INCEPTION

Historique :

Docker est une plateforme créée en 2010 par le franco-américain Solomon Hykes (ancien élève d'Épitech). Le projet est devenu open source depuis 2013, et est dirigé par l'entreprise Docker inc. Un docker est à l'origine une personne qui travaille dans les docks (l'endroit du port où s'échange la marchandise), c'est lié au nom de l'outil qu'utilise Docker, le conteneur.

Il reprend le principe du conteneur linux, qui est un processus isolé avec un environnement différent, mais qui va s'appuyer sur certains éléments de la machine hôte, il n'y a donc pas de virtualisation (comme une VM)

Différence virtualisation / conteneurisation :

La virtualisation isole les VM au niveau matériel (CPU, RAM, Disque), ils peuvent fournir plus de ressources que nécessaire, ça se fait avec un hyperviseur.

Avec la conteneurisation, l'isolation se fait au niveau de l'OS, le container se base directement sur le noyau de l'OS. C'est un outil qui existait déjà avant Docker.

Comment ça marche :

Une fois que l'on a créé notre fichier Dockerfile, on doit d'abord en faire une image (on doit la « build ») (vous pouvez voire ça comme une compilation). Une fois que cette image est créée, vous pouvez en créer un container, ce container correspond à une instance de l'image. Ce container peut ensuite être lancer, ou arrêter.

La taille d'une image dépend de ses « layers » ('couche', en anglais), il n'y a que trois instructions qui peuvent créer des layers, RUN, COPY, et ADD (voire supra). Les autres commandes s'appellent des layers intermédiaires. Lorsqu'on modifie un layer il n'y aura que lui et ceux d'après, qui seront recompilés, dans l'image.

Pour approfondir :

Le conteneur va utiliser des « namespaces » pour que son processus soit séparé du reste de l'ordinateur. De plus, les « groupes de contrôles » limitent les conteneurs dans leur consommations de mémoire, de réseau… Un site qui peut aider : https://devopssec.fr/article/cest-quoi-un-conteneur

Commandes pratiques :

Mot-clé	Fonction	Source
docker build <chemin>	Créé l'image à partir du Dockerfile dans le chemin précisé. L'option « -t <nom d'image> » permet de lui donner un nom	https://docs.docker.com/engine/reference/commandline/build/
docker run <nom d'image>	Créé et démarre le container, à partir de l'image	https://docs.docker.com/engine/reference/commandline/run/
docker exec <nom du container> <commande>	Exécute la commande dans le container. Avec l'option « -it », cela attache le container au terminal	https://docs.docker.com/engine/reference/commandline/exec/
docker ps	« process statue », liste les containers	https://docs.docker.com/engine/reference/commandline/ps/
docker logs <nom du container>	Donne les logs du container (utile si le container ne se lance qu'une fois)	https://docs.docker.com/engine/reference/commandline/logs/

Makefile :

L'option -f après la commande « docker-compose », permet de spécifier la chemin du docker-compose (https://docs.docker.com/engine/reference/commandline/compose/#use--f-to-specify-the-name-and-path-of-one-or-more-compose-files)

Il faut supprimer le contenu des volumes lors du « clean » (rm -rf « chemin volume »)

Suivant la version, il faut mettre ou pas, le « - » entre « docker » et « compose »

Mot-clé	Fonction	Source
docker-compose build	Créé les images, à partir des Dockerfiles	https://docs.docker.com/engine/reference/commandline/compose_build/
docker-compose create	Créé les conteneurs à partir des images	https://docs.docker.com/engine/reference/commandline/compose_create/
docker-compose start	Démarre les conteneurs	https://docs.docker.com/engine/reference/commandline/compose_start/
docker-compose up	Créé les conteneurs à partir des images et démarre les conteneurs, fusion des deux commandes précédentes. Avec l'option « -d », lance le container en arrière-plan	https://docs.docker.com/engine/reference/commandline/compose_up/
docker-compose down	Stoppe et détruit le container, fusion des deux commandes suivantes.	https://docs.docker.com/engine/reference/commandline/compose_down/
docker-compose stop	Stoppe le container.	https://docs.docker.com/engine/reference/commandline/compose_stop/
docker-compose rm	Détruit le container.	https://docs.docker.com/engine/reference/commandline/compose_rm/
docker system prune	Détruit les images, les containers, et les networks	https://docs.docker.com/config/pruning/#prune-everything

docker volume prune	Détruit les volumes	https://docs.docker.com/config/pruning/#prune-volumes

Docker-compose

Service :

Mot-clé	Fonction	Lien
container_name: <name>	Définit le nom du conteneur	https://docs.docker.com/compose/compose-file/#container_name
depends_on: <service>	Ordonne de commencer lorsque le service en question est finit	https://docs.docker.com/compose/compose-file/#depends_on
restart : <arg>	Définit ce que doit faire le docker lorsqu'il s'arrête, les arguments sont : - no : par défaut, ne fais rien - allways : redémarre à chaque fois qu'il s'éteint - on-failure : redémarre que si il y a un problème - unless-stopped :redémarre que s'il n'est pas stopper à la main (le docker pouvant s'arrêter naturellement s'il n'est pas lancé en daemon	https://docs.docker.com/compose/compose-file/#restart
volumes: - <volume>:<chemin>	Fait correspondre le volume (qui peut être un volume définit, ou un chemin de l'hôte) à un chemin du container	https://docs.docker.com/compose/compose-file/#volumes
networks : - <name of network>	Définit le ou les networks qu'utilisera le container	https://docs.docker.com/compose/compose-file/#networks
port : - "<port de l'hôte>:<port du container>"	Expose le port d'un container (à faire que pour nginx)	https://docs.docker.com/compose/compose-file/#ports
init : <opt>	Détruit les processus zombies, si l'option est à « true ». Expliqué ici : https://www.linkedin.com/pulse/docker-et-les-processus-zombie-karine-poulet/?originalSubdomain=fr	https://docs.docker.com/compose/compose-file/compose-file-v3/#init
environnement : <key> :<value>	Définit les variables d'environnements du container	https://docs.docker.com/compose/compose-file/#environment
env_file: <.env file>	Ajoute les variables du fichier .env aux variables d'environnement du container	https://docs.docker.com/compose/compose-file/#env_file

Service→build :

Mot-clés	Fonction	Source
dockerfile : <name>	Autorise à prendre un autre nom pour le fichier à lire (par défaut, c'est « Dockerfile »	https://docs.docker.com/compose/compose-file/build/#dockerfile
context : <chemin>	Demande le chemin où se trouve le Dockerfile (obligatoire)	https://docs.docker.com/compose/compose-file/build/#context-required
args :	Définit un argument pour le build (est repris	https://docs.docker.com/compose/com

<key>: <value>	par le ARG du dockerfile)	pose-file/build/#args

Volume :

Fonctionnement :

Le volume est le mécanisme préféré pour enregistrer les données utilisés par les dockers, le volume correspond à un fichier partagé entre l'hôte et le container, le container y mettra toutes les données « persistante », il stockera le reste (tmpfs) dans une mémoire prévu pour le container. (https://docs.docker.com/storage/volumes)

Mot-clés	Fonction	Source
driver: <driver-name>	Définit le driver du volume, par défaut, celui-ci est « local », ce qui signifie que le volume sera stocké dans l'hôte	https://docs.docker.com/compose/compose-file/#driver-1
driver_opts:	Définit les options pour le driver du volume	https://docs.docker.com/compose/compose-file/#driver_opts-1

Volume→driver_opts :

La création de volume correspond à la commande linux « mount », ces paramètres en sont aussi ses arguments

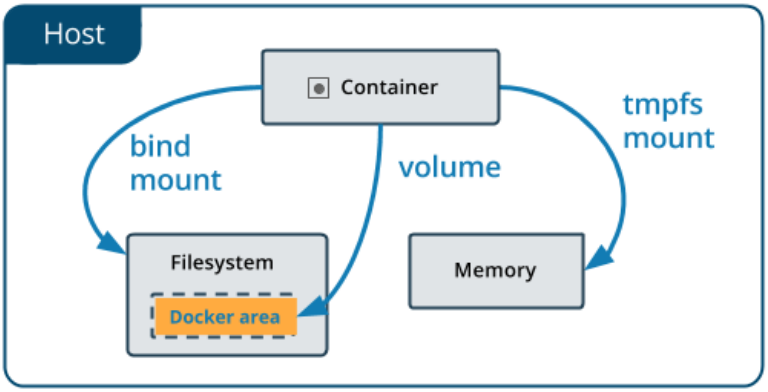
type: <type opt>	Définit la partition du disque lié au point de montage (mettre à none, si on en a pas)	https://unix.stackexchange.com/questions/136646/what-is-type-none-for-mount-points-and-why-are-mount-points-of-the-same-type-b/136689#136689
o: <option>	Définit les options de montages, elles peuvent être « read-only » (ro), « read-write » (rw), ou reliée (on peut écrire-lire sur le volume depuis l'host (bind))	https://man7.org/linux/man-pages/man8/mount.8.html
device: <chemin>	Le chemin où seront placés les volumes (le sujet demande /home/login/data)	https://docs.docker.com/storage/volumes/#block-storage-devices

La commande pour lister les volumes est « docker volume ls »

Pour approfondir :

Avant le volume, on utilisait les « bind mount », ceux-ci étaient directement dans les fichiers système de l'hôte, et devaient être gérés par l'hôte. Tandis que les volumes peuvent être inaccessibles (donc protégés), et autonomes.

Networks :



Fonctionnement :

Le network est le dernier layer, qui autorise les services à communiquer entre eux

Les trois drivers les plus connus sont :

-le driver bridge, il y en a automatiquement un dés l'installation de docker, celui-ci sera le driver par défaut, il va correspondre à une switch

-le driver none, il retire l'interface réseau du container

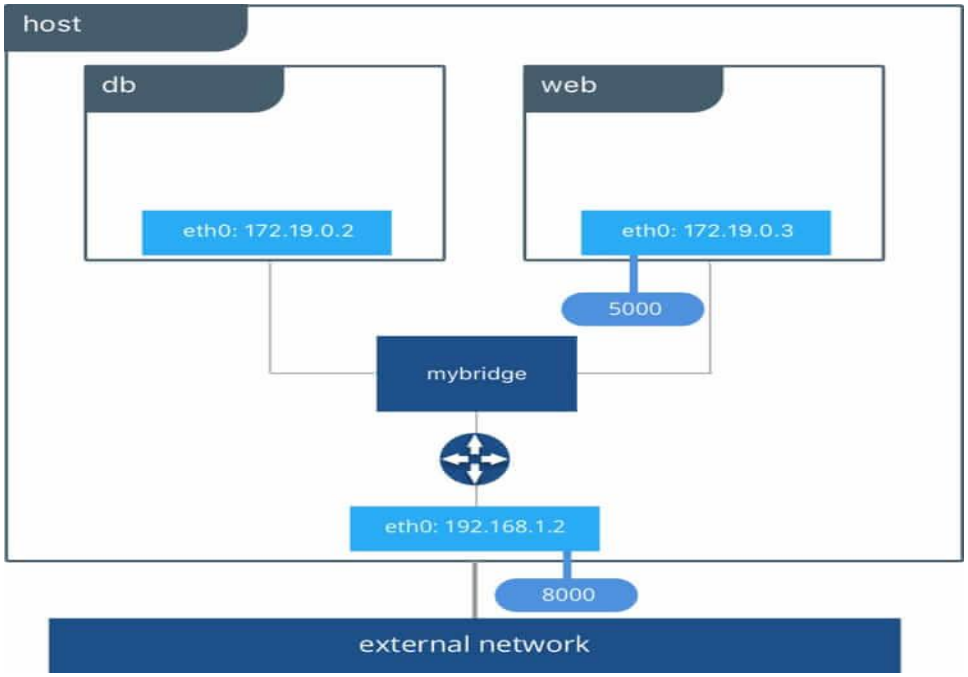
-le driver host, qui permet au container d'utiliser la même interface que l'hôte, ce qui fait qu'il aura la même IP.

Voici un lien qui vous montre d'autre networks : https://devopssec.fr/article/fonctionnement-manipulation-reseau-docker

La commande pour lister les networks est « docker network ls »

Il faudra pour chaque container, les faire lire sur l'adresse « 0.0.0.0. » cela leur permettra de recevoir des informations venant de chaque réseau, il faudra aussi « exposer » le bon port.

Exemple de configuration en network bridge :



Pour approfondir :

La documentation sur la communication entre les containers : https://docs.docker.com/network/iptables/#setting-the-default-bind-address-for-containers

Dockerfile

Mot-clé	Fonction	Documentation
FROM <nom_d'image>	Définit une image d'entrée au docker (pour nous, c'est « debian:10 »)	https://docs.docker.com/engine/reference/builder/#from
WORKDIR <chemin>	Définit le chemin de « commencement ». Correspond au « cd »	https://docs.docker.com/engine/reference/builder/#workdir
ADD <src> <dest>	Copie src dans dest, src peut être une URL, un fichier compressé, qu'il va décompressé. (cette commande est ancienne et déconseillée, mieux vaut utiliser curl et COPY)	https://docs.docker.com/engine/reference/builder/#add
RUN <commande>	Exécute la commande dans un nouveau layer	https://docs.docker.com/engine/reference/builder/#run
ENTRYPOINT [<executable>, <param>]	Lance l'executable avec les paramètres en argument. Ne peut pas se faire override.	https://docs.docker.com/engine/reference/builder/#entrypoint
CMD [<executable>, <param>] CMD[<param1>]	Lance l'exécutable avec les paramètres en argument Peut lancer que les arguments, si l'exécutable est dans l'ENTRYPOINT. Peut se faire override en mettant le nom de la commande lors du lancement du container.	https://docs.docker.com/engine/reference/builder/#cmd
EXPOSE <port>	Définit le port d'écoute du docker	https://docs.docker.com/engine/reference/builder/#expose
ARG <name>	Récupère les arguments passés par le docker-compose	https://docs.docker.com/engine/reference/builder/#arg
COPY <src> <dest>	Copie un fichier ou dossier, du PC vers un chemin du container	https://docs.docker.com/engine/reference/builder/#copy
ENV <key>=<value>	Ajoute des variables d'environnement au container	https://docs.docker.com/engine/reference/builder/#env

MariaDB

Historique :

MariaDB a été créer par le co-fondateur de MySQL, Michael Wildenius, celui-ci avait créer MySQL du nom de sa première fille « My », et de l'acronyme « Structured Query Language », langage de requête structurée. C'est lorsque MySQL s'est fait racheté par Oracle Corporation, que Michael Wildenius va créer en 2009 MariaDB, du nom de sa deuxième fille, Maria, et de l'acronyme « Data Base », dans le but de remplacer MySQL tout en assurant l'interopérabilité. MariaDB est sous licence GPL, alors que MySQL est sous double-licence GPL et propriétaire.

Dockerfile :

Dans le dockerfile, vous devrez entre autre, pour le build :

- Installer mariadb-server (l'option '-y' permet de confirmer l'installation)
- Démarrer le service mysql (https://kontext.tech/article/569/how-to-start-or-stop-mysql-service)
- Changer les paramètres de mariadb, à l'aide du tableau suivant et de la commande « mysql --user=<user name> --execute=<commande SQL à executer> »

Mot-clés	Fonction	Source
CREATE DATABASE ;	Lance la base de donnée	https://sql.sh/cours/create-database
CREATE USER <user>@<localhost> IDENTIFIED BY <password> ;	Créé un utilisateur	https://dev.mysql.com/doc/refman/8.0/en/create-user.html#create-user-overview
ALTER USER <param> ;	Change les paramètres d'un utilisateur existant (le root existe de base) (il lui faut le nouveau mot-de-passe)	https://dev.mysql.com/doc/refman/8.0/en/alter-user.html#alter-user-overview
GRANT <privilege> TO <user> ;	Donne des privilèges à un utilisateur	https://dev.mysql.com/doc/refman/8.0/en/grant.html#grant-overview

Vous devez ensuite, pour le lancement du container :

- Lancer le daemon de MySQL (https://dev.mysql.com/doc/refman/8.0/en/mysqld.html), en définissant en paramètre le bind-address à 0.0.0.0.

Enfin, vous aurez besoin pour vérifier le fonctionnement de votre database, d'exécuter dans votre container, pour vous connecté, executez la commande « mysql » dans votre container, puis pour voire ce qu'il y a :

Mot-clés	Fonction	Source

SHOW DATABASES ;	Liste les bases de données présentes sur l'hôte	https://dev.mysql.com/doc/refman/8.0/en/show-databases.html
SHOW STATUS ;	Liste les informations sur le serveur	https://dev.mysql.com/doc/refman/8.0/en/show-status.html

Wordpress

Historique :

Wordpress est un système de gestion de contenu (content management system), gratuit, libre, et open-source. Ses principaux concurrents sont Shopify, qui s'occupe des ventes en ligne, et Wix. Wordpress est écrit en PHP, et repose par défaut sur une base de données MySQL. « PHP » est un sigle auto-référentiel, qui veut dire « PHP : Hypertext Preprocessor », principalement utilisé pour les pages dynamiques, c'est un langage orienté objet, utilisé le plus souvent côté serveur, qui va convertir le code PHP en code interprétable par le navigateur (HTML, CSS), et en données (JPEG, GIF, PNG par exemple). Le PHP était à l'origine une bibliothèque logicielle en C, créé par le groenlandais Rasmus Lerdorf, la syntaxe est très proche du C.

Le PHP a été créé en 1995 et est utilisé actuellement par 80 % des sites web, tandis que Wordpress a été créer en 2003, et est actuellement utilisé par 40 % des sites web.

En 2004, a été créer PHP-FPM, FPM veut dire « FastCGI Process Manager », CGI étant une interface de passerelle commune (Common Gateway Interface). PHP-FPM est un gestionnaire de processus, qui va permettre la communication avec un serveur (NGINX dans notre cas), et PHP, sans cette interface, PHP devrait être sur le même serveur que Nginx. Ce site vous l'explique bien https://geekflare.com/fr/php-fpm-optimization/.

Dockerfile :

<u>Il faut dans le dockerfile pour build</u> :

- installer mariadb-client (fonctionne de la même manière que le -server, voir supra)
- installer php-fpm (pour communiquer avec nginx)
- installer php-mysql
- installer wp-cli, avec ce site https://make.wordpress.org/cli/handbook/guides/installing/ (wp-cli est l'interface de ligne de commande, « Command Line Interface »)
- modifier le fichier de config de fpm (etc/php/7.3/fpm/pool.d/www.conf), pour qu'il n'écoute plus sur « /run/php/php7.3-fpm.sock », mais « 0.0.0.0:9000 ». Pour cela, vous pouvez placer votre propre fichier de config modifié à la place de l'ancien (en utilisant COPY), ou utiliser la commande « sed » de bash
- se mettre dans le dossier « /var/www/html», et utiliser les commandes du CLI, pour le reste de l'installation

L'option --allow-root permet de donner les permissions de root à la commande. Pensez au Shebang.

Mots-clés	Fonction	Source
wp core download	Télécharge les fichiers d'installation.	https://developer.wordpress.org/cli/commands/core/download/
wp config create	Créé le fichier de config, il faut mettre en paramètre les coordonnées de la database (il faut le mettre dans un .sh executé par CMD ou ENTRYPOINT, car il doit	https://developer.wordpress.org/cli/commands/config/create/

	prendre en paramètre la database, il faut donc qu'elle soit lancée)	
wp core install	Lance le processus d'installation standard Regardez avec la source pour bien configurer l'admin	https://developer.wordpress.org/cli/commands/core/install/
wp user create	Créé un nouvel utilisateur (comme demandé dans le sujet). Regardez aussi la source	https://developer.wordpress.org/cli/commands/user/create/

Il faut enfin exécuter la commande wordpress avec l'argument « daemon off »

Nginx

Historique :

Nginx est un logiciel libre de serveur web ainsi qu'un proxy inverse (cela signifie par rapport au proxy, qu'il se met en intermédiaire entre deux hôtes pour surveiller les échanges, Nginx fait parti du réseau interne du serveur). Son but principal va être de chiffrer en SSL, mais il permet aussi de :

- protéger le serveur web des attaques
- utiliser la mémoire cache pour alléger les charges des serveurs web
- de compresser le site qui va être envoyé.

Il est le serveur web le plus utilisé depuis 2019, devant Apache, et Microsoft IIS. Il a été créer par le russo-kazakhstannais Igor Sysoev, en 2004, pour le moteur de recherche russe Rambler. Ce n'est que lorsqu'il a été traduit en anglais en 2006, qu'il s'est fait connaître en occident.

En 1983 est breveté par trois membres du MIT le chiffrement RSA, du nom de ses trois inventeurs, R. Rivest, A. Shamir, L. Adleman, c'est un chiffrement asymétrique, qui nécessite une clé privée et une clé publique (https://fr.wikipedia.org/wiki/Chiffrement_RSA).

OpenSSL (Secure Socket Layer), créé en 1998, va générer un certificat, qui contient une clé publique (générée par OpenSSL), et des informations (nom de la société, adresse postale…).

TLS (Transport Layer Security) est simplement le successeur de SSL, ils sont souvent confondus.

Dockerfile :

Il faut dans le dockerfile pour build :

- installer openssl
- créer les dossiers où seront placés le certificat et la clé SSL
- générer le certificat SSL, et par la même commande, la clé SSL (https://www.openssl.org/docs/man1.1.1/man1/req.html) (il faudra utiliser -node, pour empêcher qu'ils demandent une passphrase)
- installer nginx
- copier le fichier de configuration de nginx dans /etc/nginx/sites-available/default, vous allez voir comment le configurer infra, vous pouvez aussi vous aider de ces deux sites :
https://wordpress.org/documentation/article/nginx/#https-in-nginx
https://unit.nginx.org/configuration/#php

Il faut pour la fabrication du container :

- de même que pour wordpress, executer la commande « nginx » avec le daemon off. Cela l'empêchera de se lancer en arrière plan. (https://nginx.org/en/docs/ngx_core_module.html#daemon
https://www.nginx.com/resources/wiki/start/topics/tutorials/commandline/)

Pour aller plus loin :
https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/

Si mes explications sont nuls, regardez ce site : https://www.it-connect.fr/debian-comment-installer-nginx-en-tant-que-serveur-web/

Fichier de config :

Mot-clés	Fonction	Source
listen <port d'entrée> <opt> ;	Spécifie le port de lecture d'nginx, si il y a l'option « ssl », décrit que le type de contenu sera SSL	http://nginx.org/en/docs/http/ngx_http_core_module.html#listen
ssl_protocols <protocole> ;	Autorise les protocoles spécifiés (le sujet demande TLSv1.2 ou TLSv1.3)	https://nginx.org/en/docs/http/ngx_http_ssl_module.html#ssl_protocols
ssl_certificate <path to file> ;	Définit l'endroit où se trouve le certificat SSL	https://nginx.org/en/docs/http/ngx_http_ssl_module.html#ssl_certificate
ssl_certificate_key <path to file> ;	Définit l'endroit où se trouve la clé SSL	https://nginx.org/en/docs/http/ngx_http_ssl_module.html#ssl_certificate_key
root <path> ;	Définit le repertoire racine (doit correspondre au repertoire où il y a les volumes)	http://nginx.org/en/docs/http/ngx_http_core_module.html#root
index <path to file> ;	Définit les pages qui vont êtres utilisés si on ne précise rien dans l'URL (dans notre cas, ce sera index.php)	https://nginx.org/en/docs/http/ngx_http_index_module.html
location <option> { configuration }	Définit le passage à prendre pour un certain URL (pour notre cas, tout les URL qui finissent par « php » doivent utiliser le fastcgi (pour communiquer avec le serveur wordpress)	https://nginx.org/en/docs/http/ngx_http_core_module.html#location

FastCGI du fichier de config (ce site va vous aider https://nginx.org/en/docs/beginners_guide.html#fastcgi) :

Mot-clés	Fonction	Source
fastcgi_pass <adress> ;	Définit l'adresse du serveur fastcgi, dans notre cas, le serveur s'appelle « wordpress », et on l'écoute sur le port 9000	https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_pass
include fastcgi_params;	Inclut les fichiers de paramètres CGI, permet d'utiliser le paramètre « SCRIPT_FILENAME » pour la fonction d'après	https://www.nginx.com/resources/wiki/start/topics/examples/phpfcgi/

fastcgi_param <paramètre> ;	fastcgi_param définit des paramètres pour php-fpm, est utilisé pour définir le nom du site sur PHP, ce site explique ce qu'il faut faire : https://serverfault.com/questions/465607/nginx-document-rootfastcgi-script-name-vs-request-filename	https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_param

Finalisation

Il faudra modifier le fichier /etc/hosts de la VM, pour faire corresponde l'URL « login.42.fr » au localhost.

Pour vous connecter, mettez « login.42.fr » sur votre navigateur, « /wp-login.php » pour vous connecter en root, ou user.

Si vous voulez transférer vos fichiers depuis votre hôte jusqu'à la VM, utilisez la commande « scp » (https://linuxcommandlibrary.com/man/scp)

Si vous voulez manipuler vos fichiers de VM (pour la correction) depuis le vs-code de votre hôte, suivez ce guide : https://code.visualstudio.com/docs/remote/ssh-tutorial

Bonne pratique

Si vous voulez optimiser un peu vos docker, pour que vos images soient moins lourdes, vous pouvez :

- faire le moins de Layer possible, au lieu de faire deux RUN, par exemple, faites un RUN avec les « && »
- ignorer tout les fichiers présents dans dans le répertoire de votre Dockerfile, que vous n'utilisez pas dans votre docker, en les mettant dans un .dockerignore
- supprimer les dépendances de vos installations qui ne sont pas utilisés

La meilleur manière d'utiliser « ENTRYPOINT » est de mettre la commande dans l' « ENTRYPOINT » et de mettre les flags dans « CMD »

Si vous voulez plus d’infos, demandez à ajossera et offrez-lui un cookie