

SERVIDORES WEB Y ALTA DISPONIBILIDAD (2021-2022)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Memoria Práctica 5

Pedro Antonio Mayorgas Parejo

28 de mayo de 2022

Índice

1	Base de datos básica	3
1.1	Proceso de instalación de la base de datos en m1-pedroamp y m2-pedroamp.	3
1.2	Creación del usuario, las tablas y inserción de los datos.	3
1.3	Creación de tablas con el usuario y inserción de datos.	5
1.4	Sentencias SQL utilizadas y su clasificación dentro del lenguaje.	13
2	Copia de seguridad manual de base de datos con mysqldump	15
2.1	Requisitos de mysqldump	15
2.2	Qué produce mysqldump	15
2.2.1	Bloqueo/desbloqueo de tablas manual	15
2.3	Uso de mysqldump básico	18
2.4	Bloqueo de tablas automático con mysqldump	20
2.5	Transmisión de los datos a la réplica.	22
3	Servidores Primario y Réplica	28
3.1	Configurando el servidor Primary	28
3.2	Inicio del proceso de réplica.	32
4	Servidores Primario y Primario	35
5	GTID Global Transaction ID	40
5.1	Problemas con algunos errores de SQL	42
6	Iptables para SQL	45
6.1	¿Cómo he realizado el análisis de puertos TCP para saber cómo se conectan?	45
6.2	m1-pedroamp	47
6.3	m2-pedroamp	49
7	Bibliografía	52

1. Base de datos básica

1.1. Proceso de instalación de la base de datos en m1-pedroamp y m2-pedroamp.

Para realizar la instalación de la base de datos necesitamos ejecutar en Debian GNU/Linux el siguiente comando.

```
1 # Actualizamos la cache de apt
2 $ sudo apt-get update
3 $ sudo apt-get install mariadb-server mariadb-client
```

```
debian@m1-pedroamp:~$ sudo apt-get install mariadb-server mariadb-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdbd-mariadb-perl libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmariadb3 libmpfr6 libsigsegv2 libsnappy1v5
  libterm-readkey-perl libtimedate-perl liburi-perl lsof mariadb-client-10.5 mariadb-client-core-10.5 mariadb-common mariadb-server-10.5 mariadb-server-core-10.5 mysql-common rsync
Suggested packages:
  gawk-doc libldb-perl libnet-daemon-perl libsql-statement-perl libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx mariadb-test netcat-openbsd
The following NEW packages will be installed:
  galera-4 gawk libcgi-fast-perl libcgi-pm-perl libclone-perl libconfig-inifiles-perl libdbd-mariadb-perl libdbi-perl libencode-locale-perl libfcgi-bin libfcgi-perl libfcgi0ldbl
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmariadb3 libmpfr6 libsigsegv2 libsnappy1v5
  libterm-readkey-perl libtimedate-perl liburi-perl lsof mariadb-client mariadb-client-10.5 mariadb-client-core-10.5 mariadb-common mariadb-server mariadb-server-10.5
  mariadb-server-core-10.5 mysql-common rsync
0 upgraded, 36 newly installed, 0 to remove and 0 not upgraded.
Need to get 19.5 MB of archives.
After this operation, 160 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Figura 1: Instalación del RDBMS MariaDB un fork de MySQL.

1.2. Creación del usuario, las tablas y inserción de los datos.

Aquí tenemos todo el fichero de DCL que define un usuario, base de datos y le da permisos para poder ejecutar sentencias sobre ella y sus tablas.

```

1  — DML DROPPING USER AND DATABASE
2  DROP USER IF EXISTS 'tiendabd'@'localhost';
3  DROP DATABASE IF EXISTS tiendabd;
4
5  — Parte de DCL Data Control Language
6  CREATE USER 'tiendabd'@'localhost' IDENTIFIED BY 'password';
7
8  — DML CREATING DATABASE
9  CREATE DATABASE tiendabd;
10
11 — DCL GRANTING PRIVILEGES TO USER ON DATABASE AND SQL SENTENCES
12 GRANT ALL PRIVILEGES ON tiendabd.* TO 'tiendabd'@'localhost';
13
14 — COMMITING PRIVILEGES TO USER ON RDBMS
15 FLUSH PRIVILEGES;

```

```

debiang@pedro:~$ sudo mariadb -u root < DDLDDL.sql
debiang@pedro:~$ sudo mariadb -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 37
Server version: 10.5.15-MariaDB-0+deb11u1 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 |
+-----+
| information_schema |
| mysql |
| performance_schema |
| tiendabd |
+-----+
4 rows in set (0.001 sec)

MariaDB [(none)]> SELECT User from mysql.user;
+-----+
| User |
+-----+
| mariadb.sys |
| mysql |
| root |
| tiendabd |
+-----+
4 rows in set (0.002 sec)

```

Figura 2: Definición del usuario correcta.

Las sentencias para verificar la creación de la base de datos y el usuario son las siguientes:

```

1  — Verificacion de base de datos creada
2  SHOW DATABASES;
3  — Verificacion del usuario
4  SELECT User FROM mysql.user;

```

```

MariaDB [(none)]> SHOW GRANTS FOR 'tiendabd'@'localhost';
+-----+
| Grants for tiendabd@localhost |
+-----+
| GRANT USAGE ON *.* TO 'tiendabd'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C060DEE42FD1618BB99005ADCA2EC9D1E19' |
| GRANT ALL PRIVILEGES ON 'tiendabd'.* TO 'tiendabd'@'localhost' |
+-----+
2 rows in set (0.000 sec)

```

Figura 3: Verificación de privilegios del usuario.

```

1  — Verificacion de base de datos creada
2  SHOW GRANTS FOR 'tiendabd'@'localhost';

```

1.3. Creación de tablas con el usuario y inserción de datos.

Creo un fichero .SQL que me permita crear de manera automática las tablas sin tener que ir a la consola de mysql y hacerlo manualmente.

En esta parte del fichero usamos el DDL con la sentencia DROP y un verificador de IF para que ejecute esa sentencia si se cumple el EXISTS. Esto es para evitar que haya problemas con la ejecución del fichero de SQL en el RDBMS.

```
1 CREATE TABLE EMPLEADO (
2     ID_EMPLEADO INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
3     NOMBRE VARCHAR(10) ,
4     APELLIDO VARCHAR(10) ,
5     PASSWORD VARCHAR(30)
6 );
7
8
9 CREATE TABLE REPARTIDOR (
10     ID_REPARTIDOR INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
11     NOMBRE VARCHAR(10) ,
12     PASSWORD VARCHAR(30)
13 );
14
15
16 CREATE TABLE CIUDADES (
17     COD_POSTAL INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
18     NOMBRE VARCHAR(20)
19 );
20
21 CREATE TABLE CLIENTE(
22     ID_CLIENTE INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
23     NOMBRE VARCHAR(10) ,
24     TELEFONO INT(10) UNSIGNED,
25     DIRECCION VARCHAR(30) ,
26     COD_POSTAL INT(6) UNSIGNED,
27     PASSWORD VARCHAR(30) ,
28     ID_EMPLEADO_ALTA INT(4) UNSIGNED,
29     ID_EMPLEADO_BAJA INT(4) UNSIGNED,
30
31     CONSTRAINT UN_TELEFONO UNIQUE(TELEFONO) ,
32     CONSTRAINT COD_POSTAL_FK FOREIGN KEY (COD_POSTAL) REFERENCES CIUDADES(
33         COD_POSTAL) ,
34     CONSTRAINT ID_ALTA_FK FOREIGN KEY (ID_EMPLEADO_ALTA) REFERENCES
35         EMPLEADO(ID_EMPLEADO) ,
36     CONSTRAINT ID_BAJA_FK FOREIGN KEY (ID_EMPLEADO_BAJA) REFERENCES
37         EMPLEADO(ID_EMPLEADO)
38 );
39
40 — Tablas de log del TRIGGER de INSERCIÓN BEFORE CLIENTE
41 CREATE TABLE ALTA_CLIENTE (
42     ID_CLIENTE INT(4) UNSIGNED,
43     ID_EMPLEADO INT(4) UNSIGNED,
44     FECHA_ALTA DATE,
```

```

43     CONSTRAINT ID_CLIENTE_ALTA_FK FOREIGN KEY (ID_CLIENTE) REFERENCES
        CLIENTE(ID_CLIENTE) ,
44     CONSTRAINT ID_EMPLEADO_ALTA_FK FOREIGN KEY (ID_EMPLEADO) REFERENCES
        EMPLEADO(ID_EMPLEADO)
45 );
46
47 — Tablas de log del TRIGGER de UPDATE BEFORE CLIENTE
48 CREATE TABLE BAJA_CLIENTE (
49     ID_CLIENTE INT(4) UNSIGNED,
50     ID_EMPLEADO INT(4) UNSIGNED,
51     FECHA_BAJA DATE,
52
53     CONSTRAINT ID_CLIENTE_BAJA_FK FOREIGN KEY (ID_CLIENTE) REFERENCES
        CLIENTE(ID_CLIENTE) ,
54     CONSTRAINT ID_EMPLEADO_BAJA_FK FOREIGN KEY (ID_EMPLEADO) REFERENCES
        EMPLEADO(ID_EMPLEADO)
55 );
56 — Posible CONSTRAINT CANTIDAD_MAYOR_QUE_0 CHECK(CANTIDAD>0)
57
58 CREATE TABLE STOCK (
59     ID_PRODUCTO INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
60     NOMBRE VARCHAR(30) ,
61     CANTIDAD INT(4)
62 );
63
64 CREATE TABLE MOD_STOCK(
65     ID_PRODUCTO INT(4) UNSIGNED,
66     FECHA_MODIFICACION DATE,
67     VIEJA_CANTIDAD INT(4) UNSIGNED,
68     NUEVA_CANTIDAD INT(4) UNSIGNED,
69
70     CONSTRAINT ID_PRODUCTO_MOD_FK FOREIGN KEY (ID_PRODUCTO) REFERENCES
        STOCK(ID_PRODUCTO)
71 );
72
73 CREATE TABLE PEDIDO (
74     ID_PEDIDO INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
75     ID_CLIENTE INT(4) UNSIGNED,
76     ID_PRODUCTO INT(4) UNSIGNED,
77     CANTIDAD INT(4) UNSIGNED,
78     ID_REPARTIDOR INT(4) UNSIGNED,
79     FECHA_PEDIDO DATE,
80     FECHA_ENTREGA_PROGRAMADA DATE,
81     ESTADO ENUM( 'A la espera' , 'En reparto' , 'Repartido' , 'Cancelado' ) ,
82
83     CONSTRAINT ID_CLIENTE_PEDIDO_FK FOREIGN KEY (ID_CLIENTE) REFERENCES
        CLIENTE(ID_CLIENTE) ,
84     CONSTRAINT ID_PRODUCTO_PEDIDO_FK FOREIGN KEY (ID_PRODUCTO) REFERENCES
        STOCK(ID_PRODUCTO) ,
85     CONSTRAINT ID_REPARTIDOR_PEDIDO_FK FOREIGN KEY (ID_REPARTIDOR)
        REFERENCES REPARTIDOR(ID_REPARTIDOR)
86 );

```

En el resto del fichero siempre se va a ejecutar ya que si existe esta tabla ya ha sido borrada

anteriormente (recuerda hacer un backup antes de borrar las tablas). Son muy similares, algunas tienen CONSTRAINT para claves foráneas de referencia, hay CONSTRAINT de UNIQUE que hace que el RDBMS no permita la inserción de filas que utilicen un n^o de teléfono existente.

```

1  —Primero creamos las tablas no relacionadas, es decir aquellas que no
    tienen claves FK.
2  —Las tablas son:
3  — EMPLEADO
4  — CLIENTE
5  — CIUDADES
6  — STOCK
7  — REPARTIDOR
8  — PEDIDO
9  DROP TABLE IF EXISTS ALTA_CLIENTE; — Relacion con CLIENTE y EMPLEADO por
    FK
10 DROP TABLE IF EXISTS BAJA_CLIENTE; — Relacion con CLIENTE y EMPLEADO por
    FK
11 DROP TABLE IF EXISTS PEDIDO; — Relacion con CLIENTE, EMPLEADO REPARTIDOR
    STOCK por FK
12 DROP TABLE IF EXISTS CLIENTE; — Relacion con CIUDADES por codigo postal de
    FK y con EMPLEADO
13 DROP TABLE IF EXISTS MOD_STOCK; —Relacion con STOCK ya que lleva el conteo
    de las actualizaciones de este
14
15 — TABLAS NO RELACIONADAS
16 DROP TABLE IF EXISTS EMPLEADO;
17 DROP TABLE IF EXISTS CIUDADES;
18 DROP TABLE IF EXISTS STOCK;
19 DROP TABLE IF EXISTS REPARTIDOR;
20
21
22 CREATE TABLE EMPLEADO (
23     ID_EMPLEADO INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
24     NOMBRE VARCHAR(10) ,
25     APELLIDO VARCHAR(10) ,
26     PASSWORD VARCHAR(30)
27 );
28
29
30 CREATE TABLE REPARTIDOR (
31     ID_REPARTIDOR INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
32     NOMBRE VARCHAR(10) ,
33     PASSWORD VARCHAR(30)
34 );
35
36
37 CREATE TABLE CIUDADES (
38     COD_POSTAL INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
39     NOMBRE VARCHAR(20)
40 );
41
42 CREATE TABLE CLIENTE(
43     ID_CLIENTE INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
44     NOMBRE VARCHAR(10) ,
45     TELEFONO INT(10) UNSIGNED,
46     DIRECCION VARCHAR(30) ,
47     COD_POSTAL INT(6) UNSIGNED,
48     PASSWORD VARCHAR(30) ,

```



```

49     ID_EMPLEADO_ALTA INT(4) UNSIGNED,
50     ID_EMPLEADO_BAJA INT(4) UNSIGNED,
51
52     CONSTRAINT UN_TELEFONO UNIQUE(TELEFONO) ,
53     CONSTRAINT COD_POSTAL_FK FOREIGN KEY (COD_POSTAL) REFERENCES CIUDADES(
54         COD_POSTAL) ,
55     CONSTRAINT ID_ALTA_FK FOREIGN KEY (ID_EMPLEADO_ALTA) REFERENCES
56         EMPLEADO(ID_EMPLEADO) ,
57     CONSTRAINT ID_BAJA_FK FOREIGN KEY (ID_EMPLEADO_BAJA) REFERENCES
58         EMPLEADO(ID_EMPLEADO)
59 );
60
61 — Tablas de log del TRIGGER de INSERCIÓN BEFORE CLIENTE
62 CREATE TABLE ALTA_CLIENTE (
63     ID_CLIENTE INT(4) UNSIGNED,
64     ID_EMPLEADO INT(4) UNSIGNED,
65     FECHA_ALTA DATE,
66
67     CONSTRAINT ID_CLIENTE_ALTA_FK FOREIGN KEY (ID_CLIENTE) REFERENCES
68         CLIENTE(ID_CLIENTE) ,
69     CONSTRAINT ID_EMPLEADO_ALTA_FK FOREIGN KEY (ID_EMPLEADO) REFERENCES
70         EMPLEADO(ID_EMPLEADO)
71 );
72
73 — Tablas de log del TRIGGER de UPDATE BEFORE CLIENTE
74 CREATE TABLE BAJA_CLIENTE (
75     ID_CLIENTE INT(4) UNSIGNED,
76     ID_EMPLEADO INT(4) UNSIGNED,
77     FECHA_BAJA DATE,
78
79     CONSTRAINT ID_CLIENTE_BAJA_FK FOREIGN KEY (ID_CLIENTE) REFERENCES
80         CLIENTE(ID_CLIENTE) ,
81     CONSTRAINT ID_EMPLEADO_BAJA_FK FOREIGN KEY (ID_EMPLEADO) REFERENCES
82         EMPLEADO(ID_EMPLEADO)
83 );
84
85 — Posible CONSTRAINT CANTIDAD_MAYOR_QUE_0 CHECK(CANTIDAD>0)
86
87 CREATE TABLE STOCK (
88     ID_PRODUCTO INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
89     NOMBRE VARCHAR(30) ,
90     CANTIDAD INT(4)
91 );
92
93 CREATE TABLE MOD_STOCK(
94     ID_PRODUCTO INT(4) UNSIGNED,
95     FECHA_MODIFICACION DATE,
96     VIEJA_CANTIDAD INT(4) UNSIGNED,
97     NUEVA_CANTIDAD INT(4) UNSIGNED,
98
99     CONSTRAINT ID_PRODUCTO_MOD_FK FOREIGN KEY (ID_PRODUCTO) REFERENCES
100         STOCK(ID_PRODUCTO)
101 );
102
103 CREATE TABLE PEDIDO (

```

```

95     ID_PEDIDO INT(4) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
96     ID_CLIENTE INT(4) UNSIGNED,
97     ID_PRODUCTO INT(4) UNSIGNED,
98     CANTIDAD INT(4) UNSIGNED,
99     ID_REPARTIDOR INT(4) UNSIGNED,
100    FECHA_PEDIDO DATE,
101    FECHA_ENTREGA_PROGRAMADA DATE,
102    ESTADO ENUM('A la espera', 'En reparto', 'Repartido', 'Cancelado'),
103
104    CONSTRAINT ID_CLIENTE_PEDIDO_FK FOREIGN KEY (ID_CLIENTE) REFERENCES
        CLIENTE(ID_CLIENTE) ,
105    CONSTRAINT ID_PRODUCTO_PEDIDO_FK FOREIGN KEY (ID_PRODUCTO) REFERENCES
        STOCK(ID_PRODUCTO) ,
106    CONSTRAINT ID_REPARTIDOR_PEDIDO_FK FOREIGN KEY (ID_REPARTIDOR)
        REFERENCES REPARTIDOR(ID_REPARTIDOR)
107 );

1 # Para poder realizar la insercion necesitamos especificar la base de datos
  desde la consola de comandos. Con el parametro -b.
2 mysql -u tiendabd -p -b tiendabd < DDL.sql

1 — Indicamos el uso de la base de datos de tiendabd para poder ver las
  tablas
2 USE tiendabd;
3 — Mostramos las tablas de la base de datos.
4 SHOW TABLES;

```

```

debian@ml-pedroamp:~$ mysql -u tiendabd -p -b tiendabd < DDL.sql
Enter password:
debian@ml-pedroamp:~$ mysql -u tiendabd -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 43
Server version: 10.5.15-MariaDB-0+deb11u1 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)]> USE tiendabd
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 [tiendabd]> SHOW TABLES;
+-----+
| Tables in tiendabd |
+-----+
| ALTA_CLIENTE       |
| BAJA_CLIENTE       |
| CIUDADES           |
| CLIENTE            |
| EMPLEADO           |
| MOD_STOCK          |
| PEDIDO             |
| REPARTIDOR         |
| STOCK              |
+-----+
9 rows in set (0.001 sec)

MariaDB [tiendabd]> █

```

Figura 4: Creación y verificación de las tablas con el usuario creado.

Aquí tenemos los datos que vamos a utilizar para inserción de datos en las tablas en el RDBMS.

```

1 INSERT INTO EMPLEADO (NOMBRE, APELLIDO, PASSWORD) VALUES ( 'Empleado', '
    Prueba', '1234' );
2 INSERT INTO CIUDADES (COD_POSTAL, NOMBRE) VALUES (18000, 'GRANADA');
3 INSERT INTO CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD,
    ID_EMPLEADO_ALTA) VALUES ( 'cliente', 123456789, 'calle etsiit', 18000,
    '1234', 1 );
4 INSERT INTO CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD,
    ID_EMPLEADO_ALTA) VALUES ( 'Cliente2', 112345678, 'calle molotov', 18000,
    '12345', 1 );
5 INSERT INTO CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD,
    ID_EMPLEADO_ALTA) VALUES ( 'cliente3', 234567890, 'calle azor', 18000,
    '4321', 1 );
6 INSERT INTO CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD,
    ID_EMPLEADO_ALTA) VALUES ( 'Cliente4', 132316789, 'calle rpg', 18000,
    '112345', 1 );
7
8 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Camisetas L', 10 );
9 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Camisetas M', 10 );
10 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Camisetas XL', 10 );
11 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Camisetas XXL', 10 );
12 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Pantalones L', 10 );
13 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Pantalones M', 10 );
14 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Pantalones XL', 10 );
15 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Pantalones XXL', 10 );
16 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Jersey L', 10 );
17 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Jersey M', 10 );
18 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Jersey XL', 10 );
19 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Jersey XXL', 10 );
20 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Patatas L', 10 );
21 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Patatas M', 10 );
22 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Patatas XL', 10 );
23 INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ( 'Patatas XXL', 10 );

1 # Para poder realizar la insercion de los datos necesitamos especificar la
    base de datos que contenga dichas tablas. Con el parametro -b.
2 mysql -u tiendabd -p -b tiendabd < DDLdata.sql

1 — Indicamos el uso de la base de datos de tiendabd para poder ver las
    tablas
2 USE tiendabd;
3 — Mostramos una informacion de los datos insertados en las tablas de la
    base de datos.
4 SELECT * FROM STOCK;
```

```

debian@ml-pedroamp:~$ mysql -u tiendabd -p -b tiendabd < DDLdata.sql
Enter password:
debian@ml-pedroamp:~$ mysql -u tiendabd -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 46
Server version: 10.5.15-MariaDB-0+deb11u1 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)]> USE tiendabd
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 [tiendabd]> SELECT * FROM STOCK;
+-----+-----+-----+
| ID_PRODUCTO | NOMBRE | CANTIDAD |
+-----+-----+-----+
| 1 | Camisetas L | 10 |
| 2 | Camisetas M | 10 |
| 3 | Camisetas XL | 10 |
| 4 | Camisetas XXL | 10 |
| 5 | Pantalones L | 10 |
| 6 | Pantalones M | 10 |
| 7 | Pantalones XL | 10 |
| 8 | Pantalones XXL | 10 |
| 9 | Jersey L | 10 |
| 10 | Jersey M | 10 |
| 11 | Jersey XL | 10 |
| 12 | Jersey XXL | 10 |
| 13 | Patatas L | 10 |
| 14 | Patatas M | 10 |
| 15 | Patatas XL | 10 |
| 16 | Patatas XXL | 10 |
+-----+-----+-----+
16 rows in set (0.001 sec)

MariaDB [tiendabd]> █

```

Figura 5: Inserción de datos en las tablas con el usuario y verificación.

- 1 — Verificacion de base de datos creada
- 2 SHOW GRANTS FOR 'tiendabd'@'localhost' ;

1.4. Sentencias SQL utilizadas y su clasificación dentro del lenguaje.

Las sentencias de **DCL Data Control Language**, usados son:

1. Sentencia **GRANT ALL PRIVILEGES USER 'user'@'IP %' ON [DATABASE|*]. [TABLE|*]** : Permite a un usuario específico, desde una dirección IP específica (generalmente Localhost o % todas 0.0.0.0) realizar todas las sentencias sobre la base de datos específica o todas (*) y si queremos una mayor granularidad sobre unas tablas específica o todas *.
 - a) Se puede especificar qué sentencias puede ejecutar el usuario con mayor granularidad.
 - b) Si le damos acceso a todas las bases de datos con *.* y le damos permiso de realizar toda las operaciones posibles desde DDL, DML, etc... **El usuario es equivalente a al usuario root.**

Las sentencias de **DDL Data Definition Language**, usados son:

1. Sentencia **CREATE [DATABASE|TABLE|PROCEDURE|VIEW|USER...]**: Permite la creación de un objeto del tipo especificado (necesita sus parámetros) dentro del RDBMS (Relational Database Management System).
2. Sentencia **DROP [DATABASE|TABLE|PROCEDURE|VIEW|USER...]**: Elimina el objeto creado dentro del RDBMS, hay una serie de requisitos a cumplir para poder borrar el objeto, por ej: Para borrar una tabla relacionada se debe borrar primero la tabla que contiene las claves foráneas (FOREIGN KEY) que hace referencia a otra tabla. Si intentamos borrar primero la tabla referenciada, el RDBMS no nos permite borrarla por mantener la integridad Relacional de unos datos que desaparecerían.
3. Sentencia **ALTER [DATABASE|TABLE|PROCEDURE|VIEW|...]**: Modifica el objeto, añadiendo partes, las borra, o las rectifica. Es decir si hacemos un ALTER TABLE, podemos añadir una columna adicional, añadir un CONSTRAINT (RESTRICCIÓN) nuevo, o simplemente rectificar un tipo de dato de las columnas que almacena.

Las sentencias de **DML Data Manipulation Language**, usadas son:

1. Sentencia **SELECT COLUMN FROM TABLE [WHERE CLAUSES];**: Permite la visualización de la tabla y las columnas seleccionadas. Hay caracteres comodín que permiten la visualización de toda la tabla.
2. Sentencia **DROP [DATABASE|TABLE|PROCEDURE|VIEW|...]**: Elimina el objeto creado dentro del RDBMS, hay una serie de requisitos a cumplir para poder borrar el objeto, por ej: Para borrar una tabla relacionada se debe borrar primero la tabla que contiene las claves foráneas (FOREIGN KEY) que hace referencia a otra tabla. Si intentamos borrar primero la tabla referenciada, el RDBMS

no nos permite borrarla por mantener la integridad Relacional de unos datos que desaparecerían.

3. Sentencia **ALTER** [**DATABASE**|**TABLE**|**PROCEDURE**|**VIEW**|...]: Modifica el objeto, añadiendo partes, las borra, o las rectifica. Es decir si hacemos un **ALTER TABLE**, podemos añadir una columna adicional, añadir un **CONSTRAINT** (RESTRICCIÓN) nuevo, o simplemente rectificar un tipo de dato de las columnas que almacena.

2. Copia de seguridad manual de base de datos con mysqldump

2.1. Requisitos de mysqldump

Para la utilización de mysqldump necesita que el usuario que lo ejecute:

- Tenga el privilegio de ejecución de sentencias SELECT para hacer dump de las tablas.
- Tenga el privilegio SHOW VIEW para hacer dump de VIEWS.
- Tenga el privilegio de TRIGGER, para poder hacer dump de los triggers.
- Tenga el privilegio de LOCK TABLES, si no usa el argumento de `--single-transaction`.
- Tenga el privilegio de PROCESS, si no usa el argumento de `--no-tablespaces`.

2.2. Qué produce mysqldump

Mysqldump **produce un backup lógico**. Esto es que genera un fichero que reproduce toda la estructura de la base de datos, como sus tablas y los datos. Sin tener que realizar una copia física de los ficheros físicos localizados en el sistema **esto es un backup físico realizado por el comando mysqlbackup**.

Ventajas y desventajas del backup lógico:

- **Ventaja:** En el fichero lógico que genera mysqldump podemos. Modificar los datos que contenga para rectificar algo, añadir otros, etc.. lo cual es una ventaja si queremos que nuestra base de datos se rectifique con una nueva lógica de columnas pero respetando el tipo de dato existente, etc...
- **Desventaja:** El fichero lógico, necesita ser procesado por el RDBMS, por lo que es un proceso que se hace "secuencial", es decir ejecuta todas las sentencias que el fichero lógico tiene requiriendo una carga de cómputo grande y de IO en discos. En el caso de los backup físicos, la recuperación es "más rápida", ya que no tiene que ser procesado, si no que solo se tiene que copiar el fichero binario y enlazar con el RDBMS.

2.2.1. Bloqueo/desbloqueo de tablas manual

Uno de los pasos necesarios para realizar copias lógicas de la base de datos y/o sus tablas es la necesidad de bloquear la realización de más transacciones sobre dichos objetos del RDBMS. Esto es para que la integridad de los datos y la relacional. Se mantenga durante el proceso de la copia, ya que se podría realizar transacciones durante el proceso que dejaran las tablas a 'medias' que afectan a dicha integridad.

```

1  — Bloqueo de tablas de la propia base de datos.
2  LOCK TABLES TABLE_NAME READ[, TABLE_NAME2 READ,...];
3
4  — Bloqueo de tablas de la propia base de datos.
5  FLUSH TABLES WITH READ LOCK
6
7  — Desbloqueo de las tablas bloqueadas, el READ LOCK suele liberarse cuando
   se cierra la conexion que hace el lock
8  UNLOCK TABLES

```

1. **LOCK TABLES TABLE_NAME READ[, TABLE_NAME2 READ,...];**
 bloquea todas las escrituras en las tablas específicas dentro de una base de datos específica. Es decir que debemos estar en una base de datos seleccionada con USE DATABASE;

a) **LOCK TABLE[S]** Hace un commit implícito de las transacciones activas. Haciendo que no haya unas transacciones concurrentes a la espera de que se desbloquee la tabla. Esto preserva la integridad de los datos en las tablas transaccionales.

2. **FLUSH TABLES WITH READ LOCK** Necesita los permisos de FLUSH TABLES y RELOAD. Que en estos momentos solo lo tiene root, adquiere un READ LOCK global de todas las tablas en el RDBMS.

3. **UNLOCK TABLES** Desbloquea todas las tablas bloqueadas en la base de datos.

NOTA: La sesión de SQL debe mantenerse abierta, de lo contrario al finalizar dicha sesión el lock se libera permitiendo fuera de la sesión que se sigan modificando las tablas. Si se necesita el READ LOCK o el LOCK TABLES necesitamos dos consolas SSH. 1 para el LOCK y otra para ejecutar con seguridad mysqldump.

```

MariaDB [tiendabd]> FLUSH TABLES WITH READ LOCK;
ERROR 1227 (42000): Access denied; you need (at least one of) the RELOAD privilege(s) for this operation

```

Figura 6: No tenemos el permiso para hacer el READ LOCK para no tener que especificar las tablas de una base de datos.


```

MariaDB [tiendabd]> SHOW TABLES;
+-----+
| Tables_in_tiendabd |
+-----+
| ALTA_CLIENTE        |
| BAJA_CLIENTE        |
| CIUDADES            |
| CLIENTE             |
| EMPLEADO            |
| MOD_STOCK           |
| PEDIDO              |
| REPARTIDOR          |
| STOCK              |
+-----+
9 rows in set (0.001 sec)

MariaDB [tiendabd]> LOCK TABLES ALTA_CLIENTE READ, BAJA_CLIENTE READ, CIUDADES READ, CLIENTE READ, EMPLEADO READ, MOD_STOCK READ, PEDIDO READ, REPARTIDOR READ, STOCK READ;
Query OK, 0 rows affected (0.000 sec)

MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Camisetas XL', 10);
ERROR 1099 (HY000): Table 'STOCK' was locked with a READ lock and can't be updated
MariaDB [tiendabd]> UNLOCK TABLES;
Query OK, 0 rows affected (0.000 sec)

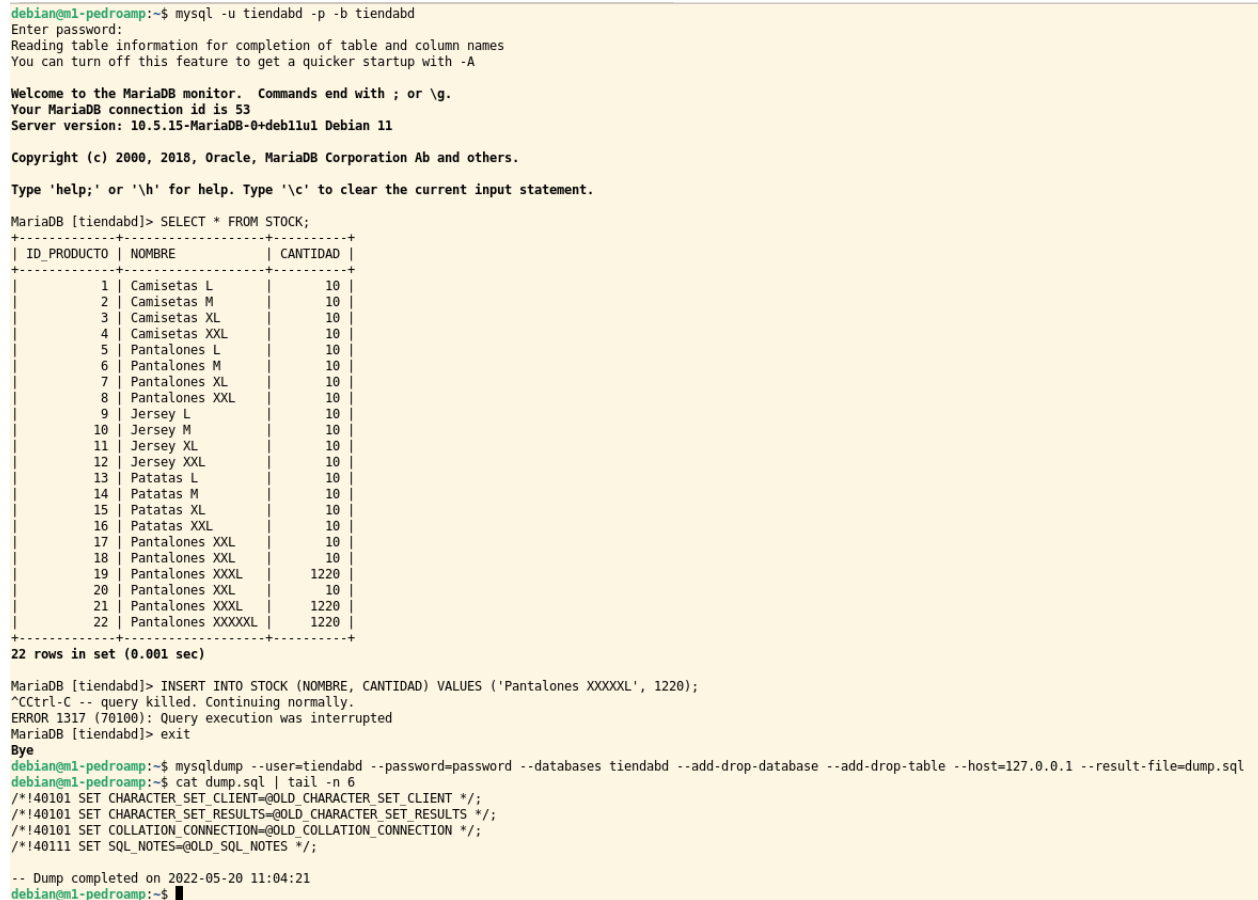
```

Figura 7: Bloqueo de las tablas de la base de datos y verificación de que efectivamente no puede bloquearse.

2.3. Uso de mysqldump básico

Comando básico a ejecutar para hacer un dump de la base de datos, que necesita un lock de dicha tabla especificado en una sesión de la base de datos

```
1 # Para una base de datos concreta
2 mysqldump --user=tiendabd --password=password --databases tiendabd --
  add-drop-database --add-drop-table [--replace] --host=127.0.0.1 --
  result-file=dump.sql
```



```
debian@ml-pedroamp:~$ mysql -u tiendabd -p -b tiendabd
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 53
Server version: 10.5.15-MariaDB-0+deb11u1 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 [tiendabd]> SELECT * FROM STOCK;
+-----+-----+-----+
| ID_PRODUCTO | NOMBRE | CANTIDAD |
+-----+-----+-----+
| 1 | Camisetas L | 10 |
| 2 | Camisetas M | 10 |
| 3 | Camisetas XL | 10 |
| 4 | Camisetas XXL | 10 |
| 5 | Pantalones L | 10 |
| 6 | Pantalones M | 10 |
| 7 | Pantalones XL | 10 |
| 8 | Pantalones XXL | 10 |
| 9 | Jersey L | 10 |
| 10 | Jersey M | 10 |
| 11 | Jersey XL | 10 |
| 12 | Jersey XXL | 10 |
| 13 | Patatas L | 10 |
| 14 | Patatas M | 10 |
| 15 | Patatas XL | 10 |
| 16 | Patatas XXL | 10 |
| 17 | Pantalones XXL | 10 |
| 18 | Pantalones XXL | 10 |
| 19 | Pantalones XXXL | 1220 |
| 20 | Pantalones XXL | 10 |
| 21 | Pantalones XXXL | 1220 |
| 22 | Pantalones XXXXL | 1220 |
+-----+-----+-----+
22 rows in set (0.001 sec)

MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Pantalones XXXXL', 1220);
^C^C^C -- query killed. Continuing normally.
ERROR 1317 (70100): Query execution was interrupted
MariaDB [tiendabd]> exit
Bye
debian@ml-pedroamp:~$ mysqldump --user=tiendabd --password=password --databases tiendabd --add-drop-database --add-drop-table --host=127.0.0.1 --result-file=dump.sql
debian@ml-pedroamp:~$ cat dump.sql | tail -n 6
/*!40101 SET CHARACTER SET CLIENT=@OLD_CHARACTER SET CLIENT */;
/*!40101 SET CHARACTER SET RESULTS=@OLD_CHARACTER SET RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL NOTES=@OLD_SQL NOTES */;

-- Dump completed on 2022-05-20 11:04:21
debian@ml-pedroamp:~$
```

Figura 8: mysqldump de la base de datos específica con todas sus tablas y donde podemos ver que el read lock es efectivo.

```
peter@mark-6: ~
debian@ml1-pedroamp: ~

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)]> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> quit
Bye
debian@ml1-pedroamp:~$ mysql -u tiendabd -p -b tiendabd -e "INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Pantalones XXL', 10);"
Enter password:
debian@ml1-pedroamp:~$ mysql -u tiendabd -p -b tiendabd -e "INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Pantalones XXL', 10);"
Enter password:
ERROR 1064 (20000): Access denied for user 'tiendabd@localhost' (using password: YES)
debian@ml1-pedroamp:~$ mysql -u tiendabd -p -b tiendabd
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 42
Server version: 10.5.15-MariaDB-0+deb11u1 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 [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Pantalones XXXL', 1220);
Query OK, 1 row affected (0.002 sec)

MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Pantalones XXXXXL', 1220);
Query OK, 1 row affected (0.002 sec)

MariaDB [tiendabd]> quit
Bye
debian@ml1-pedroamp:~$ sudo mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 43
Server version: 10.5.15-MariaDB-0+deb11u1 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)]> LOCK TABLES WITH READ LOCK;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'WITH READ LOCK' at line 1
MariaDB [(none)]> exit
Bye
debian@ml1-pedroamp:~$ sudo mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 44
Server version: 10.5.15-MariaDB-0+deb11u1 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)]> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.001 sec)

debian@ml1-pedroamp:~$ mysql -u tiendabd -p -b tiendabd
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 53
Server version: 10.5.15-MariaDB-0+deb11u1 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 [tiendabd]> SELECT * FROM STOCK;
+----+-----+-----+
| ID_PRODUCTO | NOMBRE | CANTIDAD |
+----+-----+-----+
| 1 | Camisetas L | 10 |
| 2 | Camisetas M | 10 |
| 3 | Camisetas XL | 10 |
| 4 | Camisetas XXL | 10 |
| 5 | Pantalones L | 10 |
| 6 | Pantalones M | 10 |
| 7 | Pantalones XL | 10 |
| 8 | Pantalones XXL | 10 |
| 9 | Jersey L | 10 |
| 10 | Jersey M | 10 |
| 11 | Jersey XL | 10 |
| 12 | Jersey XXL | 10 |
| 13 | Patatas L | 10 |
| 14 | Patatas M | 10 |
| 15 | Patatas XL | 10 |
| 16 | Patatas XXL | 10 |
| 17 | Pantalones XXL | 10 |
| 18 | Pantalones XXL | 10 |
| 19 | Pantalones XXXL | 1220 |
| 20 | Pantalones XXL | 10 |
| 21 | Pantalones XXXL | 1220 |
| 22 | Pantalones XXXXXL | 1220 |
+----+-----+-----+
22 rows in set (0.001 sec)

MariaDB [tiendabd]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Pantalones XXXXXL', 1220);
^C^C^C -- query killed. Continuing normally.
ERROR 1317 (70100): Query execution was interrupted
MariaDB [tiendabd]> exit
Bye
debian@ml1-pedroamp:~$ mysqldump --user=tiendabd --password=password --databases tiendabd --add-drop-database --add-drop-table --host=1
27.0.0.1 --result-file=dump.sql
debian@ml1-pedroamp:~$ cat dump.sql | tail -n 6
/*401801 SET CHARACTER SET CLIENT=@OLD_CHARACTER SET CLIENT */;
/*401801 SET CHARACTER SET RESULTS=@OLD_CHARACTER SET RESULTS */;
/*401801 SET COLLATION CONNECTION=@OLD_COLLATION CONNECTION */;
/*401811 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2022-05-20 11:04:21
debian@ml1-pedroamp:~$
```

Figura 9: Read lock en paralelo. Se muestra a la izquierda abajo. El READ LOCK que permite la extracción segura de los datos.

```
Bye
debian@ml1-pedroamp:~$ sudo mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 44
Server version: 10.5.15-MariaDB-0+deb11u1 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)]> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> UNLOCK TABLES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]>
```

Figura 10: Read lock liberado.

Comando parecido al anterior pero que no selecciona solo la base de datos de tiendabd si no todas las bases de datos posibles (Necesario el usuario root).

- 1 # Para todas las bases de datos en el RDBMS
- 2 mysqldump --user=root --password=password --all-databases --add-drop-database --add-drop-table [--replace] --host=127.0.0.1 --result-file=dump.sql

Comando para hacer un dump de backup lógico de una tabla solo.

- 1 # Para todas las bases de datos en el RDBMS

```
2      mysqldump --user=root --password=password --databases tiendabd STOCK
      --add-drop-table [--replace] --host=127.0.0.1 --result-file=
      dumptable.sql
```

Explicación del funcionamiento de los parámetros pasados:

- **--user=[user]** Indicamos el usuario que debe tener cumplimentadas las condiciones para poder ejecutar mysqldump.
- **--password=[password]** Indicamos la contraseña del usuario.
- **--databases [database table0 table1 table2] | --all-databases** Especifica una o todas las bases de datos a realizar el dump de backup lógico. En el fichero lógico se crea un CREATE DATABASE. Table0 a tableN es una especificación más concreta donde se le indica a mysqldump que haga dump de las tablas especificadas de esa base de datos.
- **--add-drop-database --add-drop-table [no incluido --add-drop-trigger]** Este parámetro le indica a mysqldump que haga un DROP detrás de cada sentencia de CREATE DATABASE TABLE o TRIGGER. Esto es para evitar que en bases de datos de réplica, se tenga que hacer un proceso manual donde tengamos los sysdba, que eliminar las antiguas bases de datos, tablas, etc... Manualmente, es lo que hago yo en los ficheros ya adjuntados en esta memoria, me parece muy útil este parámetro para regenerar bases de datos y sus tablas (con sus datos) asociadas.
- **--replace** Es un parámetro con el cual si hacemos un --add-drop-table es opcional (esto es en mi experiencia, no sale en la especificación como tal). Ya que la tabla que aloja las filas de datos, deja de existir con el DROP TABLE generado por mysqldump. Entonces este parámetro, serviría para los casos en los que no borras bases de datos o tablas previamente. Es decir ***solo actualizas datos, mysqldump crea sentencias REPLACE en vez de INSERT por cada fila que encuentre en cada tabla.*** REPLACE funciona de la misma manera que un INSERT, solo que si existe en la tabla una clave primaria que sea igual que la que intenta "INSERTAR", la reemplaza directamente. Resultando en que se conserva la clave primaria y los mismos datos (esto último depende de si el REPLACE y sus valores son iguales. Si no lo son se actualizan a la última versión con el replace).
- **--hostname=[hostname or IP]** Indica la dirección del servidor de Mariadb (MySQL también) objetivo al que aplicar el mysqldump.
- **--result-file=[pathfile.sql]** Indica la ruta o si se omite el nombre del fichero SQL que contiene el dump del backup lógico.

2.4. Bloqueo de tablas automático con mysqldump

Como sabemos el uso de READ LOCK o LOCK TABLES implica tener una sesión de SQL abierta en otra consola. Lo cual es muy engorroso a la hora de hacer una doble

autenticación por lo que el administrador de la base de datos. Tiene que estar pendiente de las dos cosas a la vez así como de verificar que todo vaya correctamente. Eso se puede prevenir con la opción de **-add-locks**. Que es lo mismo que hacer un LOCK TABLES sobre las tablas afectadas por la base de datos.

```
1      # Para una base de datos concreta
2      mysqldump --user=tiendabd --password=password --databases tiendabd --
          add-drop-database --add-drop-table --add-locks [--replace] --host
          =127.0.0.1 --result-file=dump.sql
```

2.5. Transmisión de los datos a la réplica.

En la guía de trabajo, se ha especificado un método manual con SCP, donde interviene el proceso de copiar el fichero a mano desde donde está el dump en el servidor primario al Servidor Replica. Para finalmente tener que ejecutar en una sesión SSH en el Servidor Replica, mysqldump para obtener la información actualizada.

Pero he desarrollado y probado una alternativa mejor:

Implica la realización de un **SSH port-forwarding Local** donde el Servidor Primario (como Cliente SSH), que hace el dump. Trae de manera 'local' el puerto de SQL del Servidor Replica (Servidor SSH) que recibe el dump. La conexión al ser realizada con SSH. Asegura la confidencialidad y integridad gracias a la seguridad de SSL. Evitamos también el proceso de doble autenticación en el servidor Réplica, obteniendo una consola de SSH, para luego ir a la ruta del dump y que el cliente de SQL lo formatee para la base de datos.

El procedimiento resumido sin detalles es:

El cliente crea un socket TCP/UDP en el puerto de escucha indicado. Donde debe recibir las peticiones a reenviar hacia el servidor SSH. El proceso es similar a cuando hacemos port-forwarding en una NAT. En este caso el cliente SSH se comporta como un router de frontera NAT, donde las peticiones que les lleguen a su dirección y puerto indicados. Son automáticamente reenviadas con una previa conversión de puertos si Origen es distinto a Destino (Forward), a través del tunel SSH donde el servidor auténtico las recibe y envía la respuesta de vuelta a través del mismo tunel.

Virtualmente en el localhost del Primary, con el tunel de SSH. Es como si tuvieran dos RDBMS ejecutándose en el mismo sistema.

Lo visualizamos abajo de manera más clara con el script.

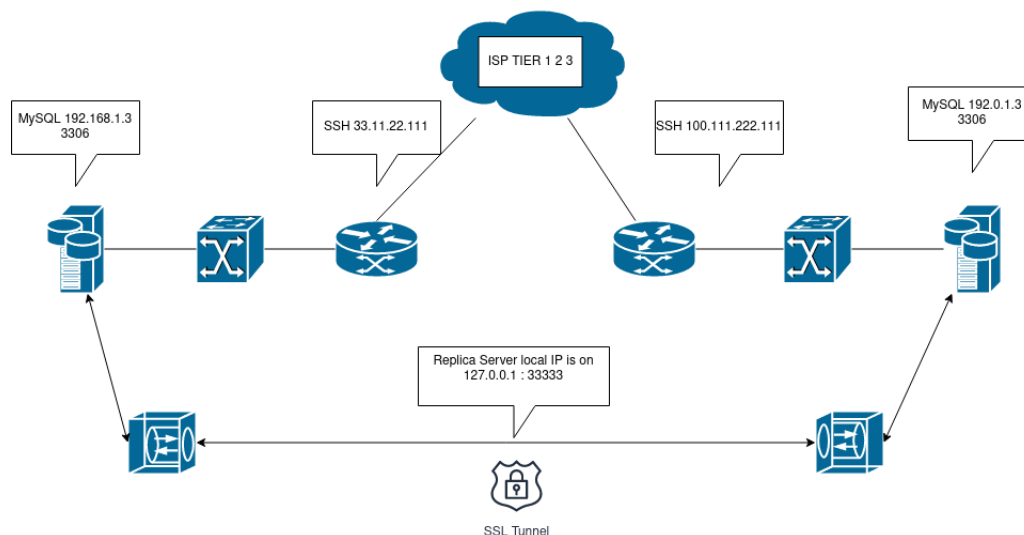


Figura 11: SSH port-forwarding ilustración, se puede visualizar que el tunel es como si tuvieran un cable directo tal y como haría un Router - Servidor en una NAT.

- 1 `# Comando basico de Port-Forward en Local. Esto es que se crea un socket local que se redirecciona al remoto en ese puerto indicado para que se comporte como si estuviera en el mismo sistema.`
- 2 `# O si lo prefieren como los Router NAT donde la NAT Es el equipo local que redirecciona las peticiones al remoto con el servicio.`
- 3 `ssh -fNT -L localhost:3307:localhost:3306 debian@m2-pedroamp`

El funcionamiento paso por paso de la foto:

- En la primera parte tenemos que hacer una "Sesión de cliente-servidor SSH", para crear el tunel y luego SSH se encarga de hacer Port-Forward desde el puerto 33333 en localhost hacia el puerto 3306 del Localhost del Host indicado de m2-pedroamp. **Se mejora si usamos clave pública para no autenticarnos con contraseña y poder ponerlo en CRON.**
- En la segunda parte: Ejecutamos **Lsof -i -n** que me enseñe los ficheros de sockets de TCP/UDP de redes abiertos en el sistema junto con su PID del proceso. Sus parámetros son:
 - **-n impide la conversión a hostname, es decir muestra la IP.** El PID es ideal saberlo para poder mandar un Kill de Signal de **SIGTERM** que finaliza el proceso en segundo plano liberando el socket al terminar la conexión ya que está en segundo plano la sesión de SSH.
- En la tercera y última parte podemos verificar que se ha restaurado en la base de datos remota el dump. Todo eso realizado a través del tunel que se comporta como si el RDBMS estuviera también en localhost.

```

debian@ml-pedroamp:~$ sudo mysql --host=127.0.0.1 --port=33333 -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 35
Server version: 10.5.15-MariaDB-0+deb11u1 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 |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.002 sec)

MariaDB [(none)]> SELECT USER, HOST FROM mysql.user;
+-----+-----+
| User | Host |
+-----+-----+
| mariadb.sys | localhost |
| mysql | localhost |
| root | localhost |
+-----+-----+
3 rows in set (0.003 sec)

MariaDB [(none)]> █

```

Figura 12: SSH port-forwarding el tunel se ha creado y se puede ver que hay un proceso que indica que se ha abierto un puerto concreto que es visible.

El proceso conocido como **Local Forwarding** sus ajustes descritos son los siguientes:

1. Existe un cliente y un servidor SSH que tienen capacidad de realizar tuneles de SSH. Realizan una conexión clásica y mantienen el tunel conectado.
2. Para configurar el lado del cliente en escucha, se le indica **SourceDirection:SourcePort**. Hace que en el lado cliente, se prepare un puerto (Socket) TCP/UDP configurado en modo Listen en la dirección que se le ha indicado (generalmente Localhost). Luego todas las peticiones que reciba el en dicha Dirección y Puerto, se reenvían automáticamente a la parte indicada del lado "Servidor". Esto es como en los routers, donde su IP pública y su puerto local es utilizado desde fuera como método para contactar a un servicio que tiene detrás de una NAT que no es directamente accesible. Si el puerto de Destino es diferente convierte el Puerto de Destino (En el cual está escuchando), al auténtico del servidor.
 - Esto ocurre muy a menudo si utilizamos un servicio que esté en el mismo puerto en ambos lados como en nuestro caso el de SQL el 3306. Con su socket ya ocupado. Entonces el cliente debe escuchar las peticiones en otro diferente, como ejemplo: 33333. Que luego cuando recibe la petición edita su capa de transporte (TCP/UDP) y pone el puerto auténtico del servicio.
 - Si se omite la SourceDirection o como se indica en el manpage bind-address. Se utiliza la dirección localhost por defecto en la escucha local.
3. En el lado del servidor donde es el destino de la redirección (FORWARD) de los paquetes. Su configuración específica es **DestinationDirection:DestinationPort**. Las peticiones que reciba a través del tunel establecido por el cliente, cuando llegan al lado del servidor este las recibe a la dirección indicada y las manda de vuelta

como si el cliente estuviera en una red local. Con la particularidad, que con los túneles, podemos hacer port-forwarding, de servicios que solo funcionan en localhost. MySQL y MariaDB en su configuración de bind-address por defecto no funcionan con la IP privada, solamente en localhost. Es útil indicar una dirección IP privada particular donde por razones de seguridad, se limite un servicio a funcionar en dicha IP:PORT a pesar de que el servidor tenga más direcciones y interfaces.

4. El -fNT final indica que:
 - -f Indica que la petición de SSH pasa a ser en background después de la ejecución del comando.
 - -N No ejecuta una comandos remotos.
 - -T Desactiva la reserva de una pseudo-tty consola.

Nota: Podemos utilizar el script con CRON para que vaya replicando la base de datos cada cierto tiempo. Sin tener que utilizar binlog del modo Primary Réplica. Ya que el fichero dump contiene DROPs que permiten resetear la base de datos entera evitando que haya duplicidades o colisiones (con mayor costo en el sistema RDBMS Réplica al tener que borrar y luego reinsertar un dato), pero que deja el mismo resultado que en el Primary.

```

1  #!/bin/bash
2
3  # Database authentication
4  userdb='tiendabd '
5  passworddb='password '
6  database='tiendabd '
7
8  # Ports and Port-Forwarding config
9  localport='3306 '
10 portdb2='3306 '
11 portforwarddb2='33333 '
12 localhost='127.0.0.1 '
13 userhostnameserver='debian@m2-pedroamp '
14
15 # Misc Dump config
16 dumpfilename='dump.sql '
17
18 echo "Dumping localdatabase"
19 mysqldump --user="$userdb" --password="$passworddb" --databases $database
    --add-drop-database --add-drop-table --add-locks --host="$localhost" --
    port="$localport" --result-file=dump.sql
20
21 echo "Creating SSH tunnel and Local Port-Forward Replica database"
22 ssh -fNT -L $localhost:$portforwarddb2:$localhost:$portdb2
    $userhostnameserver
23
24 echo "Show port running"
25 sudo lsof -i -n | egrep $portforwarddb2
26 killport='sudo lsof -i -n | egrep $portforwarddb2 | awk '{print $2}''
27
28 echo "Dumping on Replica DB"
29 echo "Creating USER AND DATABASE"
30 sudo mysql --host="$localhost" --port="$portforwarddb2" -u root -p"
    $passworddb" < DDLCL.sql
31
32 echo "Recreating Database on Replica"
33 sudo mysql --host="$localhost" --port="$portforwarddb2" -u $userdb -p"
    $passworddb" < dump.sql
34
35 echo "SHOW Database"
36 sudo mysql --host="$localhost" --port="$portforwarddb2" -u $userdb -p"
    $passworddb" -e 'SHOW DATABASES;'
37
38 echo "SHOW TABLES"
39 sudo mysql --host="$localhost" --port="$portforwarddb2" -u $userdb -p"
    $passworddb" -D "$database" -e 'SHOW TABLES;'
40
41 echo "SELECT TABLE STOCK"
42 sudo mysql --host="$localhost" --port="$portforwarddb2" -u $userdb -p"
    $passworddb" -D "$database" -e 'SELECT * FROM STOCK;'
43
44 echo "Terminating Port Forward"
45 kill $killport

```

Podemos ver la ejecución del script y el resultado de que se ha ejecutado correctamente:

```
debian@ml-pedroamp:~$ ./script.sh
Dumping localdatabase
Creating SSH tunnel and Local Port-Forward Replica database
debian@ml2-pedroamp's password:
Show port running
ssh      2485  debian    4u  IPv4  26481      0t0  TCP 127.0.0.1:33333 (LISTEN)
Dumping on Replica DB
Creating USER AND DATABASE
Recreating Database on Replica
SHOW Database
+-----+
| Database |
+-----+
| information_schema |
| tiendabd |
+-----+
SHOW TABLES
+-----+
| Tables_in_tierendabd |
+-----+
| ALTA_CLIENTE |
| BAJA_CLIENTE |
| CIUDADES |
| CLIENTE |
| EMPLEADO |
| MOD_STOCK |
| PEDIDO |
| REPARTIDOR |
| STOCK |
+-----+
SELECT TABLE STOCK
+-----+
| ID_PRODUCTO | NOMBRE | CANTIDAD |
+-----+
| 1 | Camisetas L | 10 |
| 2 | Camisetas M | 10 |
| 3 | Camisetas XL | 10 |
| 4 | Camisetas XXL | 10 |
| 5 | Pantalones L | 10 |
| 6 | Pantalones M | 10 |
| 7 | Pantalones XL | 10 |
| 8 | Pantalones XXL | 10 |
| 9 | Jersey L | 10 |
| 10 | Jersey M | 10 |
| 11 | Jersey XL | 10 |
| 12 | Jersey XXL | 10 |
| 13 | Patatas L | 10 |
| 14 | Patatas M | 10 |
| 15 | Patatas XL | 10 |
| 16 | Patatas XXL | 10 |
+-----+
Terminating Port Forward
debian@ml-pedroamp:~$
```

Figura 13: Script semiautomático mejorable de hacer un dump a una base de datos Replica.

3. Servidores Primario y Réplica

En esta sección se tratará el tema de cómo realizar dos servidores Primario y Réplica

3.1. Configurando el servidor Primary

El servidor primary elegido es *m1-pedroamp*.

Debemos ir al fichero **my.cnf** localizado en `/etc/mysql/my.cnf`. Donde la directiva se describe como lo siguiente:

```
1 # Inicio de la directiva de MariaDB
2 [mariadb]
3 # Log binario activado
4 log-bin
5 # Identificador del servidor
6 server_id=1
7 # Formato del nombre del log, es decir el prefijo que deben tomar tanto los
   PID como los ficheros de log.
8 log-basename=m1-pedroamp
9 # Formato de log binario. See Below
10 binlog-format=mixed
```

binlog-format Es una directiva que le indica al RDBMS (SGBD). Cómo almacena un registro de los cambios realizados en sus base de datos. El formato del binlog tiene un factor determinante, ya que funciona como un sistema de copias de seguridad diferencial. En concreto en sus especificaciones se indica qué impacto tiene usar uno de los *2 formatos de binlog*.

Nota: Algunos autores definen el uso de Row-based logging como Row-Based Replication y Statement-Based Logging como Statement-Based Replication. En referencia a que esto se utiliza más cuando levantamos servidores Primary-Replica.

Esto ocurre ya que el servidor Replica, lee del Primario este log para ejecutar las mismas sentencias SQL que se han ejecutado en el Primario. Es similar a cómo Mysqldump trabaja, solo que al ser a nivel binario no tiene que ejecutar un intérprete de SQL que traduzca ese lenguaje al modo binario.

1. Si ejecutamos sentencias SQL simples que actualizan muchas filas. Se recomienda utilizar de binlog: **Row-Based Logging**. Realiza lo siguiente:
 - Para indicar a la base de datos que lo utilice, ponemos: **binlog-format=ROW**.
 - Las sentencias DML Select Insert Update Delete **no** son registradas en el log binario. Pero a cambio, el resultado de cada ejecución de esa sentencia, se registra como resultado final en el binlog. Esto es si hacemos un Insert, el insert no se guarda, pero el binlog almacena el resultado diferencial final. Es decir, si hacemos INSERT INTO TABLE VALUES (1,...). Se crea un TABLE ->ROW 1021321 ->1,...

- Las sentencias DDL CREATE ALTER DROP TRUNCATE **sí** son registrados en el log binario.
 - Este es el método más seguro de registrar toda la información ocurrida en la base de datos. Pero registra todos los cambios ocurridos en una Tabla misma y en filas específicas.
2. En cambio si usamos **Statement-Based Logging**: Se registran todas las sentencias SQL tal y como se han ejecutado sobre la base de datos. Con el añadido de que las tablas temporales (Temporary Tables - O tablas desechables temporales) se registran en el servidor de Réplica también.
- Para indicar a la base de datos que lo utilice, ponemos: **binlog-format=STATEMENT**.
 - Fuerza a la Réplica a mantener con una latencia mínima con el mismo motor de almacenamiento, la misma información definitiva y temporal con el Primario. Donde en algunos casos en el Réplica no puede ser utilizado para insertar o manipular información. Solo se ve en modo "solo lectura" por problemas con como el servidor Réplica debe tratar de seguir replicando el primario y se empieza a diferenciar en una sola sentencia que modifique, cree, o borre, Algún dato definitivo o temporal. Esto ocurre porque las sentencias son Determinísticas, deben producir el mismo resultado en las tablas donde se almacenan. Entonces cuando registramos esa sentencia en el binlog del Slave donde se produciría un cambio inseguro como INSERT, donde tenemos una columna que tiene un ID de clave Primaria de AUTO_INCREMENT. Esto produce un resultado paralelo que es inseguro, porque en el caso de que en el Primary se haga lo mismo el AUTO_INCREMENT va a producir que a la hora de replicarlo hayan dos filas con la misma clave primaria pero con dos datos diferentes al tener que ejecutar la sentencia SQL replicada como tal.
 - Método recomendado para los que quieran mantener el binlog en un tamaño mínimo de binlog.
3. **Mixed Logging**: Usa una combinación de los dos anteriores. Siendo el por defecto el Statement-Based Logging. Pero si el servidor determina que hay una sentencia SQL que no es segura por el Statement-Based Logging como por ejemplo el ejemplo anterior, lo almacena en modo Row-Based Logging en binario. *No especifican el cómo se decide esto. En la documentación de MariaDB y la de MySQL. Solo que cuando leen sentencias no seguras en el réplica se pasa a binario pero no dice cómo traza eso para evitar problemas futuros en futuras replicasiones.*

```
#
[client-server]
# Port or socket location where to connect
# port = 3306
socket = /run/mysqld/mysqld.sock

# Import all .cnf files from configuration di
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/

[mariadb]
log-bin
server-id=1
log-basename=m1-pedroamp
binlog-format=mixed
```

Figura 14: Configuración del servidor primario.

Finalmente reiniciamos el servicio de MariaDB o MySQL.

- 1 `sudo systemctl restart mariadb.service`

Luego podemos ejecutar en MariaDB en la consola el siguiente comando para verificar que se ha producido el cambio.

- 1 `SHOW VARIABLES LIKE 'binlog_format'`

```
debian@m1-pedroamp:~$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.5.15-MariaDB-0+deb11u1-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 VARIABLES LIKE 'binlog_format';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| binlog_format | MIXED |
+-----+-----+
1 row in set (0.002 sec)

MariaDB [(none)]>
```

Figura 15: Configuración del servidor primario. Verificando cambios.

Luego tenemos que crear un usuario de Replicación para que el servidor replica pueda obtener acceso a los permisos de leer el binlog a modo de poder realizar la replicación. *Necesitamos el usuario root.*

- 1 `CREATE USER 'replication_user'@'%' IDENTIFIED BY 'password';`
- 2 `GRANT REPLICATION SLAVE ON *.* TO 'replication_user'@'%';`
- 3 `FLUSH PRIVILEGES;`

```
MariaDB [(none)]> CREATE USER 'replication_user'@'%' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> GRANT REPLICATION SLAVE ON *.* TO 'replication_user'@'%';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)
```

Figura 16: Creando usuario de replicación.

Luego tenemos que preparar el servidor para que acepte accesos fuera de la red. Tenemos que ir al fichero `/etc/mysql/mariadb.conf.d/50-server.cnf`, para ir a la directiva `bind-address`. Que permita escuchar peticiones en la dirección privada del servidor.

```
GNU nano 5.4 /etc/mysql/mariadb.conf.d/50-server.cnf
#skip-name-resolve

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
```

Figura 17: Configurando el SGBD para permitir escuchar peticiones en la dirección IP privada.

En el servidor replica debemos ir también al fichero `my.cnf` localizado en `/etc/mysql/` `my.cnf`. Y poner lo siguiente:

```
[mysqld]
log-bin
server_id=2
```

Figura 18: Configurando el SGBD del réplica.

- **NOTA:** Las claves del Replica y del Primary deben ser únicas.
- El replica no necesita indicarle el formato del binlog. Simplemente solo que utilice el modo del primary.

Finalmente reiniciamos el servicio en el servidor Réplica que aplique los cambios.

```
1 sudo systemctl restart mariadb.service
```

```
debian@m2-pedroamp:~$ cat /etc/mysql/my.cnf | tail -n 3
[mysqld]
log-bin
server_id=1
debian@m2-pedroamp:~$ sudo systemctl restart mariadb.service
^[[Adebian@m2-pedroamp:~$ sudo systemctl status mariadb.service
● mariadb.service - MariaDB 10.5.15 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-05-23 19:33:53 UTC; 2s ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 1598 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysql (code=exited, status=0/SUCCESS)
   Process: 1599 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Process: 1601 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera recovery ] && VAR= [ ] VAR='cd /usr/bin/./; /usr/bin/galera_recovery'; [
   Process: 1672 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Process: 1674 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
   Main PID: 1655 (mariadbd)
   Status: "Taking your SQL requests now..."
     Tasks: 17 (limit: 4678)
    Memory: 69.3M
       CPU: 274ms
    CGroup: /system.slice/mariadb.service
           └─1655 /usr/sbin/mariadbd

May 23 19:33:53 m2-pedroamp mariadbd[1655]: Version: '10.5.15-MariaDB-0+deb11u1-log' socket: '/run/mysqlld/mysqlld.sock' port: 3306 Debia
May 23 19:33:53 m2-pedroamp systemd[1]: Started MariaDB 10.5.15 database server.
May 23 19:33:53 m2-pedroamp /etc/mysql/debian-start[1676]: Upgrading MySQL tables if necessary.
May 23 19:33:53 m2-pedroamp mariadbd[1655]: 2022-05-23 19:33:53 4 [Warning] Access denied for user 'root'@'localhost' (using password: NO)
May 23 19:33:53 m2-pedroamp /etc/mysql/debian-start[1679]: Looking for 'mysql' as: /usr/bin/mysql
May 23 19:33:53 m2-pedroamp /etc/mysql/debian-start[1679]: Reading data from the MariaDB server failed. Got the following error when ex
May 23 19:33:53 m2-pedroamp /etc/mysql/debian-start[1679]: ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:
May 23 19:33:53 m2-pedroamp /etc/mysql/debian-start[1679]: FATAL ERROR: Upgrade failed
May 23 19:33:53 m2-pedroamp mariadbd[1655]: 2022-05-23 19:33:53 5 [Warning] Access denied for user 'root'@'localhost' (using password: NO)
May 23 19:33:53 m2-pedroamp debian-start[1688]: ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
lines 1-28/28 (END)
```

Figura 19: Configurando el SGBD del réplica - Reiniciando el servicio.

3.2. Inicio del proceso de réplica.

Para iniciar el proceso de réplica debemos hacer un **READ LOCK** en el servidor primario. Para que se prepare para pasar sus datos al réplica. Hacemos el **READ LOCK** para que no haya más cambios y ejecutamos el script de copia automática de los dumps. Luego, necesitamos verificar la posición del LOG y su nombre. Para configurar la réplica correctamente.

- 1 LOCK TABLES WITH **READ LOCK**;
- 2 SHOW MASTER STATUS;

```
debian@m1-pedroamp:~$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.5.15-MariaDB-0+deb11u1-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)]> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| m1-pedroamp-bin.000003 | 348 | | |
+-----+-----+-----+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> █
```

Figura 20: Preparando la réplica del servidor MariaDB verificando el fichero binlog.

Luego en el terminal de m2-pedroamp, debemos poner los siguientes comandos:

- **CHANGE MASTER TO** Indicamos los parámetros que permiten al servidor replica funcionar. Son:
 1. **MASTER_HOST='IP HOSTNAME'** Indicamos la dirección del servidor primario.
 2. **MASTER_USER='replication_user'** Indicamos el usuario con los permisos de Lectura del binlog para el Replica (SLAVE tal y como se pone en su sentencia).
 3. **MASTER_PASSWORD='password'** Indicamos la contraseña del usuario de réplica.
 4. **MASTER_PORT=3306** Indicamos el puerto de SQL del servidor primario.
 5. **MASTER_LOG_FILE=binlog.0000...** Indicamos el nombre del binlog en el master que debemos replicar.
 6. **MASTER_LOG_POST=POS** Indicamos la posición del fichero log, debemos comprobarlo con el **SHOW MASTER STATUS** en otra consola.


```

1 CHANGE MASTER TO
2   MASTER_HOST='192.168.122.70',
3   MASTER_USER='replication_user',
4   MASTER_PASSWORD='password',
5   MASTER_PORT=3306,
6   MASTER_LOG_FILE='m1-pedroamp-bin.000003',
7   MASTER_LOG_POS=348,
8   MASTER_CONNECT_RETRY=10;
9 — Iniciamos el servidor Replica
10 START SLAVE;
11 — Verificamos el estado del servidor replica
12 SHOW SLAVE STATUS \G;

```

```

MariaDB [(none)]> CHANGE MASTER TO
-> MASTER_HOST='192.168.122.70',
-> MASTER_USER='replication_user',
-> MASTER_PASSWORD='password',
-> MASTER_PORT=3306,
-> MASTER_LOG_FILE='m1-pedroamp-bin.000003',
-> MASTER_LOG_POS=348,
-> MASTER_CONNECT_RETRY=10;
Query OK, 0 rows affected (0.008 sec)

MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> SHOW SLAVE STATUS \G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.122.70
Master_User: replication_user
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: m1-pedroamp-bin.000003
Read_Master_Log_Pos: 348
Relay_Log_File: mysqld-relay-bin.000002
Relay_Log_Pos: 561
Relay_Master_Log_File: m1-pedroamp-bin.000003
Slave_IO_Running: Yes
Slave_SQL_Running: Yes

```

Figura 21: Servidor réplica trabajando - Podemos ver que tienen la misma posición del log binario.

NOTA: Para verificar que todo funciona correctamente, debemos ver las variables siguientes de **SHOW SLAVE STATUS \G**;

- **Slave_IO_Running: YES**
- **Slave_SQL_Running: YES**
- **Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates**

Se puede verificar que las réplicas trabajan rápidamente y obtienen los datos al instante en cuanto se termine de escribir el binlog del primario.

```
13 | Patatas L | 10 |
14 | Patatas M | 10 |
15 | Patatas XL | 10 |
16 | Patatas XXL | 10 |
-----
terminating Port Forward
ebian@ml-pedrocamp:~$ sudo mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.5.15-MariaDB-0+deb11u1-log Debian 11

Copyright (c) 2000, 2019, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SHOW MASTER STATUS;
-----
File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
-----
m1-pedrocamp-bin.000003 | 348 | | |
-----
row in set (0.000 sec)

MariaDB [(none)]> USE tiendab;
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 [tiendab]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XXL', 10);
Very OK, 1 row affected (0.002 sec)

MariaDB [tiendab]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XXXXXXL', 10);
Very OK, 1 row affected (0.029 sec)

MariaDB [tiendab]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XXXXXXL', 10);
Very OK, 1 row affected (0.002 sec)

MariaDB [tiendab]> INSERT INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XXXXL', 10);
Very OK, 1 row affected (0.002 sec)

MariaDB [tiendab]> exit
-> Ctrl-C -- exit!
borted
ebian@ml-pedrocamp:~$ sudo mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 17
Server version: 10.5.15-MariaDB-0+deb11u1-log Debian 11

Copyright (c) 2000, 2019, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SHOW MASTER STATUS;
-----
File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
-----
m1-pedrocamp-bin.000003 | 1313 | | |
-----
row in set (0.000 sec)

MariaDB [(none)]> 
```

```
MariaDB [(none)]> SHOW SLAVE STATUS \G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.122.70
Master_User: replication_user
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: m1-pedrocamp-bin.000003
Read_Master_Log_Pos: 1313
Relay_Log_File: mysql-relay-bin.000002
Relay_Log_Pos: 1320
Relay_Master_Log_File: m1-pedrocamp-bin.000003
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
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: 1313
Relay_Log_Space: 1836
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:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 1
Master_SSL_Crl:
Master_SSL_Crlpath:
Using Gtid: No
Gtid_IO_Pos:
Replicate_Do_Domain_Ids:
Replicate_Ignore_Domain_Ids:
Parallel_Mode: optimistic
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
Slave_Oid_Groups: 0
Slave_Non_Transactional_Groups: 0
Slave_Transactional_Groups: 4
1 row in set (0.000 sec)

ERROR: No query specified
MariaDB [(none)]> 
```

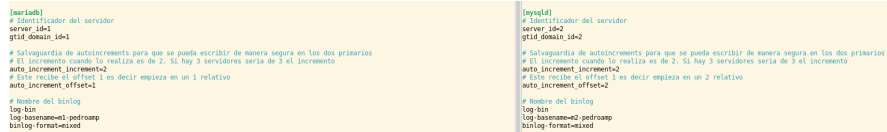
Figura 22: Servidor réplica trabajando - Podemos ver que al realizar cambios cambia la posición del log binario en la réplica al ponerse igual.

4. Servidores Primario y Primario

En esta sección se tratará el tema de cómo realizar dos servidores Primario y Primario. Primero en m2-pedroamp debemos parar el modo Replica antes de empezar el procedimiento. Para ello debemos poner la siguiente sentencia SQL:

```
1 RESET SLAVE;
```

Luego tenemos que configurar en los dos servidores primarios siguiente:



```
[m1-pedroamp]
# Identificador del servidor
server_id=1
gtid_domain_id=1

# Salvaguardia de autoincremento para que se pueda escribir de manera segura en los dos primarios
# El incremento cuando lo realiza es de 2. Si hay 3 servidores sería de 3 el incremento
auto_increment_increment=2
# Esto indica el offset 1 es decir empieza en un 1 relativo
auto_increment_offset=1

# Nombre del binlog
log_bin
log_slave_updates=1
binlog-format=mixed

[m2-pedroamp]
# Identificador del servidor
server_id=2
gtid_domain_id=2

# Salvaguardia de autoincremento para que se pueda escribir de manera segura en los dos primarios
# El incremento cuando lo realiza es de 2. Si hay 3 servidores sería de 3 el incremento
auto_increment_increment=2
# Esto indica el offset 2 es decir empieza en un 2 relativo
auto_increment_offset=2

# Nombre del binlog
log_bin
log_slave_updates=2
binlog-format=mixed
```

Figura 23: Configuración del Primario m1-pedroamp y m2-pedroamp para permitir el Primary Primary.

- **En los ID** Ponemos 1 para el id del primario m1 y 2 para el id del primario m2
- **Los offset (margen) de autoincremento** Como tenemos dos servidores de escritura primarios, debemos indicarle que se incremente en 2 para evitar problemas de EM donde escriban en la misma columna de datos. Empezando en orden 1 relativo para m1-pedroamp y 2 relativo para m2-pedroamp. Con relativo es, que si existen tablas ya que sea N+1 y N+2 al momento de la sincronización de la réplica inicial.
- **En el logbin** Se indica que existe un logbin primario del m2-pedroamp, antes solo tenía un relay-logbin que registraba lo ocurrido del primario. Pero aquí pasa a ser maestro así que gestiona su logbin.
- **GTID** Ver la sección de GTID.

Reiniciamos el servicio MariaDB server para aplicar los cambios

NOTA: Debemos activar el bind-address= 0.0.0.0 En m2-pedroamp, de lo contrario no permitirá al servidor m1-pedroamp sincronizarse. **NOTA2:** El usuario réplica está en ambas bases de datos recordando que del binlog se copia hasta la información de usuarios uniéndose a una misma base de datos mysql (que es donde se guarda la tabla de usuarios). Por lo tanto al sincronizarse estando de Esclavo antes m2-pedroamp ya tiene esa información. **NOTA3:** Si no se ha sincronizado antes la base de datos en el otro primario, se requiere hacer el mysqldump para que puedan ponerse al día en toda la información.

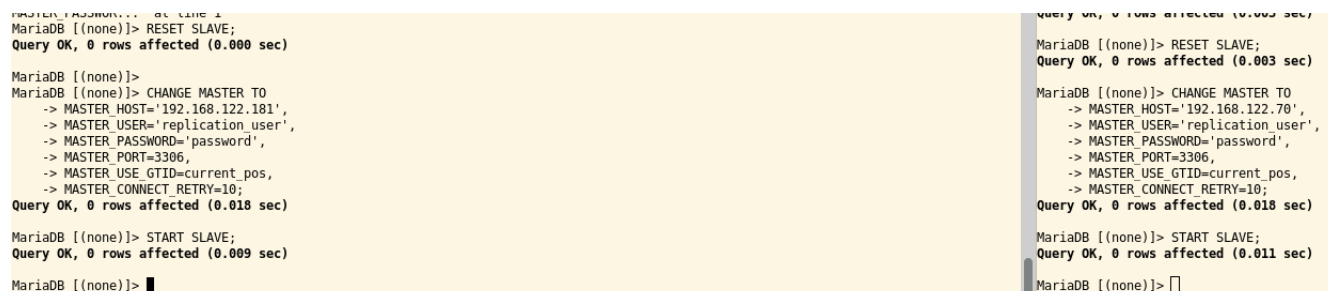
Luego finalmente en ambas consolas ponemos lo siguientes:

```
1 # m1-pedroamp
2
3 RESET SLAVE;
4 #Ver la seccion de GTID
```

```

5 # Limpia el GTID del servidor actual. Para que pueda obtener uno nuevo con
  el current_pos
6 SET GLOBAL gtid_slave_pos = "";
7
8 CHANGE MASTER TO
9 MASTER_HOST='192.168.122.119',
10 MASTER_USER='replication_user',
11 MASTER_PASSWORD='password',
12 MASTER_PORT=3306,
13 MASTER_USE_GTID=current_pos,
14 MASTER_CONNECT_RETRY=10;
15
16 START SLAVE;
17 # m2-pedroamp
18
19 RESET SLAVE;
20 # Ver la seccion de GTID
21 # Limpia el GTID del servidor actual. Para que pueda obtener uno nuevo con
  el current_pos
22 SET GLOBAL gtid_slave_pos = "";
23
24 CHANGE MASTER TO
25 MASTER_HOST='192.168.122.70',
26 MASTER_USER='replication_user',
27 MASTER_PASSWORD='password',
28 MASTER_PORT=3306,
29 MASTER_USE_GTID=current_pos,
30 MASTER_CONNECT_RETRY=10;
31
32 START SLAVE;

```



The image shows two side-by-side terminal windows. The left window shows the following commands and outputs:

```

MariaDB [(none)]> RESET SLAVE;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]>
MariaDB [(none)]> CHANGE MASTER TO
-> MASTER_HOST='192.168.122.181',
-> MASTER_USER='replication_user',
-> MASTER_PASSWORD='password',
-> MASTER_PORT=3306,
-> MASTER_USE_GTID=current_pos,
-> MASTER_CONNECT_RETRY=10;
Query OK, 0 rows affected (0.018 sec)

MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]>

```

The right window shows the following commands and outputs:

```

MariaDB [(none)]> RESET SLAVE;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> CHANGE MASTER TO
-> MASTER_HOST='192.168.122.70',
-> MASTER_USER='replication_user',
-> MASTER_PASSWORD='password',
-> MASTER_PORT=3306,
-> MASTER_USE_GTID=current_pos,
-> MASTER_CONNECT_RETRY=10;
Query OK, 0 rows affected (0.018 sec)

MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.011 sec)

MariaDB [(none)]>

```

Figura 24: Configuración del Primario m1-pedroamp y m2-pedroamp para permitir el Primary Primary.

Finalmente podemos probar que funciona con lo siguiente:

```

1 INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL,
  PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m2',132312789, 'calle rpg', 18000,
  '112345',1);
2 INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL,
  PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m1',132356789, 'calle rpg', 18000,
  '112345',1);

```

```

3
4 INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL,
  PASSWORD, ID_EMPLEADO_ALTA) VALUES ( 'm2r',232312789, 'calle rpg',
  18000, '112345',1);
5 INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL,
  PASSWORD, ID_EMPLEADO_ALTA) VALUES ( 'm1r',232356789, 'calle rpg',
  18000, '112345',1);
6
7 # Para visualizar el incremento con el offset.
8 INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL,
  PASSWORD, ID_EMPLEADO_ALTA) VALUES ( 'm1rr',232336789, 'calle rpg',
  18000, '112345',1);
9 INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL,
  PASSWORD, ID_EMPLEADO_ALTA) VALUES ( 'm1rlr',232336789, 'calle rpg',
  18000, '112345',1);

```

<pre> MariaDB [tiendadb]> INSERT INTO CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m1',123256789, 'calle rpg', 18000, '112345',1); Query OK, 1 row affected (0.002 sec) MariaDB [tiendadb]> SELECT * FROM CLIENTE; +----+-----+-----+-----+-----+-----+-----+-----+ ID_CLIENTE NOMBRE TELEFONO DIRECCION COD_POSTAL PASSWORD ID_EMPLEADO_ALTA ID_EMPLEADO_BAJA +----+-----+-----+-----+-----+-----+-----+-----+ 1 cliente 123456789 calle etsiit 18000 1234 1 NULL 2 cliente2 112345678 calle molotov 18000 12345 1 NULL 3 cliente3 234567890 calle azor 18000 4321 1 NULL 4 cliente4 123216789 calle rpg 18000 112345 1 NULL 10 m2 123212789 calle rpg 18000 112345 1 NULL 11 m1 123256789 calle rpg 18000 112345 1 NULL +----+-----+-----+-----+-----+-----+-----+-----+ 6 rows in set (0.001 sec) MariaDB [tiendadb]> SELECT * FROM CLIENTE; +----+-----+-----+-----+-----+-----+-----+-----+ ID_CLIENTE NOMBRE TELEFONO DIRECCION COD_POSTAL PASSWORD ID_EMPLEADO_ALTA ID_EMPLEADO_BAJA +----+-----+-----+-----+-----+-----+-----+-----+ 1 cliente 123456789 calle etsiit 18000 1234 1 NULL 2 cliente2 112345678 calle molotov 18000 12345 1 NULL 3 cliente3 234567890 calle azor 18000 4321 1 NULL 4 cliente4 123216789 calle rpg 18000 112345 1 NULL 10 m2 123212789 calle rpg 18000 112345 1 NULL 11 m1 123256789 calle rpg 18000 112345 1 NULL 12 m2r 232312789 calle rpg 18000 112345 1 NULL +----+-----+-----+-----+-----+-----+-----+-----+ 7 rows in set (0.001 sec) MariaDB [tiendadb]> INSERT INTO tiendadb.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m1r',2 32356789, 'calle rpg', 18000, '112345',1); Query OK, 1 row affected (0.002 sec) MariaDB [tiendadb]> SELECT * FROM CLIENTE; +----+-----+-----+-----+-----+-----+-----+-----+ ID_CLIENTE NOMBRE TELEFONO DIRECCION COD_POSTAL PASSWORD ID_EMPLEADO_ALTA ID_EMPLEADO_BAJA +----+-----+-----+-----+-----+-----+-----+-----+ 1 cliente 123456789 calle etsiit 18000 1234 1 NULL 2 cliente2 112345678 calle molotov 18000 12345 1 NULL 3 cliente3 234567890 calle azor 18000 4321 1 NULL 4 cliente4 123216789 calle rpg 18000 112345 1 NULL 10 m2 123212789 calle rpg 18000 112345 1 NULL 11 m1 123256789 calle rpg 18000 112345 1 NULL 12 m2r 232312789 calle rpg 18000 112345 1 NULL 13 m1r 232356789 calle rpg 18000 112345 1 NULL +----+-----+-----+-----+-----+-----+-----+-----+ 8 rows in set (0.001 sec) MariaDB [tiendadb]> </pre>	<pre> 6 rows in set (0.001 sec) MariaDB [(none)]> INSERT INTO CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m2r',232312789, 'calle rpg', 18000, '112345',1); ERROR 1046 (3D000): No database selected MariaDB [(none)]> SELECT * FROM tiendadb.CLIENTE; +----+-----+-----+-----+-----+-----+-----+-----+ ID_CLIENTE NOMBRE TELEFONO DIRECCION COD_POSTAL PASSWORD ID_EMPLEADO_ALTA ID_EMPLEADO_BAJA +----+-----+-----+-----+-----+-----+-----+-----+ 1 cliente 123456789 calle etsiit 18000 1234 1 NULL 2 cliente2 112345678 calle molotov 18000 12345 1 NULL 3 cliente3 234567890 calle azor 18000 4321 1 NULL 4 cliente4 123216789 calle rpg 18000 112345 1 NULL 10 m2 123212789 calle rpg 18000 112345 1 NULL 11 m1 123256789 calle rpg 18000 112345 1 NULL +----+-----+-----+-----+-----+-----+-----+-----+ 6 rows in set (0.000 sec) MariaDB [(none)]> INSERT INTO tiendadb.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m2r',232 312789, 'calle rpg', 18000, '112345',1); Query OK, 1 row affected (0.002 sec) MariaDB [(none)]> SELECT * FROM tiendadb.CLIENTE; +----+-----+-----+-----+-----+-----+-----+-----+ ID_CLIENTE NOMBRE TELEFONO DIRECCION COD_POSTAL PASSWORD ID_EMPLEADO_ALTA ID_EMPLEADO_BAJA +----+-----+-----+-----+-----+-----+-----+-----+ 1 cliente 123456789 calle etsiit 18000 1234 1 NULL 2 cliente2 112345678 calle molotov 18000 12345 1 NULL 3 cliente3 234567890 calle azor 18000 4321 1 NULL 4 cliente4 123216789 calle rpg 18000 112345 1 NULL 10 m2 123212789 calle rpg 18000 112345 1 NULL 11 m1 123256789 calle rpg 18000 112345 1 NULL 12 m2r 232312789 calle rpg 18000 112345 1 NULL +----+-----+-----+-----+-----+-----+-----+-----+ 7 rows in set (0.001 sec) MariaDB [(none)]> SELECT * FROM tiendadb.CLIENTE; +----+-----+-----+-----+-----+-----+-----+-----+ ID_CLIENTE NOMBRE TELEFONO DIRECCION COD_POSTAL PASSWORD ID_EMPLEADO_ALTA ID_EMPLEADO_BAJA +----+-----+-----+-----+-----+-----+-----+-----+ 1 cliente 123456789 calle etsiit 18000 1234 1 NULL 2 cliente2 112345678 calle molotov 18000 12345 1 NULL 3 cliente3 234567890 calle azor 18000 4321 1 NULL 4 cliente4 123216789 calle rpg 18000 112345 1 NULL 10 m2 123212789 calle rpg 18000 112345 1 NULL 11 m1 123256789 calle rpg 18000 112345 1 NULL 12 m2r 232312789 calle rpg 18000 112345 1 NULL 13 m1r 232356789 calle rpg 18000 112345 1 NULL +----+-----+-----+-----+-----+-----+-----+-----+ 8 rows in set (0.001 sec) MariaDB [(none)]> </pre>
---	---

Figura 25: Pruebas del Primary Primary.

Al principio podemos ver que las tablas son iguales. Luego en el m2, indicado a la derecha. Se inserta una fila encabezada por m2r para su mejor distinción. Luego a continuación a la izquierda podemos visualizar que automáticamente se ha llevado al otro servidor.

Finalmente en el m1 a la izquierda se vuelve a insertar una fila de datos encabezada por m1r. Que al ejecutarse y a continuación se replica en el m2.

¿Cómo sabemos que no es el mismo RDBMS? Si nos fijamos en un pequeño detalle, vemos que el incremento de m1 tiene un offset (esto es donde comienza) de 1 y un incremento de 2, entonces si el primer dato de m1 es 11 su incremento de 2 lo ha llevado a 13. Luego en m2 tiene un offset de 2 y un incremento de 2. Por lo que se encarga de los pares.

NOTA: No realmente uno se va a encargar de los pares y otro de los impares. Depende de donde esté el último incremento de antes de hacer los dos servidores Primarios. Esto es decir si estamos en 1023 m1 va a poder insertar el 1024 y m2 inserta el 1025.

Esto se ha indicado en la configuración para evitar colisiones de datos autoincrementales, donde un servidor se encarga de $N+1$ incrementos y el otro parte de $N+2$ incrementos. La demostración la tenemos en la siguiente página, donde solo hacemos inserciones en m1-pedroamp y podemos ver que se incrementa solo los impares.

```
debiandm1-pedroamp:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 10.5.15-MariaDB-0-debian11-log Debian 11

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SELECT * FROM tiendabd.CLIENTE;
+----+-----+-----+-----+-----+-----+-----+-----+
| ID_CLIENTE | NOMBRE | TELEFONO | DIRECCION | COD_POSTAL | PASSWORD | ID_EMPLEADO_ALTA | ID_EMPLEADO_BAJA |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | cliente | 123456789 | calle etsiit | 18000 | 1234 | 1 | NULL |
| 2 | cliente2 | 112345678 | calle molotov | 18000 | 12345 | 1 | NULL |
| 3 | cliente3 | 234567890 | calle azor | 18000 | 4321 | 1 | NULL |
| 4 | cliente4 | 132316789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 10 | m2 | 132312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 11 | m1 | 132356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 12 | m2r | 232312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 13 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
0 rows in set (0.005 sec)

MariaDB [(none)]> INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m1r',232356789, 'calle rpg', 18000, '112345',1);
Query OK, 1 row affected (0.030 sec)

MariaDB [(none)]> INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m1r',232356789, 'calle rpg', 18000, '112345',1);
ERROR 1062 (23000): Duplicate entry '232356789' for key 'UN TELEFONO'
MariaDB [(none)]> INSERT INTO tiendabd.CLIENTE (NOMBRE, TELEFONO, DIRECCION, COD_POSTAL, PASSWORD, ID_EMPLEADO_ALTA) VALUES ('m1r',232356789, 'calle rpg', 18000, '112345',1);
Query OK, 1 row affected (0.022 sec)

MariaDB [(none)]> SELECT * FROM tiendabd.CLIENTE;
+----+-----+-----+-----+-----+-----+-----+-----+
| ID_CLIENTE | NOMBRE | TELEFONO | DIRECCION | COD_POSTAL | PASSWORD | ID_EMPLEADO_ALTA | ID_EMPLEADO_BAJA |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | cliente | 123456789 | calle etsiit | 18000 | 1234 | 1 | NULL |
| 2 | cliente2 | 112345678 | calle molotov | 18000 | 12345 | 1 | NULL |
| 3 | cliente3 | 234567890 | calle azor | 18000 | 4321 | 1 | NULL |
| 4 | cliente4 | 132316789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 10 | m2 | 132312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 11 | m1 | 132356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 12 | m2r | 232312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 13 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 15 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.001 sec)

MariaDB [(none)]>
```

```
debiandm2-pedroampcopy:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 12
Server version: 10.5.15-MariaDB-0-debian11-log Debian 11

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SELECT * FROM tiendabd.CLIENTE;
+----+-----+-----+-----+-----+-----+-----+-----+
| ID_CLIENTE | NOMBRE | TELEFONO | DIRECCION | COD_POSTAL | PASSWORD | ID_EMPLEADO_ALTA | ID_EMPLEADO_BAJA |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | cliente | 123456789 | calle etsiit | 18000 | 1234 | 1 | NULL |
| 2 | cliente2 | 112345678 | calle molotov | 18000 | 12345 | 1 | NULL |
| 3 | cliente3 | 234567890 | calle azor | 18000 | 4321 | 1 | NULL |
| 4 | cliente4 | 132316789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 10 | m2 | 132312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 11 | m1 | 132356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 12 | m2r | 232312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 13 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.005 sec)

MariaDB [(none)]> SELECT * FROM tiendabd.CLIENTE;
+----+-----+-----+-----+-----+-----+-----+-----+
| ID_CLIENTE | NOMBRE | TELEFONO | DIRECCION | COD_POSTAL | PASSWORD | ID_EMPLEADO_ALTA | ID_EMPLEADO_BAJA |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | cliente | 123456789 | calle etsiit | 18000 | 1234 | 1 | NULL |
| 2 | cliente2 | 112345678 | calle molotov | 18000 | 12345 | 1 | NULL |
| 3 | cliente3 | 234567890 | calle azor | 18000 | 4321 | 1 | NULL |
| 4 | cliente4 | 132316789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 10 | m2 | 132312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 11 | m1 | 132356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 12 | m2r | 232312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 13 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 15 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
9 rows in set (0.001 sec)

MariaDB [(none)]> SELECT * FROM tiendabd.CLIENTE;
+----+-----+-----+-----+-----+-----+-----+-----+
| ID_CLIENTE | NOMBRE | TELEFONO | DIRECCION | COD_POSTAL | PASSWORD | ID_EMPLEADO_ALTA | ID_EMPLEADO_BAJA |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1 | cliente | 123456789 | calle etsiit | 18000 | 1234 | 1 | NULL |
| 2 | cliente2 | 112345678 | calle molotov | 18000 | 12345 | 1 | NULL |
| 3 | cliente3 | 234567890 | calle azor | 18000 | 4321 | 1 | NULL |
| 4 | cliente4 | 132316789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 10 | m2 | 132312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 11 | m1 | 132356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 12 | m2r | 232312789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 13 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 15 | m1r | 232356789 | calle rpg | 18000 | 112345 | 1 | NULL |
| 19 | m1r | 23232789 | calle rpg | 18000 | 112345 | 1 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.001 sec)

MariaDB [(none)]>
```

Figura 26: Pruebas del Primary Primary. Inserciones solo en m1

5. GTID Global Transaction ID

Aquí explicaremos un poco el por qué no se utiliza la posición del Binlog ahora con la configuración Primaria Primaria y antes sí.

Es un tipo de modo de trabajo nuevo que ha aparecido en versiones de MariaDB 10 en adelante (No indica la versión específica). Su propósito es sustituir al modo antiguo de replicación. Con la mentalidad de permitir que los servidores sean Primarios y Réplica al mismo tiempo.

Resumiendo el proceso antiguo y sus limitaciones: En el servidor Primario al realizar actualizaciones en la base de datos con operaciones DML y DDL. Son escritas en el binlog suyo. Pero el réplica debe trazar y registrar esos cambios en su binlog (al ser primario también). Además de actualizar la posición del binlog en la que está para poder comprobar que ha habido cambios. Pero esto solo puede ser posible replicando solo a un Primario, es decir perdería esa información si cambia de Primario y replica otro Primario diferente aunque tenga la misma información (pudiéndose en mi opinión producir colisiones de información).

GTID cambia en que ya eso no es necesario, es decir apuntar al tope del binlog y registrar siempre el último cambio. Ahora cualquier operación se registra como un Event Group (evento de grupo), que es similar a como se registra una transacción en BD, o un snapshot, o un git commit, ... Cada event group tiene un nº único ID que lo identifica y las réplicas (Primarios también). Pueden comparar y obtener cada event group para estar al día.

En la documentación se enumeran dos beneficios de usar GTID:

1. Facilidad de cambiar de Primario por parte de la Réplica. **Ya que el servidor Réplica sabe que tiene que separar una instancia llamada Domain ID que es parte (GTID).** Cómo visualizarlo si tenemos dos Primarios con dos informaciones diferentes:
 - En uno le asignaríamos un Domain ID de 1, que indica que su Dominio es 1.
 - En otro le asignaríamos un Domain ID de 2, que indica que su Dominio es 2.
 - Ahora se puede obtener Réplica de ambos servidores primarios. Ya que en el estilo antiguo de Replicación, solo podemos indicarle al servidor Réplica. El nombre del fichero réplica del Primario y su offset (intervalo). Ahora no necesita conocer el offset y el fichero. Solo el GTID bajo un Domain ID y el nº de secuencia que le indica hasta donde debe replicar desde lo que tiene.
2. Si la Réplica cae, existe con el método antiguo la posibilidad de que se desincronice la posición del binlog y hay información que no es segura ante caídas. Como las tablas no transaccionales y los DDL. En cambio con GTID al ser una transacción en sí mismo, pues sí puede recuperarse la réplica.

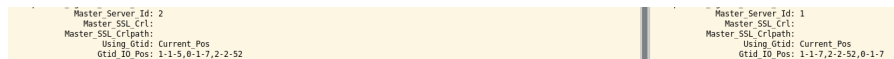


Figura 27: GTID de m1 y m2

El GTID Gtid_IO_Pos: **Nos centramos en este GTID:1-1-7,2-2-52,0-1-7** de ambos es similar:

1. En la configuración le hemos dicho que su Domain ID es 1. Que es el n unsigned int de la izquierda del todo.
2. El n del medio es el Server que es el identificador único del servidor en el Domain ID.
3. El n derecho es el Event Group ID que registra la posición de los Eventos, si hay cambios se empieza la réplica desde donde se tenga el último Event Group ID.

Aparentemente el Server ID debe ser único. Pero en mi caso aparece con el mismo ID en ambos servidores, es posible que al ser ambos Primary y Réplica a la vez no necesiten un Server ID distinto. Solamente seguir los Event Group ID. El Server ID en la mayoría de la documentación es distinto si hay una Réplica que no sea Primaria, es decir que tengamos 2 Primarios y 1 Réplica. La réplica entonces "forzaría.^a que se identifiquen de manera única para no repetir la información.

En un GTID aparece distinto Event Group ID Es debido a que ha ocurrido una serie de transacciones (2 de hecho) que son las que he indicado anteriormente, que en el m2 indica que tiene entradas duplicadas y estas no afectan en nada a m1 que está con la misma información. Entonces por eso va 2 pasos adelantados.

Es posible que me haya equivocado interpretando la información de la documentación. Esta es muy densa y tiene demasiadas variables que gestionar de hecho hay variables diferentes para master y slave. Que se configuran para motivos distintos que deben ser estudiados para saber que realizar correctamente. Hay muy poca información relacionada que explique de manera sencilla ya que se está implementando recientemente. Pero GTID parece ser el paso futuro a los sistemas de Réplica.



Figura 28: GTID - Diferencias con MySQL y MariaDB. No son compatibles.

5.1. Problemas con algunos errores de SQL

Tenemos que resolver errores de la réplica del segundo Primary que no ejecuta bien algunas sentencias SQL y debemos ignorarlas ya que son errores de duplicidad. Por lo que debemos ejecutar las siguientes sentencias hasta que se logre despejar el error y el SLAVE_SQL_RUNNING Funcione correctamente.

```
1 STOP SLAVE;  
2 SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;  
3 START SLAVE;  
4 #VERIFICAMOS QUE FUNCIONA YA BIEN  
5 SHOW SLAVE STATUS \G;
```

NOTA: Podemos visualizar que los errores de duplicación hacen que se pare la replicación. Los motivos son claros, tiene entradas duplicadas. Para ello ejecutamos los comandos indicados en la página anterior para que pueda continuar trabajando con normalidad. Ya que es una indicación en el caso de que esos datos que aparecen como duplicados, se recoja por parte del administrador de la base de datos para evitar la pérdida de estos.

```
MariaDB [(none)]> SHOW SLAVE STATUS \G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.122.70
Master_User: replication_user
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: m1-pedroamp-bin.000007
Read_Master_Log_Pos: 1339
Relay_Log_File: m2-pedroamp-relay-bin.000002
Relay_Log_Pos: 938
Relay_Master_Log_File: m1-pedroamp-bin.000003
Slave_IO_Running: Yes
Slave_SQL_Running: No
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 1062
Last_Error: Error 'Duplicate entry '20' for key 'PRIMARY'' on query. Default database: 'tiendabd'. Query: 'INSERT
INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XXXXL', 10)
Skip_Counter: 0
Exec_Master_Log_Pos: 1072
Relay_Log_Space: 5333
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: NULL
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 1062
Last_SQL_Error: Error 'Duplicate entry '20' for key 'PRIMARY'' on query. Default database: 'tiendabd'. Query: 'INSERT
INTO STOCK (NOMBRE, CANTIDAD) VALUES ('Patatas XXXXL', 10)'
```

Figura 29: Errores de un servidor primary.

Aquí podemos ver los servidores sin errores, este es la pantalla que debería mostrar para que la réplica funcione correctamente entre los dos Primarios.

```

MariaDB [tiendabd]> SHOW SLAVE STATUS \G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.122.119
Master_User: replication_user
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: m2-pedroamp-bin.000001
Read_Master_Log_Pos: 13792
Relay_Log_File: m1-pedroamp-relay-bin.000002
Relay_Log_Pos: 14097
Relay_Master_Log_File: m2-pedroamp-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
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: 13792
Relay_Log_Space: 14412
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:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 2
Master_SSL_Crl:
Master_SSL_Crlpath:
Using_Gtid: Current_Pos
Gtid_IO_Pos: 1-1-3,2-2-52,0-1-7
Replicate_Do_Domain_Ids:
Replicate_Ignore_Domain_Ids:
Parallel_Mode: optimistic
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
Slave_DDL_Groups: 0
Slave_Non_Transactional_Groups: 3
Slave_Transactional_Groups: 16
1 row in set (0.000 sec)

ERROR: No query specified
MariaDB [tiendabd]>

```

```

query OK, 0 rows affected, 1 warning (0.000 sec)
MariaDB [(none)]> SHOW SLAVE STATUS \G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.122.70
Master_User: replication_user
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: m1-pedroamp-bin.000007
Read_Master_Log_Pos: 1675
Relay_Log_File: m2-pedroamp-relay-bin.000006
Relay_Log_Pos: 1980
Relay_Master_Log_File: m1-pedroamp-bin.000007
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
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: 1675
Relay_Log_Space: 2692
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:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 1
Master_SSL_Crl:
Master_SSL_Crlpath:
Using_Gtid: Current_Pos
Gtid_IO_Pos: 1-1-5,2-2-51,0-1-7
Replicate_Do_Domain_Ids:
Replicate_Ignore_Domain_Ids:
Parallel_Mode: optimistic
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
Slave_DDL_Groups: 4
Slave_Non_Transactional_Groups: 2
Slave_Transactional_Groups: 11
1 row in set (0.000 sec)

ERROR: No query specified
MariaDB [(none)]>

```

Figura 30: Ambos servidores funcionan correctamente. A la izquierda es el m2-pedroamp y a la derecha m1-pedroamp

NOTA: Aquí podemos ver que los binlog que se referencian tanto uno como otro es el binlog usado como master. Por lo que no necesitan referenciarse a mano en el fichero `/etc/mysql/my.cnf`.

6. Iptables para SQL

En las iptables solo hemos añadido reglas de entrada y salida del mismo servidor. Es decir:

1. Hemos ampliado las variables de direcciones IP, en las cuales se especifica qué servidores tienen permitido hacer operaciones de Réplica de SQL y contactar por el puerto abierto de SQL. Reduciendo la exposición de la base de datos, a pesar de estar 'abierta'. Es decir el puerto 3306 antes de iptables podría ser conocido por cualquier host. Ahora ya solo se pueden leer en localhost y entre los dos servidores.
2. Necesitamos una salida para el servidor Réplica (siendo Primario o Réplica) hacia el puerto de MariaDB (MySQL) del otro servidor Primario. Debido a que este debe contactar con el otro Primario para poder obtener el binlog.
3. Necesitamos permitir la entrada del otro servidor Réplica (que es Primario) su puerto de MariaDB (MySQL) Primario. Ya que necesita recibir las peticiones de obtener el binlog para hacer la réplica.

Las reglas parecen redundantes, pero no es así ya que tenemos que especificar qué paquetes permitimos que salgan y entren.

Todo paquete que salga de un servidor Réplica con destino al servidor Primario (a replicar). Que tenga como destino el puerto TCP/3306 del servidor primario, está permitido. Todo paquete entrante desde el servidor Réplica que tenga como destino el puerto TCP/3306 del propio servidor primario está permitido.

6.1. ¿Cómo he realizado el análisis de puertos TCP para saber cómo se conectan?

Para el análisis de puertos y cómo se conecta he utilizado el siguiente comando:

```
1 sudo netstat -tqn
```

Donde cada parámetro implica

1. **t** TCP connections
2. **p** Muestra el PID del programa que realiza o recibe conexiones en los puertos indicados.
3. **n** Utiliza las direcciones IP en vez de hostnames.

Con ese comando podemos ver las conexiones que se realizan con MariaDB. Los clientes al leer el binlog para ser Réplica se configuran con puertos efímeros y tienen como destino el puerto 3306. A la inversa funciona igual, el servidor replicado primero se consulta al segundo servidor pasando a ser cliente.

<pre> debian@ml-pedroamp:~\$ sudo netstat -tln Active Internet connections (w/o servers) Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name tcp 0 0 192.168.122.70:3306 192.168.122.119:34960 ESTABLISHED 643/mariadb tcp 0 0 192.168.122.70:22 192.168.122.1:42776 ESTABLISHED 693/sshd: debian [p tcp 0 0 192.168.122.70:52012 192.168.122.119:3306 ESTABLISHED 643/mariadb </pre>	<pre> debian@ml-pedroampcopy:~\$ sudo netstat -tln Active Internet connections (w/o servers) Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name tcp 0 0 192.168.122.119:3306 192.168.122.70:52012 ESTABLISHED 606/mariadb tcp 0 0 192.168.122.119:22 192.168.122.1:55812 ESTABLISHED 654/sshd: debian [p tcp 0 0 192.168.122.119:34960 192.168.122.70:3306 ESTABLISHED 606/mariadb </pre>
--	---

Figura 31: Conexiones que se realizan.

Si bloqueamos por ejemplo la salida de un Servidor, lo que ocurre es lo siguiente.

<pre> debian@ml-pedroamp:~\$ sudo netstat -tln Active Internet connections (w/o servers) Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name tcp 0 0 192.168.122.70:3306 192.168.122.119:34960 ESTABLISHED 643/mariadb tcp 0 0 192.168.122.70:52012 192.168.122.119:3306 ESTABLISHED 643/mariadb tcp 0 0 192.168.122.70:22 192.168.122.1:42776 ESTABLISHED 693/sshd: debian [p debian@ml-pedroamp:~\$ sudo mysql -u root -p -e "SHOW SLAVE STATUS \G" head -n 14 Enter password: ***** 1. row ***** Slave_IO_State: Waiting for master to send event Master_Host: 192.168.122.119 Master_User: replication user Master_Port: 3306 Connect_Retry: 10 Master_Log_File: m2-pedroamp-bin.000009 Read_Master_Log_Pos: 348 Relay_Log_File: m1-pedroamp-relay-bin.000003 Relay_Log_Pos: 653 Relay_Master_Log_File: m2-pedroamp-bin.000009 Slave_IO_Running: Yes Slave_SQL_Running: Yes Replicate_Do_DB: </pre>	<pre> debian@ml-pedroampcopy:~\$ sudo tail -n 10 iptables.sh iptables -A OUTPUT -p tcp -d 199.232.182.132 --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT iptables -A INPUT -p tcp -s 199.232.182.132 --sport 80 -m state --state ESTABLISHED -j ACCEPT # Regla de DNS iptables -A OUTPUT -p udp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT iptables -A INPUT -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT # Salida para contactar con el otro Primario Esclavo iptables -A INPUT -p tcp -s \$(ml) --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT iptables -A OUTPUT -p tcp -d \$(ml) --sport 3306 -m state --state ESTABLISHED -j ACCEPT debian@ml-pedroampcopy:~\$ sudo mysql -u root -p -e "SHOW SLAVE STATUS \G" head -n 14 Enter password: ***** 1. row ***** Slave_IO_State: Connecting to master Master_Host: 192.168.122.70 Master_User: replication user Master_Port: 3306 Connect_Retry: 10 Master_Log_File: m1-pedroamp-bin.000017 Read_Master_Log_Pos: 364 Relay_Log_File: m2-pedroamp-relay-bin.000001 Relay_Log_Pos: 4 Relay_Master_Log_File: m1-pedroamp-bin.000017 Slave_IO_Running: Connecting Slave_SQL_Running: Yes Replicate_Do_DB: </pre>
--	--

Figura 32: El servidor m2 que tiene por defecto firewall de whitelist sin permitir la salida para consultar no sabe nada del otro master. Sin embargo el otro servidor si sabe que ocurre en el otro M2 y puede actualizar la información de M2. Sin embargo el M2 se irá quedando atrás al no poder actualizar.

6.2. m1-pedroamp

El fichero entero de iptables aplicado a m1 es:

```
1  #!/bin/bash
2
3  # VARIABLES for IP
4  m2="192.168.122.119"
5  m3="192.168.122.57"
6
7  # Flushing iptables
8  iptables --flush
9
10 # White Listing Firewall
11 iptables -P INPUT DROP
12 iptables -P OUTPUT DROP
13 iptables -P FORWARD DROP
14
15 # Permitir conexiones en localhost
16 iptables -A INPUT -i lo -j ACCEPT
17 iptables -A OUTPUT -o lo -j ACCEPT
18
19 # SSH rules
20 # La cadena de INPUT siguiendo el TCP Handshake se establece un registro en
    la tabla de estados de IPTABLES (cortafuegos con estado STATEFUL)
21 # en la que toda la conexion nueva *NEW debe ser registrada en dicha tabla
    y pasa a ser como conexion
22 # establecida. Dicho registro en el estado ESTABLISHED permite la siguiente
    cadena en la cual sale una conexion de
23 # SSH permitida a traves del puerto 22.
24
25 iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j
    ACCEPT
26 iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
27
28 # ICMP echo PING RULES ALLOW
29 iptables -A INPUT -p icmp --icmp-type echo-request -m state --state NEW,
    ESTABLISHED,RELATED -j ACCEPT
30 iptables -A OUTPUT -p icmp --icmp-type echo-reply -m state --state
    ESTABLISHED,RELATED -j ACCEPT
31
32 # Regla de contacto con desde el balanceador a sus servidores
33 # Solo se permite el contacto del balanceador para el puerto HTTPs 443
34 iptables -A INPUT -p tcp -s ${m3} --dport 443 -m state --state NEW,
    ESTABLISHED -j ACCEPT
35 iptables -A OUTPUT -p tcp -d ${m3} --sport 443 -m state --state ESTABLISHED
    -j ACCEPT
36
37 # Regla para permitir el uso de un repositorio de APT
38 iptables -A OUTPUT -p tcp -d 199.232.182.132 --dport 80 -m state --state
    NEW,ESTABLISHED -j ACCEPT
39 iptables -A INPUT -p tcp -s 199.232.182.132 --sport 80 -m state --state
    ESTABLISHED -j ACCEPT
40
```

```

41 # Regla de DNS
42 iptables -A OUTPUT -p udp --dport 53 -m state --state NEW,ESTABLISHED -j
    ACCEPT
43 iptables -A INPUT -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
44
45 # Regla de MariaDB
46 # Entrada del contacto con el otro Primario Esclavo
47 iptables -A OUTPUT -p tcp -d ${m2} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
48 iptables -A INPUT -p tcp -s ${m2} --sport 3306 -m state --state ESTABLISHED
    -j ACCEPT
49
50 # Salida para contactar con el otro Primario Esclavo
51 iptables -A INPUT -p tcp -s ${m2} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
52 iptables -A OUTPUT -p tcp -d ${m2} --sport 3306 -m state --state
    ESTABLISHED -j ACCEPT

```

De las cuales las líneas más importantes son:

```

1 # Regla de MariaDB
2 # Entrada del contacto con el otro Primario Esclavo
3 iptables -A OUTPUT -p tcp -d ${m2} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
4 iptables -A INPUT -p tcp -s ${m2} --sport 3306 -m state --state ESTABLISHED
    -j ACCEPT
5
6 # Salida para contactar con el otro Primario Esclavo
7 iptables -A INPUT -p tcp -s ${m2} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
8 iptables -A OUTPUT -p tcp -d ${m2} --sport 3306 -m state --state
    ESTABLISHED -j ACCEPT

```

Tal y como hemos dicho en la subsección anterior, es importante permitir tanto la entrada como la salida.

6.3. m2-pedroamp

El fichero entero de iptables aplicado a m2 es:

```
1  #!/bin/bash
2
3  # VARIABLES for IP
4  m1="192.168.122.70"
5  m3="192.168.122.57"
6
7  # Flushing iptables
8  iptables --flush
9
10 # White Listing Firewall
11 iptables -P INPUT DROP
12 iptables -P OUTPUT DROP
13 iptables -P FORWARD DROP
14
15 # Permitir conexiones en localhost
16 iptables -A INPUT -i lo -j ACCEPT
17 iptables -A OUTPUT -o lo -j ACCEPT
18
19 # SSH rules
20 # La cadena de INPUT siguiendo el TCP Handshake se establece un registro en
    la tabla de estados de IPTABLES (cortafuegos con estado STATEFUL)
21 # en la que toda la conexion nueva *NEW debe ser registrada en dicha tabla
    y pasa a ser como conexion
22 # establecida. Dicho registro en el estado ESTABLISHED permite la siguiente
    cadena en la cual sale una conexion de
23 # SSH permitida a traves del puerto 22.
24
25 iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j
    ACCEPT
26 iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
27
28 # ICMP echo PING RULES ALLOW
29 iptables -A INPUT -p icmp --icmp-type echo-request -m state --state NEW,
    ESTABLISHED,RELATED -j ACCEPT
30 iptables -A OUTPUT -p icmp --icmp-type echo-reply -m state --state
    ESTABLISHED,RELATED -j ACCEPT
31
32 # Regla de contacto con desde el balanceador a sus servidores
33 # Solo se permite el contacto del balanceador para el puerto HTTPs 443
34 iptables -A INPUT -p tcp -s ${m3} --dport 443 -m state --state NEW,
    ESTABLISHED -j ACCEPT
35 iptables -A OUTPUT -p tcp -d ${m3} --sport 443 -m state --state ESTABLISHED
    -j ACCEPT
36
37 # Regla para permitir el uso de un repositorio de APT
38 iptables -A OUTPUT -p tcp -d 199.232.182.132 --dport 80 -m state --state
    NEW,ESTABLISHED -j ACCEPT
39 iptables -A INPUT -p tcp -s 199.232.182.132 --sport 80 -m state --state
    ESTABLISHED -j ACCEPT
40
```

```

41 # Regla de DNS
42 iptables -A OUTPUT -p udp --dport 53 -m state --state NEW,ESTABLISHED -j
    ACCEPT
43 iptables -A INPUT -p udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
44
45 # Regla de MariaDB
46 # Entrada del contacto con el otro Primario Esclavo
47 iptables -A OUTPUT -p tcp -d ${m1} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
48 iptables -A INPUT -p tcp -s ${m1} --sport 3306 -m state --state ESTABLISHED
    -j ACCEPT
49
50 # Salida para contactar con el otro Primario Esclavo
51 iptables -A INPUT -p tcp -s ${m1} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
52 iptables -A OUTPUT -p tcp -d ${m1} --sport 3306 -m state --state
    ESTABLISHED -j ACCEPT

```

De las cuales las líneas más importantes son:

```

1 # Regla de MariaDB
2 # Entrada del contacto con el otro Primario Esclavo
3 iptables -A OUTPUT -p tcp -d ${m1} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
4 iptables -A INPUT -p tcp -s ${m1} --sport 3306 -m state --state ESTABLISHED
    -j ACCEPT
5
6 # Salida para contactar con el otro Primario Esclavo
7 iptables -A INPUT -p tcp -s ${m1} --dport 3306 -m state --state NEW,
    ESTABLISHED -j ACCEPT
8 iptables -A OUTPUT -p tcp -d ${m1} --sport 3306 -m state --state
    ESTABLISHED -j ACCEPT

```

Tal y como hemos dicho en la subsección anterior, es importante permitir tanto la entrada como la salida.

```
debian@m1-pedroamp:~$ tail -n 7 iptables.sh
# Entrada del contacto con el otro Primario Esclavo
iptables -A OUTPUT -p tcp -d $(m2) --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -s $(m2) --sport 3306 -m state --state ESTABLISHED -j ACCEPT

# Salida para contactar con el otro Primario Esclavo
iptables -A INPUT -p tcp -s $(m2) --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -d $(m2) --sport 3306 -m state --state ESTABLISHED -j ACCEPT
debian@m1-pedroamp:~$ sudo mysql -u root -p -e "SHOW SLAVE STATUS \G" | head -n 14
Enter password:
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.122.119
Master_User: replication_user
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: m2-pedroamp-bin.000011
Read_Master_Log_Pos: 348
Relay_Log_File: m1-pedroamp-relay-bin.000002
Relay_Log_Pos: 653
Relay_Master_Log_File: m2-pedroamp-bin.000011
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
debian@m1-pedroamp:~$ sudo iptables -L | grep sql
ACCEPT tcp -- m2-pedroampcopy anywhere tcp spt:mysql state ESTABLISHED
ACCEPT tcp -- m2-pedroampcopy anywhere tcp dpt:mysql state NEW,ESTABLISHED
ACCEPT tcp -- anywhere m2-pedroampcopy tcp dpt:mysql state NEW,ESTABLISHED
ACCEPT tcp -- anywhere m2-pedroampcopy tcp spt:mysql state ESTABLISHED
debian@m1-pedroamp:~$ sudo netstat -ttn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0 192.168.122.70:52024 192.168.122.119:3306 ESTABLISHED 851/mariadb
tcp 0 0 0 192.168.122.70:22 192.168.122.1:42776 ESTABLISHED 693/sshd: debian [p
tcp 0 0 0 192.168.122.70:3306 192.168.122.119:34976 ESTABLISHED 851/mariadb
debian@m1-pedroamp:~$

debian@m2-pedroampcopy:~$ tail -n 7 iptables.sh
# Entrada del contacto con el otro Primario Esclavo
iptables -A OUTPUT -p tcp -d $(m1) --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -s $(m1) --sport 3306 -m state --state ESTABLISHED -j ACCEPT

# Salida para contactar con el otro Primario Esclavo
iptables -A INPUT -p tcp -s $(m1) --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -d $(m1) --sport 3306 -m state --state ESTABLISHED -j ACCEPT
debian@m2-pedroampcopy:~$ sudo mysql -u root -p -e "SHOW SLAVE STATUS \G" | head -n 14
Enter password:
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.122.70
Master_User: replication_user
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: m1-pedroamp-bin.000018
Read_Master_Log_Pos: 364
Relay_Log_File: m2-pedroamp-relay-bin.000003
Relay_Log_Pos: 669
Relay_Master_Log_File: m1-pedroamp-bin.000018
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
debian@m2-pedroampcopy:~$ sudo iptables -L | grep sql
ACCEPT tcp -- m1-pedroamp anywhere tcp spt:mysql state ESTABLISHED
ACCEPT tcp -- m1-pedroamp anywhere tcp dpt:mysql state NEW,ESTABLISHED
ACCEPT tcp -- anywhere m1-pedroamp tcp dpt:mysql state NEW,ESTABLISHED
ACCEPT tcp -- anywhere m1-pedroamp tcp spt:mysql state ESTABLISHED
debian@m2-pedroampcopy:~$ sudo netstat -ttn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0 192.168.122.119:34976 192.168.122.70:3306 ESTABLISHED 1140/mariadb
tcp 0 0 0 192.168.122.119:22 192.168.122.1:55812 ESTABLISHED 654/sshd: debian [p
tcp 0 0 0 192.168.122.119:3306 192.168.122.70:52024 ESTABLISHED 1140/mariadb
debian@m2-pedroampcopy:~$
```

Figura 33: Funcionamiento de la BDs con las reglas de iptables.

7. Bibliografía

- ¹ https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_logical_backup
- ² https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_physical_backup
- ³ <https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html>
- ⁴ <http://dirk-loss.de/ssh-port-forwarding.htm>
- ⁵ <https://dev.mysql.com/doc/refman/8.0/en/windows-and-ssh.html>
- ⁶ https://mariadb.com/kb/en/server-system-variables/#bind_address
- ⁷ <https://dev.mysql.com/doc/refman/5.7/en/binary-log-mixed.html>
- ⁸ <https://www.databasejournal.com/mysql/comparing-mysql-statement-based-and-row-based-replication/>
- ⁹ <https://mariadb.com/kb/en/binary-log-formats/>
- ¹⁰ <https://mariadb.com/kb/en/setting-up-replication/>
- ¹¹ <https://dev.mysql.com/doc/refman/5.7/en/group-replication-primary-secondary-replication.html>
- ¹² <https://fromdual.com/mariadb-master-master-gtid-based-replication-with-keepalived-vip>
- ¹³ <https://serverfault.com/questions/872911/slave-sql-running-no-mysql-replication-stopped-working>
- ¹⁴ <https://mariadb.com/kb/en/gtid/>

¹Copia lógica.

²Copia física.

³Referencia de mysqldump y todos sus parámetros.

⁴Referencia de SSH Port-Forward y el manpage también debe ser consultado.

⁵Referencia de SSH Port-Forward con el MySQL.

⁶bind-address MySQL.

⁷binlog mixed mode MySQL.

⁸Diferencias entre Row-based y Statement-Based.

⁹Formatos de Binlog.

¹⁰Replicación Primaria y Réplica.

¹¹Replicación Primaria y Réplica. Agrupada

¹²Configuración primary primary. Con GTID

¹³Solucionar errores de colisión con algunas sentencias en un servidor primary que para la réplica

¹⁴Solucionar errores de colisión con algunas sentencias en un servidor primary que para la réplica