HE-ARC > MAS-RAD > CAS-IDD > DSS

---

05 - Virtualisation et containers - Exercices

Raphaël P. Barazzutti - 18 février 2024

Exercice 1 : Comprendre la virtualisation

1. Différence entre un hyperviseur de type 1 (bare metal) et un hyperviseur de type 2 (hosted)
    - Hyperviseur de type 1 (bare metal)
        - S’exécute directement sur la machine physique, sans système d’exploitation hôte intermédiaire.
        - Accède quasi-nativement aux ressources matérielles, ce qui offre de meilleures performances.
        - Généralement utilisé en production et dans les data centers (ex.: VMware ESXi, Microsoft Hyper-V, Xen).
        - Avantage principal : performance élevée et accès direct au hardware.
        - Inconvénient : configuration plus complexe, maintenance pointue.
    - Hyperviseur de type 2 (hosted)
        - Fonctionne au-dessus d’un système d’exploitation hôte (Windows, Linux, macOS).
        - Performances un peu moins bonnes car dépend d’une couche logicielle supplémentaire (l’OS hôte).
        - Exemple : VirtualBox, VMware Workstation.
        - Avantage principal : mise en place plus simple pour le développement ou les tests.
        - Inconvénient : performances moindres, moins optimisé pour les gros environnements de production.
2. Cas pratique (recherche)
    - Type 1 :
        - VMware ESXi : solutions dans les grands data centers pour virtualiser de nombreux serveurs physiques.
        - Microsoft Hyper-V (mode bare metal) : intégré dans Windows Server, adapté aux environnements Windows.
        - Proxmox : solution basée sur KVM (directement intégré dans le noyau Linux) et fonctionnant sur une base de Linux Debian
        - Cas d’usage : data center où l’on souhaite la haute performance et la haute disponibilité.

- Type 2 :
  - VirtualBox : pratique pour les développeurs qui veulent rapidement monter des environnements de test.
  - VMware Workstation : usage poste client pour simuler plusieurs OS sur un même PC.
  - Cas d'usage : environnement de test ou de démonstration, besoin de souplesse sur un PC personnel.

Exercice 2 : Différences entre machines virtuelles et conteneurs

1. Question de compréhension théorique
   - Gestion des ressources
     - *VM* : chaque VM embarque son propre OS invité, ce qui alourdit la consommation en RAM et CPU.
     - *Conteneur* : partage le noyau de l'OS hôte, seul le système de fichiers et les processus sont isolés, donc plus léger.
   - Temps de démarrage
     - *VM* : démarrage d'un OS complet, plus long (plusieurs secondes, voire minutes).
     - *Conteneur* : initialisation très rapide (souvent moins d'une seconde) car le noyau est déjà en exécution.
   - Isolation du système
     - *VM* : isolation forte (chaque VM a son OS dédié), idéal pour héberger des environnements hétérogènes (Linux + Windows).
     - *Conteneur* : isolation au niveau des processus (via cgroups, namespaces). Plus léger, mais si l'OS de l'hôte est Linux, tous les conteneurs utilisent un noyau Linux.
   - Exemple d'application
     - Mieux en conteneur : un microservice web léger (Node.js, Go, Python Flask). Besoin de scale rapide, faible overhead.
     - Mieux en VM : une appli qui nécessite un OS complet différent (ex. Windows Server + Active Directory) ou un isolement critique.
2. Mise en situation (déploiement de 50 instances)
   - Avantage conteneurs : démarrage rapide, consommation moindre en ressources, orchestration plus simple (via Docker Compose ou Kubernetes).
   - VM : surcoût plus important (chaque VM embarque un OS), démarrage plus lent.

Exercice 3 : Créer une image Docker minimale

1. Écriture d'un Dockerfile
   - Exemple de Dockerfile (basé sur alpine pour un mini-serveur HTTP avec busybox):

```
# Nom du fichier : Dockerfile
FROM alpine:latest

# Installation des "extra" de busybox pour avoir le serveur
        httpd
RUN apk add busybox-extras

# Ajout de notre fichier html
RUN mkdir -p /var/www \
    && echo "Bonjour Docker" > /var/www/index.html

# Exposition du port 8080 (par exemple)
EXPOSE 8080

# Commande de lancement
CMD ["httpd", "-f", "-p", "8080", "-h", "/var/www"]
```

- Construction de l'image :

  ```
  docker build -t ex4 .
  ```

- Contrôle de la taille :

  ```
  docker images ex4
  ```

  (valeur obtenue ~10.5 MB)

2. Optimisation La commande `apk add` ramène la liste de paquets disponibles et l'enregistre dans Alpine.

   Une façon de faire est de lui demander de ne pas l'enregistrer avec `apk --no-cache add`

   Dans notre exemple en remplaçant la ligne par:

   ```
   RUN apk add --no-cache busybox-extras
   ```

   L'image ne fait que 8.0 MB

Exercice 4 : Docker Compose – application multi-conteneurs

fichier zip annexe contenant le code source.