLI varchar(8) NOT NULL,
    ->     FOREIGN KEY (idArt) REFERENCES Article(idArt),
    ->     FOREIGN KEY (idCLI) REFERENCES Client(idCLI)
    -> );
Query OK, 0 rows affected (0,004 sec)
```

Pour vérifier que notre base de données a bien été créé nous le vérifions avec un **SHOW DATABASES;** et pour l'utiliser nous faisons un **USE Amazon;**

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| Amazon             |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0,001 sec)

MariaDB [(none)]> USE Amazon;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

Pour vérifier la création des tables dans notre base de données Amazon nous faisons un **SHOW TABLES;**

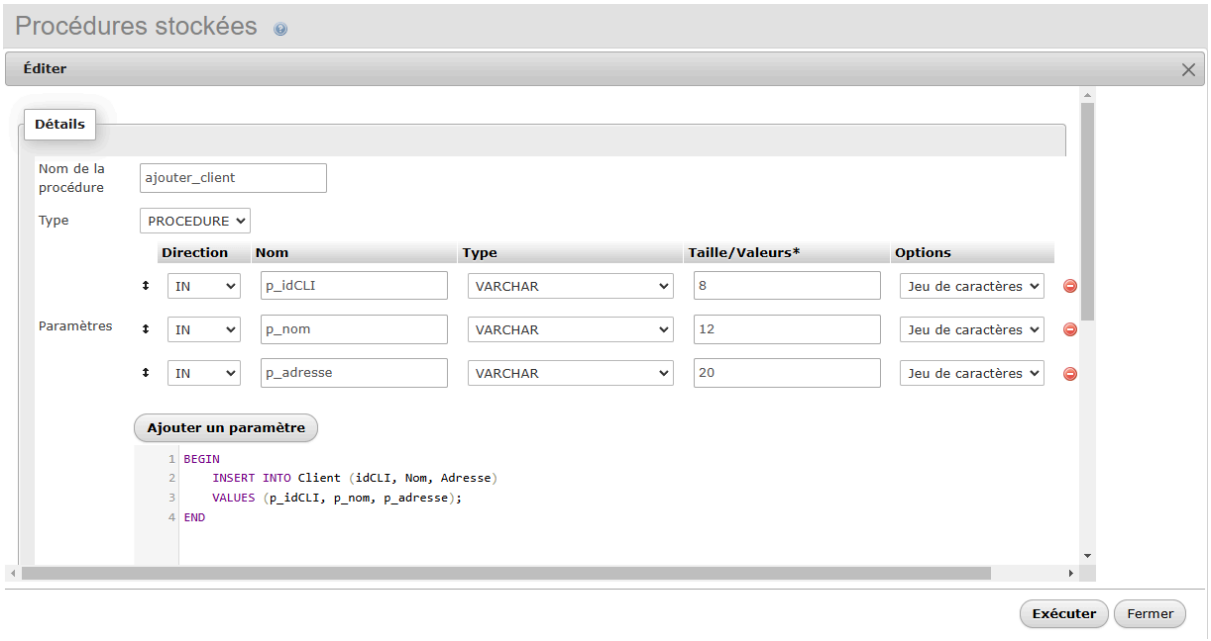
```
MariaDB [Amazon]> SHOW TABLES;
+-----------------+
| Tables_in_Amazon |
+-----------------+
| Article         |
| Client          |
| Commande        |
+-----------------+
3 rows in set (0,000 sec)
```

Nous allons insérer quelques données dans notre base de données Amazon.

```sql
INSERT INTO Client (idCLI, Nom, Adresse) VALUES
('CLI00001', 'Dupont', '10 Bis de Paris'),
('CLI00002', 'Martin', '5 Ter d'Avenue '),
('CLI00003', 'Lemoine', '25 Quater Boulevard ');

INSERT INTO Article (idArt, Quantite, PriUnit) VALUES
('ART00001', 100, 50),
('ART00002', 200, 30),
('ART00003', 150, 75);

INSERT INTO Commande (Date, Prix, idArt, idCLI) VALUES
('2024-12-01', 500.00, 'ART00001', 'CLI00001'),
('2024-12-02', 900.00, 'ART00002', 'CLI00002'),
('2024-12-03', 2250.00, 'ART00003', 'CLI00003');
```

III - Création d'une procédure stockée

Nous allons mettre en place une procédure stockée qui permet d'ajouter un client plus simplement dans notre base de données.



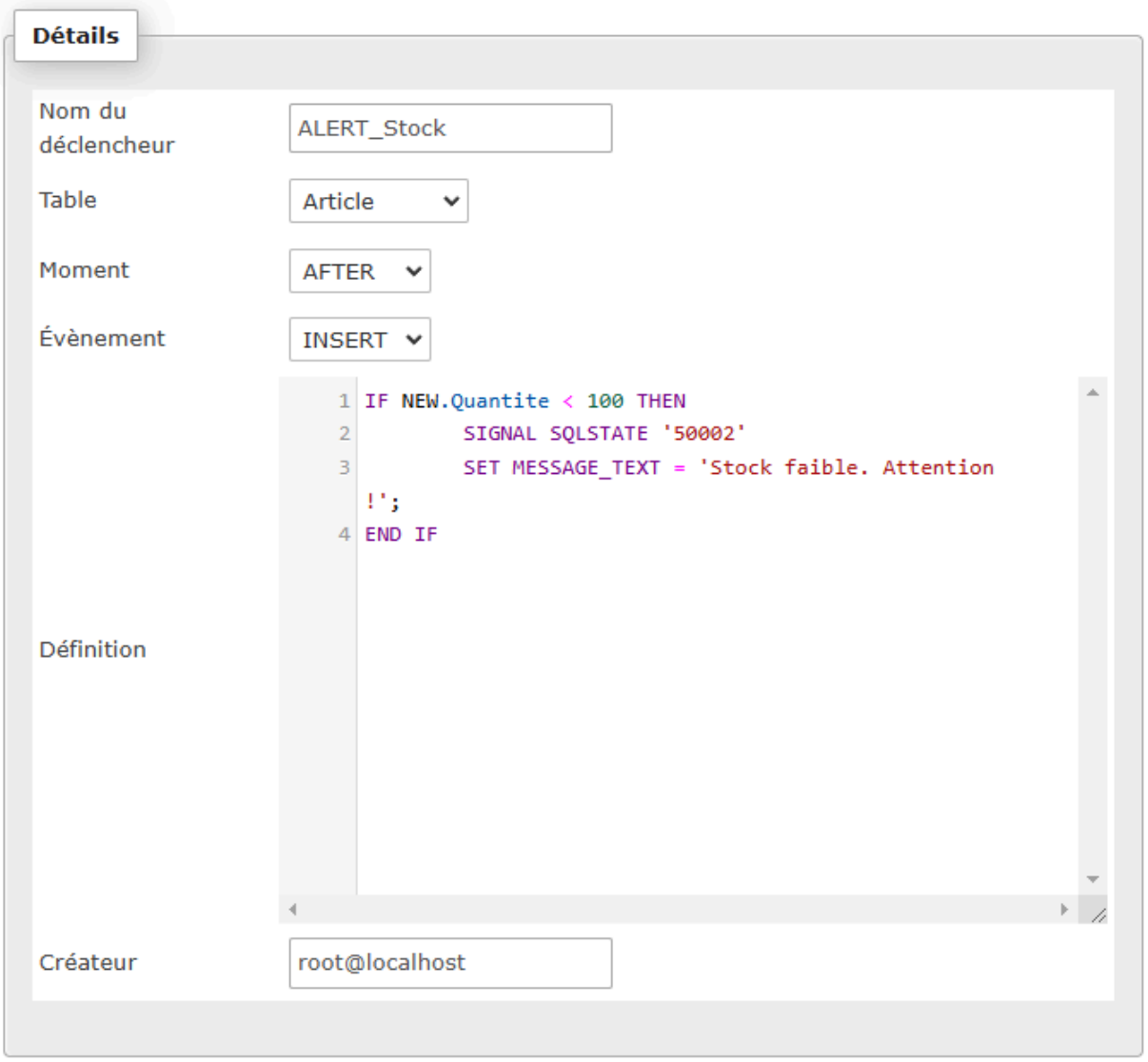
Nous appelons notre procédure stockée avec **CALL ajouter_client.**

```
MariaDB [Amazon]> SELECT * FROM Client;
+----------+---------+----------------------+
| idCLI    | Nom     | Adresse              |
+----------+---------+----------------------+
| CLI00001 | Dupont  | 10 Bis de Paris      |
| CLI00002 | Martin  | 5 Ter d'Avenue       |
| CLI00003 | Lemoine | 25 Quater Boulevard  |
+----------+---------+----------------------+
3 rows in set (0,002 sec)

MariaDB [Amazon]> CALL ajouter_client ('CLI0004','lebon','15 sexiet');
Query OK, 1 row affected (0,007 sec)

MariaDB [Amazon]> SELECT * FROM Client;
+----------+---------+----------------------+
| idCLI    | Nom     | Adresse              |
+----------+---------+----------------------+
| CLI00001 | Dupont  | 10 Bis de Paris      |
| CLI00002 | Martin  | 5 Ter d'Avenue       |
| CLI00003 | Lemoine | 25 Quater Boulevard  |
| CLI0004  | lebon   | 15 sexiet            |
+----------+---------+----------------------+
4 rows in set (0,002 sec)
```

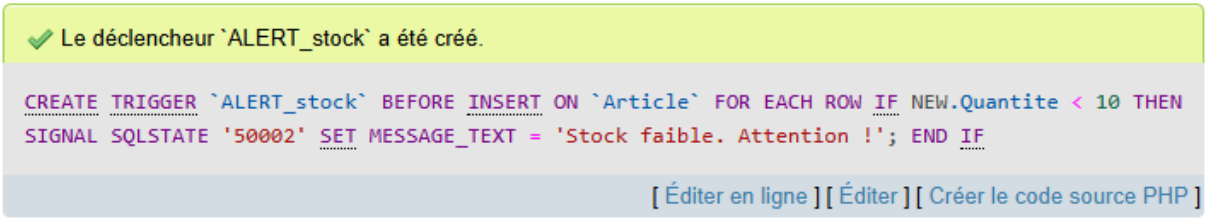
On peut voir ci-dessus que notre procédure stockée qui permet d'ajouter un client fonctionne bien.

IV - Création du trigger

Le trigger aura pour but ici de nous indiquer lorsque l'on entrera une donnée de dire si le nombre en stock est suffisant ou non. Il marche ici uniquement à sa création.



Notre trigger a bien ete cree.



Pour tester notre trigger nous allons insérer un article avec une quantité inférieure à 100.

```
MariaDB [Amazon]> INSERT INTO Article (idArt, Quantite, PriUnit) VALUES ('ART00004', 5, 40);
ERROR 1644 (50002): Stock faible. Attention !
```

Nous pouvons apercevoir que notre trigger **ALERT_Stock** fonctionne parfaitement.

V - Gestion des droits des utilisateurs

Nous allons créer des utilisateurs et leur attribuer des droits sur certaines tables et pas d'autres.

```
MariaDB [Amazon]> CREATE USER 'gestionnaire'@'localhost' IDENTIFIED BY 'class';
Query OK, 0 rows affected (0,006 sec)

MariaDB [Amazon]> CREATE USER 'consultant'@'localhost' IDENTIFIED BY 'class';
Query OK, 0 rows affected (0,005 sec)
```

L'utilisateur gestionnaire créer aura des privilèges sur la table commande de la base de donnée Amazon tandis que l'utilisateur consultant aura des privilèges sur les table Article et Client.

```
MariaDB [Amazon]> GRANT ALL PRIVILEGES ON Amazon.Commande TO 'gestionnaire'@'localhost';
Query OK, 0 rows affected (0,005 sec)

MariaDB [Amazon]> GRANT ALL PRIVILEGES ON Amazon.Article TO 'consultant'@'localhost';
Query OK, 0 rows affected (0,005 sec)

MariaDB [Amazon]> GRANT ALL PRIVILEGES ON Amazon.Client TO 'consultant'@'localhost';
Query OK, 0 rows affected (0,005 sec)

MariaDB [Amazon]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,001 sec)
```

Pour vérifier ceci nous allons nous connecter avec l'utilisateur gestionnaire avec **mysql -u gestionnaire -p,** le mot de passe est "class" comme nous l'avons définie plus haut.

```
root@MySQL:/home/userrt# mysql -u gestionnaire -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 205
Server version: 10.5.26-MariaDB-0+deb11u2-log Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| Amazon             |
| information_schema |
+--------------------+
2 rows in set (0,001 sec)

MariaDB [(none)]> USE Amazon;
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
MariaDB [Amazon]> SHOW TABLES;
+------------------+
| Tables_in_Amazon |
+------------------+
| Commande         |
+------------------+
1 row in set (0,000 sec)

MariaDB [Amazon]> SELECT * FROM Commande;
+--------+------------+------+----------+----------+
| NumCom | Date       | Prix | idArt    | idCLI    |
+--------+------------+------+----------+----------+
|      1 | 2024-12-01 |  500 | ART00001 | CLI00001 |
|      2 | 2024-12-02 |  900 | ART00002 | CLI00002 |
|      3 | 2024-12-03 | 2250 | ART00003 | CLI00003 |
+--------+------------+------+----------+----------+
3 rows in set (0,001 sec)
```

Comme on peut voir ci dessus nous pouvons voir que l'utilisateur gestionnaire a accès qu' à la table Commande et le consultant a accès qu'aux tables Article et Client.

VI - Mise en place de la réplication bidirectionnelle

Nous allons mettre en place une réplication bidirectionnelle de notre base de données.
Pour faire cela nous allons mettre en place un deuxième serveur nommé Slave.
Nous allons modifier le fichier **/etc/mysql/mariadb.conf.d/50-server.cnf** sur le serveur Maître en ajoutant **replicate-do-db = Amazon**, **log-bin = /var/lib/mysql/mysql-bin** et commentez **bind-address.**

```
  GNU nano 5.4                     /etc/mysql/mariadb.conf.d/50-server.cnf *
#slow_query_log_file    = /var/log/mysql/mariadb-slow.log
#long_query_time        = 10
#log_slow_verbosity     = query_plan,explain
#log-queries-not-using-indexes
#min_examined_row_limit = 1000

# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
server-id               = 1
log_bin                 = /var/lib/mysql/mysql-bin
replicate-do-db          = Amazon
expire_logs_days        = 10
#max_binlog_size        = 100M
#bind-address           =127.0.0.1
```

Sur le serveur Slave nous allons faire pareil sauf pour server-id qui sera à 2.

```
  GNU nano 5.4                    /etc/mysql/mariadb.conf.d/50-server.cnf *
#slow_query_log_file    = /var/log/mysql/mariadb-slow.log
#long_query_time        = 10
#log_slow_verbosity     = query_plan,explain
#log-queries-not-using-indexes
#min_examined_row_limit = 1000

# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
server-id               = 2
log_bin                 = /var/lib/mysql/mysql-bin
replicate-do-db          = Amazon
expire_logs_days        = 10
#max_binlog_size         = 100M
#bind-address            =127.0.0.1
```

Nous créons un utilisateur pour la réplication qui se nomme **'reception'@'%',** nous mettons **%** car notre serveur change souvent d'adresse IP.

```
MariaDB [(none)]> CREATE USER 'reception'@'%' IDENTIFIED BY 'class';
Query OK, 0 rows affected (0,006 sec)

MariaDB [(none)]> GRANT REPLICATION SLAVE ON *.* TO 'reception'@'%';
Query OK, 0 rows affected (0,005 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,001 sec)
```

Ensuite nous passons à la configuration de la réplication.

Sur le serveur Maître on met l'adresse IP du serveur Slave

```
MariaDB [(none)]> CHANGE MASTER TO
    ->      MASTER_HOST='192.168.1.78',
    ->      MASTER_USER='reception',
    ->      MASTER_PASSWORD='class',
    ->      MASTER_LOG_FILE='mysql-bin.000001',
    ->      MASTER_LOG_POS=0;
Query OK, 0 rows affected (0,026 sec)

MariaDB [(none)]>
MariaDB [(none)]>
MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0,001 sec)
```

Sur le Slave on met l'adresse IP du serveur Maître.

```
MariaDB [(none)]> CHANGE MASTER TO
    ->     MASTER_HOST='192.168.1.61',
    ->     MASTER_USER='reception',
    ->     MASTER_PASSWORD='class',
    ->     MASTER_LOG_FILE='mysql-bin.000001',
    ->     MASTER_LOG_POS=0;
Query OK, 0 rows affected (0,022 sec)

MariaDB [(none)]>
MariaDB [(none)]>
MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0,001 sec)
```

Pour vérifier que tout fonctionne, on fait un **SHOW SLAVE STATUS\G.** Nous pouvons voir que sur le serveur Maître et Slave que **Slave_SQL_Running** est en **"Yes"** et que **Slave_IO_Running est en "Connecting"** alors qu'il est censé être en **"Yes".**

```
MariaDB [(none)]> SHOW SLAVE STATUS\G;
*************************** 1. row ***************************
               Slave_IO_State: Connecting to master
                  Master_Host: 192.168.1.78
                  Master_User: reception
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000001
          Read_Master_Log_Pos: 4
               Relay_Log_File: mysqld-relay-bin.000001
                Relay_Log_Pos: 4
        Relay_Master_Log_File: mysql-bin.000001
             Slave_IO_Running: Connecting
            Slave_SQL_Running: Yes
              Replicate_Do_DB: Amazon
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
```

Pour le mettre en "Yes" nous allons vérifier les ping entre les deux serveurs Maître et Slave et autoriser toute les connexions entrant sur le port 3306.

```
root@MySQL:/home/userrt# ping 192.168.1.78
PING 192.168.1.78 (192.168.1.78) 56(84) bytes of data.
64 bytes from 192.168.1.78: icmp_seq=1 ttl=64 time=20.0 ms
64 bytes from 192.168.1.78: icmp_seq=2 ttl=64 time=1.36 ms
^C
--- 192.168.1.78 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.361/10.695/20.030/9.334 ms
root@MySQL:/home/userrt#
```

Nous voyons que les ping entre les deux serveurs fonctionnent bien.

Sur le serveur Maître et Slave nous allons autoriser toute les connexions entrant sur le port 3306 avec un **sudo ufw allow 3306.**

```
root@MySQL:/home/userrt# sudo ufw allow 3306
Rules updated
Rules updated (v6)
```

Nous démarrons nos deux serveur avec la commande **systemctl restart mysql**

Nous allons vérifier que tout fonctionne bien avec un **SHOW SLAVE STATUS\G** et que **Slave_IO_Running** est bien en "**Yes".**

```
MariaDB [(none)]> SHOW SLAVE STATUS\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.78
                  Master_User: reception
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000004
          Read_Master_Log_Pos: 342
               Relay_Log_File: mysqld-relay-bin.000007
                Relay_Log_Pos: 641
        Relay_Master_Log_File: mysql-bin.000004
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: Amazon
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 342
              Relay_Log_Space: 1336
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
```

Nous pouvons voir sur ci dessus que tout fonctionne **Slave_IO_Running** est bien en "**Yes".**

Sur les serveurs Maître nous allons verrouiller la lecture et l'écriture de toutes les tables nous permettant de faire une sauvegarde (snapshot) de notre base de données avec un **FLUSH TABLES WITH READ LOCK.**

```
MariaDB [(none)]> FLUSH TABLES WITH READ LOCK;
```

Nous sauvegardons sur le serveur Maître notre base de données dans un fichier qui est en .sql avec un **mysqldump -u root -p --databases Amazon --master-data=2 > Amazon.sql** et nous envoyons ce fichier sur le Slave avec **scp Amazon.sql userrt@192.168.1.78:/tmp.**

```
root@MySQL:/home/userrt# mysqldump -u root -p --databases Amazon --master-data=2 > Amazon.sql
Enter password:
root@MySQL:/home/userrt# scp Amazon.sql userrt@192.168.1.78:/tmp
userrt@192.168.1.78's password:
Amazon.sql
```

Une fois que notre sauvegarde et terminer nous allons pouvoir retirer le READ Lock sur le serveur Maître de notre base de données avec **UNLOCK TABLES;**

```
MariaDB [(none)]> UNLOCK TABLES;
```

Sur le serveur Slave nous allons importer la base de données dans notre service MySQL avec un **mysql -u root -p </tmp/Amazon.sql .**

```
root@MySQL:/home/userrt# mysql -u root -p < /tmp/Amazon.sql
Enter password:
```

Sur le serveur Slave on peut voir avec un show databases que notre base de données a été importée avec succès.

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| Amazon             |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0,001 sec)
```

Maintenant nous allons tester la réplication en insérant une nouvelle table dans notre base de données Amazon sur le serveur Maître qui se nomme test_replication.

Voici la notre base de données sur le serveur Slave avant l'insertion d'une nouvelle table :

```
Database changed
MariaDB [Amazon]> SHOW TABLES;
+------------------+
| Tables_in_Amazon |
+------------------+
| Article          |
| Client           |
| Commande         |
| test_replication |
+------------------+
4 rows in set (0,001 sec)
```

Insertion de notre tables test_replication dans notre base de données Amazon sur le serveur Maître.

```
MariaDB [Amazon]> USE Amazon;
Database changed
MariaDB [Amazon]> CREATE TABLE IF NOT EXISTS test_replication (id INT AUTO_INCREMENT PRIMARY KEY, messag
 VARCHAR(255));
Query OK, 0 rows affected, 1 warning (0,000 sec)

MariaDB [Amazon]> INSERT INTO test_replication (message) VALUES ('Test réplication MySQL');
Query OK, 1 row affected (0,002 sec)
```

Ici nous voyons que notre table test_replication ajouter via le serveur Maître est bien sur notre serveur Slave.

```
MariaDB [(none)]> USE Amazon;
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
MariaDB [Amazon]> SHOW TABLES;
+------------------+
| Tables_in_Amazon |
+------------------+
| Article          |
| Client           |
| Commande         |
| test_replication |
+------------------+
4 rows in set (0,001 sec)
```