C09-docker - COURS LINUX - MTN

Guillaume ASTIER

26/02/16

Table des matières

La virtualisation

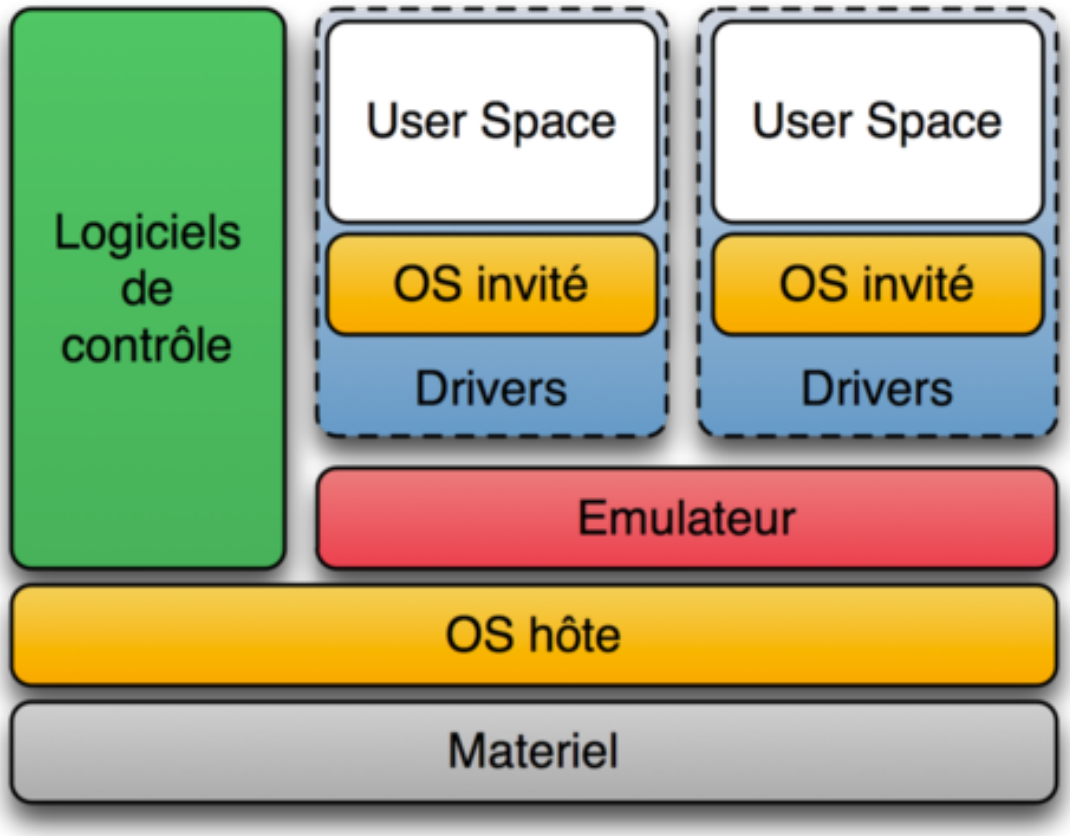
Le fonctionnement de la virtualisation

Mutualisation de plusieurs serveurs virtuels depuis un serveur physique grâce à un logiciel nommé l'hyperviseur. L'hyperviseur permet d'émuler intégralement les différentes ressources matérielles

d'un serveur physique :

- l'unité centrale
- le CPU
- la RAM
- le disque dur
- carte réseau
- etc …

Les VM (Virtual Machine) accèdent à des ressources matérielles selon leurs besoins (par exemple plus de puissance processeur et plus de !mémoire vive mais avec moins d'espace disque) Mes modifications de ses caractéristiques ce fait simplement et s'adapte au besoin.



Les avantages de virtualisation

- Consacrer les ressources adaptées selon les applications qu'on souhaite mettre en place.
- Les machines virtuelles reste simples à manier.
- Il est possible par exemple de basculer une VM d'un lieu à l'autre voir même de sauvegarder et de dupliquer une VM à volonté sans aucun impact visible pour les utilisateurs.
- La virtualisation réduit les dépenses en abaissant le besoin de systèmes matériels physiques.
- Elle permet ainsi de réduire la quantité d'équipement nécessaire et les coûts de maintenance d'alimentation et de refroidissement des composants.
- Les machines virtuelles apportent également une aisance à l'administration car un matériel virtuel n'est pas sujet aux défaillances.
- Les administrateurs profitent des environnements virtuels pour faciliter les sauvegardes, la reprise après catastrophe.

Les inconvénients de la virtualisation

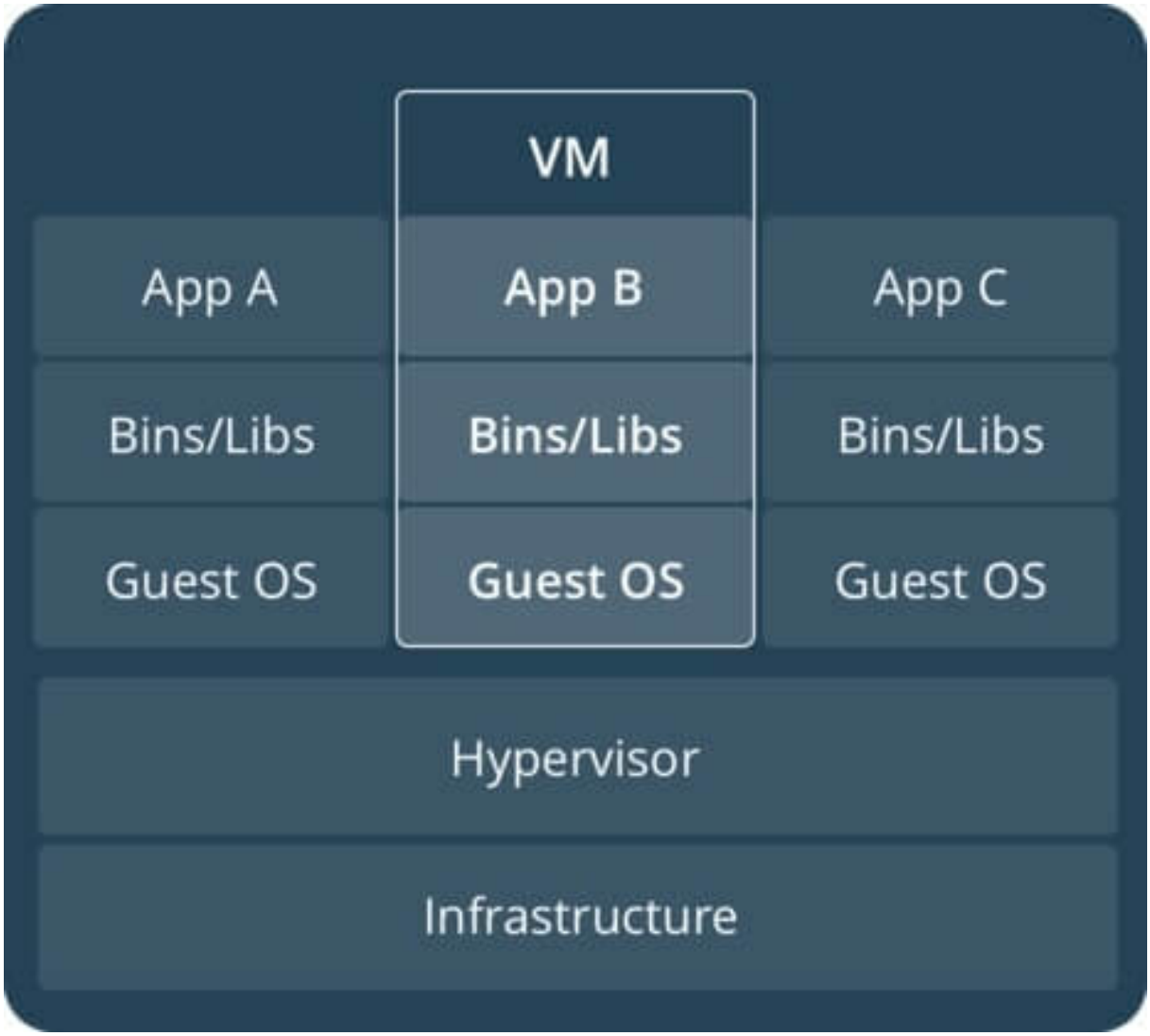
- Le fait d'accéder aux ressources de façon virtuelle affaiblie les performances, cela est dû car on passe par une couche d'abstraction matérielle qui malheureusement doit faire des interprétations entre le matériel en place et celui simulé dans la machine virtuelle.
- Comme expliqué plus haut la virtualisation consiste à faire fonctionner sur un seul ordinateur physique plusieurs VMs avec différents systèmes d'exploitation, comme s'ils fonctionnaient sur des ordinateurs distincts.
- Malheureusement cette couche d'OS consomme à lui tout seul énormément de ressources alors qu'au final, ce qui nous intéresse c'est la ou les applications qui vont tourner dessus.

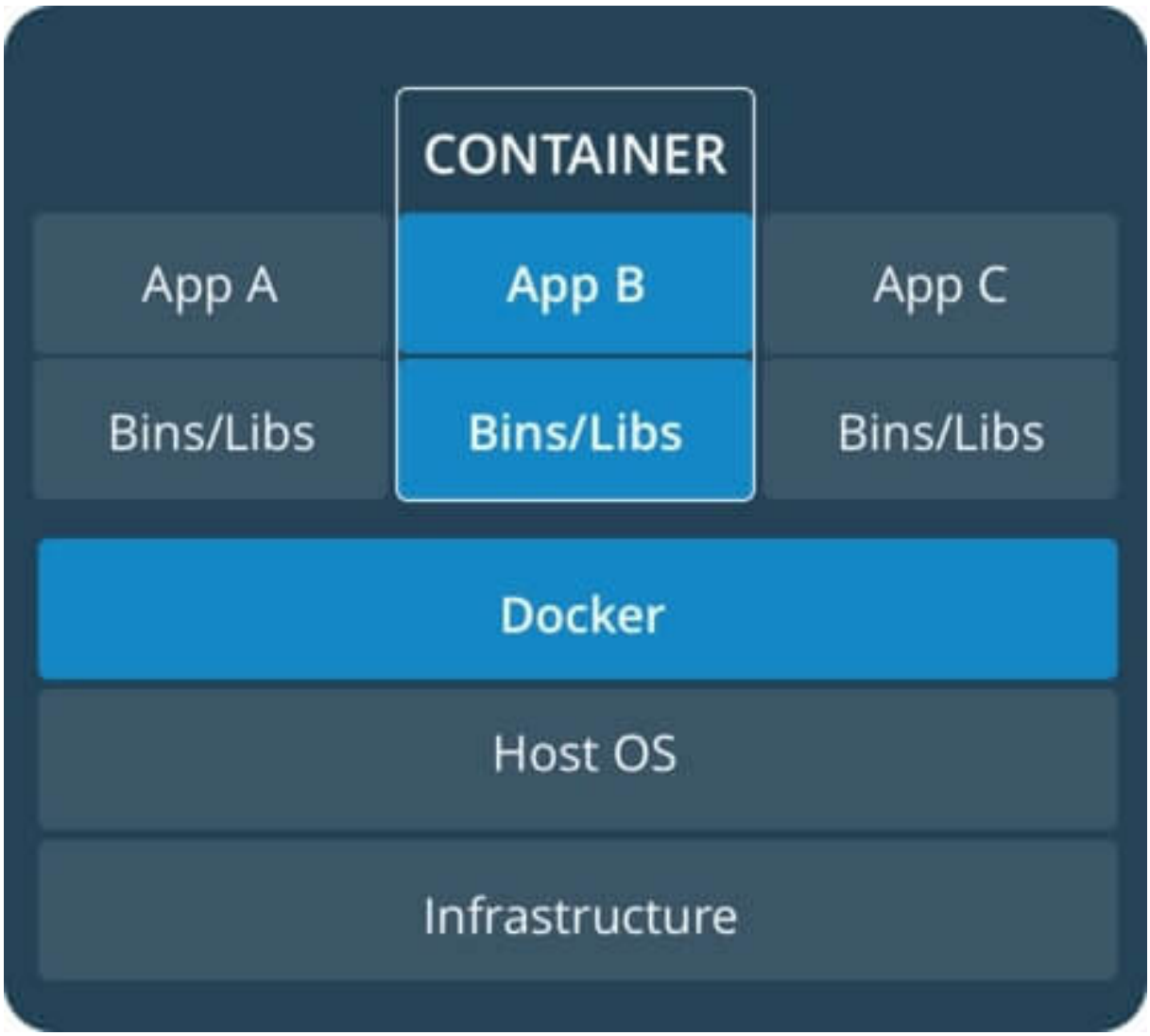
Container

Isolation

- Dans le cas de la virtualisation l'isolation des VMs se fait au niveau matérielles (CPU/RAM/-Disque) avec un accès virtuel aux ressources de l'hôte via un hyperviseur.
- Généralement les ordinateurs virtuels fournissent un environnement avec plus de ressources que la plupart des applications n'en ont besoin.
- Par contre dans le cas de la conteneurisation, l'isolation se fait au niveau du système d'exploitation.
- Un conteneur va s'exécuter sous Linux de manière native et va partager le noyau de la machine hôte avec d'autres conteneurs.

- Ne prenant pas plus de mémoire que tout autre exécutable, ce qui le rend léger.





Avantages docker / virtu

- Comme vu plus haut les machines virtuelles intègrent elles-mêmes un OS pouvant aller jusqu'à des Giga-octets.
- Ce n'est pas le cas du conteneur. Le conteneur appel directement l'OS pour réaliser ses appels système et exécuter ses applications.
- Il est beaucoup moins gourmand en ressources
- Le déploiement est un des points clés à prendre en compte de nos jours.
- On peut déplacer les conteneurs d'un environnement à l'autre très rapidement
- On peut bien sur faire la même chose pour une machine virtuelle en la déplaçant entièrement de serveurs en serveurs

- Un déploiement de VM d'OS à OS rendra le déploiement beaucoup plus lent

Les avantages de conteneurisation

La conteneurisation est de plus en plus populaire car les conteneurs sont :

- Flexible: même les applications les plus complexes peuvent être conteneurisées.
- Léger: les conteneurs exploitent et partagent le noyau hôte.
- Interchangeable: vous pouvez déployer des mises à jour à la volée
- Portable: vous pouvez créer localement, déployer sur le cloud et exécuter n'importe où votre application.
- Évolutif: vous pouvez augmenter et distribuer automatiquement les réplicas (les clones) de conteneur.
- Empilable (stackable): Vous pouvez empiler des services verticalement et à la volée.

fichier / cmd Docker

Dockerfile

Le Dockerfile permet la construction d'une image. Les commandes internes sont :

- ADD Add local or remote files and directories.
- ARG Use build-time variables.
- CMD Specify default commands.
- COPY Copy files and directories.
- ENTRYPOINT Specify default executable.
- ENV Set environment variables.
- EXPOSE Describe which ports your application is listening on.
- FROM Create a new build stage from a base image.
- HEALTHCHECK Check a container's health on startup.

- LABEL Add metadata to an image.
- MAINTAINER Specify the author of an image.
- ONBUILD Specify instructions for when the image is used in a build.
- RUN Execute build commands.
- SHELL Set the default shell of an image.

- STOPSIGNAL Specify the system call signal for exiting a container.
- USER Set user and group ID.
- VOLUME Create volume mounts.
- WORKDIR Change working directory.

Build

pour Builder une image avec le Dockerfile il faut lancer la commande :

```
# se placer au même niveau que le Dockerfile
docker build -t [image name] .
```

Run

Pour lancer une instance docker il faut executer la commande :

```
docker run [image name]

# Avec des options supplémentaire (mode daemon, nom du process docker, partage de port, partage de volume hôte/instance)
docker run -d  --name=WebMe -p 333:4444 -v /home/isen/docker/newpyth/www:/var/www pythone
# le -d indique daemon (en "arrière plan")

docker run -ti  --name=WebMe -p 333:4444 -v /home/isen/docker/newpyth/www:/var/www pythone
# le -t indique le mode terminal. Affichera tous les logs
# le -i indique interactif et vous permet d'intéragire avec le process

#Pour stop une instance :
docker stop [instance name]
```

list les instances en cours

```
root@docker:/home/isen/docker/newpyth# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
85ea168c3323 pythone:latest "/bin/sh -c /sbin/st..." 6 minutes ago Up 5 minutes 0.0.0.0:80-\>4444/tcp WebOne
# Stop de l'instance
root@docker:/home/isen/docker/newpyth# docker stop e55976589871
e55976589871
# ou avec le nom de l'instance
```

```
root@docker:/home/isen/docker/newpyth# docker stop WebOne
WebOne
# Suppression de l'instance
root@docker:/home/isen/docker/newpyth# docker rm e55976589871
e55976589871
# Ou avec le nom de l'instance
root@docker:/home/isen/docker/newpyth# docker rm
WebOne
```

Ex python web Dockerfile

```
    root@docker:/home/isen/docker/newpyth# cat Dockerfile
    FROM debian:latest
    RUN apt update
    COPY ./start.sh /sbin/start.sh
    RUN chmod +x /sbin/start.sh
    RUN apt install python -y
    RUN mkdir /var/www
    CMD /sbin/start.sh
```

Ex python web build

```
    root@docker:/home/isen/docker/newpyth# docker build . -t pythone
    Sending build context to Docker daemon  5.632kB
    Step 1/7 : FROM debian:latest
     ---> c4905f2a4f97
    Step 2/7 : RUN apt update
     ---> Using cache
     ---> 156d9dcd3d3a
    [...]
    Step 7/7 : CMD /sbin/start.sh
     ---> Using cache
     ---> 864b75e537dc
    Successfully built 864b75e537dc
    Successfully tagged pythone:latest
```

Ex python web run

```
root@docker:~/newpyth# docker run -d  --name=WebMe -p 333:4444 \
-v $(pwd)/newpyth/www:/var/www pythone

e05ea5918438b417c7544bf5a6d3daf9c232e21927f9c7407cb4556ac002776f

root@docker:~/newpyth# docker ps
```

```
CONTAINER ID IMAGE   COMMAND     CREATED  STATUS  PORTS         NAMES
e05ea5918438 pythone "start.…sh" 7sec ago Up 5sec 333->4444/tcp WebMe
```

Ex python web : Vérification du service

```
root@docker:/home/isen/docker/newpyth# cat www/index.html
<pre>

I am alive !!!

</pre>
root@docker:/home/isen/docker/newpyth# curl http://localhost:333
<pre>

I am alive !!!

</pre>
```

docker compose

docker-compose.yml

Les options peuvent être rébarbative dans le cadre d'un grand nombre d'instance.

Nous pouvons utiliser un "orchestrateur" comme docker-compose

Ce dernier à travers un fichier de configuration en yml (docker-compose.yml) permet de démarrer les instances qui y sont configurées.

ex python web docker-compose.yml

Ici nous allons configuré sur la base de l'image pythone 2 instances docker sur les port 80 et 443

```
root@docker:/home/isen/docker/newpyth# cat docker-compose.yml
version: "3.3"
services:
  WebOne:
    image: pythone:latest
    volumes:
      - "./www:/var/www"
    ports:
      - "80:4444"
  WebTwo:
    image: pythone:latest
```

```
12          volumes:
13            - "./www:/var/www"
14          ports:
15            - "443:4444"
```

Ex python web : docker-compose run

```
1  root@docker:/home/isen/docker/newpyth# docker-compose up -d
2  Starting newpyth_WebTwo_1 ... done
3  Starting newpyth_WebOne_1 ... done
4  root@docker:/home/isen/docker/newpyth# docker ps
5  CONTAINER ID IMAGE   COMMAND CREATED  STATUS PORTS         NAMES
6  85ea168c3323 pythone "…st"   1min ago 3sec   80->4444/tcp
       newpyth_WebOne_1
7  e55976589871 pythone "…st"   1min ago 3sec   443->4444/tcp
       newpyth_WebTwo_1
```

Ex python web : docker-compose stop

```
1      root@docker:/home/isen/docker/newpyth# docker-compose down
2      Stopping newpyth_WebTwo_1 ... done
3      Stopping newpyth_WebOne_1 ... done
4      Removing newpyth_WebTwo_1 ... done
5      Removing newpyth_WebOne_1 ... done
6      Removing network newpyth_default
```

Ex python web : Verification

```
1      root@docker:/home/isen/docker/newpyth# curl -sq http://localhost:80
2      <pre>
3
4      I am alive !!!
5
6      </pre>
7      root@docker:/home/isen/docker/newpyth# curl -sq http://localhost
           :443
8      <pre>
9
10     I am alive !!!
11
12     </pre>
```

docker connexion sur instance

docker permet de se “connecter” sur une instance avec son shell ou du moins lancer un processus autre que celui pour le quel il a été créé

```
1
2     root@docker:/home/isen/docker/newpyth# docker ps
3  CONTAINER ID IMAGE   COMMAND CREATED  STATUS PORTS         NAMES
4  85ea168c3323 pythone "…st"   1min ago 3sec   80->4444/tcp
      newpyth_WebOne_1
5  e55976589871 pythone "…st"   1min ago 3sec   443->4444/tcp
      newpyth_WebTwo_1
6
7  root@e55976589871:/# ps axf
8      PID TTY      STAT   TIME COMMAND
9       93 pts/0    Ss     0:00 bash
10     100 pts/0    R+     0:00  \_ ps axf
11       1 ?        Ss     0:00 /bin/sh -c /sbin/start.sh
12       8 ?        S      0:00 /bin/bash /sbin/start.sh
13       9 ?        S      0:00  \_ python -m SimpleHTTPServer 4444
```

docker lancement de processus

Dans l’exemple precedent nous avons lancé un processus bash permettant de se “connecter” en shell sur une instance docker.

/bin/bash étant un binaire comme un autre il est tout a fait possible de lancer autre chose.

```
1  # Acune instance ne tourne acutellement sur le serveur.
2  root@docker:/home/isen/docker/newpyth# docker ps
3  CONTAINER ID IMAGE COMMAND CREATED SATUS PORTS NAMES
```

```
1  # Verification de la présence de l'image
2  root@docker:~/newpyth# docker image ls | grep  pythone
3  pythone latest     8698631e54b9   7 years ago     15.2MB
4
5  # Lancement de l'instance pythone en mode par default
6  root@docker:/home/isen/docker/newpyth# docker run -d pythone
7  # Lancement de l'instance pythone avec un autre binaire
8  root@docker:/home/isen/docker/newpyth# docker run -d date
9  lun. 15 avril 2024 22:05:17 CEST
```