femto-st SCIENCES & TECHNOLOGIES

Solutions de virtualisation et de conteneurisation

Frédéric LASSABE

cnrs
UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ
UNIVERSITÉ DE FRANCHE-COMTÉ
SUP MICRO TECH ENSMM
utbm université de technologie Belfort-Montbéliard

La virtualisation

- Principe de virtualiser une machine
  - Processeur
  - Mémoire
  - Stockage
  - Autres périphériques
- Gestion par des instructions spécifiques du processeur
- Plusieurs VM dans une machine physique
- Repose sur un hyperviseur

Hyperviseurs

- VirtualPC (windows)
- VMWare (client, et pro sur datacenter)
- VirtualBox (poste de travail)
- Proxmox (datacenter)
- KVM (Linux, client et datacenter)

Conteneurisation

- Exécution d'un processus dans un environnement fermé
- Séparation des ressources de l'hôte
- Pas de virtualisation
- Certaines plateformes de virtualisation proposent aussi des conteneurs :
  - Proxmox : conteneurs LXC
- Exemples :
  - docker
  - podman

Conteneurisation

podman-compose

Choix de podman

- Similaire à docker (mêmes commandes)
- Ne demande pas d'être root
- Commande de base : **docker**
- Quelques manipulations avec LXC également

Cas d'usage

- Site web PHP, servi par Apache avec base de données mariadb.
- Choix 1 :
  - Un seul conteneur : apache, mariadb, php
- Choix 2 :
  - Un conteneur mariadb
  - Un conteneur apache/php

Lister les besoins

- Lister les besoins en logiciels
  - apache2 libapache2-mod-php php php-mysql mariadb-server mariadb-client
- Quels sont les ports utilisés
  - Port web 80
- Quelles sont les ressources
  - Fichiers du site
  - Fichier SQL de constitution de la BDD
  - Fichier script de déploiement

Étapes

- Écrire le dockerfile
  - Installation des packages
  - Configuration du port web
  - Appel au script de déploiement
- Écrire le script de déploiement
  - Copie des fichiers
  - Lancement de la BDD
  - Importer structure et données de la BDD
  - Lancer serveur web
- Écrire les fichiers du site

Exemple simple

```
1  FROM debian:latest
2
3  WORKDIR /home/
4
5  CMD echo 'Hello world!'
```

Image de conteneur basé sur debian ; affiche Hello world ! puis quitte.

Mots-clé importants

FROM	Image sur laquelle on base l'image créée. Ici : Debian
WORKDIR	Répertoire depuis lequel seront lancées les commandes. Important pour les chemins relatifs
CMD	Une seule CMD par fichier ! La commande que le conteneur exécutera au démarrage

Besoins pour serveur web

- Serveur Web (par ex : Apache)
- PHP
- Base de données (Mysql ou MariaDB)
- Connecteur PHP/BDD
- Module Apache PHP

Accès rapide au source

```
1  FROM debian:latest
2
3  WORKDIR /home/
4
5  RUN apt update
6  RUN apt install apache2 libapache2-mod-php \
7  php php-mysql mariadb-server mariadb-client -y
8
9  EXPOSE 80
10
11 CMD bash /home/start.sh
```

Nouveaux mots-clé

RUN Commandes lancées à la création du conteneur (une seule fois par commande build)

EXPOSE Ouvre un port accessible par l'hôte qui lance le conteneur. Permet l'accès à un serveur du conteneur.

Création de l'image

```
podman build -f Dockerfile -t my_image:latest
```

- Construit l'image
- nommée "my_image :latest" (option t)
- à partir de la configuration dans le fichier Dockerfile (option f)
- Avec tous les packages logiciels pour le web
- Lance la commande à partir d'un script

Script

```
#!/usr/bin/bash

echo "Starting"
cp -r /php/* /var/www/html/
service mariadb start
mysql < /home/test.sql
mysqladmin -u root password toto123
apache2ctl -DFOREGROUND
```

Prérequis

- Script qui repose sur plusieurs options
- Code PHP dans répertoire php, mappé avec option v
- Idem pour SQL dans /home
- Mappe (par ex) le port 8080 de l'hôte sur le port 80 du CT (option p)
- Option –rm pour supprimer le CT quand il a terminé
- Mappage dossier avec option :O pour lecture seule avec modification interne

Commande d'exécution du conteneur

```
podman run --rm \
 -p 8080:80 \
 -v $PWD/php/:/php:O \
 -v $PWD/resources/:/home:O \
 site:latest
```

Conteneurisation

podman-compose

Rôle

- Composer des services entre eux
- Définition plus compacte que Dockerfile
- Plusieurs services/conteneurs par fichier
- Syntaxe en Yaml (fichier suffixé .yml)
- Indique des dépendances de services
- Pour lancer dans l'ordre correct

Exemple de fichier compose – partie 1

```
1  version: '3.4'
2
3  services:
4    wordpress:
5      image: wordpress:${WP_VERSION}
6      ports: 8080:80
7      privileged: true
8      environment:
9        - WORDPRESS_DB_HOST=db
10       - WORDPRESS_DB_USER=${DB_USER}
11       - WORDPRESS_DB_PASSWORD=${DB_PASSWD}
12       - WORDPRESS_DB_NAME=${DB_NAME}
13     volumes:
```

Exemple de fichier compose – partie 2

```
1       - ./wordpress:/var/www/html
2     depends_on:
3       - db
4
5   db:
6     image: mariadb:${MARIADB_VERSION}
7     privileged: true
8     environment:
9       - MYSQL_ROOT_PASSWORD=${DB_ROOT_PASSWD}
10      - MYSQL_DATABASE=${DB_NAME}
11      - MYSQL_USER=${DB_USER}
12      - MYSQL_PASSWORD=${DB_PASSWD}
13    volumes:
14      - ./mysql:/var/lib/mysql
```

Remarque sur la syntaxe

- **image** définit l'image du conteneur (ex : wordpress, mariadb)
- **ports** définit le mappage des ports (option p de podman)
- **privileged : true** requis pour podman-compose
- **environment** variables passées au shell du conteneur
- **volumes** volumes mappés sur le disque de l'hôte
- **depends_on** dépendance à d'autres CT
- Rq : référence à **db** comme hôte MariaDB dans wordpress (substitution par IP du CT db)

Variables

- Exemple requiert des variables pour les id BDD
- Nom d'utilisateur, nom de BDD, etc.
- Dans un fichier nommé .env

```
1 WP_VERSION=5.7
2 MARIADB_VERSION=10.4
3 DB_USER=wordpress
4 DB_PASSWD=blahblah
5 DB_NAME=wordpress
6 DB_ROOT_PASSWD=blahblah
```

Conclusion

- Virtualisation utile pour des systèmes complets
- Souvent utilisée dans les services réseau
- Conteneurisation utile et fréquente dans le développement de web services
- Notamment microservices (chaque CT a un rôle, composition via API web)
- Plusieurs types de containeurs
- Commandes similaires entre podman et docker