

## MISSION2

### Introduction

Dans cette mission, le but était de configurer deux serveurs, WEB et WEB2, pour qu'ils soient capables d'héberger des sites web. Pour cela, on a suivi quatre étapes : d'abord l'installation du serveur Apache2, puis celle de SQLite3, ensuite MariaDB, et enfin phpMyAdmin. Chaque étape nous a permis d'apprendre et de comprendre comment fonctionnent les serveurs web et les bases de données.

### 2.0. Utilité d'un LAMP

Voici pourquoi il est essentiel d'installer et de configurer un LAMP (Linux, Apache, MariaDB, PHP) dans le cadre de cette mission.

#### Pourquoi utiliser un LAMP ?

- **Linux** : Fournit un environnement stable et sécurisé pour héberger des applications et des sites web.
- **Apache** : Permet de servir les pages web en tant que serveur HTTP.
- **MariaDB** : Gère les bases de données nécessaires pour stocker et organiser les informations des applications.
- **PHP** : Permet d'exécuter des scripts côté serveur et de créer des pages dynamiques.

En combinant ces outils, on peut créer un serveur web complet, capable d'héberger des applications fonctionnelles, sécurisées et adaptées à des besoins variés. Un LAMP est donc une base idéale pour tout projet web professionnel ou éducatif.

### Partie 1 - Configuration du serveur WEB2

Premièrement, il fallait copier le premier template, c'est-à-dire le conteneur template, pour en créer un nouveau nommé web2. Ainsi, Apache 2 serait directement installé. Cependant, nous avons rencontré un problème, car le conteneur template démarre automatiquement. Pour que la copie fonctionne, il faut d'abord éteindre le conteneur template avec la commande suivante :

```
root@srv-g5 lxc-stop -n template
```

Une fois le conteneur arrêté, nous pouvons le copier pour en créer web2 avec la commande :

```
root@srv-g5 lxc-copy -n template -N web2
```

Ainsi, Apache 2 sera directement installé dans le conteneur web2 après configuration. Pour cela, il faut modifier l'adresse IP du serveur. La nouvelle adresse sera 10.30.80.81.

Accéder au conteneur web2 :

```
root@srv-g5 lxc-attach -n web2
```

Modifier le fichier de configuration réseau :

```
root@web2 nano /etc/network/interfaces
```

Enfin, vérifier si le serveur fonctionne en accédant à l'adresse 10.30.80.81 depuis un navigateur.



Une fois cette étape terminée, nous pouvons passer à la partie suivante.

## Partie 2 - Installation et configuration de SQLite3

Tout d'abord, il faut installer SQLite3 et les modules nécessaires pour PHP et Apache2. On peut faire ça avec la commande suivante :

Tout d'abord, il faut installer SQLite3 et les modules nécessaires pour PHP et Apache2. On peut faire ça avec la commande suivante :

```
root@srv-g5 apt update
```

```
root@srv-g5 apt install sqlite3 php-sqlite3 libapache2-mod-php
```

Cela va installer SQLite3 (le moteur de base de données SQL), le module PHP pour SQLite3 et le module PHP pour Apache2. Si PHP n'était pas déjà installé, il sera installé aussi.

Ensuite, crée un fichier PHP pour tester que PHP et les modules SQLite3 sont bien installés. Crée le fichier **info.php** dans le répertoire `/var/www/html` :

```
root@srv-g5 nano /var/www/html/info.php
```

Après avoir créé le fichier, vérifie si la librairie PDO pour SQLite3 est bien activée en accédant à l'URL suivante dans un navigateur :

```
http://10.31.81.80/info.php
```

Ensuite, on crée une base de données SQLite3 nommée **myCDS.db** dans le répertoire `/var/www/html` :

```
root@srv-g5 sqlite3 /var/www/html/myCDS.db
```

pour quitter la base de donnée utilisé cette commande

```
quit
```

Dans cette base de données **myCDS.db**, crée deux tables : **artist** et **cd**. Utilise les commandes suivantes dans SQLite3 pour créer les tables :

```
CREATE TABLE artist (art_id INTEGER PRIMARY KEY, art_name TEXT);
```

```
CREATE TABLE cd (cd_id INTEGER PRIMARY KEY, art_id INTEGER NOT NULL,
cd_title TEXT NOT NULL, cd_date TEXT);
```

Ensuite, insère des données dans les deux tables. Par exemple :

```
INSERT INTO artist (art_name) VALUES ('Peter Gabriel');
```

```
INSERT INTO artist (art_name) VALUES ('Bruce Hornsby');
```

```
INSERT INTO artist (art_name) VALUES ('Lyle Lovett');
```

```
INSERT INTO artist (art_name) VALUES ('Beach Boys');
```

```
INSERT INTO cd (art_id, cd_title, cd_date) VALUES (1, 'Us', '1992');
```

```
INSERT INTO cd (art_id, cd_title, cd_date) VALUES (2, 'The Way It Is',
'1986');
```

```
INSERT INTO cd (art_id, cd_title, cd_date) VALUES (2, 'Scenes from the
Southside', '1990');
```

```
INSERT INTO cd (art_id, cd_title, cd_date) VALUES (1, 'Security', '1990');
```

```
INSERT INTO cd (art_id, cd_title, cd_date) VALUES (3, 'Joshua Judges Ruth',
'1992');
```

```
INSERT INTO cd (art_id, cd_title, cd_date) VALUES (4, 'Pet Sounds', '1966');
```

Et ainsi la base **myCDS.db** est créée avec les tables nécessaires et inséré des données dedans et ne



Connecté à la base

Les artistes

1. Peter Gabriel
 

Albums de Peter Gabriel :

  1. Us (1992)
  4. Security (1990)
2. Bruce Hornsby
 

Albums de Bruce Hornsby :

  2. The Way It Is (1986)
  3. Scenes from the Southside (1990)
3. Lyle Lovett
 

Albums de Lyle Lovett :

  5. Joshua Judges Ruth (1992)
4. Beach Boys
 

Albums de Beach Boys :

  6. Pet Sounds (1966)

pas oublier de reeboot .

### Partie 3 - Installation et configuration de MariaDB

Tout d'abord, il faut installer MariaDB et le module PHP pour MySQL. Utilise la commande suivante pour installer tout ça :

```
root@srv-g5 apt update
```

```
root@srv-g5 apt install mariadb-server php-mysql
```

Cela va installer MariaDB (SGBD MariaDB) et le module PHP pour MySQL (et MariaDB). Si PHP n'était pas déjà installé, il sera installé aussi.

Si Apache ne fonctionne pas, le redémarrer en le reboot :

Ensuite, on vérifie que le module PDO pour MySQL est bien activé. Pour cela, accède à la page `info.php` dans le navigateur à l'adresse suivante

```
http://10.31.81.80/info.php
```

Vérifie ensuite que MariaDB fonctionne en exécutant la commande suivante :

```
root@srv-g5 systemctl status mariadb
```

Afin de finaliser l'installation de MariaDB, on lance le script **mysql\_secure\_installation** pour choisir un mot de passe pour l'utilisateur **root** de MariaDB. Utilise la commande suivante :

```
root@srv-g5 mariadb-secure-installation
```

Voici les principales auquel on devras répondre :

- **Configure the VALIDATE PASSWORD PLUGIN ?**
- **Set root password?**
- **Remove anonymous users?**
- **Disallow root login remotely?**
- **Remove test database and access to it?**
- **Reload privilege tables now?**

Après avoir répondu à ces questions, on auras un **mot de passe root** de MariaDB.

Ensuite, se connecter à MariaDB avec le client en ligne de commande :

```
root@srv-g5 mysql -u root -p
```

Entrez le mot de passe root que tu viens de définir.

Pour créer un compte **dba** conformément au cahier des charges, utiliser les commandes suivantes dans MariaDB :

```
MariaDB [(none)]> create user 'dba'@'localhost' identified by 'drowssap';
```

```
MariaDB [(none)]> grant all privileges on *.* to 'dba'@'localhost' with
```

```
grant option;
```

```
MariaDB [(none)]> flush privileges;
```

Ainsi, le compte **dba** est créé et a tous les privilèges nécessaires.

## Partie 4 - Installation et configuration de phpMyAdmin

Premièrement, il fallait installer les modules PHP nécessaires pour que phpMyAdmin fonctionne correctement. Voici la commande utilisée :

```
root@web2 apt install php-json php-mbstring php-zip php-gd php-xml php-curl
```

Une fois les modules installés, Apache 2 doit être redémarré pour appliquer les modifications.

Ensuite, nous avons téléchargé la dernière version de phpMyAdmin depuis le site officiel à l'aide de `wget`. Pour cela, il fallait d'abord s'assurer que `wget` était installé :

```
root@web2 apt update && apt install wget
```

Une fois `wget` installé, voici la commande utilisée pour télécharger phpMyAdmin :

```
root@web2 wget  
https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-all-languages  
.tar.gz
```

Après le téléchargement, nous avons déplacé l'archive dans le répertoire `/var/www/html/` :

```
root@web2 mv phpMyAdmin-5.2.1-all-languages.tar.gz /var/www/html/
```

Ensuite, l'archive a été extraite dans ce répertoire :

```
root@web2 tar xzvf /var/www/html/phpMyAdmin-5.2.1-all-languages.tar.gz
```

Le dossier extrait a été renommé en **phpmyadmin** pour simplifier l'accès :

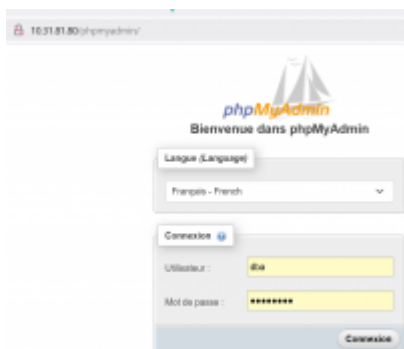
```
root@web2 mv /var/www/html/phpMyAdmin-5.2.1-all-languages  
/var/www/html/phpmyadmin
```

Enfin, nous avons donné la propriété du dossier et de tout son contenu à l'utilisateur `www-data` pour permettre à Apache de fonctionner correctement :

```
root@web2 chown -R www-data:www-data /var/www/html/phpmyadmin
```

À cette étape, phpMyAdmin est installé et accessible via le navigateur à l'adresse suivante :

```
http://10.31.81.80/phpmyadmin
```



Pour vérifier, nous nous sommes connectés avec le compte **dba** et avons créé une base de données MariaDB, identique à celle utilisée pour SQLite3.

### Création de la base de données via MariaDB

Pour créer la base de données et y insérer des données directement en ligne de commande :

Se connecter à MariaDB :

```
root@web2 mysql -u dba -p
```

Créer une base de données :

### Création d'un script PHP pour tester la base de données

Nous avons créé un script PHP pour vérifier la connexion à la base de données. Voici comment faire :

Créer un fichier PHP dans `/var/www/html/` :

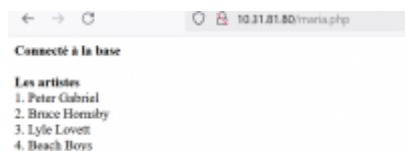
```
root@web2 nano /var/www/html/test_mariadb.php
```

Sauvegarder et quitter. Enfin, accéder au script depuis un navigateur :\\

<code>

```
http://10.31.81.80/test_mariadb.php
```

Si tout est configuré correctement, les données insérées dans la table devraient s'afficher.



Une fois ces tests réalisés, phpMyAdmin et MariaDB fonctionnent parfaitement.

From:

<https://sisr2.beaupeyrat.com/> - **Documentations SIO2 option SISR**

Permanent link:

<https://sisr2.beaupeyrat.com/doku.php?id=sisr1-g5:lamp>

Last update: **2025/12/05 11:16**

