Activité 2 : Déploiement d'un site Web dynamique

Le lycée dans lequel vous évoluez met à disposition des élèves et des professeurs de nombreux services dont des services applicatifs avec de nombreuses applications Web qui nécessitent l'accès à des bases de données relationnelles. **L'idée générale est de migrer l'ensemble de ces applications, actuellement en production sur des machines virtuelles, dans des conteneurs Docker.** Les premiers tests sont réalisés avec l'application « geststage » développée en PHP/MySQL.

L'objectif de cette activité est donc de déployer via un conteneur Docker un site web dynamique intégrant des pages PHP en liaison avec une base de données MySQL/MariaDB. Nous allons, pour cela :

1. **faire évoluer l'image construite dans l'activité 1** en y ajoutant les outils nécessaires à ce service web dynamique à savoir PHP ;
2. **utiliser une nouvelle image** pour le serveur de base de données ;
3. **gérer la problématique de la persistance des données** en ce qui concerne les pages Web de l'application et la base de données ;
4. **lier les deux conteneurs**.

Nous aurions pu utiliser un seul conteneur avec tous les outils *LAMP* mais Docker recommande de séparer chaque service dans un conteneur différent. Il sera ainsi possible de rendre l'environnement plus modulaire en permettant de tester plusieurs versions de service Web, de PHP et/ou d'une base de données (MariaDB et MySQL par exemple).

Nous choisirons donc d'installer le serveur web (Apache et PHP) séparément du serveur de base de données, dans 2 conteneurs différents. Il aurait été bien sûr possible de créer un troisième conteneur pour PHP (avec possibilité dans ce cas de tester plus facilement différentes versions du service Web et de PHP).

La problématique de la persistance des données

Les images servant de base au démarrage d'une instance de conteneur sont en lecture seule et toutes les modifications sur le conteneur sont réalisées dans une couche supplémentaire. Ainsi, les données ajoutées dans un conteneur disparaissent avec lui lors de sa destruction. Pour sauvegarder les modifications dans une autre image, il est nécessaire d'utiliser la commande *commit*.

Cette méthode est adaptée lorsque l'utilisateur est en phase de construction d'une image ou pour déployer un micro-service ne gérant pas de données, mais ne l'est pas pour l'exploitation d'une image existante quand il s'agit, par exemple, d'utiliser le conteneur pour un service en ligne où les données sont modifiées sans cesse ou bien s'il est important de conserver certaines données comme les traces (*logs*).

Dans notre cas, chaque site Web dynamique nécessite à minima :

- un espace de stockage des fichiers Web du site ;
- un espace de stockage pour la base de données.

Docker a prévu plusieurs mécanismes pour gérer les données persistantes, nous mettrons en pratique celui le plus couramment utilisé et surtout celui adapté à la situation qui consiste **à utiliser un stockage externe** au conteneur (sur l'hôte ou sur n'importe quel autre support, typiquement des baies SAN ou NAS.) appelé **volume de données** (plutôt que le stockage dans le système de fichiers par défaut du conteneur) pour les pages Web du site.

Nous travaillerons à partir de l'image sauvegardée dans l'activité précédente et nous utiliserons pour nos tests, l'application Web de gestion des stages fournie sous forme de fichier « zip » comprenant les fichiers sources de l'application et le script SQL de création et d'alimentation de la base de données MySQL.

Vous disposez également de la documentation suivante :

- **document 1** : Gestion de la persistance des données avec les volumes ;
- **document 2** : Utilisation d'un serveur de base de données conteneurisé.

Travail à faire

Avant de continuer, il est conseillé de réaliser toutes les manipulations détaillées dans les documents. *Vous pourrez ensuite supprimer tous les conteneurs créés et toutes les bases de données (en supprimant les volumes correspondants).*

Travail à faire 1 Installation de l'application Web

- Démarrez le conteneur en interactif avec un shell en lui donnant le nom « servWeb ».

- Mettez à jour les paquets du conteneur (ar/debian:stretch-apache2).

Notre site contient d'une part des pages PHP -notre serveur Web doit être capable de les interpréter - et utilise d'autre part une base de données MySQL/MariaDB. Vous installerez PHP dans le conteneur comprenant le service *web* Apache2 mais le service de base de données sera exploité dans un conteneur dédié.

- Installez les paquets php7.0 et php7.0-mysql.

- Créez une nouvelle image (vos_initiales/debian:stretch-apache2-php7) à partir du conteneur modifié.

 **C'est cette nouvelle image que nous allons utiliser.**

Nous devons maintenant déposer à la racine du serveur Web les fichiers sources de l'application que nous voulons mettre en ligne. En cas d'évolution de l'application ou de correction de *bug*, ces fichiers sources peuvent être modifiés. Pour ne pas à avoir à modifier continuellement l'image, **il est nécessaire de déposer ces fichiers dans un dossier du serveur hôte.**

- Créez sur l'hôte le dossier /var/www/html_docker et déposez les fichiers sources de l'application dedans.

- Lancez un conteneur (servweb) à partir de cette nouvelle image permettant d'accéder à l'application via l'URL **http://adresseIPDocker:8001/geststages/.** Vous devriez obtenir l'écran suivant :

Travail à faire 2 Liaison de l'application avec la base de données

Il est maintenant nécessaire de **lier** cette application web à la base de données qu'elle exploite. Le système de gestion de base de données MariaDB (*fork* libre de MySQL) sera dans son propre conteneur et **l'accès à la base de données ne sera possible qu'à partir de l'application intégrée au conteneur apache/PHP.**

Les bases de données seront externalisées dans un volume sur l'hôte **/var/lib/mysql_docker**.

Les deux premières manipulations demandées ne servent qu'au test du bon fonctionnement du conteneur en exécution.

Téléchargez et lancez le conteneur *mariadb* avec un mot de passe par défaut pour *root* à « mysqlAdmin » et un nom de conteneur « servbd ». Le port 3306 de la machine host *debian* sera redirigé vers ce conteneur en cours d'exécution.

À partir d'une machine cliente (Ubuntu de votre collègue à côté par exemple), testez l'accès à la base de données.

L'objectif étant de connecter le site web du conteneur apache/PHP à la base de données, il n'est pas utile de rendre visible la base de données aux clients en dehors de la machine hôte ou d'un autre conteneur docker contenant une application destinée justement à exploiter cette base de données. Par défaut, lorsqu'on démarre un conteneur à partir de l'image mariadb, le port 3306 est ouvert (il est « exposé »). if suffit de créer un lien (*–link*) entre les deux conteneurs qui utilise par défaut ce port exposé (voir document 2.2).

Stoppez et supprimez le conteneur *servbd* précédemment créé. Créez un nouveau conteneur *servbd* **sans redirection de port** et testez l'accès à partir d'une autre instance de l'image *mariadb.*

Le script d'initialisation de la base de données est fourni : *geststages.sql*. Il inclut tous les éléments nécessaires (création de la base de données, de l'utilisateur ayant les droits sur la base de données, des tables, des contraintes ainsi que les commandes SQL d'insertion des données).

On constate que l'application Web a accès à la base de données « bdd_geststages » avec l'utilisateur « usergs » qui a pour mot de passe « mdpGS ».

Utilisez une instance de l'image *mariadb* pour initialiser la base de données. *Le script « geststages.sql » sera déposé dans le dossier « /mnt/scripts » de l'hôte (dossier mappé sur le dossier « scripts » du conteneur)*

Au niveau de l'application, l'accès à la base de données est configuré dans « include/fonction/f_bdd.php ».

Extrait du fichier **/var/www/html_docker/geststages/include/function/f_bdd.php :**

```
...
        $bdd = new PDO('mysql:host=servbd;port=3306;dbname=bdd_geststages;charset=utf8', 'usergs', 'mdpGS', array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
...
```

Ainsi, l'application web a accès à la base de données « bdd_geststages » avec l'utilisateur « usergs » qui a pour mot de passe « mdpGS ». Ce sont ces éléments qui sont créés lors de l'importation du fichier SQL.
Par ailleurs, l'accès à la base de données se fait sur l'hôte « servbd » ⇒ Si vous avez suivi les consignes, **vous avez théoriquement nommé le conteneur d'origine avec ce nom sinon vous devez utiliser ce nom en alias dans la commande de lancement du conteneur.**

- Stoppez et supprimez le conteneur *servweb* déployé en fin de première partie de cette activité.

- Re-lancez un nouveau conteneur *web* lié au conteneur accueillant la base de données.

- Testez l'application Web.

Pour l'authentification sur l'application, les *login* et les mots de passe sont volontairement en clair dans la base de données (tables "etudiant" et "professeur"). Vous pouvez tester la connexion avec l'étudiant « benpas01 » qui a pour mot de passe « benpas01 ».

http://192.168.0.120:8001/geststages/

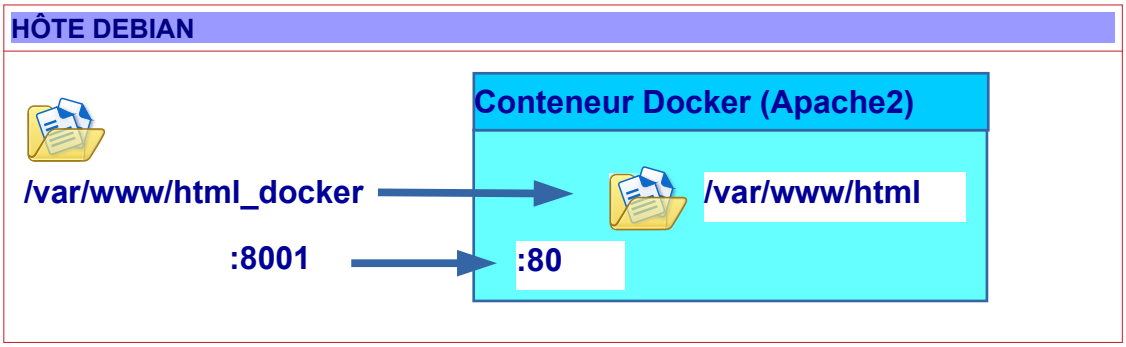
Les deux conteneurs sont maintenant déployés sur la machine de production.

Document 1 – Gestion de la persistance des données avec les volumes

Un volume est un répertoire défini pour y stocker les données soit utilisables par un conteneur (via un mappage) soit issues d'un conteneur sachant que ce stockage persistera même lorsque le conteneur sera supprimé.

L'idée consiste ici à monter une arborescence de fichiers de l'hôte à l'intérieur du container. Il est ainsi possible de publier un site Web dont les fichiers sources sont déposés dans un dossier du serveur hôte à travers un conteneur. Docker utilise pour cela le **paramètre -v** :

```
docker run --name servweb -d -v /var/www/html_docker:/var/www/htm -p 192.168.0.120:8001:80  ar/debian:stretch-apache2 /usr/sbin/apache2ctl  -DFOREGROUND
```



Docker initialise le volume de données et crée le dossier tant sur l'hôte que dans le conteneur si ce dernier n'existe pas. **Au moment de l'initialisation**, si des fichiers sont présents dans le dossier du conteneur, ceux-ci sont copiés dans le dossier de l'hôte.

Il suffit donc maintenant de déposer dans **/var/www/htm_docker** de l'hôte un fichier *index.html* personnalisé afin de le distinguer de la page par défaut d'Apache et de vérifier que c'est bien cette page qui est publiée :

http://adresseIPDocker:8001

Il est possible de multiplier le paramètre -v autant de fois que nécessaire et d'utiliser ce ou ces volumes à partir d'autres conteneurs (opération que nous ne verrons pas dans cette activité).

Il est aussi intéressant de pouvoir externaliser dans un répertoire de l'hôte des données du conteneur afin de pouvoir les conserver ou de les partager avec d'autres conteneurs. Par exemple, les données contenues dans le répertoire **/var/log/apache2** du conteneur seront externalisées et ne seront plus stockées dans le conteneur lui-même.

```
docker run --name servweb -d -v /var/www/html_docker:/var/www/html -v /var/log/apache2 -p 192.168.0.120:8001:80  ar/debian:stretch-apache2 /usr/sbin/apache2ctl  -DFOREGROUND
```

Le contenu de /var/log/apache2 est copié dans **/var/lib/docker/volumes/id_du_conteneur/_data** et les nouvelles données seront écrites directement dans ce volume.

Ce volume de données initialisé à la création du conteneur est persistant même si le conteneur est supprimé. Docker ne supprimera jamais automatiquement un volume. Pour effacer des volumes qui ne sont liés à aucun conteneur : **docker volume rm $(docker volume ls -qf dangling=true)**

Document 2 - Utilisation d'un serveur de base de données conteneurisé

2.1 – Téléchargement de l'image et création du conteneur

L'idée est ici de lancer un conteneur *mariadb* qui externalise ses bases dans **/var/lib/mysql_docker/** et de le lier à un ou plusieurs autres conteneurs applicatifs destinés à exploiter les données.

Pour un conteneur MariaDB/MySQL, le volume de données à exposer correspond au dossier **/var/lib/mysql** du conteneur.

Nous utiliserons l'image *mariadb* officielle. Celle-ci « expose » par défaut le port standard 3306 et permet l'accès au serveur de bases de données à partir d'un hôte distant. Le mot de passe que l'on veut utiliser pour *root* peut être passé au conteneur via la variable d'environnement MYSQL_ROOT_PASSWORD.

Le lien suivant https://hub.docker.com/_/mariadb/ donne toutes les indications d'utilisation du conteneur.

Les noms de certaines variables donnés dans ce lien ne sont plus bons. Pour connaître les variables utilisables dans un conteneur : **docker run <id_conteneur> env**

1. Téléchargement de la dernière version du docker *mariadb* officiel

```
docker pull mariadb
```

Cette étape peut être combinée avec la suivante puisque l'opération *run* télécharge l'image si celle-ci n'est pas trouvée en local.

2. Lancement du conteneur que nous nommerons servbd avec :
   - un mot de passe pour root initialisé à « mysqlAdmin » ;
   - un volume local : /var/lib/mysql_docker.

```
docker run -v /var/lib/mysql_docker :/var/lib/mysql -e MYSQL_ROOT_PASSWORD=mysqlAdmin --name servbd -d mariadb
```

C'est **l'option -e** qui permet de définir les variables nécessaires : ici le mot de passe pour *root* (administrateur du système de gestion des bases de données) qui va permettre par exemple d'administrer le serveur en ligne de commande (voir 2.2) ou via une application comme phpMyAdmin (voir 2.3).

Sur l'hôte, on constate que les éléments présents dans /var/lib/mysql ont bien été copiés dans /var/lib/mysql_docker (qui a été créé automatiquement par *docker*) : **ls /var/lib/mysql_docker/**

```
aria_log.00000001 ibdata1 ibtmp1 performance_schema aria_log_control  ib_logfile0  multi-master.info tc.log ib_buffer_pool ib_logfile1  mysql
```

2.2 – Connexion à MariaDB à partir du client en ligne de commande MariaDB via une nouvelle instance de l'image en liaison avec le conteneur d'origine (servbd).

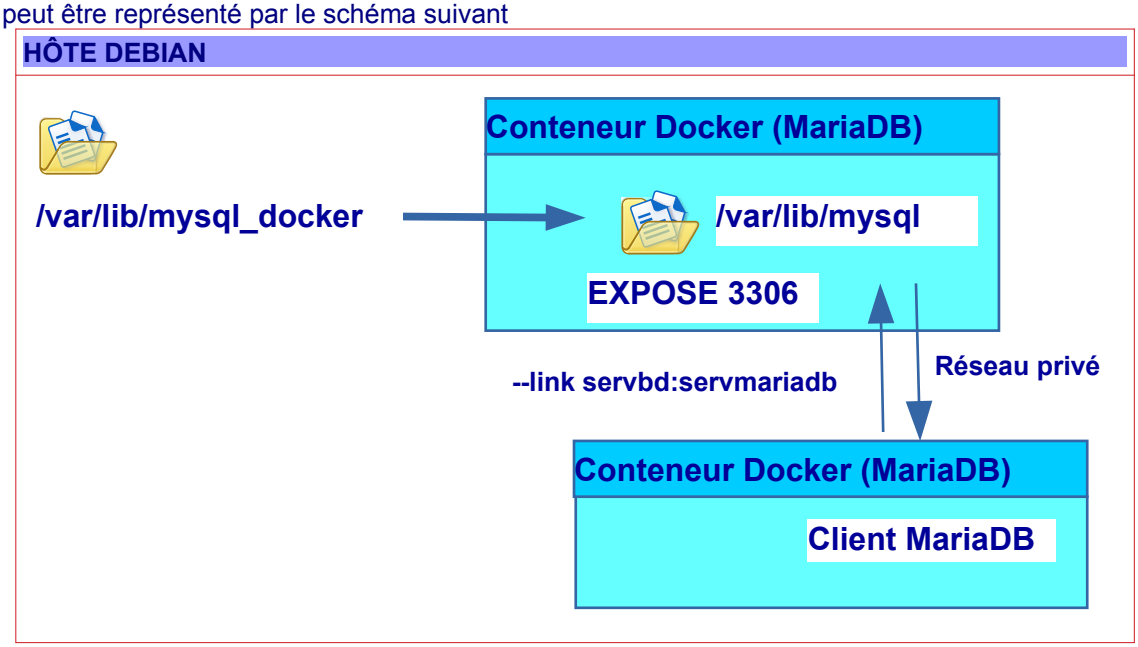
Il est rappelé que cette image a été conçue pour exposer le port MySQL standard (3306) et permettre la connexion à partir d'un hôte distant via notamment la redirection de ports entre l'hôte et le conteneur défini par l'option -p.

Valeur de « EXPOSE » et liaisons entre conteneurs

Une des bonnes pratiques de Docker est d'avoir un conteneur par service. Ceux-ci ont donc besoin de communiquer entre eux. L'opérateur *link* permet à un conteneur d'avoir accès **directement** au service d'un autre conteneur via un port sur un réseau privé défini par la valeur de EXPOSE sur le conteneur cible (ici 3306) ; port qu'il n'est alors pas utile de mapper s'il n'y a aucune raison de rendre la base de données accessible en dehors de docker.

À noter également que Docker va modifier le fichier /etc/hosts. Ainsi, on pourra accéder à ce container via son nom.

Ceci peut être représenté par le schéma suivant



La commande à utiliser est la suivante :

```
docker run -it --rm --link servbd:servmariadb mariadb bash -c 'exec mysql -h "$SERVMARIADB_PORT_3306_TCP_ADDR" -u root -pmysqlAdmin'
```

```
Your MariaDB connection id is 4
Server version: 10.3.8-MariaDB-1:10.3.8+maria~bionic...
...
MariaDB [(none)]>create database testdocker;
Query OK, 1 row affected (0.001 sec)
```

La commande démarre un nouveau conteneur *mariadb* en le liant au conteneur d'origine. C'est **l'opérateur --link (ou -l)** qui permet de faire cela en prenant en argument **le nom_du_conteneur_d_origine:alias_de_ce_conteneur** où alias_de_ce_conteneur est un nom d'alias libre qui peut être le même que celui du conteneur d'origine (voir encadré ci-après).

La nouvelle base a été créée dans /var/lib/mysql_docker : **ls /var/lib/mysql_docker/**

```
aria_log.00000001  ibdata1 ibtmp1 performance_schema aria_log_control  ib_logfile0 multi-master.info tc.log ib_buffer_pool ib_logfile1 mysql testdocker
```

Nous remarquons également le paramètre --rm qui supprime le conteneur dès que la commande a été exécutée (typiquement ici dès qu'on quitte le client mariadb) : ce conteneur ne sert qu'à lancer un client en ligne de commande ; dès que lon a terminé nos commandes sur le serveur de base de données le conteneur ne doit plus être actif.

Il est rappelé que du moment que l'on a géré la persistance des données, si l'on supprime le conteneur serveur « servbd » et qu'on lance des nouvelles instances (une pour le serveur et l'autre pour l'accès client), nous pouvons voir que la base de données créée précédemment existe toujours :

docker rm servbd

docker run **-v /var/lib/mysql_docker :/var/lib/mysql** -e MYSQL_ROOT_PASSWORD=mysqlAdmin --name servbd -d mariadb

docker run -it --rm --link servbd:servmariadb mariadb bash -c 'exec mysql -h "$SERVMARIADB_PORT_3306_TCP_ADDR" -u root -pmysqlAdmin'

```
...
MariaDB [(none)]>show databases;
..
testdocker
...
```

Pour connaître les noms de variables et leurs contenus utilisées par le conteneur d'origine :
docker run --link servbd:mariadb mariadb env
Nous voyons qu'on aurait pu utiliser la variable "MARIADB_ENV_MYSQL_ROOT_PASSWORD" en lieu et place du mot de passe.

Pour pouvoir utiliser un script (pour par exemple initialiser une base), il est nécessaire de pouvoir y accéder à travers la machine hôte, ce qui est possible via les volumes :

1. Créez le dossier /mnt/scripts
2. Déplacez le script de création et d'alimentation de votre base (par exemple intbase.sql) de données dans ce dossier (on suppose ici que le script contient la commande de création de la base)
3. Utilisez la commande suivante :

```
docker run -it --rm --link servbd:servmariadb -v /mnt/scripts:/scripts mariadb bash -c 'exec mysql -h "$SERVMARIADB_PORT_3306_TCP_ADDR" -u root -pmysqlAdmin < /scripts/'initbase.sql'
```

Quid des alias

Docker lance le conteneur en ajoutant dans le fichier *hosts* une entrée pour le conteneur passé en paramètres de --link (que l'on peut consulter si on lance le conteneur en interactif avec une console) :
docker run -it --rm --link servbd:servmariadb mariadb bash
root@934340604d93:/# cat /etc/hosts
127.0.0.1 localhost
...
172.17.0.3 servmariadb 9ec3e25d6cee servbd
172.17.0.6 934340604d93

Il est donc possible d'utiliser l'alias dans une configuration. De même, les variables d'environnement pour désigner ces conteneurs sont automatiquement créées dans le conteneur appelant –link, pour pouvoir être exploitées dans des scripts par exemple :
root@934340604d93:/# env

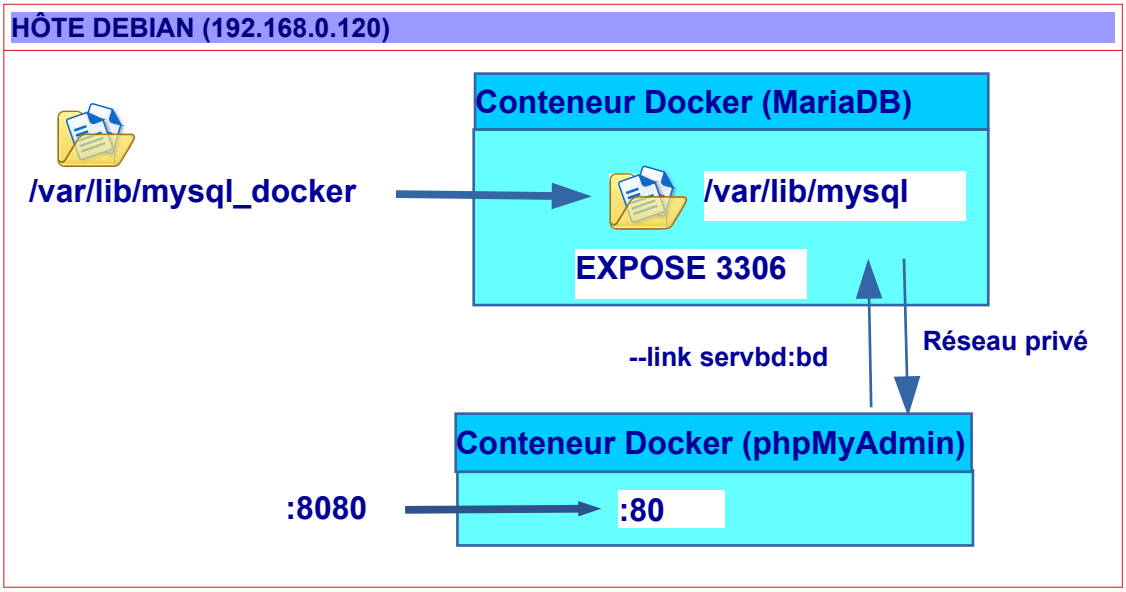
SERVMARIADB_PORT_3306_TCP_PORT=3306
SERVMARIADB_PORT=tcp://172.17.0.3:3306
HOME=/root
SERVMARIADB_ENV_MYSQL_ROOT_PASSWORD=mysqlAdmin

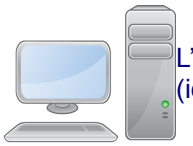
2.3 - Connexion à MySQL à partir d'une application tournant dans un autre conteneur

C'est le même principe que précédemment. Il faut maintenant démarrer le conteneur d'application en le liant au conteneur MariaDB (préalablement lancé) selon le schéma suivant :

```
docker run --name <nom_conteneur> -d --link <nom_conteneur_lié>:<alias> [autres options] nom_conteneur_application_qui_utilise_mariadb
```

Par exemple, utilisation d'un conteneur « phpmyadmin » :
docker run --name pma -d --link servbd:bd -p 8080:80 -e PMA_HOST=servbd phpmyadmin/phpmyadmin

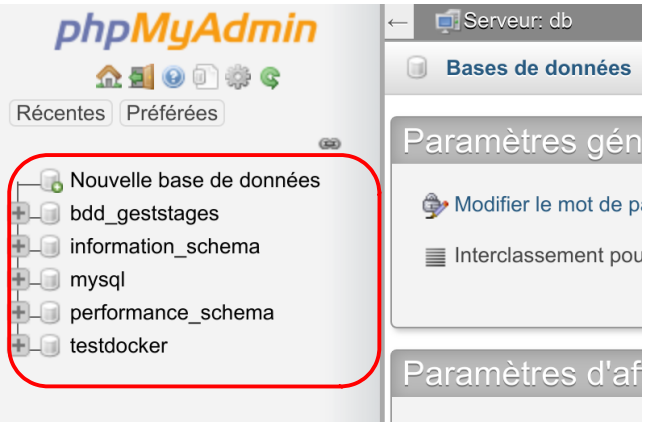




L'accès se fait via l'URL suivante : http://adresseIPhote:8080
(ici http://192.168.0.120:8080)

L'interface de *phpMyAdmin* devrait apparaître, il suffit de se connecter avec l'utilisateur *root* et le mot de passe défini lors de la création du conteneur (mysqlAdmin ici).

Bases de données présentes dans le dossier /var/lib/mysql_docker de l'hôte.



2.4 - Connexion à MySQL via un port mappé sur l'hôte

Comme nous l'avons vu dans l'activité précédente, il est possible de lancer le conteneur en mappant un port (port 3306 par exemple) de la machine hôte *debian* qui sera redirigé vers ce conteneur en cours d'exécution.

```
docker run -p 3306:3306 -v /var/lib/mysql_docker:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=mysqlAdmin --name servbd -d mariadb
```

Il est alors possible de se connecter au serveur de base de données de manière classique avec un client MariaDB/MySQL installé sur la machine hôte ou sur un autre poste.

$mysql -h 192.168.0.120 -u root -pmysqlAdmin
Your MariaDB connection id is 8
Server version: 10.3.8-MariaDB-1:10.3.8+maria~bionic...
...
MariaDB [(none)]>

À noter qu'il est également possible de se connecter **à partir d'un client MariaDB/MySQL de l'hôte** sans faire un mappage de port si l'on connaît l'adresse IP du conteneur. Pour connaître cette dernière : **$docker inspect -f "{{.NetworkSettings.IPAddress}}" servbd** ⇒ 172.17.0.4.
$mysql -h 172.17.0.4 -u root -pmysqlAdmin
Your MariaDB connection id is 8
Server version: 10.3.8-MariaDB-1:10.3.8+maria~bionic...
...
MariaDB [(none)]>