

mysql / mariadb

Docs SQL / Serveur SQL...

- [AutoMysqlBackup](#)
- [Commandes memo](#)
- [mysql/mariadb - Configuration et optimisation](#)

AutoMysqlBackup

→ Script .sh permettant de faire des backup auto des base AutoMysqlBackup

Pas default le script sauvegarde toutes les bases

Installation

→ Installer le paquet `Automysqlbackup`

```
apt install automysqlbackup
```

Configuration

Tout se configure dans `/etc/default/automysqlbackup` Fichier bien documenté pour se repérer

Pour changer le répertoire de svg, changer la valeur de 'BACKUPDIR' dans le fichier

→ **Lancer une backup manuelle:**

Lancer le script suivant `/usr/sbin/automysqlbackup`

Le script s'exécute automatiquement tous les jours via cron

Docs

→ [Ubuntu Wiki](#)

→ [Doc du Projet](#)

Commandes memo

Tables

Show Table Schema

```
DESCRIBE TABLE_NAME;
```

Create table

```
CREATE TABLE nom_de_la_table  
(  
    colonne1 type_donnees,  
    colonne2 type_donnees,  
    colonne3 type_donnees,  
    colonne4 type_donnees  
)
```

→ [SQL.SH Create Tables](#)

Users & Permissions

Create User & Del User

```
CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';  
DROP USER 'username'@'localhost';
```

Users Password

Change Password User

```
UPDATE mysql.user SET Password=PASSWORD('new-password-here') WHERE USER='user-name-here' AND  
Host='host-name-here';
```

Privileges

Add Privileges:

```
GRANT ALL PRIVILEGES ON * . * TO 'newuser'@'localhost';
```

- ALL PRIVILEGES- as we saw previously, this would allow a MySQL user full access to a designated database (or if no database is selected, global access across the system)
- CREATE- allows them to create new tables or databases
- DROP- allows them to them to delete tables or databases
- DELETE- allows them to delete rows from tables
- SERT- allows them to insert rows into tables
- SELECT- allows them to use the SELECT command to read through databases
- UPDATE- allow them to update table rows
- GRANT OPTION- allows them to grant or remove other users' privileges

Revoke Privileges

```
REVOKE type_of_permission ON database_name.table_name FROM 'username'@'localhost';
```

Show user privileges

```
SHOW GRANTS username;
```

Manipulation données

Ajout données

Pour [ajouter des données](#) dans une table existante:

```
INSERT INTO myTable VALUES ('value1','value2')
```

Avec cette première syntaxe, il faut obligatoirement mettre toutes les données, dans le bon ordre des colonnes.

Pour insérer des données et ne remplir que certaine colonnes (les autres colonnes prendront une valeur nulle, ou celle par défaut):

```
INSERT INTO mytable (colonne_1, colonne_2,...) VALUES ('valeur1',valeur2',...)
```

Il n'est pas nécessaire que l'ordre des colonnes soit respecté.

Commandes diverse

Insérer des données dans une table, en récupérant des valeur de cette table avec une condition, et ajouter une valeur fixe.

Par exemple, si j'ai une table mappant l'id d'un utilisateur avec l'id d'un objet stocké dans une autre table, afin de relier les 2.

Je veux ajouter dans ma table "user_object" des valeurs, où je copie les droits d'un autre user. Je récupère dans la table toutes les valeurs avec l'id de cet user(id 1), et j'ajoute les données dans la table en me basant sur cette requête + l'id de l'user que je veux ajouter (id 2).

```
INSERT INTO user_object (id_user, id_object) SELECT id_user,2 FROM user_object WHERE value2=1
```

Docs

→ [SQL.SH Create Tables](#)

→ [How To Create a New User and Grant Permissions in MySQL](#)

→ [MySQL Change a User Password](#)

→ [Mémo SQL pour la Data Science](#)

→ [Jointure SQL](#)

→ [CheatSheet SQL](#)

mysql/mariadb - Configuration et optimisation

Cette documentation explique la configuration et la customisation d'un serveur mariadb, afin d'améliorer ses performances.

Elle fait office de mémo pour la compréhension de ces paramétrages

Ici la configuration est à destination d'un serveur avec 16G de RAM. Elle sera donc à adapter en fonction des machine, avec plus ou moins de RAM.

Fichiers de configuration

Ces paramètres sont à mettre dans les fichiers de config de mariadb.

Sous debian:

- `/etc/mysql/mariadb.cnf`
ou
- `/etc/mysql/mariadb.conf.d/50-server.cnf` (config spécifique à mariadb)

Sous Redhat Entreprise Linux et dérivée:

- `/etc/my.cnf`

Fichier de configuration d'exemple

Exemple pour un système disposant de 16 Go de RAM

```
[mysqld]
# Settings généraux
user = mysql
pid-file = /run/mysqld/mysqld.pid
basedir = /usr
datadir = /var/lib/mysql
tmpdir = /tmp
# Désactivation résolution de nom
skip-name-resolve
# Interface d'écoute
bind-address = 127.0.0.1
```

```
# Paramétrages performances
innodb_buffer_pool_size = 8G
innodb_buffer_pool_instances = 8
innodb_log_file_size = 512M
thread_cache_size = 16
query_cache_size = 128M
query_cache_type = 1
tmp_table_size = 1024M
max_heap_table_size = 1024M
table_open_cache = 4096
table_definition_cache = 4096

# Paramétrage connexions
max_connections = 500
max_user_connections = 50
```

Explication des paramétrages pour la customisation des performances:

- **innodb_buffer_pool_size:** Mémoire RAM utilisée pour le cache innodb. Cela diminue les accès disques, améliorant les performances. Il est recommandé de mettre la moitié de la RAM (ici il y a 16G sur le système)
- **innodb_buffer_pool_instances:** Nb d'instances "buffer pool" d'innodb. Mettre autant d'instance que de RAM affectée (ici 8)
- **innodb_log_file_size:** Taille fichiers journaux d'innodb. L'augmentation de ce paramètres, permet d'améliorer les perfs
- **thread_cache_size:** Nombre de threads à mettre en cache, pour les réutiliser. Cela permet de réutiliser le thread mis en cache, si un client se reconnecte
- **query_cache_size:** Cache des requêtes. Le résultat de requêtes SQL sera mis en cache, afin d'accélérer les performances
- **query_cache_type = 1:** Activation du cache de requêtes
- **max_connections:** Nombre max de connexions acceptées par le serveur. Au delà, les nouvelles connexions seront refusées
- **max_user_connections:** Nombre max d'utilisateurs connectés
- **tmp_table_size et max_heap_table_size:** Cache mémoire des tables temporaires
- **table_open_cache et table_definition_cache:** Nombre des tables SQL à garder ouverte

Après toute modification de la configuration, il faudra redémarrer le service afin qu'ils soient pris en compte:

```
systemctl restart mariadb
```

Ressources

- [Several Ways to Optimize MySQL/MariaDB for Performance](#)