RESSOURCES POUR DOCKER ET DOCKER COMPOSE

Raspberry Pi - Debian Bullseye
Ressources basiques et avancées

Tutoriel **DOCKER** - RASPBERRY PI

David GOÏTRÉ

Table des matières

Introduction

DOCKER peut installer des conteneurs uniquement en ligne de commandes avec **docker-compose**. Ou **dockerfile**. Voici une liste non exhaustive de plusieurs applications prêtent à l'emploi.

1. Les fichiers docker-compose et dockerfile

Liste des fichiers **docker-compose** et **dockerfile** proposés en téléchargement libre.

Intitulé	Liens de téléchargement
Installation de ADGUARD	ADGUARD Docker-compose
Installation de CLONEZILLA	CLONEZILLA Docker-compose
Installation de DIUN	DIUN Docker-compose
Installation de DUPLICATI	DUPLICATI Docker-compose
Installation de GRAFANA	GRAFANA Docker-compose
Installation de LAMP	LAMP Docker-compose
Installation de NGINX	NGINX Docker-compose
Installation de WIREGUARD	WIREGUARD Docker-compose
Installation de LAMP	LAMP Dockerfile
Installation de DEBIAN WEB	DEBIAN Dockerfile

2. Installation d'une application via docker-compose

Le fichier docker-compose.yml au format YAML décrit tous les **docker run** possibles avec images, noms, volumes, liens, ports, variables d'environnements, réseaux... Il n'est dès lors plus nécessaire de mémoriser et introduire tous ces paramètres en ligne de commande : ils sont tous inventoriés en liste dans ce fichier unique. Si l'on maîtrise nos images, conteneurs et paramètres d'exécution, on peut facilement rédiger ce fichier.

a) Contenu d'un **docker-compose**

image : image docker à utiliser
container_name : nom du conteneur
environment : variables d'environnement à passer au conteneur
ports : correspondance des ports ouverts. Toujours les entourer de quotes
volumes : volumes à créer entre la machine hôte et le conteneur
build : si l'image doit être construite à partir d'un fichier Dockerfile
depends_on : si le conteneur dépend d'un autre pour son exécution
links : liens entre services

Attention l'indentation sur tout le fichier doit être respecter à la ligne près, sans quoi le déploiement échouera.

b) Installer de Docker-compose

```
# sudo docker-compose -v  # affiche la version
# sudo apt install docker-compose  # installation de docker-compose
```

c) Créer un dossier et y créer un fichier **docker-compose.yml** dedans

```
# sudo mkdir nomdudossier
# cd nomdudossier
# sudo nano docker-compose.yml
```

d) Construire un docker-compose.xml

```
version: '3'
 services:
   web:
   image: nginx:latest
   ports:
    -  "8080:80"
   volumes:
    - ./src:/usr/share/nginx/html
   links:
    - php
   php:
   image: php:7-fpm
```

d) Pousser le fichier **docker-compose.yml** via **docker-compose**

```
# sudo docker-compose -f docker-compose.yml up --detach
```

3. Installation d'une application via dockerfile

Le **Dockerfile** est un moyen de construire son image via un fichier docker file. Dans ce fichier Dockerfile, nous allons trouver l'ensemble de la recette décrivant l'image Docker dont on a besoin pour notre projet.

La première chose que l'on doit faire est de créer un fichier nommé **dockerfile**, puis de définir dans celui-ci l'image que l'on va utiliser comme base.

a) Contenu d'un dockerfile

```
FROM image:tag        # crée un calque à partir de l'image et sa version via Docker
LABEL version=1.0"    # Métadonnées de l'image
ARG DOCROOT           # Variable temporaire, utilisée dans le dockerfile
RUN apt install xxx   # exécute des commandes dans le conteneur
ADD test.txt path/    # ajoute des fichiers à partir du répertoire actuel de notre client Docker
COPY test.txt path/   # ajoute des fichiers à partir du répertoire actuel de notre client Docker
WORKDIR /dossier      # défini le répertoire de travail
CMD npm run start     # défini la commande par défaut lors de l'exécution du conteneur Docker
```

COPY et **ADD** agissent de la même manière, mais contrairement à **ADD**, **COPY** ne permet pas d'importer des documents venant d'une source distante telle qu'une URL. Dans la plupart des cas, on utilise **COPY** afin d'éviter des désagréments causés par l'utilisation de liens externes autorisés par **ADD**.

b) Créer un dossier et y créer un fichier **dockerfile** dedans

```
# sudo mkdir nomdudossier
# cd nomdudossier
# sudo nano dockerfile
```

c) Construire un **dockerfile** : pour l'exemple ce sera Debian avec des utilitaires

```
FROM debian:latest
LABEL version="1.0" maintainer=https://GDidées.eu
ARG APT_FLAGS="-yq"
RUN apt update ${APT_FLAGS} \
&& apt install apt-utils -yq \
&& apt install nano -yq \
&& apt install unzip -yq \
&& apt install axel -yq \
&& apt install apache2 libapache2-mod-php -yq \
&& apt install php php-cli -yq \
&& apt install php-mysql -yq \
&& apt install mariadb-server -y
EXPOSE 80
RUN /etc/init.d/apache2 start
RUN echo "<?php phpinfo(); ?>" > /var/www/html/test.php
ADD information /home/
CMD cat information
CMD 'sleep' 'infinity'
```

d) Lancer le dockerfile

```
# sudo docker build -t <nomimage> .  # le point est le dossier où se trouve le dockerfile
# sudo docker images  # vérifie la création de l'image
```

e) Créer le conteneur

```
# sudo docker run -d -p 8004:80 <nomimage>
```

4. Liens annexes

Liste de contenu à télécharger pour Docker et Portainer

- Ressources officiel de docker
- Documentation officiel de dockerfile

5. Conclusion

Les applications citées dans ce document sont utilisées par **Docker-compose** et **Dockerfile** pour gérer ses conteneurs, images, données, etc…

Destiné au RaspberryPi (Raspbian) avec Docker, **Docker-compose** ou **Dockerfile**, peuvent parfaitement installer une application complète dans un conteneur ou un **stack portainer** très facilement.