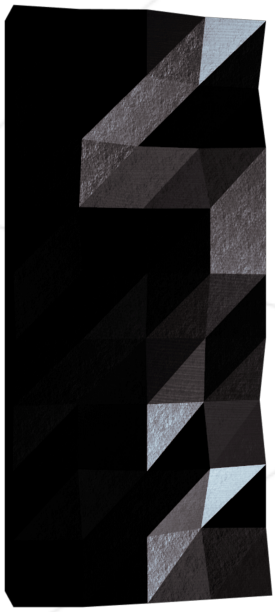


Inception

Résumé: Ce document est un sujet d'Administration Système.

Table des matières

Chapitre I

Préambule

Chapitre II

Introduction

Ce projet a pour but d'approfondir vos connaissances en vous faisant utiliser Docker. Vous allez virtualiser plusieurs images Docker en les créant dans votre nouvelle machine virtuelle personnelle.

Chapitre III

Consignes générales

- Vous devez rendre tous les fichiers nécessaires à la configuration de votre projet dans un dossier `srcs`.
- Vous devez également rendre un `Makefile` qui doit se trouver à la racine de votre répertoire. Il doit permettre de mettre en place toute votre application (c'est-à-dire build les images Docker via `docker-compose`).
- Ce sujet requiert de mettre en pratique des notions que, selon votre parcours, vous n'avez possiblement pas encore abordées. Nous vous conseillons donc de ne pas avoir peur de lire beaucoup de documentation sur l'utilisation de Docker, ainsi que sur tout ce que vous jugerez utile pour réaliser ce projet.

Chapitre IV

Partie obligatoire

Ce projet consistera à vous faire mettre en place une mini-infrastructure de différents services en suivant des règles spécifiques. L'intégralité de ce projet est à réaliser dans une machine virtuelle. Pour ce faire, vous devrez obligatoirement utiliser `docker-compose`.

Chaque image Docker devra obligatoirement porter le même nom que le service concerné.
Chaque service devra tourner dans un container dédié.
Pour des raisons de performance, les containers devront être build au choix : soit sous Alpine Linux avec l'avant-dernière version stable, soit sous Debian Buster.
Aussi, ils devront chacun posséder leur propre `Dockerfile` écrit par vos soins. Les `Dockerfiles` seront appelés dans votre fichier `docker-compose.yaml` par votre `Makefile`.
Vous devrez donc build vous-même les images Docker que vous utiliserez. Il est bien entendu interdit d'en prendre des toutes faites, de même qu'utiliser des services tels que DockerHub (Alpine et Debian étant exclus de cette règle).

Vous allez donc devoir mettre en place :

- Un container Docker contenant NGINX avec TLSv1.2 ou TLSv1.3 uniquement.
- Un container Docker contenant WordPress + php-fpm (installé et configuré).
- Un container Docker contenant MariaDB.
- Un volume contenant votre base de données WordPress.
- Un second volume contenant les fichiers de votre site WordPress.
- Un `docker-network` qui fera le lien entre vos containers.

Vos containers devront redémarrer en cas de crash.

`Un container Docker n'est pas une machine virtuelle, il n'est donc pas recommandé d'utiliser des "hacky patch" à base de 'tail -f' et compagnie lors de son exécution. Informez-vous sur le fonctionnement des 'daemons' et dans quels cas y avoir recours est pertinent.`

L'utilisation de network: host ou --link est bien entendu interdite. La ligne network(s) doit être présente dans votre fichier docker-compose.yml.

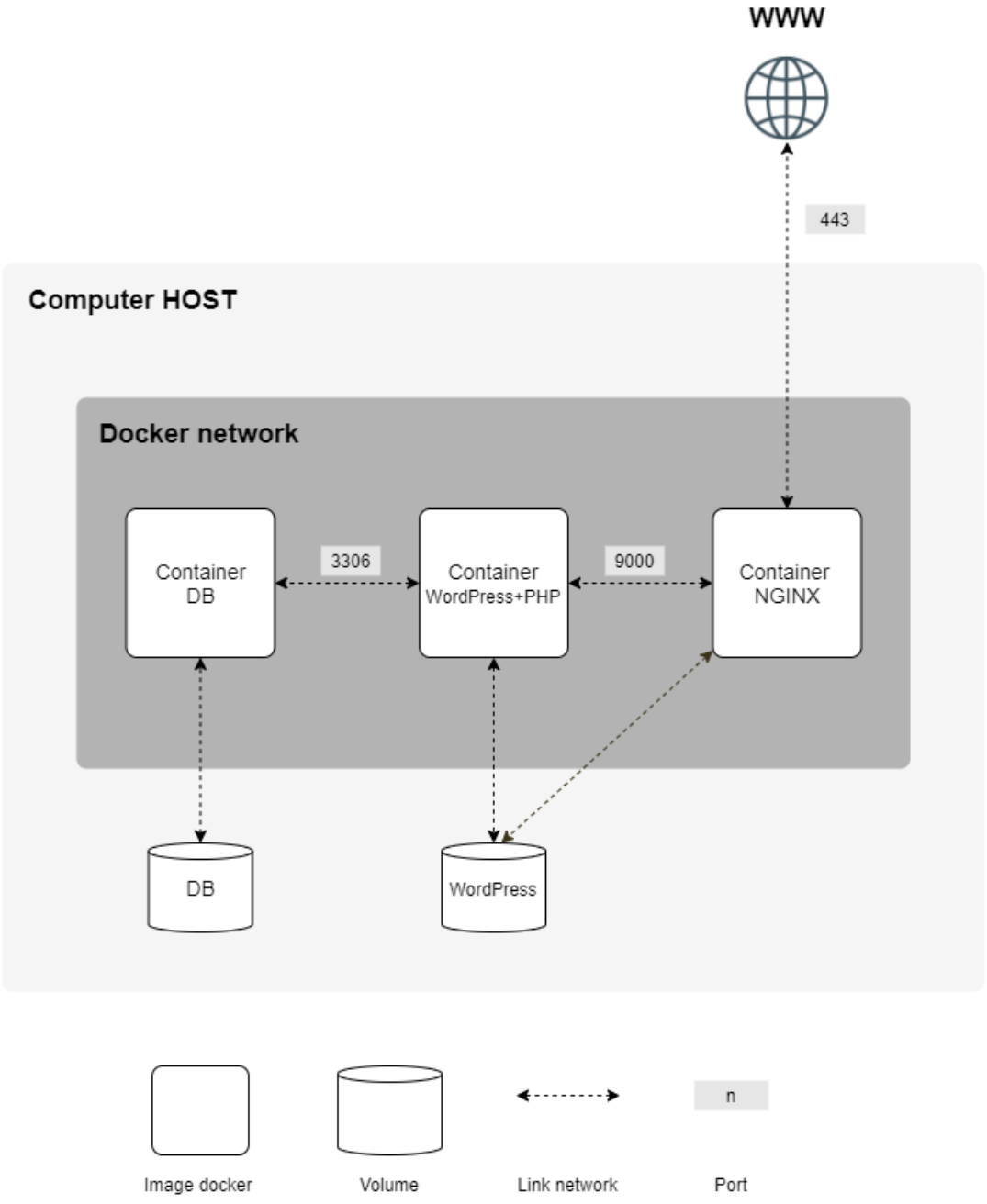
- Deux utilisateurs seront présents dans votre base de données WordPress, l'un d'eux étant le compte Admin. L'username de l'Admin ne devra pas comporter "admin" ou "Admin" (exemples : admin, administrator, administrateur, administratrice, admin-123, etc. ...).

Vos volumes seront disponibles dans le dossier /home/login/data de la machine hôte utilisant Docker. Bien entendu, remplacez le "login" par le vôtre.

Pour des questions de lisibilité, il faudra configurer votre nom de domaine afin qu'il pointe vers votre adresse IP locale.
Ce nom de domaine sera `login.42.fr`. A nouveau, vous utiliserez votre login.
Par exemple, si votre login est wil, `wil.42.fr` redirigera vers l'adresse IP pointant vers le site web de wil.

Le tag latest est interdit.
Aucun mot de passe ne doit être présent dans vos Dockerfiles.
L'utilisation des variables d'environnement est obligatoire.
La mise en place d'un fichier .env afin de stocker vos variables d'environnement est fortement conseillée. Ce fichier devra être rangé à la racine de votre dossier srcs.
Votre container NGINX doit être le seul point d'entrée de votre infrastructure par le port 443 uniquement en utilisant le protocole TLSv1.2 ou TLSv1.3.

Voici un exemple, sous forme de schéma, de ce que vous devrez mettre en place :



Voici un exemple de structure attendue dans votre rendu :

```
$> ls -alR
total XX
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 .
drwxrwxrwt 17 wil wil 4096 avril 42 20:42 ..
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 Makefile
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 srcs

./srcs:
total XX
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 .
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 ..
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 docker-compose.yml
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 .env
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 requirements

./srcs/requirements:
total XX
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 .
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 ..
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 bonus
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 mariadb
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 nginx
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 tools
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 wordpress

./srcs/requirements/mariadb:
total XX
drwxrwxr-x 4 wil wil 4096 avril 42 20:45 .
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 ..
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 conf
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 Dockerfile
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 .dockerignore
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 tools
[...]
./srcs/requirements/nginx:
total XX
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 .
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 ..
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 conf
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 Dockerfile
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 .dockerignore
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 tools
[...]

$> cat srcs/.env
DOMAIN_NAME=wil.42.fr
# certificates
CERTS_=./XXXXXXXXXXX
# MYSQL SETUP
MYSQL_ROOT_PASSWORD=XXXXXXXXXXX
MYSQL_USER=XXXXXXXXXXX
MYSQL_PASSWORD=XXXXXXXXXXX
[...]
$>
```

Chapitre V

Partie Bonus

Pour ce projet, nous souhaitons que les bonus restent simples.

Il vous faudra écrire un `Dockerfile` par service supplémentaire. Ainsi, chaque service tournera dans un container dédié et, si besoin, possédera son propre volume.

Liste de bonus :

- Mettre en place `redis cache` pour votre WordPress, afin de gérer le cache proprement.
- Mettre en place un `serveur FTP` pointant vers le volume de votre site Wordpress.
- Création d'un petit site statique, dans le langage de votre choix sauf PHP (Oui, PHP est exclu !). Par exemple, un site vitrine ou un site de présentation de votre CV.
- Mettre en place `Adminer`.
- Mettre en place un service de votre choix qui vous semble utile. Durant la soutenance, vous aurez à justifier ce choix.

Dans le cadre des bonus, vous avez la possibilité de mettre en place d'autres services. Dans ce cas, il pourra y avoir plus de ports ouverts selon vos besoins.

Les bonus ne seront évalués que si la partie obligatoire est PARFAITE. Par parfaite, nous entendons complète et sans aucun dysfonctionnement. Si vous n'avez pas réussi TOUS les points de la partie obligatoire, votre partie bonus ne sera pas prise en compte.

Chapitre VI

Rendu et peer-évaluation

Rendez votre travail dans votre dépôt `Git` comme d'habitude. Seul le travail présent dans votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.