Résumé Virtualisation

# Leçon

## Containers et images

Image : package qui inclut tout ce qui est strictement nécessaire à l'exécution d'une application :

* Le code
* L'exécution
* Les variables d'environnement
* Les bibliothèques
* Les fichiers de configuration

Container : fabriqué en exécutant une **image** (son instance en gros).

**docker container XXX id\_nom**

* ps → afficher tous les containers
* pause / unpause → mettre en pause
* run / rm *IMAGE* → créer un serveur à partir d’une image
  + run -p *PORTLOCAL:PORTCONTAINER* → précise les ports
  + run -d → exécute en asynchrone
  + run –*name* → précise le nom
  + -v *DOSSLOCAL | DOSSCONTAINER →* permet de mapper le dossier serveur dans le dossier hote
  + --name *SERV*
* start / stop → démarrer / stopper
* exec *id\_nom commande* → exécuter une commande (ex : bin/bash)
  + -it :

**stop** supprime le conteneur, ce qu’il contient, y compris les données stockées, fichiers de configuration modifiés directement dans le conteneur, mais pas les éventuelles données stockées sur le système hôte si des dossiers de l’hôte avaient été mappés comme volumes, ni les volumes créés et liés au conteneur.

Pour rendre persistantes les données contenues :

* “Mapper” un dossier de l’hôte dans le conteneur et enregistrer toutes les données que l’on souhaite voir disponible et pérenne dans le dossier mapper dans le conteneur.
* Créer un volume, et stocker les informations dans ce volume.

## Réseaux

**docker network XXX**

* create
* connect

## Bases de données

**mysql -u NOMUTILISATEUR -p** → connecte à la base de donnée depuis une session utilisateur dans le container

**show tables**

**show databases**

**create NOMDB** → créé la base de données

**use NOMDB** → utilise la base de données

**quit**

## Autre

Installer nano :

**apt-get update** → maj des applications installées

**apt-get install nano** → installer nano

# TD-TP

## TD1

On lance un container nginx : **docker run –p 8000:80 –d nginx**

Pour trouver le dossier contenant les fichiers sources on doit ouvrir le fichier de configuration. Pour cela, on doit ouvrir une session de terminal utilisateur : **docker exec –t –i webserver /bin/bash**

On accède ensuite au fichier dans /etc/nginx/conf.d/default.conf :

Une image contenant texte

Description générée automatiquement

→ les fichiers source se trouvent donc dans /usr/share/nginx/html

*Lorsqu’on héberge un site sans mapping :*

* *La modification des fichiers sources est fastidieuse*
* *La sauvegarde des informations du site sera elle aussi fastidieuse*

→ Créer un conteneur tout en mappant un dossier spécifique du conteneur à un dossier du système hôte :

Docker run -p 8000:80 -d -v f:\BUT\docker\webserver:/usr/share/nginx/html --name webserver nginx

## TP1

On cherche sur docker hub un container apache et php 7 en tapant “apache php” et en cochant sponsor OSS à gauche.

**docker images** pour voir les images crées

**docker run -p 9002:80 -d -v C:\Temp\docker\webserver\:/usr/local/apache2/htdocs --name webserver httpd** → Mappe httpd dans le serveur nommé webserver dans le dossier C:\Temp\docker\webserver\.

recherche d’un docker basé sur apache 2.4 depuis docker hub :

Une image contenant texte

Description générée automatiquement

## TP2

on lance un container webdevops/php-apache-dev

on trouve dans le fichier de config le dossier des fichiers source : /app

on relance le container en le mappant dans C:/Temp : **docker run -p 8080:80 --name container -v C:/Temp:/app webdevops/php-apache-dev**

on créé le réseau myworkflow : **docker network create myworkflow**

ensuite on ajoute le container à ce réseau : **docker network connect myworkflow db**

on lance un container mariadb (car mysql fonctionne pas bien) en précisant le mdp (c’est obligatoire) et son réseau : **docker run -p 8081:80 -e MYSQL\_ROOT\_PASSWORD=toor --network myworkflow --name db -d mariadb**

on se connecte au container avec **docker exec -it db bin/bash**

ensuite on se connecte à la base de donnée avec **mysql -u root -p**