MySQL

Réplication MySQL

Réplication MySQL

Voir l'état d'une réplication

Il faut taper la commande

```
SHOW SLAVE STATUS\G;
```

Problème de réplication suite au redémarrage du serveur MySQL

'You cannot 'ALTER' a log table

Si vous avez ce message

```
Erreur : CRIT Slave_IO_Running: Yes Slave_SQL_Running: No Last_Error: Error 'You cannot 'ALTER' a log table if logging is enabled' on query
```

Alors connectez vous sur vos serveurs MySQL (master and slave) et tapez les commandes ci dessous :

```
STOP SLAVE;
SET GLOBAL slow_query_log = "OFF";
START SLAVE;
SET GLOBAL slow_query_log = "ON";
```

The server is not configured as slave

Dans ce genre de cas, pas besoin de pleurer. Il faut juste avoir le mot de passe de l'utilisateur replication. Si vous ne l'avez pas, changez le sur vos deux serveurs :

MySQL 1 (Master)

```
set password for 'replication'@'192.168.1.%' = password('dfihgartdccuvbg');
```

MySQL 2 (Slave)

```
set password for 'replication'@'192.168.1.%' = password('dfihgartdccuvbg');
```

Ensuite, faire un dump du serveur MySQL 1 et récupérer la position de ses logs :

Dans un premier terminal, connectez vous sur le master MySQL et tapez la commande

```
FLUSH TABLES WITH READ LOCK
```

Sans fermer le terminal (sinon vous perdrez le lock et donc des données), passez sur un autre terminal

```
mysqldump --master-data=1 -u root -p --single-transaction --routines --triggers --events --all-databases >
/var/tmp/mysql1.dump
mysql -u root -p -e "show master status;"
```

Le `show master status` devrait vous retourner des valeurs de la forme :

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000076 |   791681 |              |                  |
+------------------+----------+--------------+------------------+
```

Quittez le terminal 1 pour enlever le lock.

Réimporter le dump dans le serveur MySQL2 Une fois l'import réalisé, tapez la commande :

```
CHANGE MASTER TO MASTER_HOST='192.168.1.1', MASTER_USER='replication',
MASTER_PASSWORD='dfihgartdccuvbg', MASTER_LOG_FILE='mysql-bin.000076', MASTER_LOG_POS=791681;
```

Cette méthode peut fonctionner sur de la réplication croisée mais avec plus de risque.

Écart avec la réplication croisée MySQL

Une réplication MySQL croisée c'est quand les deux serveurs MySQL sont et `master` et `slave`. Il est donc en l'état impossible de savoir quel serveur est plus en avance sur l'autre. Dans une partie des cas, la réplication croisée n'est utilisé que lorsque un des deux serveurs se plante, le second serveur prend la relève. Cela est soit géré dans le code de l'application, soit avec un transfert d'IP entre les deux serveurs (soit pas du tout géré mais chuuuut).

Pour réparer la réplication croisée, on va devoir verouiller à l'écriture les tables. On a rien sans rien même si une majorité des gens pensent qu'il est possible de ne gérer aucune coupure pour réparer la réplication...

Dans l'exemple ci dessous, on a deux serveurs MySQL

SQL1 : 192.168.1.101 SQL2 : 192.168.1.102

On se connecte donc sur les deux serveurs MySQL en root et on tape :

```
STOP SLAVE;
FLUSH TABLES WITH READ LOCK;
```

On ouvre maintenant deux autres terminal.

Sur un des deux terminal, connectez vous sur le serveur SQL1. Verifions d'abord l'ID de ce master

```
SHOW VARIABLES LIKE 'server_id';
```

Il va renvoyer cette valeur ci

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 1     |
+---------------+-------+
```

On peut donc faire un dump de la base de donnée

```
mysqldump --master-data=1 -u root -p --single-transaction --routines --triggers --events --all-databases --insert-ignore > /var/tmp/mysql1.dump
```

Remplcer bien entendu le `1` de `--master-data=1` par la valeur `server_id` (qui peut bien entendu être différente).

Sur le second terminal, connectez vous sur le serveur SQL2, verifiez également le `server_id` puis faites un dump de la base de donnée.

```
mysqldump --master-data=2 -u root -p --single-transaction --routines --triggers --events --all-databases --insert-ignore > /var/tmp/mysql2.dump
```

Le `--insert-ignore` aura pour effet de modifier les entrées différente (genre une entrée supprimé, une entrée mise à jour...) mais n'effacera pas les entrées déjà existantes (aucun `DROP TABLE` ou autre). Cela n'est donc pas destructif.

Une fois les deux dump fait, transférer les sur l'autre serveur (le dump de SQL2 devra aller sur SQL1 et vice versa).

Sur SQL1

```
mysql -u root -p < /var/tmp/mysql2.dump
mysql -u root -p -e "RESET SLAVE"
```

Sur SQL2

```
mysql -u root -p < /var/tmp/mysql1.dump
mysql -u root -p -e "RESET SLAVE"
```

Une fois les deux imports réalisés et le `RESET SLAVE` executé, retournee sur les deux premiers terminaux et fermez les.

Il vous reste donc plus que deux terminaux, connectés respectivement sur SQL1 et SQL2. Sur chacun des terminaux tapez :

```
mysql -u root -p -e "START SLAVE"
```

Puis verifiez si tout vas bien avec

```
SHOW SLAVE STATUS\G;
```

Normalement tout est réglé.

Sauvegarde d'un serveur MySQL

Sauvegarde d'un serveur MySQL

```
mysqldump -u backup --password=xxx --single-transaction --routines --triggers --events --all-databases
```

Ajouter `--master-data=1` s'il s'agit d'une réplication

Activer les slow logs MySQL

Dans la vie d'un site web en production, le temps de réponse à la base peut augmenter pour divers raisons.

Pour identifier les requêtes il existe deux méthodes.

Méthode à chaud

Connectez vous à votre base de donnée avec des droits avancé (si possible root).

Activez les slow logs

```
SET GLOBAL slow_query_log = 'ON';
```

Definissez le temps d'execution qu'une requête peut avoir avant d'être considéré comme lente

```
SET GLOBAL long_query_time = 5;
```

En remplaçant 5 par la valeur que vous souhaitez en seconde.

Choississez dans quel dossier doit être écrit ces logs

```
SET GLOBAL slow_query_log_file = '/var/log/mysql/mysql-slow.log';
```

Puis testez

```
SELECT SLEEP(10);
```

Vous devriez avoir le résultat dans le fichier `/var/log/mysql/mysql-slow.log`.

Pour le désactivez passer la variable `slow_query_log` à `OFF`

```
SET GLOBAL slow_query_log = 'OFF';
```

Méthode à froid

Éditez le fichier `/etc/mysql/my.cnf`

Ajoutez les lignes ci dessous :

```
slow_query_log = 1
slow_query_log_file = /var/log/mysql/mysql-slow.log
long_query_time = 5
```

`slow_query_log` Active les slow query `slow_query_log_file` Définie le chemin dans lequel MySQL va écrire ces logs `long_query_time` Définie le temps d'execution qu'une requête peut avoir avant d'être considéré comme lente

Relancez MySQL

```
systemctl restart mysql
```

Pour testez, connectez vous sur votre base de données puis tapez la commande suivante

```
SELECT SLEEP(10);
```

Vous devriez voir une occurence appraitre dans le fichier `/var/log/mysql/mysql-slow.log`

Pour désactivez ça, retirez les directives ajoputé dans `/etc/mysql/my.cnf` et redémarrez MySQL.