

Upgrade de MySQL 5.7 a MySQL 8.0 - DirectAdmin

Introducción

Un cambio muy necesario (por más que se empeñen algunos) es la actualización a MySQL 8.0. Si seguimos interesados por seguir trabajando con MySQL, es necesario, ya que los cambios, la seguridad, y la eficacia del motor MySQL en la 8.0 supera con creces al viejo motor 5.7

Pero una de las cosas que nos trajo, las distintas bifurcaciones (forks) de MySQL (Percona, MariaDB, y otros) es que ya no son tan compatibles, o cuando menos existen ciertos cambios que nos traen de cabeza a los administradores de sistemas, sobre todo cuando nos centramos en el trabajo como desarrolladores.

“ Muchos desarrolladores no conocen en profundidad los sistemas, y eso es fuente de problemas de mantenimiento y despliegue.

Escenario

El escenario de este artículo es:

Documentación original

La documentación original para actualizar MySQL 5.7 a MySQL 8.0 con Custom Builds, es [How to upgrade mysql with custombuild 2.0](#)

```
cd /usr/local/directadmin/custombuild
./build set mysql 8.0
./build set mysql_inst mysql
./build set mysql_backup yes
```

```
./build update
```

```
./build mysql
```

Mención importante es hacer un backup antes de nada, o bien tener la opción `mysql_backup=yes` en el fichero `/usr/local/directadmin/custombuild/options.conf` mediante el seteo `./build set mysql_backup yes`

Posibles problemas y sus soluciones

Todo debería ir bien pero siempre podemos encontrarnos con algo que se escapo de las manos. Algo que alguien toco, algo que tocamos, algo modificado, que ha dejado la actualización con problemas de permisos, uno de los handicaps de este cambio junto con el de las variables de entorno de MySQL, relativas a ciertos cambios respecto de los valores por defecto, nulos, y otras cuestiones.

No funciona webmail ni PhpMyAdmin

Cuando pasa esto, no te lo esperas. Primero porque ves que puedes entrar perfectamente al panel de DirectAdmin de tu servidor, pero sin embargo te falla el acceso a estos sitios tan importantes.

El que ocurra esto es una pista muy importante porque demuestra que la autenticación está correcta, ya que ves el panel de control. Así pues, el problema está en algo que afecta a estas dos piezas.

Lo normal es que acudas a la reconstrucción (rebuild) de ambos elementos, pero eso no es sino una pérdida de tiempo, salvo que leamos un cierto detalle si hacemos caso de un tip habitual de DirectAdmin que pasa por la eliminación de la base de datos de roundcube y la instalación desde 0 de roundcube

```
mysql> DROP DATABASE da_roundcube;
```

```
Query OK, 15 rows affected (0.08 sec)
```

```
mysql> exit
```

```
Bye
```

```
root@server-xerintel01:/usr/local/directadmin/custombuild# ./build roundcube
```

```
Inserting data to mysql and creating database/user for roundcube...
```

```
Found MySQL version 8.0
```

```
Granting access: GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER,LOCK
```

```
TABLES,INDEX,REFERENCES ON da_roundcube.* TO 'da_roundcube'@'localhost';
```

```
Setting password: ALTER USER 'da_roundcube'@'localhost' IDENTIFIED WITH mysql_native_password BY  
'yuCf7CImtO5';
```

```
Database created, da_roundcube password is yuCf7CImtO5
```

Editing roundcube configuration...

Roundcube 1.4.11 has been installed successfully.

WARNING: Changed defaults (These config options have new default values):

- 'skin'
- 'smtp_port'
- 'smtp_user'
- 'smtp_pass'
- 'jquery_ui_skin_map'

Executing database schema update.

ERROR: SQLSTATE[HY000] [2054] Server sent charset unknown to the client. Please, report to the developers

ERROR: Failed to connect to database

El problema está en el juego de caracteres () por defecto no compatible así que mejor lo cambiamos en el fichero `/etc/my.cnf` y hacemos un restart del servidor mysqld.

```
[mysqld]
...
# character-set-server=utf8mb4
character-set-server=utf8
```

Una vez reiniciado nuestro servidor mysql, tendremos tanto Roundcube como PHPMyAdmin operativos.

Otros problemas derivados de las variables de entorno

Es muy típico en estos upgrades o en los de MariaDB, encontrarnos con problemas derivados de los cambios de tratamiento de ciertos aspectos de estos servidores Bases de Datos, llamados **Server SQL Modes** y documentados de forma oficial en [5.1.11 Server SQL Modes](#)

Son muy comunes: Los datos inválidos que antaño se soportaban en MySQL, convirtiéndose en '0000-00-00' Los ANSI_QUOTES El error de división por 0

Las soluciones para muchos programadores como norma general, serán la vía rápida y la vía rapidísima.

Vía rapidísima

Tras una búsqueda en Google, DuckduckGo que les llevará lectura en stackoverflow, haran un copy & paste y desactivaran TODO lo que ha avanzado MySQL en la protección y homogeneización del desarrollo de bases de datos SQL, y por supuesto de la optimización.

Es decir, desactivaran todo lo que dice el post, y como les funciona, pues ya está. Hasta otro día en el que por una migración, desastre, o lo que sea que haya modificado el comportamiento de

MySQL, se repita. Por ende, ni siquiera estará anotado en los requerimientos del software, y con toda seguridad será un quebradero de cabeza para el administrador de sistemas que lidie con ello.

Vía rápida

En la misma búsqueda, vemos el tema, y tratamos de analizarlo. Bien, somos conscientes del problema, y buscamos las variables adecuadas al problema que deberemos cambiar.

Por ejemplo, tenemos problemas con datos nulos o incorrectos en un datetime.

```
"Incorrect date value: '0000-00-00' for column 'datsent' at row 1"
```

La solución rápida pasaría por modificar los modos SQL a **NO_ZERO_IN_DATE,NO_ZERO_DATE**

Lo primero es guardar la configuración actual

```
mysql> SELECT @@GLOBAL.sql_mode;

+-----+
| @@GLOBAL.sql_mode |
+-----+
|
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZE
RO,NO_ENGINE_SUBSTITUTION |
+-----+

1 row in set (0.00 sec)
```

Lo segundo es removerlo los modos relativos al problema en tiempo de ejecución

```
mysql> SET GLOBAL sql_mode =
'ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
mysql> SET SESSION sql_mode =
'ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

Lo tercero es añadirlo al fichero my.cnf para que en el siguiente reinicio los valores esten de la forma deseada.

```
[mysqld]
sql_modes=ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
```

Por supuesto añadir a la documentación del proyecto la necesidad de que conservemos o tengamos esos modelos de SQL

Vía correcta

Lamentablemente los anteriores caminos no son sino un parche que tarde o temprano nos pasará factura, además de que suponen una mala praxis profesional, ya que la diferencia entre hacer las cosas bien y no hacerlas bien, es también tiempo, y el tiempo es dinero.

¿Qué es pues lo correcto?

Lo correcto es modificar lo que está incorrecto, que muchas veces no es mucho más complicado que una o dos horas de un programador junior.

No vamos a explicar el cómo, detalladamente pues las variantes son muchas, pero básicamente diremos que debemos actualizar las filas actuales de fechas a 0 a valores válidos ('1970-01-02' es un valor válido) o bien permitir que la columna tenga valores nulos.

Nota adicional sobre backups

Un caso muy doloroso, es cuando por razones de urgencia debemos migrar o restaurar nuestra base de datos en otro sistema (ejemplo, un incendio en nuestro centro de datos de OVH).

La solución pasa por una modificación temporal de los modos sql.

[Error 1292 y Error 1217](#) fue un artículo que escribe hace tiempo a colofón de estos problemas

La cuestión es añadir al principio de nuestro fichero sql

```
/* Añadir al principio */
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS;
SET FOREIGN_KEY_CHECKS=0;
-- save current setting of sql_mode
SET @old_sql_mode := @@sql_mode ;

-- derive a new value by removing NO_ZERO_DATE and NO_ZERO_IN_DATE
SET @new_sql_mode := @old_sql_mode ;
SET @new_sql_mode := TRIM(BOTH ',' FROM REPLACE(CONCAT(',',@new_sql_mode,','),',NO_ZERO_DATE,',' ,','));
SET @new_sql_mode := TRIM(BOTH ',' FROM REPLACE(CONCAT(',',@new_sql_mode,','),',NO_ZERO_IN_DATE,',' ,','));
SET @@sql_mode := @new_sql_mode ;
```

Y al final

```
echo 'SET @@sql_mode := @old_sql_mode; ' >> dump.sql
echo 'SET @@FOREIGN_KEY_CHECKS := @OLD_FOREIGN_KEY_CHECKS; ' >> dump.sql
```

Enlaces

- [Upgrade mysql from 5.7 to Mysql 8.0 some issues](#)
- [How to set sql_mode in my.cnf in MySQL 8?](#)

Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como esta, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

Revision #5

Created 28 May 2021 17:51:56 by Abkrim

Updated 29 May 2021 05:07:04 by Abkrim