

# 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

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 (nótese que vamos a trabajar como root, de ahí el carácter # indicando el prompt de root):

```
# 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 > /root/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 root@maquina1:/root/ejemplodb.sql /root/
```

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 creemos 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 < /root/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

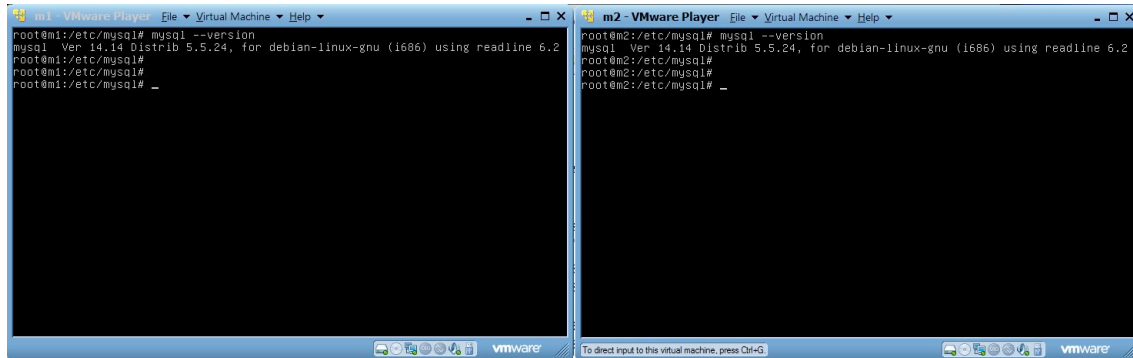
La opción anterior funciona perfectamente, pero es algo que realiza un operador a mano.

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:

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` y modificamos los siguientes parámetros:

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. Por ejemplo al reiniciar el servicio, si 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 da ningún error, vamos a la configuración de mysql el esclavo (mismo archivo).

La configuración es similar a la del maestro, con la diferencia de que el `server-id` en esta ocasión será 2, y que podemos configurar los datos del master si estamos en versiones de mysql inferiores a la 5.5 en este mismo archivo de configuración:

```
Master-host = 192.168.45.140
```

```
Master-user = usuariobd
```

```
Master-password = 123456
```

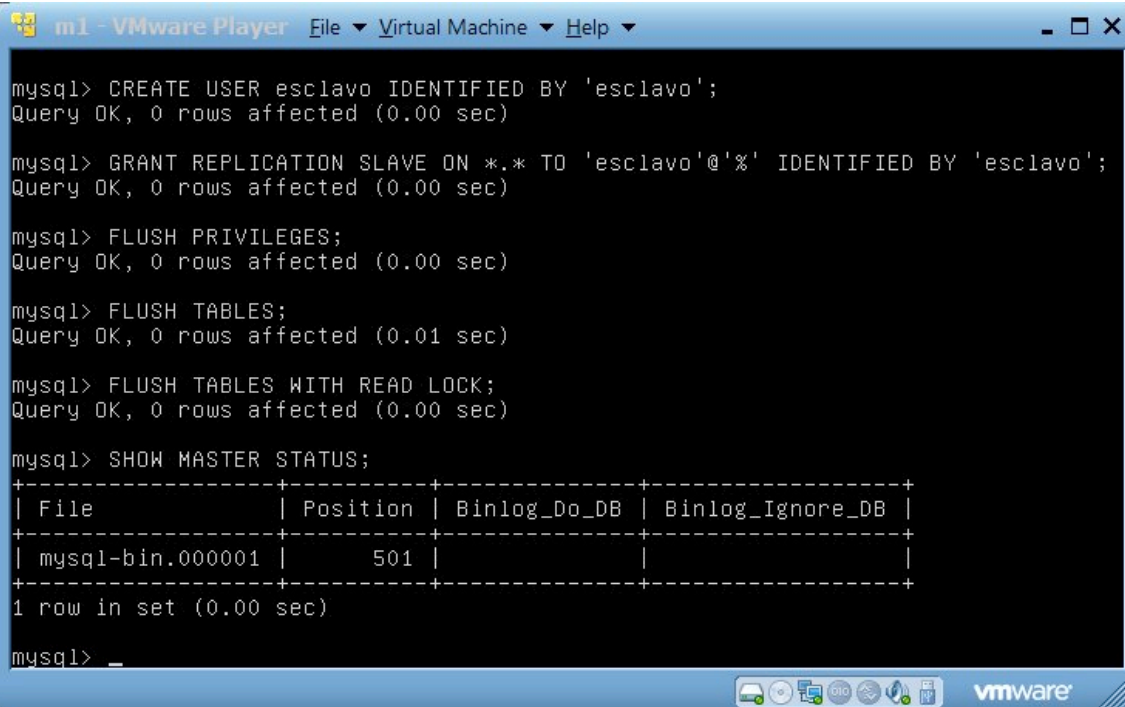
Si trabajamos en 5.5 o superiores nos dará un error al reiniciar el servicio. Eso significa que estos datos se deben añadir desde el servicio mysql (según veremos un poco más adelante).

Reiniciamos el servicio en el esclavo:

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

y si no da ningún error, volvemos al maestro para crear un usuario y darle permisos de acceso para la replicación. Entramos en mysql y ejecutamos la siguiente sentencia (ojo con la IP del esclavo):

```
CREATE USER esclavo IDENTIFIED BY 'esclavo';
GRANT REPLICATION SLAVE ON *.* TO 'esclavo'@'%' IDENTIFIED BY 'esclavo';
FLUSH PRIVILEGES;
FLUSH TABLES;
FLUSH TABLES WITH READ LOCK;
```



```
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>
```

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

```
SHOW MASTER STATUS;
```

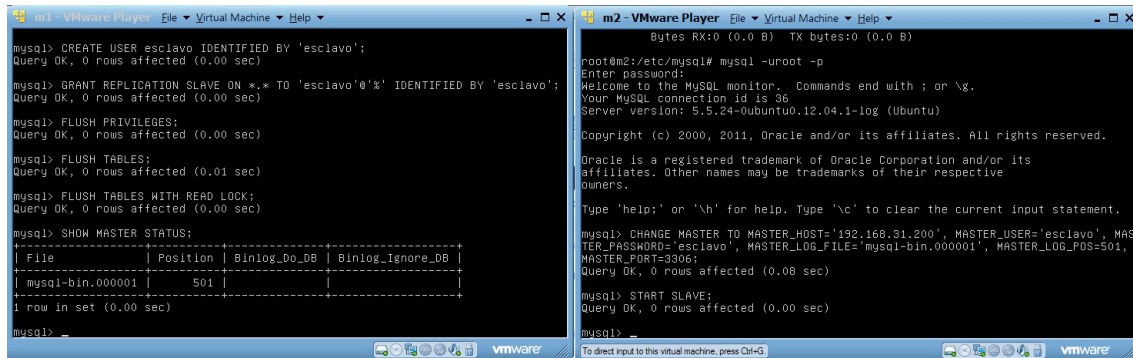
Volvemos a la máquina esclava, entramos en mysql y le damos los datos del maestro. Estos datos se pueden introducir desde el archivo de configuración si trabajamos con versiones inferiores a mysql 5.5. Si no es así, en el entorno de mysql ejecutamos la siguiente sentencia (ojo con la IP, "master\_log\_file" y del "master\_log\_pos" del maestro):

```
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 replicar los datos que se introduzcan/modifiquen/borren en el servidor maestro:

START SLAVE;



```
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)

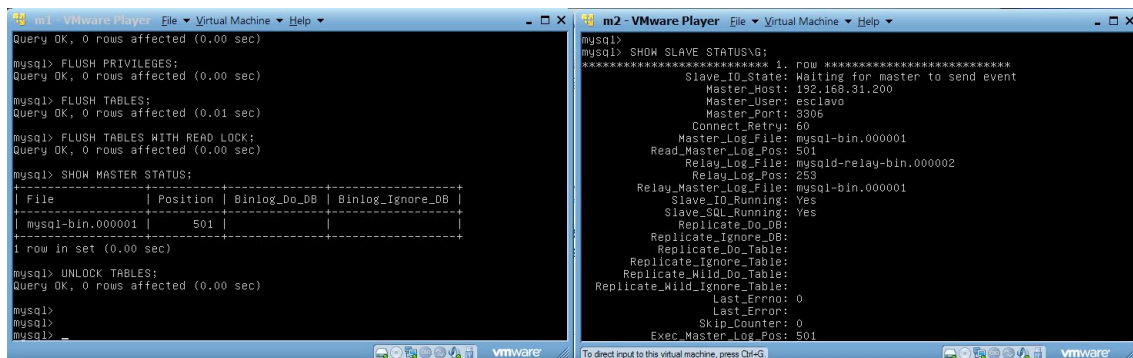
mysql>
```

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.

UNLOCK TABLES;

Ahora si queremos saber que todo funciona perfectamente y que el esclavo no tiene ningún problema para acceder al maestro y copiar la base de datos, nos vamos al esclavo y con la siguiente orden “SHOW SLAVE STATUS\G” y viendo que el valor de la variable “Seconds\_Behind\_Master” es distinto de null todo funciona perfectamente, sino, es que existen errores y las demás variables te lo dicen para que lo arregles y todo funcione perfectamente. Como se muestra en la siguiente captura de pantalla el valor de la variable es 0 y por lo tanto no tenemos ningún error y todo funciona perfectamente.



```
m1 - VMware Player
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)

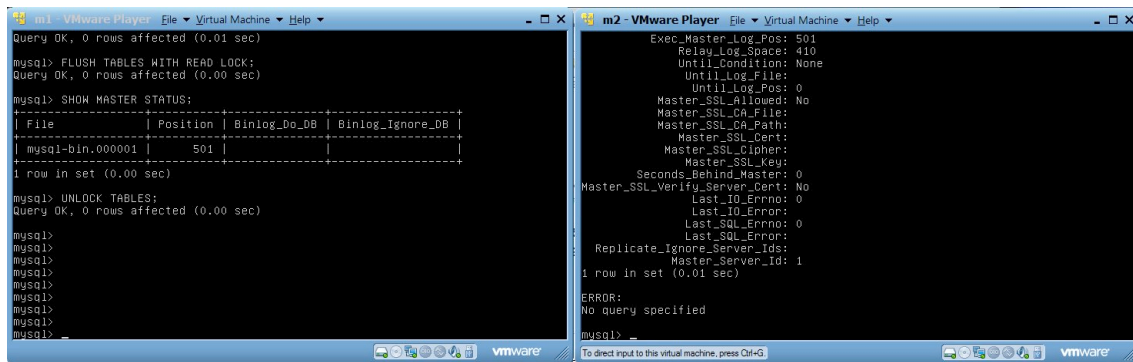
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.00 sec)

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

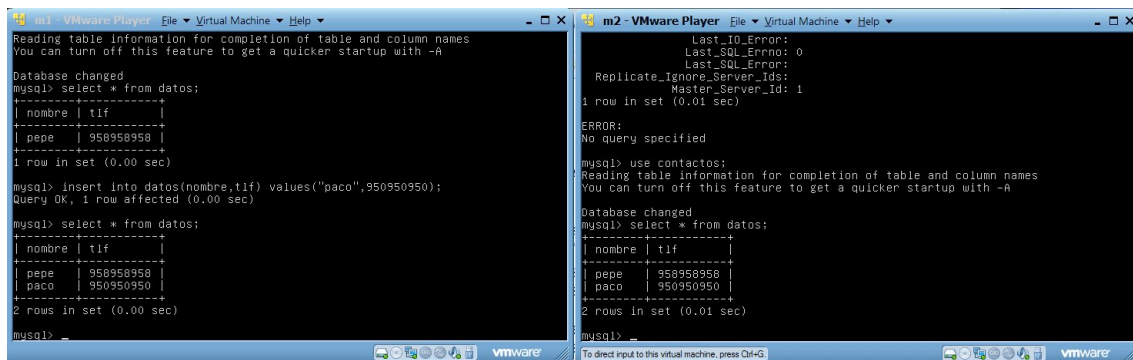
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql>
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 demostrar que todo esto que hemos realizado es verdad y que funciona lo único que tenemos que hacer es irnos al maestro introducir nuevos datos a la base de datos y luego irnos al esclavo y mirar esa misma base de datos y comprobar que las dos máquinas tienen la misma base de datos.



## 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:

- Realizar copias de seguridad de una BD completa usando mysqldump.
- Restaurar dicha copia en la segunda máquina (clonado manual de la BD).
- Realizar la configuración maestro-esclavo de los servidores MySQL para que la replicación de datos se realice automáticamente.

Como resultado de la práctica 5 se mostrará al profesor el funcionamiento del proceso de clonado (manual y automático) de la información en una BD MySQL. 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 solo archivo de texto en el que se muestre la información requerida. La entrega se realizará subiendo el archivo TXT al repositorio SWAP2015 en la cuenta de github del alumno, nombrando el archivo como:

P5\_\_nombre-apellido1-apellido2.txt

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.