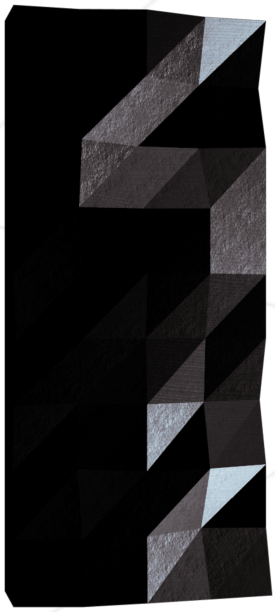


Inception

Resumen: Este documento trata de administración de sistemas.

Versión: 1

Índice general

Capítulo I

Preámbulo

Capítulo II

Introducción

Este proyecto busca expandir tu conocimiento en la administración de sistemas con Docker. Deberás virtualizar varias imágenes de Docker, creándolas en tu nueva máquina virtual personal.

Capítulo III

Instrucciones generales

- Este proyecto debe realizarse en una máquina virtual.
- Todos los archivos requeridos para la configuración de tu proyecto deben estar localizados en un directorio `srcs`.
- Un `Makefile` es necesario y deberá estar localizado en la raíz de tu repositorio. Debe configurar tu aplicación por completo (es decir, debe construir tus imágenes de Docker utilizando `docker-compose.yml`).
- Este subject requiere poner en práctica conceptos que, dependiendo de tu procedencia, puedes no haber aprendido todavía. Por lo tanto, te recomendamos no dudar en leer mucha documentación relacionada con el uso de Docker, y cualquier otra cosa que puedas considerar útil para completar este proyecto.

Capítulo IV

Parte obligatoria

Este proyecto consiste en levantar una pequeña infraestructura compuesta de diferentes servicios bajo ciertas reglas. El proyecto entero debe hacerse en una máquina virtual. Deberás utilizar `docker-compose`.

Cada imagen de Docker deberá llevar el mismo nombre que el servicio al que pertenezca.
Cada servicio debe ejecutarse en un contenedor dedicado.
Por motivos de rendimiento, los contenedores deben construirse utilizando la penúltima versión estable de Alpine Linux, o Debian Buster. La elección es tuya.
Deberás escribir tus propios `Dockerfiles`, uno por servicio. Los `Dockerfiles` deben llamarse en tu `docker-compose.yaml` por tu `Makefile`.
Esto significa que debes construir tus propias imágenes de Docker para el proyecto. Está prohibido utilizar imágenes preparadas, así como utilizar servicios como DockerHub (Alpine/Debian están excluidos de esta regla).

Debes entonces configurar:

- Un contenedor de Docker que solo contenga NGINX con TLSv1.2 o TLSv1.3.
- Un contenedor de Docker que contenga solo WordPress + php-fpm (debe estar instalado y configurado) sólo (sin servicios adicionales).
- Un contenedor de Docker que tenga MariaDB sólo (sin servicios adicionales).
- Un volumen que contenga tu base de datos de WordPress.
- Un segundo volumen que contenga tus ficheros de WordPress.
- Un `docker-network` que establezca la conexión entre tus contenedores.

Tus contenedores deben reiniciarse en caso de crash.

`Un contenedor de Docker que no sea una máquina virtual. Por lo tanto, no se recomienda utilizar ningún apaño del estilo 'tail -f' cuando intentes ejecutarlo. Lee sobre el funcionamiento de los daemons y si es buena idea utilizarlos o no.`

Evidentemente, utilizar network: host, --link o links: está prohibido. La línea de network debe estar presente en tu archivo docker-compose.yml. Tus contenedores no deben iniciar con un comando ejecutándose en bucle infinito. Por lo tanto, esto también aplica a cualquier comando utilizado como entrypoint, o usado en scripts de entrypoint. Los siguientes son algunos de los posibles apaños prohibidos: tail -f, bash, sleep infinity, while true.

Lee sobre el PID 1 y las buenas prácticas para escribir Dockerfiles.

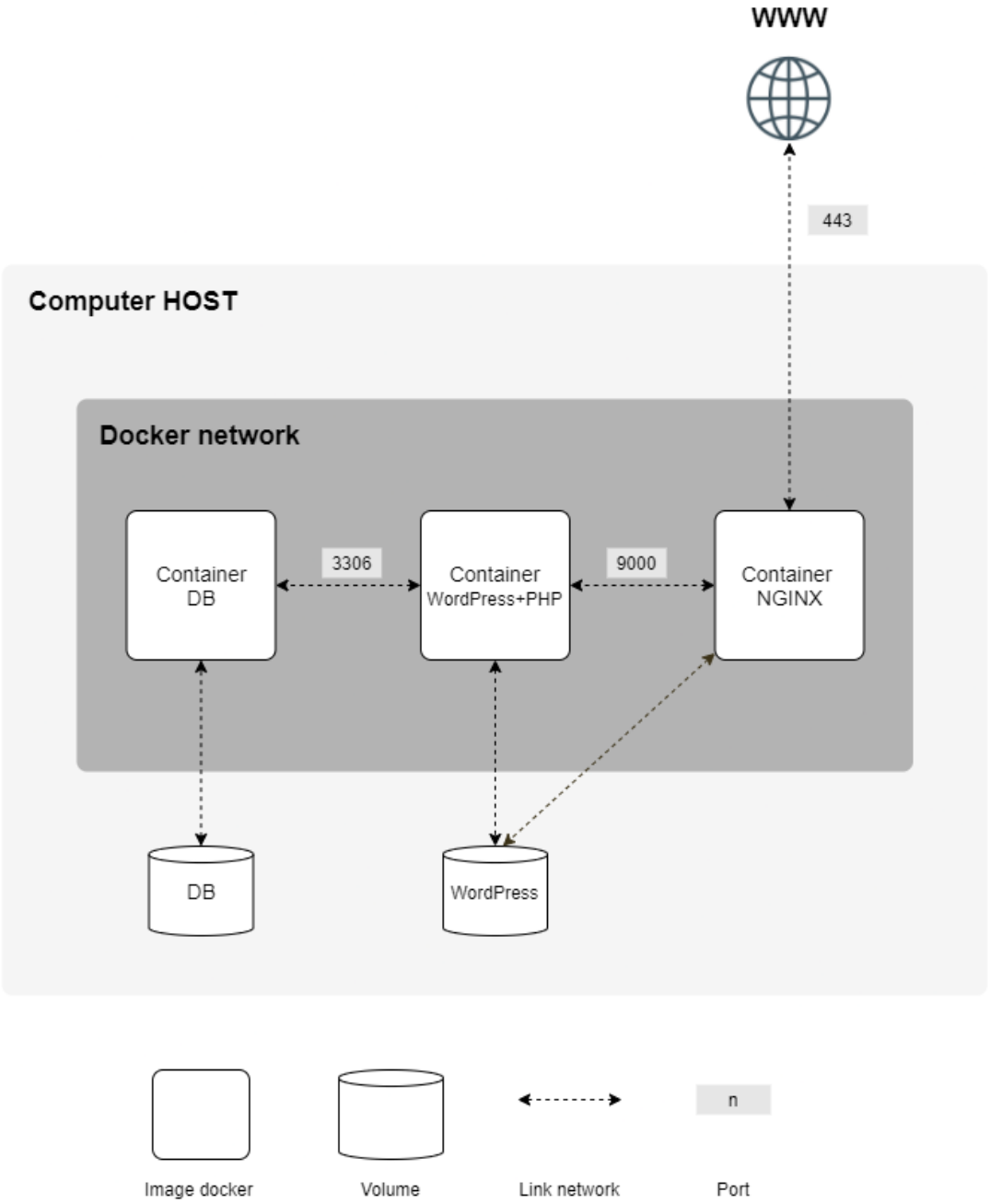
- En tu base de datos de WordPress, debe haber dos usuarios. Uno de ellos siendo el administrador. El usuario de administrador no puede contener admin/Admin o administrator/Administrator (por ejemplo, admin, administrator, Administrador, admin-123, y un largo etc).

Tus volúmenes deberán estar localizados en el directorio /home/login/data del host utilizando Docker. Por supuesto, deberás reemplazar login por el tuyo.

Para simplificar las cosas, deberás configurar tu nombre de dominio para que apunte a tu IP local.
El nombre de dominio debe ser `login.42.fr`. De nuevo, deberás usar tu propio login.
Por ejemplo, si tu login es wil, `wil.42.fr` redirigirá a la IP del sitio web de wil.

El uso de la etiqueta latest está prohibido.
No deben quedar contraseñas en tus Dockerfiles.
Es obligatorio el uso de variables de entorno.
También, se recomienda encarecidamente utilizar un archivo .env para guardar las variables de entorno. El archivo .env deberá estar localizado en la raíz del directorio srcs.
Tu contenedor NGINX debe ser el único punto de entrada a tu infraestructura, sólo a través del puerto 443 y utilizando el protocolo TLSv1.2 o TLSv1.3.

Aquí hay un diagrama de ejemplo del resultado esperado:



Debajo tienes un ejemplo del sistema de la estructura de directorios esperada:

```
\$> 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
[...]
\$>
```

Capítulo V

Parte bonus

Para este proyecto, la parte bonus intenta ser sencilla.

Un archivo Dockerfile debe ser escrito para cada servicio extra, así que cada servicio extra dispondrá de su propio contenedor y su respectivo volumen.

Lista de bonus:

- Configura `redis cache` para controlar decentemente el cache de tu sitio WordPress.
- Configura un contenedor `FTP server` apuntando al volumen de tu sitio WordPress.
- Crea un sitio web estático en el lenguaje de tu elección exceptuando PHP (sí, PHP está excluido). Por ejemplo, un resumen o un sitio que te presente.
- Configura `Adminer`.
- Configura un servicio de tu elección que consideres útil. Durante la defensa, deberás justificar tu elección.

Para completar la parte bonus, dispones de la posibilidad de configurar servicios adicionales. En este caso, puedes abrir puertos que te permitan adaptarte a las necesidades.

La parte bonus será solo corregida si la parte obligatoria está PERFECTA. Perfecta significa que la parte obligatoria está completa íntegramente y funciona sin fallos. Si no has pasado TODOS los requisitos obligatorios, tu parte bonus no será en absoluto evaluada.

Capítulo VI

Entrega y evaluación de compañeros

Entrega tu proyecto en tu repositorio `Git` como siempre. Solo el trabajo dentro de tu repositorio será evaluado durante la defensa. No dudes en comprobar dos veces los nombres de directorios y archivos para asegurarte de que son correctos.