Virtualisation & Conteneurisation

1. Introduction générale

La virtualisation et la conteneurisation sont deux technologies essentielles permettant l'exécution d'environnements logiciels isolés. Elles favorisent la portabilité, l'efficacité et la sécurité dans les systèmes informatiques modernes.

Définition – Environnement (en informatique) : Un environnement informatique désigne un ensemble de ressources (matériel, logiciel, réseau) configuré pour exécuter, tester ou développer une application. Cela peut inclure un système d'exploitation, des bibliothèques, des outils spécifiques et des configurations réseau.

2. Virtualisation

La virtualisation est le processus consistant à créer une version virtuelle d'un environnement informatique. Elle permet d'exécuter plusieurs systèmes d'exploitation (OS) sur une seule machine physique, grâce à un logiciel appelé **hyperviseur**.

Un hyperviseur est un composant clé qui permet de créer et de gérer des machines virtuelles (VM). Il en existe deux types :

- **Hyperviseur de type 1 (bare-metal)** : fonctionne directement sur le matériel physique. Exemples : VMware ESXi, Microsoft Hyper-V, Xen, KVM.
- **Hyperviseur de type 2 (hosted) :** fonctionne comme une application installée sur un OS hôte. Exemples : VirtualBox, VMware Workstation, Parallels Desktop.

Comparaison :

Critère	Hyperviseur Type 1	Hyperviseur Type 2
Niveau d'installation	Directement sur le matériel	Sur un système d'exploitation
Performances	Excellentes	Moins bonnes (surcharge hôte)
Sécurité	Plus élevée	Moins élevée
Facilité d'usage	Moins convivial	Plus simple à installer
Usage typique	Data centers, serveurs pro	Tests, développement local

Avantages généraux de la virtualisation :

- Isolation forte entre les VMs
- Exécution de plusieurs OS indépendants

Inconvénients :

- Consommation importante de ressources
- Temps de démarrage plus long

3. Conteneurisation sous Linux

La conteneurisation est une alternative plus légère à la virtualisation. Elle consiste à empaqueter une application avec toutes ses dépendances dans un **conteneur**.

Contrairement à la virtualisation, les conteneurs partagent le même noyau Linux de l'hôte, ce qui les rend plus légers et rapides à démarrer.

Principes techniques sous-jacents :

- `chroot` : restreint un processus à un sous-dossier du système de fichiers
- `namespaces` : isolent les ressources (PID, réseau, utilisateurs, etc.)
- `cgroups` : contrôlent l'utilisation des ressources (CPU, mémoire, disque)
- `unionFS` (overlay) : système de fichiers empilable

Outils populaires : Docker, Podman, LXC, containerd

Avantages :

- Rapidité
- Portabilité
- Légèreté

Limites :

- Isolation plus faible
- Dépendance au noyau de l'hôte

4. Comparatif : Virtualisation vs Conteneurisation

Critère	Virtualisation	Conteneurisation
Isolation	Forte	Moyenne
Démarrage	Lent	Rapide
Portabilité	Moyenne	Excellente
Poids	Lourd	Léger
Sécurité	Élevée	Moins isolée
Cas d'usage	Infrastructure cloud	Microservices, DevOps

5. Cas concrets d'usage

- **Virtualisation :** serveurs cloud (AWS EC2), tests multi-OS, machines critiques nécessitant une isolation forte.
- **Conteneurisation :** microservices, pipelines CI/CD, orchestration avec Kubernetes, déploiement rapide d'applications.

6. Commandes utiles

Commandes Vagrant

Commande	Utilité

Commande	Utilité
`vagrant init`	Initialise un fichier Vagrantfile pour configurer une VM
`vagrant up`	Démarre et provisionne la VM
`vagrant ssh`	Se connecte en SSH à la VM
`vagrant halt`	Éteint proprement la machine
`vagrant suspend`	Met la VM en pause
`vagrant resume`	Reprend une VM suspendue
`vagrant destroy`	Supprime complètement la VM
`vagrant reload`	Redémarre la VM et recharge les fichiers de configuration
`vagrant provision`	Re-exécute le provisioning (scripts de configuration)
`vagrant box add`	Ajoute une box (image de base)
`vagrant box list`	Liste toutes les box disponibles localement
`vagrant box remove`	Supprime une box
`vagrant status`	Affiche l'état de la VM
`vagrant plugin install`	Installe un plugin
`vagrant global-status`	Affiche le statut de toutes les VMs gérées par Vagrant sur la machine

Commandes Docker

Commande	Utilité
`docker build -t nom_image .`	Crée une image Docker à partir d'un Dockerfile
`docker pull nom_image`	Télécharge une image depuis Docker Hub
`docker push nom_image`	Pousse une image sur un registre (Docker Hub, etc.)
`docker run -it nom_image`	Lance un conteneur en mode interactif
`docker run -d -p 8080:80 nom_image`	Lance un conteneur détaché avec mappage de port
`docker ps`	Liste les conteneurs en cours d'exécution
`docker ps -a`	Liste tous les conteneurs (même arrêtés)
`docker stop id_conteneur`	Arrête un conteneur
`docker start id_conteneur`	Démarre un conteneur déjà créé
`docker restart id_conteneur`	Redémarre un conteneur
`docker rm id_conteneur`	Supprime un conteneur
`docker rmi nom_image`	Supprime une image
`docker images`	Liste toutes les images locales
`docker exec -it id bash`	Ouvre un terminal interactif dans un conteneur
`docker logs id`	Affiche les logs d'un conteneur

Commande	Utilité
`docker volume create nom_volume`	Crée un volume
`docker volume ls`	Liste les volumes
`docker network create nom_reseau`	Crée un réseau Docker personnalisé
`docker-compose up`	Lance les services définis dans un `docker-compose.yml`
`docker-compose down`	Arrête et supprime les services créés avec Docker Compose

7. QCM

- Le fichier "box" est systématiquement téléchargé quand on fait "vagrant reload" sur une VM dont l'état est "démarrée".

  → **Faux**

- Le fichier "box" est systématiquement téléchargé quand on fait "vagrant up" aussitôt après "vagrant destroy".

  → **Faux**

- Supposons qu'une VM de Vagrant est dans l'état "running", dans quel état passe-t-elle si on appelle la commmande "vagrant reload" ?

  → **démarrée**

- Supposons qu'une VM de Vagrant est dans l'état "poweroff", dans quel état passe-t-elle si on appelle la commmande "vagrant up" ?

  → **démarrée**

- Supposons qu'une VM de Vagrant est dans l'état "saved", dans quel état passe-t-elle si on appelle la commmande "vagrant reload" ?

  → **démarrée**

- Quelle commande permet de passer une VM de l'état "poweroff" à l'état "saved" ?

  → **aucune**

- Quelle commande permet de passer une VM de l'état "running" à l'état "saved" ?

  → **suspend**

- Si l'adresse IP de la VM est 192.168.1.34/24 et celle de l'hôte est 192.168.1.50/24, quel est le type de connexion réseau utilisé par la VM ?

  → **bridged**

- Compléter le Dockerfile ci-dessous afin de permettre la construction d'une image fonctionnelle.

→
```
FROM eclipse-temurin:11-jre-alpine
RUN mkdir /opt/app
COPY score-jre11.jar /opt/app
CMD ["java", "-jar", "/opt/app/score-jre11.jar"]
Voici le contenu du répertoire courant :
user@debian:~/qcm$ ls -l
total 0
-rw-r--r-- 1 user user 0  2 févr. 13:29 Dockerfile
-rw-r--r-- 1 user user 0  2 févr. 13:29 score-jre11.jar
```

- Qu'effectue la commande : docker image pull ubuntu

  → **Elle télécharge depuis le registre l'image ubuntu:latest**

- Que fait la commande : docker container commit linux-proxy ubuntu-tp3-commit-img

  → **Elle crée une image ubuntu-tp3-commit-img:latest à partir du conteneur linux-proxy**

- Etant donné les conteneurs et images suivantes:

```
ml@r408-ml:~$ docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
latest                  latest    ff7475f973fc   32 minutes ago   126MB
ubuntu-tp3-dockerfile   latest    07e3b2195b41   32 minutes ago   126MB
ubuntu-tp3-commit       latest    07e3b2195b41   32 minutes ago   126MB
postgres                latest    a26eb6069868   7 days ago       379MB
busybox                 latest    827365c7baf1   8 days ago       4.86MB
ubuntu                  22.04     6b7dfa7e8fdb   3 weeks ago      77.8MB
ubuntu                  20.04     d5447fc01ae6   3 weeks ago      72.8MB
hello-world             latest    feb5d9fea6a5   15 months ago    13.3kB

ml@r408-ml:~$ docker ps -a
CONTAINER ID   IMAGE                   COMMAND                  CREATED              STATUS

96b465a12859   ubuntu-tp3-commit       "bash"                   About a minute ago   Exited (0) About a minute ago

30da76ec49ad   ubuntu-tp3-dockerfile   "bash"                   2 minutes ago        Exited (0) 2 minutes ago

91ef4aa165e7   ubuntu-tp3-dockerfile   "bash"                   2 minutes ago        Exited (0) 2 minutes ago

9aaef3ec2b76   hello-world             "/hello"                 3 minutes ago        Exited (0) 3 minutes ago

cbcaf3a91ced   6b7dfa7e8fdb            "/bin/sh -c 'apt upd…"   32 minutes ago       Exited (1) 32 minutes ago

6eb06734c35d   busybox                 "sh"                     6 hours ago          Exited (0) 6 hours ago

5d1f090bb102   ubuntu:20.04            "bash"                   6 hours ago          Exited (0) About an hour ago
```

  Que va nous retourner l'invité de commande à l'exécution de : docker container rm 6eb06734c35d

  → **La confirmation de la suppression du conteneur 6eb06734c35d provenant d'une image busybox**

- Sélectionner les affirmations vraies concernant un conteneur Docker.

  → **Toutes les modifications apportées à son système de fichiers sur la « couche conteneur » sont perdues en cas de suppression du conteneur**

  → **C'est une instance exécutable d'une image Docker**

  → ~~Par défaut, il a accès au système de fichiers de son hôte~~

  → ~~Il peut consulter les processus de la machine hôte~~

- L'inspection de l'image busybox:latest nous permet d'extraire les informations suivantes:

```
"Config": {
"Hostname": "",
"Domainname": "",
"User": "",
"AttachStdin": false,
"AttachStdout": false,
"AttachStderr": false,
"Tty": false,
"OpenStdin": false,
"StdinOnce": false,
"Env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
],
"Cmd": [
    "sh"
],
"ArgsEscaped": true,
"Image": "",
"Volumes": null,
"WorkingDir": "/",
"Entrypoint": null,
"OnBuild": null,
"Labels": null
}
```


En vous appuyant sur ces informations, pouvez-vous indiquer comment va être considéré "ls -ali" dans la commande suivante ?
`docker container create --name busy01 busybox:latest ls -ali`

→ **Comme la commande à exécuter en lieu et place de celle par défaut dans l'image busybox:latest**

- Etant donné les images et les conteneurs suivants:

```
ml@r408-ml:~$ docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
latest                  latest    ff7475f973fc   32 minutes ago   126MB
ubuntu-tp3-dockerfile   latest    07e3b2195b41   32 minutes ago   126MB
ubuntu-tp3-commit       latest    07e3b2195b41   32 minutes ago   126MB
postgres                latest    a26eb6069868   7 days ago       379MB
busybox                 latest    827365c7baf1   8 days ago       4.86MB
ubuntu                  22.04     6b7dfa7e8fdb   3 weeks ago      77.8MB
ubuntu                  20.04     d5447fc01ae6   3 weeks ago      72.8MB
hello-world             latest    feb5d9fea6a5   15 months ago    13.3kB

ml@r408-ml:~$ docker ps -a
CONTAINER ID   IMAGE                   COMMAND                  CREATED              STATUS

96b465a12859   ubuntu-tp3-commit       "bash"                   About a minute ago   Exited (0) About a minute ago

30da76ec49ad   ubuntu-tp3-dockerfile   "bash"                   2 minutes ago        Exited (0) 2 minutes ago

91ef4aa165e7   ubuntu-tp3-dockerfile   "bash"                   2 minutes ago        Exited (0) 2 minutes ago

9aaef3ec2b76   hello-world             "/hello"                 3 minutes ago        Exited (0) 3 minutes ago

cbcaf3a91ced   6b7dfa7e8fdb            "/bin/sh -c 'apt upd…"   32 minutes ago       Exited (1) 32 minutes ago

6eb06734c35d   busybox                 "sh"                     6 hours ago          Exited (0) 6 hours ago

5d1f090bb102   ubuntu:20.04            "bash"                   6 hours ago          Exited (0) About an hour ago
```

Que va nous retourner l'invité de commande à l'exécution de : `docker image rm busybox`

→ **Une erreur car l'image busybox est actuellement utilisée par un conteneur**

- Sélectionner les affirmations vraies concernant une image Docker.

→ **C’est un modèle immuable constitué de différentes couches**

→ **C’est une méthode pour construire des conteneurs de manière reproductible**

→ ~~C’est la même chose qu'un Dockerfile~~

→ ~~C’est une instance exécutable d’un conteneur~~

- L’interface en ligne de commande (CLI) de Docker est obligatoirement sur une machine différente de l’hôte (la machine hébergeant le démon Docker).

  → **Faux**

- Quelle est la fonction de l'option -p 80:8080 dans la commande : docker container run -d -p 80:8080 --name ws-score-jre11 score:jre11

  → **Elle mappe le port TCP 8080 à l'intérieur du conteneur sur le port TCP 80 de la machine hôte**