

ACTIVITES DOCKER

Activité 1 :

1. Créer un conteneur ubuntu 18.04 avec apache, mysql, php et phpmyadmin en utilisant l'image : **mattnayner/lamp**

On veillera à mapper les ports 80 et 3306 et créer un volume app.

Pour cela on saisit: `docker run -p 80:80 -p 3306:3306 -v ${PWD}/app:/app mattnayner/lamp:latest-1804`

2. Lancer votre navigateur préféré en localhost : <http://localhost>

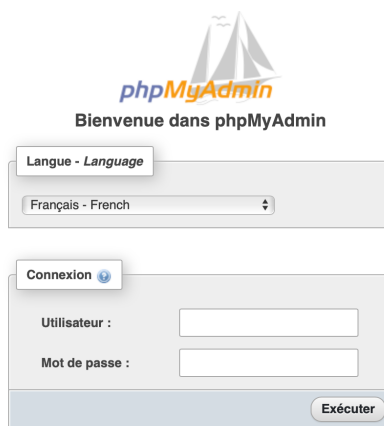
Index of /

[Name](#) [Last modified](#) [Size](#) [Description](#)

Apache/2.4.29 (Ubuntu) Server at localhost Port 80

La page précédente s'affiche, cela permet de vérifier que le serveur Apache fonctionne. Nous n'avons pas encore de page web sur le serveur.

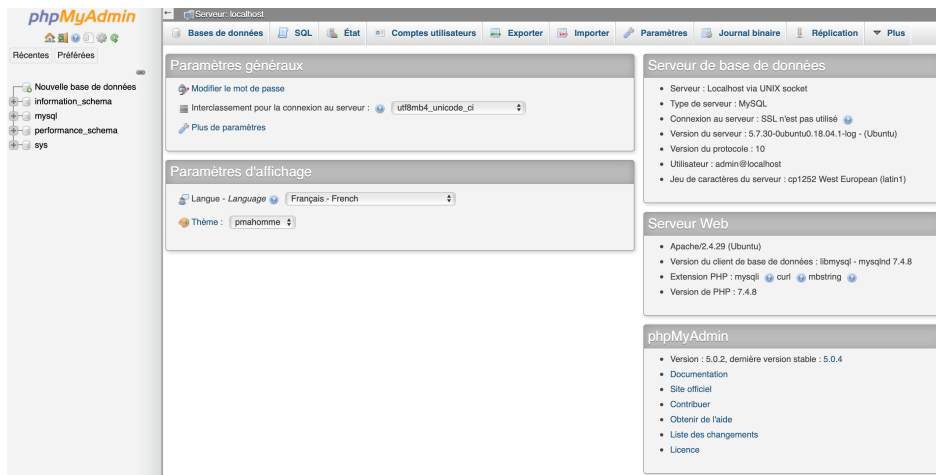
3. Saisissez l'url : <http://localhost/phpmyadmin/> dans votre navigateur.



On constate que phpmyadmin est correctement installé.

4. Saisissez Utilisateur : admin et mot de passe : transmis lors de la saisie du docker run.

On obtient la page suivante :



5. A l'aide de visual studio code, créer un fichier nommé test.php et enregistrez le dans votre répertoire `${PWD}/app`. En effet, nous avons créé précédemment un volume app que l'on a lié avec le répertoire /app du conteneur en saisissant : `v ${PWD}/app:/app`

Contenu du fichier test.php :

```
<?php
Echo « page internet de Nom Prénom dans un conteneur »
?>
```

6. Tester ensuite le site internet en saisissant : <http://localhost/test.php>
7. Relever l'adresse ip de votre pc et faites une requête vers votre site depuis le PC de votre voisin. Que constatez-vous ? Cette IP est-elle celle de votre conteneur ? expliquer.
8. Dans une autre fenêtre terminale créer à nouveau un conteneur avec la commande :

```
docker run -p 80:80 -p 3306:3306 -v ${PWD}/app2:/app
mattrayner/lamp:latest-1804
```

9. Que constatez-vous ?
10. Modifier la commande précédente comme ci-dessous :

```
docker run -p 81:80 -p 3307:3306 -v ${PWD}/app2:/app
mattrayner/lamp:latest-1804
```

11. Tester la connexion depuis le PC du voisin. Que constatez-vous ?

SCENARIO DU TP

Avant de commencer : Pourquoi Docker ?

Planche 0

Deux façons de lancer un serveur

Machine Virtuelle (VM)

- App LAMP
- OS Complet (Lourd)
- Librairies
- Matériel Virtuel

Démarrage... 2 min

Conteneur Docker

- App LAMP
- Dépendances minimales

Démarrage : 2 secondes !

Ce qu'on apprend

VM = Système complet isolé utilisant le noyau de l'hôte (lourd et rapide).
Conteneur = Processus isolé utilisant le noyau de l'hôte (léger et rapide).

Objectif du TP

Ok Whale ! Aujourd'hui, le prof veut qu'on lance une pile LAMP complète pour développer une page web.

Et il faut que je puisse voir la page depuis mon PC !

Facile. On va utiliser une image prête à l'emploi, ouvrir des ports et créer un volume pour tes fichiers.

Ce qu'on apprend

Objectif du TP : déployer un serveur web, une base de données, et rendre le tout accessible sur le réseau.

Activité 1 : Premier lancement et tests

Planche 1

Téléchargement & lancement

Terminal du PC (comme dans <IMAGE>)

```
> docker run -p 80:80 -p 3306:3306 -v $(PWD)/app:/app matttrayner/lamp:latest-1804
```

Test Apache

Regarde bien : -p ouvre les ports (ports) et -v crée un port magique entre ton dossier local et le conteneur.

local host pointe vers votre propre machine. Il est à l'écoute. Le port 80 est le port web par défaut, il est redirigé vers le conteneur.

Où, mais ton dossier app est encore vide. C'est la page par défaut.

Ca marche ! Apache répond.

Test phpMyAdmin

Et la base de données est là aussi !

Un seul conteneur peut contenir plusieurs services (ici Apache + MySQL + PHP + phpMyAdmin) qui fonctionnent ensemble.

Ce qu'on apprend

-p Hôte:Conteneur publie un port.
-v Hôte:Conteneur partage un dossier (volume) pour que les données persistent.

Planché 2 : Le Volume (Pont entre l'hôte et le conteneur)

Planche 2

Le volume, c'est le pont

C'est ça la magie du volume ! Tu utilises tes outils préférés sur ton PC (VS Code), et le conteneur voit le fichier instantanément.

Créer test.php

```
test.php
<?php
echo "Page internet de Sam CIEL dans un conteneur";
```

Ok, je crée une page PHP toute simple pour tester.

Ce qu'on apprend

On crée un fichier de code standard dans le dossier partagé sur la machine hôte.

Tester test.php

Yess ! Mon code s'exécute

Apache dans le conteneur a lu ton fichier via le volume et PHP l'a interprété.

Page internet de Sam CIEL dans un conteneur

Yess ! Mon code s'exécute.

Le développement est immédiat : modification locale -> rafraîchissement du navigateur -> résultat visible.

Ce qu'on apprend

Pas besoin de FTP ! Le volume synchronise en temps réel le dossier local de développement avec le serveur dans le conteneur.

Planché 3 : Accès depuis le voisin : IP du PC vs IP du conteneur

Planche 3

Quelle IP donner au voisin ?

Hé Voisin ! Tape http://192.168.1.23 sur ton navigateur !

Attention : tu donnes l'IP de ta machine physique (l'hôte), pas celle du conteneur (qui est cachée).

Mon adresse IP publique est celle de ton ordinateur, une seule adresse IP : 192.168.1.23.

Le voisin teste

Page internet de Sam CIEL dans un conteneur

Non, c'est l'IP de ton PC hôte. Docker redirige le port 80 du PC vers le conteneur.

Ce qu'on apprend

Le Port Mapping (redirection de port) agit comme une passerelle entre le réseau réel et le réseau virtuel de Docker.

Mini-comparatif VM vs Docker (Rappel réseau)

PC Hôte VM

Besoin d'une IP dédiée (souvent)

Utilise l'IP de l'hôte + des ports

PC Hôte Docker

Docker simplifie souvent la gestion réseau en utilisant l'IP de l'hôte et en jouant sur les ports.

Ce qu'on apprend

Pour accéder à un conteneur depuis l'extérieur, on utilise l'adresse IP de la machine hôte qui le fait tourner.

Planché 4 : Le conflit de ports

Planche 4

On relance un 2e conteneur (la tentative)

```
> docker run -p 80:80 -p 3306:3306 -v $(PWD)/app2:/app matttrayner/lamp:latest-1804
```

Je veux un deuxième site pour un autre projet. Je relance la même commande !

Ah... ça bloque ! (Le constat)

```
Erreur : Le port 80 est déjà utilisé sur cette machine.
```

Mince, ça ne marche pas.

Ce qu'on apprend

Conflit de ports. Un port spécifique sur la machine hôte (ici le 80) ne peut être écouté que par un seul service à la fois.

Logique ! Imagine le port 80 comme une place de parking. Le premier conteneur s'y est déjà garé. La place est prise.

Ce qu'on apprend

Tentative de lancer une deuxième pile LAMP identique sur la même machine.

Planche 5 : La solution (Changer les ports)

Planche 5

On corrige la commande

```
> docker run -p 81:80 -p 3307:3306 -v $(PWD)/app2:/app matttrayner/lamp:latest-1804
```

Voilà la solution : on dit à Docker : "Si quelqu'un appelle sur le port 81 de l'hôte, transfère-le au port 80 du deuxième conteneur".

Test local du 2e conteneur

Page internet de Sam...

Index of /

Super ! localhost pour le premier site, localhost:81 pour le deuxième.

Ce qu'on apprend

Le port dans l'URL (ex : :81) permet de choisir à quel conteneur on s'adresse sur la même machine.

Test depuis le voisin

Page internet de Sam...

Je peux accéder à tes deux environnements en même temps !

Deux environnements web isolés sur une seule machine, sans conflit. C'est la force de Docker !

Ce qu'on apprend

Isolation et parallélisation : Docker permet de multiplier les environnements de développement facilement sur un seul poste.

Planché 6 : Conclusion – Synthèse

Planche 6

Ce que Docker apporte (Le Top pour le Dev)

Checklist

- ✓ Démarrage éclair (Secondes vs Minutes)
- ✓ Léger (Pas d'OS complet)
- ✓ Reproductible (La même image)
- ✓ Simple à exposer (Gestion des ports)
- ✓ Idéal pour le Dév (Volumes)

Pour les TP et le développement web, c'est vraiment plus fluide qu'une VM.

Ce qu'on apprend

Docker est l'outil idéal pour les environnements de développement, de test rapide et les applications "conteneurisées" modernes.

Quand préférer une VM ? (L'autre outil)

Utilisez une VM si :

- Besoin d'un OS complet différent du vôtre
- Isolation de sécurité très forte (le noyau n'est pas partagé)
- Vous simulez un parc de machines physiques complètes

Je ne remplace pas totalement les VM. Nous sommes deux outils différents pour des usages différents !

Fin du TP ! Vous maîtrisez les bases de Docker.