SWAP: Replicación de bases de datos MySQL

David Cabezas Berrido

dxabezas@correo.ugr.es

14 de mayo de 2021

Índice

1. Preparativos

Es importante desactivar el cortafuegos antes de hacer la configuración de maestro-esclavo. Como medida preventiva, desactivamos el cortafuegos en todas las máquinas ejecutando el script `off.sh` de la práctica anterior (para *iptables*) y también `sudo ufw disable` (para UFW).

Como hicimos las reglas de *iptables* permanentes, también ejecutamos (desde root)

```
# iptables-save > /etc/iptables/rules.v4
```

2. Base de datos MySQL

Creamos la base de datos, la tabla e insertamos una tupla tal y como se describe en el guión.

Como opción avanzada, podemos exigir que algún campo no pueda ser nulo, por ejemplo el campo usuario. Esto se consigue añadiendo `NOT NULL` detás del tipo del campo. Además, a la hora de insertar una tupla podemos omitir el nombre de los atributos si ponemos los valores en el mismo orden que aparecen en la descripción (el orden que usamos cuando creamos la tabla), aunque esto no es muy recomendable (si se modificase la tabla, habría que cambiar los scripts de inserción en caso de tenerlos).

```
mysql> select * from datos;
+--------+-----------------+----------+-----------------------+
| nombre | apellidos       | usuario  | email                 |
+--------+-----------------+----------+-----------------------+
| David  | Cabezas Berrido | dxabezas | dxabezas@correo.ugr.es |
+--------+-----------------+----------+-----------------------+
1 row in set (0.00 sec)

mysql> describe datos;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| nombre    | varchar(100) | YES  |     | NULL    |       |
| apellidos | varchar(100) | YES  |     | NULL    |       |
| usuario   | varchar(100) | NO   |     | NULL    |       |
| email     | varchar(100) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> exit;
Bye
```

Figura 1: Tenemos una tupla en la tabla que hemos creado. En la descripción de la tabla podemos ver los distintos campos y el tipo de cada uno. Nos fijamos que el campo usuario no puede ser nulo (obtendremos un error si intentamos introducir una tupla con valor nulo para este atributo).

2.1. Replicar la BD con mysqldump

Antes de realizar la copia, debemos bloquear las tablas.

```
dxabezas@m1-dxabezas:~$ sudo mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.33-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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

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

mysql> flush tables with read lock;
Query OK, 0 rows affected (0.00 sec)

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

Database changed
mysql> insert into datos values("Pepe","Lopez","pepelopez","pepelopez@correo.ugr.es");
ERROR 1223 (HY000): Can't execute the query because you have a conflicting read lock
```

Figura 2: Bloqueamos las tablas y comprobamos que no se puede modificar la base de datos.

Ahora desde bash copiamos la base de datos a un archivo con

`sudo mysqldump estudiante -u root > /tmp/estudiante.sql`

Después volvemos a entrar a MySQL y desbloqueamos las tablas con `unlock tables;`.

Copiamos el archivo a la máquina 2 (IP 192.168.56.101) con:

`sudo scp /tmp/estudiante.sql dxabezas@192.168.56.101:/tmp/estudiante.sql`

A continuación, desde M2 creamos la base de datos al igual que hemos hecho en M1 (como se describe en el guión) y recuperamos la copia con

`sudo mysql -u root estudiante < /tmp/estudiante.sql`

Si entramos a MySQL, podemos obtener el mismo resultado que en la Figura 1 (ahora en M2).

En el manual encontramos algunas opciones avanzadas que pueden ser útiles. Si tenemos varias bases de datos, podemos especificar (tanto al copiar como restaurar) las que queremos copiar con la opción `--databases <nombre-db1><nombre-db2>...` o usar `--all-databases` para copiarlas todas. También podemos ahorrarnos tener que bloquear las tablas añadiendo la opción `--lock-all-tables` en el `mysqldump` (para bloquear todas las tablas de todas las bases de datos) o simplemente `--lock-tables` para bloquear sólo las tablas que copiamos. También podemos omitir las copias de algunas tablas con `--ignore-table=estudiante.datos` (en general, `<nombre db>.<nombre tabla>`) y omitir el contenido (las tuplas) de las tablas y copiar sólo la estructura con `--no-data`.

3. Replicar la BD mediante configuración maestro-esclavo

Configuración de MySQL del maestro (M1)

Editamos como root el archivo `/etc/mysql/mysql.conf.d/mysqld.cnf` como se indica en el guión:

- Comentamos `#bind-address 127.0.0.1`.
- `log_error = /var/log/mysql/error.log` (donde se almacena el log de errores).
- `server-id = 1`.
- `log_bin = /var/log/mysql/bin.log` (donde se almacenan los binarios con la información).

Reiniciamos el servicio, todo parece estar correcto.

```
dxabezas@m1-dxabezas:~$ sudo /etc/init.d/mysql restart
[ ok ] Restarting mysql (via systemctl): mysql.service.
dxabezas@m1-dxabezas:~$ sudo service mysql status
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-05-12 07:46:55 UTC; 12s ago
  Process: 4342 ExecStart=/usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid (code=exited, status=0/SUCCESS)
  Process: 4321 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
 Main PID: 4344 (mysqld)
    Tasks: 27 (limit: 1106)
   CGroup: /system.slice/mysql.service
           └─4344 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid

May 12 07:46:54 m1-dxabezas systemd[1]: Stopped MySQL Community Server.
May 12 07:46:54 m1-dxabezas systemd[1]: Starting MySQL Community Server...
May 12 07:46:55 m1-dxabezas systemd[1]: Started MySQL Community Server.
```

Configuración de MySQL del esclavo (M2)

Hacemos las mismas configuraciones, pero con `server-id = 2`. Reiniciamos y comprobamos que también parece correcto.

```
dxabezas@m2-dxabezas:~$ sudo /etc/init.d/mysql restart
[ ok ] Restarting mysql (via systemctl): mysql.service.
dxabezas@m2-dxabezas:~$ sudo service mysql status
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-05-12 07:50:10 UTC; 10s ago
  Process: 2080 ExecStart=/usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid (code=exited, status=0/SUCCESS)
  Process: 2058 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
 Main PID: 2092 (mysqld)
    Tasks: 27 (limit: 1106)
   CGroup: /system.slice/mysql.service
           └─2092 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid

May 12 07:50:08 m2-dxabezas systemd[1]: Stopped MySQL Community Server.
May 12 07:50:08 m2-dxabezas systemd[1]: Starting MySQL Community Server...
May 12 07:50:10 m2-dxabezas systemd[1]: Started MySQL Community Server.
```

Configuración de MySQL del maestro (M1)

Volvemos a M1, entramos a MySQL y creamos un usuario esclavo con los permisos necesarios para realizar la replicación y bloqueamos las tablas.

```
dxabezas@m1-dxabezas:~$ sudo mysql -u root

mysql> create user esclavo_dxabezas identified by 'esclavo_dxabezas';
mysql> grant replication slave on *.* to 'esclavo_dxabezas'@'%' identified by 'esclavo_dxabezas';
mysql> flush privileges;
mysql> flush tables;
mysql> flush tables with read lock;
```

A continuación, mostramos el estado del maestro.

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000002 |      998 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
```

Estos datos son relativos a los ficheros de `log_bin = /var/log/mysql/bin.log`, y cambian cuando se realizan modificaciones. Por tanto, NO debemos tocar la base de datos hasta que no completemos la configuración.

Configuración de MySQL del esclavo (M2)

Entramos a MySQL en M2, le damos los datos del maestro y arrancamos el esclavo.

```
dxabezas@m2-dxabezas:~$ sudo mysql -u root

mysql> change master to master_host='192.168.56.102',
       master_user='esclavo_dxabezas', master_password='esclavo_dxabezas',
       master_log_file='mysql-bin.000002', master_log_pos=998, master_port=3306;

mysql> start slave;
```

Configuración de MySQL del maestro (M1)

Volvemos una vez más a M1 para activar las tablas con

```
dxabezas@m1-dxabezas:~$ sudo mysql -u root

mysql> unlock tables;
```

Configuración de MySQL del esclavo (M2)

Comprobamos el estado del esclavo.

```
dxabezas@m2-dxabezas:~$ sudo mysql -u root

mysql> show slave status\G;
```

Observamos que aparece `Seconds_Behind_Master: 0` y todos los marcadores de errores estan a 0 o son vacíos (`Last_SQL_Error, Last_IO_Error, Last_Error,...`). Así que nos disponemos a probar la configuración introduciendo nuevos datos en el maestro para ver si el esclavo los replica.

En M1, introducimos una nueva tupla.

```
mysql> use estudiante;
mysql> insert into datos values("Pepe","Lopez","pepelopez","pepelopez@correo.ugr.es");
```

Y comprobamos que se ha añadido también en M2.

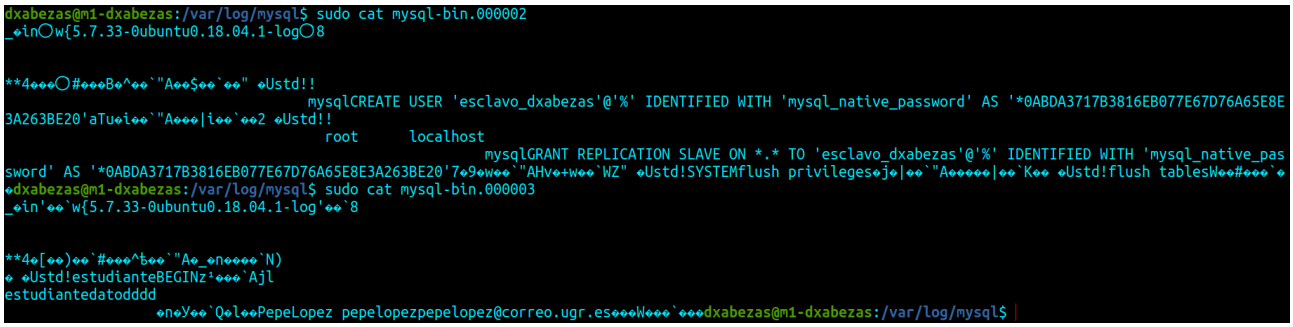
```
mysql> use estudiante;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from datos;
+--------+-----------------+-----------+--------------------------+
| nombre | apellidos       | usuario   | email                    |
+--------+-----------------+-----------+--------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es   |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es  |
+--------+-----------------+-----------+--------------------------+
2 rows in set (0.00 sec)
```

Figura 3: Base de datos de M2 tras añadir la tupla en M1.

Si nos fijamos, el estado del maestro ha cambiado.

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000003 |      474 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

Aunque los ficheros sean binarios, observamos secuencias relacionadas con las modificaciones que acabamos de hacer.



4. Replicar la BD mediante configuración maestro-maestro

Los servicios ya están configurados, y hay que repetir la creación del usuario esclavo, esta vez en M2, e indicárselo a M1.

En M2, repetimos lo mismo que hicimos en su momento con M1:

```
dxabezas@m2-dxabezas:~$ sudo mysql -u root

mysql> create user esclavo_dxabezas identified by 'esclavo_dxabezas';
mysql> grant replication slave on *.* to 'esclavo_dxabezas'@'%' identified by 'esclavo_dxabezas';
mysql> flush privileges;
mysql> flush tables;
mysql> flush tables with read lock;
```

También mostramos el estado del maestro.

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000004 |      998 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

En M1 indicamos los datos del maestro y lanzamos el esclavo.

```
dxabezas@m1-dxabezas:~$ sudo mysql -u root

mysql> change master to master_host='192.168.56.101',
master_user='esclavo_dxabezas', master_password='esclavo_dxabezas',
master_log_file='mysql-bin.000004', master_log_pos=998, master_port=3306;

mysql> start slave;
```

Ya podemos desbloquear las tablas en M2.

```
dxabezas@m2-dxabezas:~$ sudo mysql -u root

mysql> unlock tables;
```

Cuando hacemos `show slave status\G;` en M1, nos percatamos de que algún error está ocurriendo. M1 no está siendo capaz de conectarse a M2.

```
Seconds_Behind_Master: NULL
SL_Verify_Server_Cert: No
       Last_IO_Errno: 2003
       Last_IO_Error: error connecting to master 'esclavo_dxabezas@192.168.56.101:3306' - retry-time: 60  retries: 2
```

```
 Slave_IO_Running: Connecting
Slave_SQL_Running: Yes
```

Parece que el error se debía a que el cortafuegos había vuelto a activarse por algún motivo (estuve tocándolo para para la siguiente sección). Sólo con desactivarlo y volver a pedir el estado, todo parece en orden.

```
          Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
```

```
 Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

Ya estamos en condiciones de probarlo. Observamos que M1 replica las actualizaciones que realiza M2. La izquierda es M1 y la derecha M2).

```
mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | AA              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)

mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | BB              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)

mysql>
mysql>
mysql>
```

```
mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | AA              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)

mysql> update datos set apellidos='BB' where usuario='aaaa';
Query OK, 1 row affected (0.37 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | BB              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)
```

De igual forma, M2 replica las modificaciones de M1, como ya hacía previamente.

```
mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | BB              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)

mysql> update datos set apellidos='CC' where usuario='aaaa';
Query OK, 1 row affected (1.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | CC              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)
```

```
mysql> mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | BB              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)

mysql> select * from datos;
+--------+-----------------+-----------+-------------------------+
| nombre | apellidos       | usuario   | email                   |
+--------+-----------------+-----------+-------------------------+
| David  | Cabezas Berrido | dxabezas  | dxabezas@correo.ugr.es  |
| Pepe   | Lopez           | pepelopez | pepelopez@correo.ugr.es |
| A      | CC              | aaaa      | a@correo.ugr.es         |
+--------+-----------------+-----------+-------------------------+
3 rows in set (0.00 sec)

mysql>
mysql>
mysql>
```

5. Configurar *iptables* para el puerto 3306

Es deseable que la configuración funcione sin la necesidad de cortar el cortafuegos. Por tanto, recuperamos las reglas de *iptables* que configuramos en la práctica anterior (deniegan todo el tráfico salvo los servicios SHH, HTTP y HTTPS), con la excepción de que debemos abrir también el puerto 3306 para permitir el tráfico de MySQL. Además, aprovechamos para limitar estas conexiones sólo entre M1 y M2.

Todo esto lo conseguimos añadiendo las siguientes reglas en el script de *iptables*:

Para M1 sólo aceptamos tráfico con M2 (192.168.56.101):

```
iptables -A INPUT -p tcp -s 192.168.56.101 --dport 3306 -j ACCEPT
iptables -A OUTPUT -p tcp -d 192.168.56.101 --sport 3306 -j ACCEPT
```

Para M2 sólo aceptamos tráfico con M1 (192.168.56.102):

```
iptables -A INPUT -p tcp -s 192.168.56.102 --dport 3306 -j ACCEPT
iptables -A OUTPUT -p tcp -d 192.168.56.102 --sport 3306 -j ACCEPT
```

Hemos encontrado un problema implementando esto y hemos descubierto que es importante poner la **IP antes que el puerto** para que funcione correctamente.

Con esta configuración del cortafuegos, la replicación sigue funcionando correctamente.