

Práctica 5. Replicación de bases de datos MySQL

Duración: 2 sesiones

1. Objetivos de la práctica

A la hora de hacer copias de seguridad de nuestras bases de datos (BD) MySQL, una opción muy común suele ser la de usar una réplica maestro-esclavo, de manera que nuestro servidor en producción hace de maestro y otro servidor de backup hace de esclavo.

Podemos hacer copias desde el servidor de backup sin que se vea afectado el rendimiento del sistema en producción y sin interrupciones de servicio.

Tener una réplica en otro servidor también añade fiabilidad ante fallos totales del sistema en producción, los cuales, tarde o temprano, ocurrirán. Por ejemplo, podemos tener un pequeño servidor actuando como backup en nuestra oficina sincronizado mediante réplicas con nuestro sistema en producción.

Esta opción, además, añade fiabilidad ante posibles interrupciones de servicio permanentes del servidor maestro por cualquier escenario catastrófico que nos podamos imaginar. En ese caso, tendremos posiblemente decenas de clientes y servicios parados sin posibilidad de recuperar sus datos si no hemos preparado un buen plan de contingencias. Tener un servidor de backup con MySQL actuando como esclavo de replicación es una solución asequible y no consume demasiado ancho de banda en un sitio web de tráfico normal, además de que no afecta al rendimiento del maestro en el sistema en producción.

Los objetivos concretos de esta práctica son:

- Copiar archivos de copia de seguridad mediante ssh.
- Clonar manualmente BD entre máquinas.
- Configurar la estructura maestro-esclavo entre dos máquinas para realizar el clonado automático de la información.

2. Crear un tar con ficheros locales y copiarlos en un equipo remoto

Ya vimos en la práctica 2 cómo crear un tar.gz con un directorio de un equipo y dejarlo en otro mediante ssh.

Para ello, vimos que deberemos indicar al comando tar que queremos que use stdout como destino y mandar con una pipe la salida al ssh. Éste debe coger la salida del tar y escribirla en un fichero. El comando quedaría:

```
tar czf - directorio | ssh equipodestino 'cat > ~/tar.tgz'
```

De esta forma en el equipo de destino tendremos creado el archivo tar.tgz

También vimos cómo ejecutar de una forma similar el comando scp. Sin embargo, esto que puede ser útil en un momento dado, no nos servirá para sincronizar grandes cantidades de información. En el caso de BD, hay mejores formas de hacerlo.

3. Crear una BD e insertar datos

Para el resto de la práctica debemos crearnos una BD en MySQL e insertar algunos datos. Así tendremos datos con los cuales hacer las copias de seguridad. En todo momento usaremos la interfaz de línea de comandos del MySQL:

```
mysql -uroot -p
Enter password: <TECLEAR LA CLAVE SI NO ES VACÍA>
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.44 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

mysql> create database contactos;
Query OK, 1 row affected (0,00 sec)

mysql> use contactos;
Database changed

mysql> show tables;
Empty set (0,00 sec)

mysql> create table datos(nombre varchar(100),tlf int);
Query OK, 0 rows affected (0,01 sec)

mysql> show tables;
+-----+
| Tables_in_contactos |
+-----+
| datos                |
+-----+
1 row in set (0,00 sec)

mysql> insert into datos(nombre,tlf) values ("pepe",95834987);
Query OK, 1 row affected (0,00 sec)

mysql> select * from datos;
+-----+-----+
| nombre | tlf      |
+-----+-----+
| pepe   | 95834987 |
+-----+-----+
3 rows in set (0,00 sec)
```

Ya tenemos datos (un registro) insertados en nuestra BD llamada “datos”. Podemos haber insertado más registros. Veamos cómo entrar y hacer una consulta:

```
mysql -uroot -p
Enter password: <TECLEAR LA CLAVE SI NO ES VACÍA>
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.44 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

mysql> use contactos;
Database changed
```

```
mysql> select * from datos;
+-----+-----+
| nombre | tlf      |
+-----+-----+
| pepe   | 95834987 |
+-----+-----+
3 rows in set (0,00 sec)

mysql> describe datos;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nombre | varchar(100) | YES  |     | NULL    |       |
| tlf    | int(11)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,00 sec)

mysql> quit
Bye
```

4. Replicar una BD MySQL con mysqldump

MySQL ofrece la una herramienta para clonar las BD que tenemos en nuestra maquina. Esta herramienta es mysqldump.

mysqldump es parte de los programas de cliente de MySQL, que puede ser utilizado para generar copias de seguridad de BD. Puede utilizarse para volcar una o varias BD para copia de seguridad o para transferir datos a otro servidor SQL (no necesariamente un servidor MySQL). EL volcado contiene comandos SQL para crear la BD, las tablas y rellenarlas.

Esta herramienta soporta una cantidad considerable de opciones. Ejecuta como root el siguiente comando:

```
mysqldump --help
```

para obtener la lista completa. En la siguiente URL se explican con detalle todas las opciones posibles:

<http://dev.mysql.com/doc/refman/5.0/es/mysqldump.html>

Concretamente, las opciones --quick o --opt hacen que MySQL cargue el resultado entero en memoria antes de volcarlo a fichero, lo que puede ser un problema si se trata de una BD grande. Sin embargo, la opción --opt está activada por defecto.

La sintaxis de uso es:

```
mysqldump ejemplodb -u root -p > /root/ejemplodb.sql
```

Esto puede ser suficiente, pero tenemos que tener en cuenta que los datos pueden estar actualizándose constantemente en el servidor de BD principal. En este caso, antes de hacer la copia de seguridad en el archivo .SQL debemos evitar que se acceda a la BD para cambiar nada.

Así, en el servidor de BD principal (maquina1) hacemos:

```
mysql -u root -p

mysql> FLUSH TABLES WITH READ LOCK;
mysql> quit
```

Ahora ya sí podemos hacer el mysqldump para guardar los datos. En el servidor principal (maquina1) hacemos:

```
mysqldump ejemplodb -u root -p > /tmp/ejemplodb.sql
```

Como habíamos bloqueado las tablas, debemos desbloquearlas (quitar el “LOCK”):

```
mysql -u root -p
```

```
mysql> UNLOCK TABLES;
```

```
mysql> quit
```

Ya podemos ir a la máquina esclavo (maquina2, secundaria) para copiar el archivo .SQL con todos los datos salvados desde la máquina principal (maquina1):

```
scp maquina1:/tmp/ejemplodb.sql /tmp/
```

y habremos copiado desde la máquina principal (1) a la máquina secundaria (2) los datos que hay almacenados en la BD.

Es importante destacar que el archivo .SQL de copia de seguridad tiene formato de texto plano, e incluye las sentencias SQL para restaurar los datos contenidos en la BD en otra máquina. Sin embargo, la orden mysqldump no incluye en ese archivo la sentencia para crear la BD (es necesario que nosotros la creamos en la máquina secundaria en un primer paso, antes de restaurar las tablas de esa BD y los datos contenidos en éstas).

Con el archivo de copia de seguridad en el esclavo ya podemos importar la BD completa en el MySQL. Para ello, en un primer paso creamos la BD:

```
mysql -u root -p
```

```
mysql> CREATE DATABASE `ejemplodb`;
```

```
mysql> quit
```

Y en un segundo paso restauramos los datos contenidos en la BD (se crearán las tablas en el proceso):

```
mysql -u root -p ejemplodb < /tmp/ejemplodb.sql
```

Por supuesto, también podemos hacer la orden directamente usando un “pipe” a un ssh para exportar los datos al mismo tiempo (siempre y cuando en la máquina secundaria ya hubiéramos creado la BD):

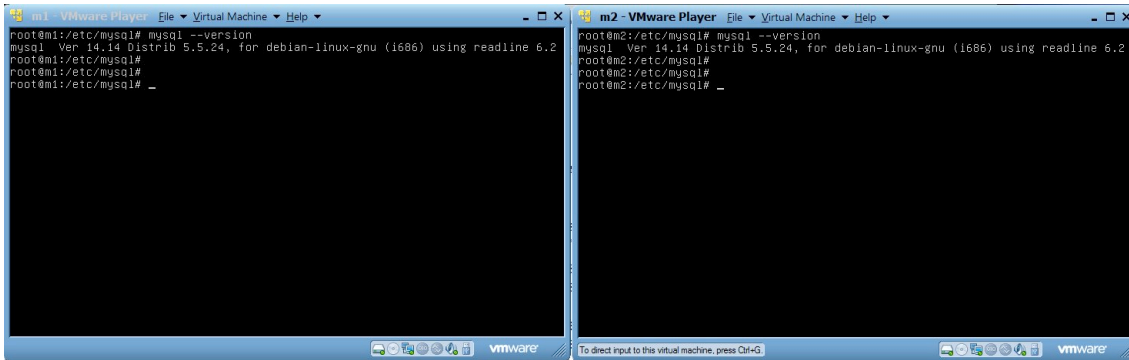
```
mysqldump ejemplodb -u root -p | ssh equipodestino mysql
```

5. Replicación de BD mediante una configuración maestro-esclavo

La opción anterior funciona perfectamente, pero es algo que realiza un operador a mano. Sin embargo, MySQL tiene la opción de configurar el demonio para hacer replicación de las BD sobre un esclavo a partir de los datos que almacena el maestro.

Se trata de un proceso automático que resulta muy adecuado en un entorno de producción real. Implica realizar algunas configuraciones, tanto en el servidor principal como en el secundario.

A continuación se detalla el proceso a realizar en ambas máquinas, para lo cual, supondremos que partimos teniendo clonadas las base de datos en ambas máquinas:



Lo primero que debemos hacer es la configuración de mysql del maestro. Para ello editamos, como root, el `/etc/mysql/my.cnf` (aunque según la versión de mysql puede que la configuración esté en el archivo `/etc/mysql/mysql.conf.d/mysqld.cnf`) para realizar las modificaciones que se describen a continuación.

Comentamos el parámetro `bind-address` que sirve para que escuche a un servidor:

```
#bind-address 127.0.0.1
```

Le indicamos el archivo donde almacenar el log de errores. De esta forma, si por ejemplo al reiniciar el servicio cometemos algún error en el archivo de configuración, en el archivo de log nos mostrará con detalle lo sucedido:

```
log_error = /var/log/mysql/error.log
```

Establecemos el identificador del servidor.

```
server-id = 1
```

El registro binario contiene toda la información que está disponible en el registro de actualizaciones, en un formato más eficiente y de una manera que es segura para las transacciones:

```
log_bin = /var/log/mysql/bin.log
```

Guardamos el documento y reiniciamos el servicio:

```
/etc/init.d/mysql restart
```

Si no nos ha dado ningún error la configuración del maestro, podemos pasar a hacer la configuración del mysql del esclavo (editar como root su archivo de configuración).

La configuración es similar a la del maestro, con la diferencia de que el `server-id` en esta ocasión será 2. Si trabajamos en versiones de mysql 5.5 o superiores (será lo habitual si la versión de Linux es actual), los datos de configuración para indicar cuál es la máquina maestra se deben añadir desde el servicio mysql (según veremos un poco más adelante, en la siguiente página).

Sólo si estamos en versiones de mysql inferiores a la 5.5 deberemos indicar esos datos relativos al master en el archivo de configuración. En ese caso añadiremos las siguientes líneas:

```
Master-host = 192.168.45.140
Master-user = usuariobd
Master-password = 123456
```

Reiniciamos el servicio en el esclavo:

```
/etc/init.d/mysql restart
```

y una vez más, si no da ningún error, habremos tenido éxito. Podemos volver al maestro para crear un usuario y darle permisos de acceso para la replicación.

Entramos en mysql y ejecutamos las siguientes sentencias:

```
mysql> CREATE USER esclavo IDENTIFIED BY 'esclavo';

mysql> GRANT REPLICATION SLAVE ON *.* TO 'esclavo'@'%'
IDENTIFIED BY 'esclavo';

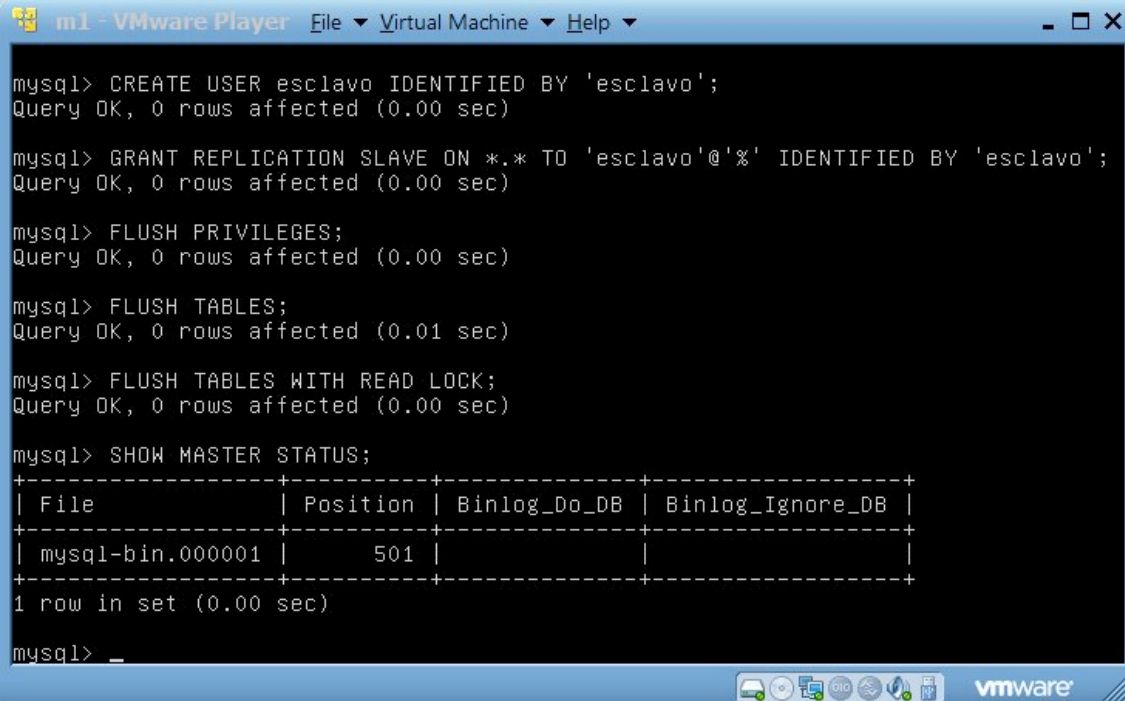
mysql> FLUSH PRIVILEGES;

mysql> FLUSH TABLES;

mysql> FLUSH TABLES WITH READ LOCK;
```

Para finalizar con la configuración en el maestro, obtenemos los datos de la BD que vamos a replicar para posteriormente usarlos en la configuración del esclavo:

```
mysql> SHOW MASTER STATUS;
```



```
mysql> CREATE USER esclavo IDENTIFIED BY 'esclavo';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT REPLICATION SLAVE ON *.* TO 'esclavo'@'%' IDENTIFIED BY 'esclavo';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH TABLES;
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000001 |      501 |              |                  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

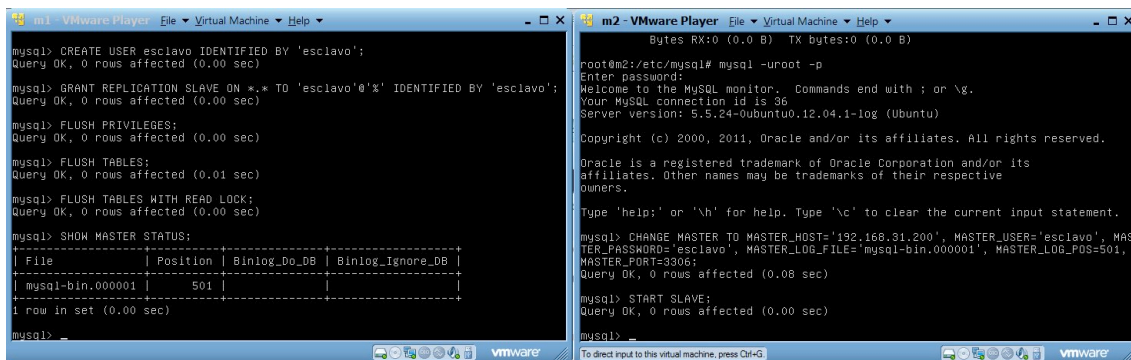
mysql> _
```

Volvemos a la máquina esclava, entramos en mysql y le damos los datos del maestro. Como indicábamos antes, sólo si trabajamos con versiones inferiores a mysql 5.5 estos datos se pueden introducir directamente en el archivo de configuración. Como lo más seguro es que estemos trabajando con una versión de mysql superior a la 5.5, entraremos en el entorno de mysql y ejecutamos la siguiente sentencia (ojo con la IP, con el valor de "master_log_file" y del "master_log_pos" del maestro):

```
mysql> CHANGE MASTER TO MASTER_HOST='192.168.31.200',
MASTER_USER='esclavo', MASTER_PASSWORD='esclavo',
MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=501,
MASTER_PORT=3306;
```

Por último, arrancamos el esclavo y ya está todo listo para que los demonios de MySQL de las dos máquinas repliquen automáticamente los datos que se introduzcan/modifiquen/borren en el servidor maestro:

```
mysql> START SLAVE;
```



The image shows two terminal windows from a VMware Player. The left window (m1) shows the setup of a MySQL master: creating a user 'esclavo', granting replication privileges, flushing privileges, flushing tables, flushing tables with read lock, and showing master status. The right window (m2) shows the setup of a MySQL slave: connecting to the master, changing master to 'esclavo', starting the slave, and showing slave status.

```
m1 - VMware Player
mysql> CREATE USER esclavo IDENTIFIED BY 'esclavo';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT REPLICATION SLAVE ON *.* TO 'esclavo'@'%' IDENTIFIED BY 'esclavo';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH TABLES;
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW MASTER STATUS;
+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+
| mysql-bin.000001 | 501 | | |
+-----+
1 row in set (0.00 sec)

mysql>

m2 - VMware Player
root@m2:/etc/mysql# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.5.24-0ubuntu0.12.04.1-log (Ubuntu)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

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> CHANGE MASTER TO MASTER_HOST='192.168.31.200', MASTER_USER='esclavo', MAS
TER_PASSWORD='esclavo', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=501,
MASTER_PORT=3306;
Query OK, 0 rows affected (0.08 sec)

mysql> START SLAVE;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Ahora, podemos hacer pruebas en el maestro y deberían replicarse en el esclavo automáticamente.

Por último, volvemos al maestro y volvemos a activar las tablas para que puedan meterse nuevos datos en el maestro:

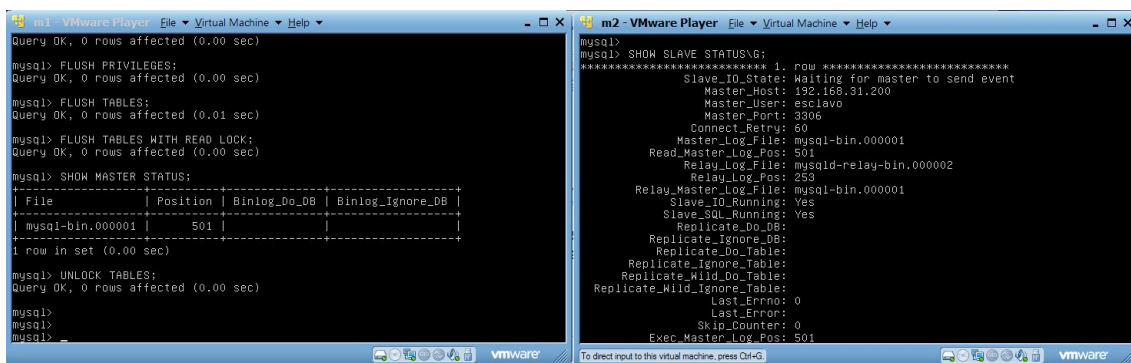
```
mysql> UNLOCK TABLES;
```

Ahora, si queremos asegurarnos de que todo funciona perfectamente y que el esclavo no tiene ningún problema para replicar la información, nos vamos al esclavo y con la siguiente orden:

```
mysql> SHOW SLAVE STATUS\G
```

revisamos si el valor de la variable “Seconds_Behind_Master” es distinto de “null”. En ese caso, todo estará funcionando perfectamente. Si no es así, es que hay algún error y los valores de las demás variables indicarán cuál es el problema y cómo arreglarlo para que todo funcione bien (un error común se debe a la falta de conexión entre máquinas porque el puerto 3306 esté bloqueado en una de las máquinas).

A continuación vemos que el valor de la variable “Seconds_Behind_Master” es 0, por lo que no hay ningún error y todo funciona como debe:



The image shows two terminal windows. The left window (m1) shows the master status after unlocking tables. The right window (m2) shows the slave status, indicating that the slave is running and the replication is successful.

```
m1 - VMware Player
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH TABLES;
Query OK, 0 rows affected (0.01 sec)

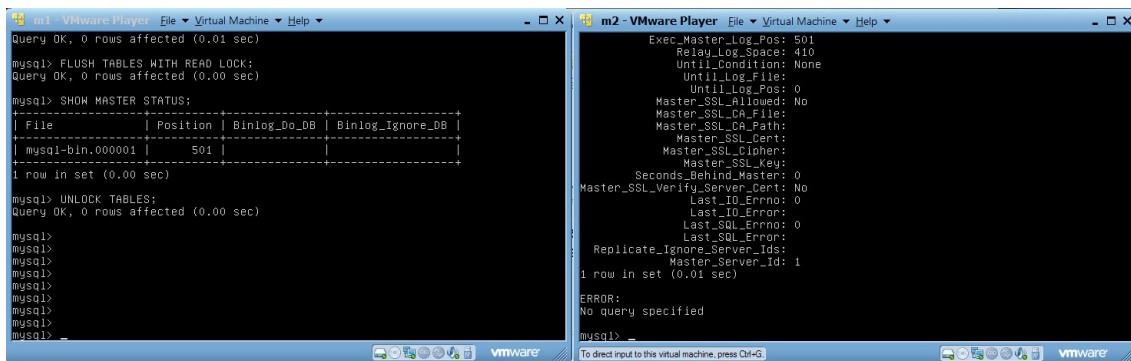
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW MASTER STATUS;
+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+
| mysql-bin.000001 | 501 | | |
+-----+
1 row in set (0.00 sec)

mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)

mysql>

m2 - VMware Player
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.31.200
Master_User: esclavo
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mysql-bin.000001
Read_Master_Log_Pos: 501
Relay_Log_File: mysqld-relay-bin.000002
Relay_Log_Pos: 253
Relay_Master_Log_File: mysql-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: 501
```



The image shows two terminal windows. The left window (m1) shows the master status after unlocking tables. The right window (m2) shows the slave status, indicating that the slave is running and the replication is successful.

```
m1 - VMware Player
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.00 sec)

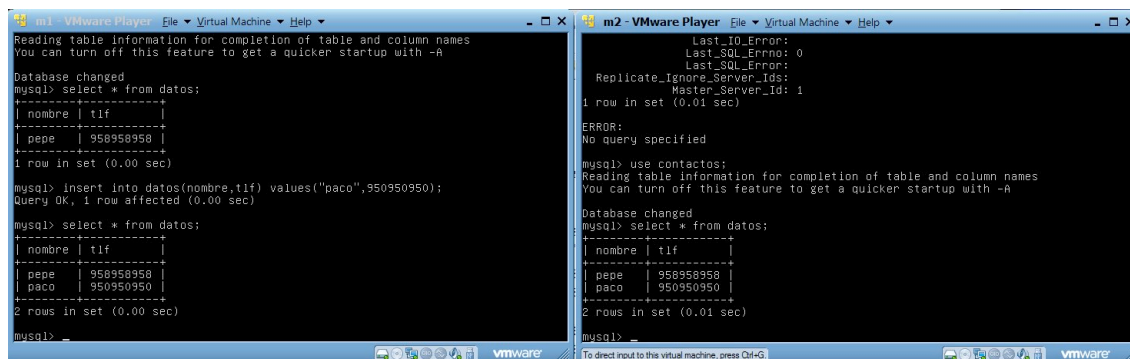
mysql> SHOW MASTER STATUS;
+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+
| mysql-bin.000001 | 501 | | |
+-----+
1 row in set (0.00 sec)

mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)

mysql>

m2 - VMware Player
mysql> SHOW SLAVE STATUS\G;
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.31.200
Master_User: esclavo
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mysql-bin.000001
Read_Master_Log_Pos: 501
Relay_Log_File: mysqld-relay-bin.000002
Relay_Log_Pos: 253
Relay_Master_Log_File: mysql-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: 501
```


Para comprobar que todo funciona, debemos ir al maestro e introducir nuevos datos a la base de datos. A continuación vamos al esclavo para revisar si la modificación se ha reflejado en la tabla modificada en el maestro:



The image shows two side-by-side terminal windows from a VMware Player. The left window, titled 'm1 - VMware Player', shows a MySQL session on the master server. It displays the 'datos' table with columns 'nombre' and 'tif'. The table initially contains two rows: 'pepe' with '958958958' and 'paco' with '950950950'. A new row is inserted: 'paco' with '950950950'. The right window, titled 'm2 - VMware Player', shows a MySQL session on the slave server. It displays the same 'datos' table, which now contains three rows: 'pepe' with '958958958', 'paco' with '950950950', and another 'paco' with '950950950'. This demonstrates that the data change on the master has been replicated to the slave.

```
m1 - MySQL> select * from datos;
+-----+-----+
| nombre | tif      |
+-----+-----+
| pepe   | 958958958 |
| paco   | 950950950 |
+-----+-----+
1 row in set (0.00 sec)

mysql> insert into datos(nombre,tif) values("paco","950950950");
Query OK, 1 row affected (0.00 sec)

mysql> select * from datos;
+-----+-----+
| nombre | tif      |
+-----+-----+
| pepe   | 958958958 |
| paco   | 950950950 |
| paco   | 950950950 |
+-----+-----+
2 rows in set (0.00 sec)

mysql>

m2 - MySQL> use contactos;
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 | tif      |
+-----+-----+
| pepe   | 958958958 |
| paco   | 950950950 |
| paco   | 950950950 |
+-----+-----+
2 rows in set (0.01 sec)

mysql>
```

Cuestiones a resolver

En esta práctica el objetivo es configurar las máquinas virtuales para trabajar de forma que se mantenga actualizada la información en una BD entre dos servidores (la máquina secundaria mantendrá siempre actualizada la información que hay en la máquina servidora principal).

Hay que **llevar a cabo las siguientes tareas obligatorias**:

1. Crear una BD con al menos una tabla y algunos datos.
2. Realizar la copia de seguridad de la BD completa usando mysqldump en la máquina principal y copiar el archivo de copia de seguridad a la máquina secundaria.
3. Restaurar dicha copia de seguridad en la segunda máquina (clonado manual de la BD), de forma que en ambas máquinas esté esa BD de forma idéntica.
4. Realizar la configuración maestro-esclavo de los servidores MySQL para que la replicación de datos se realice automáticamente.

Adicionalmente, y como tarea opcional para conseguir una mayor nota en esta práctica, se propone realizar la configuración maestro-maestro entre las dos máquinas de bases de datos.

Como resultado de la práctica 5 **se mostrará** al profesor el funcionamiento del proceso de clonado automático de la información entre bases de datos MySQL en las máquinas principal y secundaria (configuración maestro-esclavo y/o maestro-maestro, en su caso). En el documento de texto a entregar se describirá en detalle cómo se ha realizado la configuración de ambos servidores (configuraciones y comandos de terminal ejecutados en cada momento).

Normas de entrega

La práctica podrá realizarse de manera individual o por grupos de hasta 2 personas.

Se entregará como un archivo de texto en el que se muestre la información requerida. También se puede utilizar la sintaxis de Markdown para conseguir una mejor presentación e incluso integrar imágenes o capturas de pantalla. La entrega se realizará subiendo los archivos necesarios al repositorio SWAP en la cuenta de github del alumno, a una carpeta llamada "practica5".

Toda la documentación y material exigidos se entregarán en la fecha indicada por el profesor. No se recogerá ni admitirá la entrega posterior de las prácticas ni de parte de las mismas.

La detección de prácticas copiadas implicará el suspenso inmediato de todos los implicados en la copia (tanto del autor del original como de quien las copió).

Las faltas de ortografía se penalizarán con hasta 1 punto de la nota de la práctica.

Referencias

<https://support.rackspace.com/how-to/configuring-mysql-server-on-ubuntu/>
<https://dev.mysql.com/doc/refman/5.7/en/server-configuration-defaults.html>
<https://dev.mysql.com/doc/refman/5.7/en/server-configuration.html>
<http://stackoverflow.com/questions/38490785/where-is-mysql-5-7-my-cnf-file>
<http://blog.programster.org/ubuntu-16-04-default-mysql-5-7-configuration/>