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

---

05 - Virtualisation et containers - Exercices

Raphaël P. Barazzutti - 18 février 2024

Exercice 1 : Comprendre la virtualisation

1. Question de compréhension théorique
   - Expliquez la différence entre un hyperviseur de type 1 (*bare metal*) et un hyperviseur de type 2 (*hosted*).
   - Quels sont, selon vous, les avantages et inconvénients respectifs de ces deux approches?
2. Cas pratique (recherche)
   - Citez deux solutions logicielles pour chaque type d'hyperviseur (type 1 et type 2). Donnez un cas d'usage concret où vous recommanderiez chaque solution.

Exercice 2 : Différences entre machines virtuelles et conteneurs

1. Question de compréhension théorique
   - Décrivez en quoi un conteneur (type Docker) diffère d'une machine virtuelle en termes de :
     - Gestion des ressources
     - Temps de démarrage
     - Isolation du système
   - Donnez un exemple d'application qui se prêterait mieux à un déploiement sous forme de conteneur plutôt que de machine virtuelle, et inversement.
2. Mise en situation
   - Vous devez déployer 50 instances d'un service web léger. Précisez pourquoi vous opteriez (ou non) pour des conteneurs au lieu de VMs.

Exercice 3 : Créer une image Docker minimale

1. Écriture d'un Dockerfile
   - À partir d'une image de base légère (par ex. `alpine:latest`), construisez une image qui installe un petit serveur HTTP (ex. `httpd` ou `busybox httpd`) et affiche un message "Bonjour Docker".
   - Vérifiez la taille finale de l'image générée.
2. Optimisation
   - Appliquez des bonnes pratiques pour minimiser la taille :

  - Nettoyez le cache des paquets (si vous installez quoi que ce soit).
  - Utilisez si possible une unique instruction `RUN` pour grouper installation et nettoyage.
- Comparez la taille de l'image avant et après cette optimisation.

Exercice 4 : Docker Compose – application multi-conteneurs

1. Compose d'une application simple
   - Créez un fichier `docker-compose.yml` pour lancer deux conteneurs :
     1. Un conteneur backend (par exemple basé sur une `image python:alpine` qui lance un petit serveur Flask ou le petit serveur HTTP de l'exercice 3).
     2. Un conteneur reverse-proxy (basé sur une image `nginx:alpine`) qui redirige le trafic vers votre backend.
   - Assurez-vous que le conteneur frontend puisse communiquer avec le conteneur backend sur un réseau Docker dédié.
2. Tests
   - Lancez la stack avec `docker compose up -d`.
   - Vérifiez depuis votre navigateur (http://localhost) que vous voyez bien la page servie par votre backend (via le reverse proxy du frontend).