

# TP #3

## Etape 0

Stopper et supprimer tous les containers existants

Créer un répertoire **docker-tp3**

Chaque étape doit être dans un sous-répertoire nommé "etapeX" (X étant le numéro de l'étape)  
Il peut être utile au fur et à mesure des étapes d'organiser les fichiers dans des sous-répertoires (config, src, ... par exemple).

Comme dans les exemples fournis, créer un fichier `launch.sh` qui contient les commandes spécifiques à chaque étape, dans le répertoire de l'étape en question.  
Penser aussi à supprimer les containers entre chaque étape pour éviter les conflits à cause des noms ou des ports utilisés.

Initialiser un repository git dans ce répertoire.

Le TP sera à rendre sous la forme d'un repository github dont tu me donneras le lien (et les droits de consultation ;)

Si tu n'es pas très à l'aise avec git, voici un tuto assez simple pour apprendre ou avoir un rappel des bases : <https://www.ionos.fr/digitalguide/sites-internet/developpement-web/tutoriel-git/>

**Les questions 1 à 2 sont à réaliser sans Docker Compose**  
**Il est interdit d'utiliser Docker Compose avant l'étape 3**

## Etape 1

2 containers nommés comme suit :

- HTTP : 1 container avec un serveur HTTP qui écoute sur le port 8080
- SCRIPT : 1 container avec un [interpréteur PHP](#) (plus le [protocole FPM pour NGINX](#))

Une page `index.php` qui lorsqu'elle est appelée exécute la fonction `phpinfo()` et qui sera situé dans les containers dans le répertoire `/app`.

Test de validité de l'exercice : avec un navigateur voir le résultat de l'exécution du `php_info()`

Pour réaliser cette étape, tu auras besoin :

- d'organiser dans un répertoire de ton hôte Docker les différents fichiers dont tu auras besoin. Les sources du site (1 fichier php : `index.php`) sont fournies dans une archive qui accompagne ce TP. Les sources du site seront à ajouter aux 2 containers grâce à des volumes bind mount.
- de lancer des containers avec `docker container run`, un `nginx` et un `php`

- de reconfigurer le container `nginx` pour aller communiquer avec le container `php` comme dans les TP précédents avec des volumes bind mount sur le fichier de configuration de `nginx`. Les modifications à effectuer sur le fichier `default.conf` sont plus bas.
- de mettre les containers en communication via un réseau commun (pas le réseau par défaut) avec `docker network create` et aussi l'option `--network` de `docker container run`

Décommenter et remplacer les lignes 30 à 36 du fichier `default.conf` de Nginx par les suivantes :

```
location ~ \.php$ {
    root           /app;
    fastcgi_pass   script:9000;
    fastcgi_index  index.php;
    fastcgi_param  SCRIPT_FILENAME
$document_root$fastcgi_script_name;
    include        fastcgi_params;
}
```


Test de validité de cette étape : en consultant la page <http://localhost:8080/> avec un navigateur, vous devez voir ça (la version de PHP peut être différente) :

PHP 7.4.33 - phpinfo()

×

+

PHP Version 7.4.33



System	Linux d1918acd6f32 6.9.3-76060903-generic #202405300957~1736980680~22.04~44ea8a9 SMP PREEMPT_DYNAMIC Thu J x86_64
Build Date	Nov 15 2022 06:05:32
Configure Command	./configure '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20190902
PHP Extension	20190902

## Etape 2

3 containers nommés comme suit :

- HTTP : 1 container avec un serveur HTTP qui écoute sur le port 8080

- SCRIPT : 1 container avec un interpréteur PHP (plus le protocole FPM pour NGINX)
- DATA : 1 container avec un serveur de base données SQL (MariaDB)

Une page `test.php` qui lorsqu'elle est appelée va exécuter 2 requêtes CRUD (Request : lecture, Create Update Delete : écriture) sur le serveur SQL : 1 écriture et 1 lecture.


Pour réaliser cette étape, tu auras besoin :

- d'avoir fait l'étape 1
- du fichier `test.php` et du fichier `create.sql` qui sont fournis dans une archive qui accompagne ce TP.
- de lancer un container supplémentaire basé sur l'image `mariadb`.
- d'initialiser ce container de base de données grâce au répertoire `/docker-entrypoint-initdb.d` (voir le paragraphe "Initializing the database contents" sur la page de l'[image MariaDB](#))
- définir une variable d'environnement pour l'image MariaDB : `MARIADB_RANDOM_ROOT_PASSWORD`
- d'ajouter l'extension `mysqli` dans le container `php` (voir le paragraphe "How to install more PHP extensions" de la page de l'[image PHP](#)) grâce à un `Dockerfile`.

Test de validité de cette étape : en consultant la page <http://localhost:8080/> avec un navigateur, on doit voir ça (la version de PHP peut être différente) :

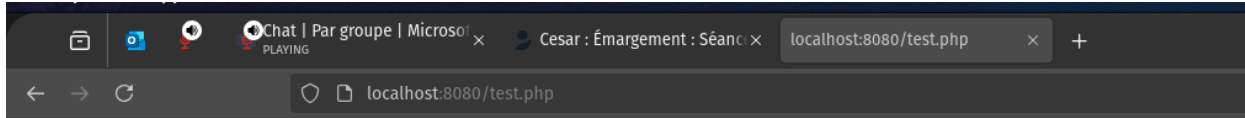
PHP 7.4.33 - phpinfo()

PHP Version 7.4.33



System	Linux d1918acd6f32 6.9.3-76060903-generic #202405300957~1736980680~22.04~44ea8a9 SMP PREEMPT_DYNAMIC Thu J x86_64
Build Date	Nov 15 2022 06:05:32
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20190902
PHP Extension	20190902

Et en consultant cet autre lien <http://localhost:8080/test.php> avec un navigateur, on doit voir ça, avec le compteur qui s'incrémente à chaque rafraîchissement de la page :



# Count updated

## Count : 6

### Etape 3

Convertir la configuration de l'étape 2 en Docker Compose

Test de validité de l'exercice : identique à l'étape 2