7. Configurer une BDD

- Postgres
  - Installer PostgreSQL
  - Processus de création d'une BDD
  - Le terminal PSQL
  - Gérer les utilisateurs
  - Sauvegarder une BDD
  - Restaurer une BDD
- MariaDB
  - Se connecter dans MYSQL
  - Faire une sauvegarde MariaDB

Postgres

Installer PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle et objet. C'est un outil libre disponible selon les termes d'une licence de type BSD. Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres, ou propriétaires tel que Oracle.

Installation de PostgreSQL

```
# Importation du repository
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'
# Importation des clés signé
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
OK
# Installation version 14 de PostgreSQL
apt-get install postgresql-common postgresql-14 postgresql-contrib-14 postgresql-doc-14
# Vérifie que le service PostgreSQL fonctionne correctement
sudo service postgresql status
```

Changer le mot de passe administrateur

Maintenant il faut mettre un mot de passe pour l’utilisateur admin de Postgres.

```
# On se connecte sur l'utilisateur postgres
sudo su - postgres
# On affecte un mot de passe
psql -c "alter user postgres with password 'StrongDBPasswordUser'"
ALTER ROLE
```

Se connecter sur PSQL

Pour accéder au terminal PSQL qui permet d’administrer les bases de données sur PostgreSQL il suffit de se connecter sur l’utilisateur postgres et d'exécuter la commande psql.

```
> sudo su - postgres
> psql
psql (14.1 (Debian 14.1-1.pgdg110+1))
Saisissez « help » pour l'aide.

postgres=#
```

Processus de création d'une BDD

1) Créer une BDD

```
CREATE DATABASE <dbname>;
```

Sortie :

```
postgres=# CREATE DATABASE exempledb;
CREATE DATABASE
```

2) Créer un utilisateur pour cette BDD

Il est préférable d'avoir un utilisateur par base donnée.

```
CREATE USER <username> WITH ENCRYPTED PASSWORD 'yourpass';
```

Sortie :

```
postgres=# CREATE USER myuser WITH ENCRYPTED PASSWORD 'mypass';
CREATE ROLE
```

3) Donner tous les droits sur la BDD

Une fois l'utilisateur et la base de donnée créé, pour que celui-ci puisse la manager il est nécessaire de donner des droits au nouvel utilisateur.

```
GRANT ALL PRIVILEGES ON DATABASE <dbname> TO <username>;
```

Sortie :

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE exempledb TO myuser;
GRANT
```

4) Donner les droits de connexion

```
postgres=# GRANT CONNECT ON DATABASE exempledb TO myuser;
GRANT
```

5) Modifier le propriétaire

Modifier le propriétaire de la BDD pour mettre notre utilisateur récemment créé :

```
ALTER DATABASE db_name OWNER TO new_owner_name;
```

56) Se connecter sur la nouvelle BDD

```
 psql -U <username> -d <dbname>
```

Le terminal PSQL

> **psql** — terminal interactif PostgreSQL

Accéder à PSQL

Il existe différente façon d'accéder au terminal interactif de PostgreSQL.

Pour se connecter avec un utilisateur et son mot de passe :

```
psql -U <username> -W
```

Liste des options PSQL

Option PSQL	Description
-U `username`	Se connecte à la base de données en tant que l'utilisateur
-W	Force psql à demander un mot de passe avant de se connecter à une base de données
-d `dbname`	Indique le nom de la base de données où se connecter
-h `host`	Indique le nom d'hôte de la machine sur lequel le serveur est en cours d'exécution. (Pour se connecter à distance)
-p `port`	Indique le port TCP a utiliser pour la connexion (par défaut: 5432)
-o *`nomfichier`*	Dirige tous les affichages de requêtes dans le fichier *`nomfichier`*.
-L *`nomfichier`*	Écrit tous les résultats des requêtes dans le fichier *`nomfichier`*

Liste des commandes PSQL

L'invite devient `-#` (pour le super utilisateur) ou `->` (pour l'utilisateur normal) pour la poursuite de la commande.

Commande PSQL	Description
\?	Pour les aides sur les commandes PSQL
\h	Pour les aides sur les commandes SQL
\du	Liste des utilisateurs
\q	Pour quitter PSQL
\l	Lister les base de données
\d	Affiche toutes les tables, indexes, vues et séquences
\dt	Affiche toutes les tables
\c `dbname` `[username]`	Pour se connecter à une base de donnée avec en option un utilisateur

Gérer les utilisateurs

Se connecter à PostgreSQL

```
psql -h localhost -p 5432 -U <username> -W
```

Il est possible que la commande psql vous sorte une erreur de type : `commande inconnue`.

Dans ce cas vous allez devoir vous connecter sur l'utilisateur Linux possédant les droits d'utilisation de la commande psql.

```
su - postgres
```

Lister les utilisateurs

Dans l'interface de psql :

```
\du
```

Créer un utilisateur

Il existe deux commandes différentes pour réaliser cette action.

```
CREATE USER <username>;
```

ou

```
CREATE ROLE <username> WITH LOGIN;
```

Créer un utilisateur avec l'utilitaire

La création interactive d'un utilisateur est une option pratique disponible uniquement pour l'utilitaire client. Pour créer un utilisateur de manière interactive, exécutez la commande suivante :

```
sudo -u postgres createuser --interactive
```

Créer un super utilisateur

> Un super utilisateur de base de données contourne toutes les vérifications, ce qui est dangereux du point de vue de la sécurité. Utilisez cette action avec précaution et évitez de travailler avec un compte super utilisateur sauf en cas de nécessité absolue.

Sur PostgreSQL le "super user" est `postgres` mais dans certain cas il est possible de vouloir créer son propre super user.

Pour créer un super utilisateur :

```
CREATE USER <username> SUPERUSER;
```

Si cet utilisateur doit avoir un mot de passe :

```
CREATE USER <username> WITH SUPERUSER PASSWORD 'passwordstring';
```

Mot de passe

Dans le cas ou vous avez créé un utilisateur sans mot de passe il possible de lui rajouter le mot de passe via la commande suivante.

```
ALTER USER <username> WITH PASSWORD '<password>';
```

Créer un utilisateur avec des droits

```
CREATE USER <name> WITH <option>;
```

Option Syntax	PSQL	Explanation
`-s` `--superuser`	`SUPERUSER`	Add the superuser privilege.
`-S` `--no-superuser`	`NOSUPERUSER`	No superuser privilege (default).
`-d` `--createdb`	`CREATEDB`	Allows the user to create databases.

`-D` `--no-createdb`	`NOCREATEDB`	Not allowed to create databases (default).
`-r` `--createrole`	`CREATEROLE`	Allows the user to make new roles.
`-R` `--no-createrole`	`NOCREATEROLE`	Not allowed to create roles (default).
`-i` `--inherit`	`INHERIT`	Automatically inherit the privileges of roles (default).
`-I` `--no-inherit`	`NOINHERIT`	Do not inherit privileges of roles.
`-l` `--login`	`LOGIN`	Allows the user to log into a session with the role name (default).
`-L` `--no-login`	`NOLOGIN`	Not allowed to log into a session with the role name.
`--replication`	`REPLICATION`	Allows initiating streaming replication and activating/deactivating backup mode.
`--no-replication`	`NOREPLICATION`	Not allowed to initiate streaming replication or backup mode (default).
`-P` `--pwprompt`	`PASSWORD '<password>'`	Initiates password creation prompt or adds provided password to the user. Avoid using this option to create a passwordless user.
/	`PASSWORD NULL`	Specifically sets the password to null. Every password authentication fails for this user.
`-c <number>` `--connection-limit=<number>`	`CONNECTION LIMIT <number>`	Sets the maximum number of connections for user. Default is without limit.

Changer les droits d'un utilisateur

Par exemple pour donner les droits de création d'une base de donnée a un utilisateur :

```
ALTER USER <username> CREATEDB;
```

Dans le même ordre d'idée on peux donner les droits super utilisateur :

```
ALTER USER <username> WITH SUPERUSER;
```

Mais aussi les enlever :

```
ALTER USER <username> WITH NOSUPERUSER;
```

Changer le mot de passe d'un utilisateur

```
ALTER USER user_name WITH PASSWORD 'new_password';
```

Supprimer un utilisateur

```
DROP USER [IF EXISTS] <username>;
```

Si l'utilisateur que vous essayez de supprimer possède des dépendances, la suppression échouera.

Vous allez devoir transférer les dépendances a un autre utilisateur.

1) Par exemple, pour transférer les objets appartenant à myuser à postgres, exécutez :

```
REASSIGN OWNED BY <old_user> TO <new_user>;
```

2) Supprimer les connexions de l'objet de la base de données à l'utilisateur avec :

```
DROP OWNED BY <username>;
```

3) Maintenant vous pouvez supprimer l'utilisateur

Supprimer un rôle

```
DROP ROLE [IF EXISTS] <name>;
```

Donner l'accès à une BDD à un utilisateur

```
GRANT ALL PRIVILEGES ON DATABASE <db_name> TO <username>;
```

Sauvegarder une BDD

Réaliser une sauvegarde d'une BDD

Pour créer une sauvegarde (**backup** en Anglais) de la base de données sur PostgreSQL c'est très simple car il existe un utilitaire `pg_dump` pour créer une sauvegarde même si la base données est en cours d'utilisation.

Les sauvegardes peuvent être produits dans des formats de script ou de fichier d'archive. elles sont constitué des fichiers en texte clair contenant les commandes SQL nécessaires pour reconstruire la base de données dans l'état où elle se trouvait au moment où elle a été enregistrée.

Tout d'abord nous allons créer un répertoire de sauvegarde pour centraliser les sauvegarde de base de données. Puis nous allons créer une sauvegarde pour vérifier que tout fonctionne. Ensuite nous allons créer un script de sauvegarde de BDD qui s'exécutera périodiquement.

Création du répertoire des sauvegardes

```
sudo mkdir -p /var/backup
```

Créer une sauvegarde

Commande pour créer une sauvegarde. Remplacer les variables par les bonnes valeur avant de l'exécuter.

${DB_USER}	Utilisateur ayant accès a la base de données
${DB_PASSWORD}	Mot de passe de l'utilisateur
${DB_HOST}	Host de la base de données : Adresse IP ou nom de domaine
${DB_NAME}	Nom de la base de données

```
/usr/bin/pg_dump postgresql://${DB_USER}:${DB_PASSWORD}@${DB_HOST}/${DB_NAME} | gzip -c > /var/backup/"${DB_NAME}-`date +%d-%m-%y_%T`".sql.gz;
```

Le fichier générer sera sous le nom suivant : **${DB_NAME}-${DATE}_${HEURE}.sql.gz**

Automatiser la sauvegarde

Le processus de remplacement des variables pour réaliser une sauvegarde peut-être long et fatiguant. Il est serait plus pratique de créer un script regroupant les démarches précédentes.

Création du script

```
sudo nano save-database
```

Contenu du script

```
#!/bin/bash

# # # # # # # # # # # # # # #
# SAVE DATABASE by gpatruno #
# # # # # # # # # # # # # # #

# définition des variables
backupdir='/etc/backup';
datefolder=`date +%m-%y`;
datescript=`date +%d-%m-%y_%T`;
DB_USER=postgres
DB_NAME=dbname
DB_PASSWORD=dbpassword
DB_HOST=localhost

echo "Répertoire de destination : $backupdir/$datefolder"
mkdir -p $backupdir/$datefolder;

echo  "PostgreSQL Backup for database : ${DB_NAME} start at " `date +%T`;
echo  "Exporting ${DB_NAME}";

# Pour conserver la version non compressé du fichier 'gzip -c fichier-de-sortie.ext > fichier-de-sortie.ext.gz'
# Sinon  compresse le fichier fichier-de-sortie et la remplace par la version compressé nommé avec l’extention *.gz 'gzip fichier-de-sortie.ext.gz'
echo "/usr/bin/pg_dump postgresql://${DB_USER}:${DB_PASSWORD}@${DB_HOST}/${DB_NAME} | gzip -c > $backupdir/$datefolder/"${DB_NAME}-$datescript".sql.gz;"
/usr/bin/pg_dump postgresql://${DB_USER}:${DB_PASSWORD}@${DB_HOST}/${DB_NAME} | gzip -c >
```

```
$backupdir/$datefolder/"${DB_NAME}-$datescript".sql.gz;

echo  "done";
echo `date +%d-%m-%y`" "`date +%T`;
```

Donner le droit d'exécution

```
sudo chmod +x save-database
```

Exécuter le script

```
./save-database
```

L'exécution du script donne la sortie suivante :

```
Répertoire de destination : /var/backup
PostgreSQL Backup for database : dbname start at  16:38:07
Exporting dbname
/usr/bin/pg_dump postgresql://postgres:dbpassword@localhost/dbname | gzip -c >
/etc/backup/dbname-03-12-21_16:38:07.sql.gz;
done
03-12-21 16:38:07
```

Exécution régulière

Afin de mettre en place une exécution régulière du script nous allons utiliser l'outil Linux appelé `Cron`. C'est un programme disponible sur les systèmes de type Unix (Linux, Mac Osx ...) permettant de planifier des taches régulières.

Ouvrir Crontab

```
crontab -e
```

Mettre le contenu suivant

```
# Exécution du script tous les jours à 2h du matin
0 2 * * * /bin/sh /script/save-database
```

La tâche `Cron` ci-dessus sera exécutée tous les jours à 2 heures du matin et lancera un script `save-database` avec l'interpréteur `bash`. Ceci maintiendra la sauvegarde chaque jour.

Restaurer une BDD

Restaurer une sauvegarde

Nous allons créer un script de restauration afin d'éviter de répéter les commandes.

Dans le même dossier que pour le script de sauvegarde nous allons créer un script **restore-database** avec en paramètre le nom du fichier de sauvegarde a utiliser.

Création du script

```
sudo nano restore-database
```

Contenu du script

```
#!/bin/bash

# # # # # # # # # # # # # # # # #
# RESTORE DATABASE by gpatruno  #
# # # # # # # # # # # # # # # # #

# Récupération de l'emplacement du fichier de sauvegarde
read -p "Emplacement du fichier de sauvegarde : " BACKUPDIR
# Récupération du nom du fichier de sauvegarde
read -p "Nom du fichier de sauvegarde : " FILENAME

# définition des variables
SCRIPTNAME="${FILENAME/.gz/""}";
DB_USER=postgres
DB_NAME=dbname
DB_PASSWORD=dbpassword

# RESTORE
echo `date +%d-%m-%y`" "`date +%T`;

# Dézip de la sauvegarde
echo "Unzip $BACKUPDIR/$FILENAME";
gunzip -c $BACKUPDIR/$FILENAME > $SCRIPTNAME
```

```
# Déconnecter tous les utilisateurs
echo "Disconnect all users from the db";
/usr/bin/psql postgresql://${DB_USER}:${DB_PASSWORD}@localhost/${DB_NAME} -c "SELECT
pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity WHERE
pg_stat_activity.datname = '$DB_NAME' AND pid <> pg_backend_pid();"

# Supprimer l'ancienne database si elle existe
echo "DROP DATABASE $DATABASE";
/usr/bin/psql postgresql://${DB_USER}:${DB_PASSWORD}@localhost -c "DROP DATABASE
\"$DB_NAME\";"

# Créer la nouvelle Base de données
echo "CREATE DATABASE $DB_NAME";
/usr/bin/psql postgresql://${DB_USER}:${DB_PASSWORD}@localhost -c "CREATE DATABASE
\"$DB_NAME\";"

# Restauration de la nouvelle BDD
echo "RESTORE DATABASE $SCRIPTNAME";
/usr/bin/psql postgresql://${DB_USER}:${DB_PASSWORD}@localhost/${DB_NAME} -f $SCRIPTNAME

echo `date +%d-%m-%y`" "`date +%T`;
```

Donner le droit d'exécution

```
sudo chmod +x restore-database
```

Puis exécuter le script et remplir les paramètres demandé au fur-à-mesures de l'avancement.

MariaDB

MariaDB

Se connecter dans MYSQL

Se connecter dans MYSQL

```
# Se connecter sans sélectionner de database
mysql -u <user> -p <password>

# Se connecter a une database sans préciser l'utilisateur
mysql -U <database> -p <password>

# Se connecter a une database avec un utilisateur en particulier
mysql -u <user> -U <database> -p
```

MySQL show database

```
SHOW DATABASES;
```

MySQL show user

```
SELECT user, host FROM mysql.user;
```

MySQL user privilège

```
SELECT * FROM information_schema.user_privileges;
```

MySQL add user privilège

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, INDEX, DROP, ALTER, CREATE TEMPORARY TABLES,
LOCK TABLES ON <db_name>.* TO '<user>'@'<host>';
```

Exemple

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, INDEX, DROP, ALTER, CREATE TEMPORARY TABLES,
LOCK TABLES ON bookstack.* TO 'userbook'@'localhost';
```

Mariadb

Faire une sauvegarde MariaDB

`mysqldump` effectue une sauvegarde logique. C'est la manière la plus flexible d'effectuer une sauvegarde et une restauration, et un bon choix lorsque la taille des données est relativement petite.

Pour les grands ensembles de données, le fichier de sauvegarde peut être volumineux, et le temps de restauration long.

mysqldump décharge les données au format SQL (il peut également décharger dans d'autres formats, tels que CSV ou XML) qui peuvent ensuite être facilement importés dans une autre base de données. Les données peuvent être importées dans d'autres versions de MariaDB, MySQL, ou même dans un autre SGBD, à condition que le dump ne contienne pas d'instructions spécifiques à la version ou au SGBD.

mysqldump vide les triggers en même temps que les tables, car ils font partie de la définition de la table. Cependant, les procédures stockées, les vues, et les événements ne le sont pas, et nécessitent des paramètres supplémentaires pour être recréés explicitement (par exemple, --routines et --events). Les procédures et les fonctions font toutefois également partie des tables du système (par exemple mysql.proc).

Sauvegarder une base de donnée

```
# Pour faire une sauvegarde de toutes les bdd mariaDB
mysqldump -u<user> -p<password> --all-databases > mariadb-dump-$(date +%F_%H-%M-%S).sql

# Pour sauvegarder une bdd en particulier
mysqldump -u<user> -p<password> --databases <database> > mariadb-dump-$(date +%F_%H-%M-%S).sql
# Une autre syntaxe équivalente
mysqldump --user="<user>" --password="<password>" --databases <database> > mariadb-dump-$(date
+%F_%H-%M-%S).sql
```