UNIVERSITÉ CAEN NORMANDIE

Rappels sur les conteneurs

Virtualisation avancée – TD

IUT GRAND OUEST NORMANDIE

Maxime Lambert

2024 / 2025

Modalités

- Travail en binôme en salle de machines.
  Vous devez former un binôme avec une personne présente dans le même groupe de TP que vous.

- Matériels ou logiciels utilisés **:**
  - Accès au serveur Proxmox de l'université (https://proxmox-iutc3.unicaen.fr) depuis un navigateur web

- L'objectif de ce TD est de revoir les bases sur les conteneurs et Docker que nous avons étudié dans le module R4.08 lors du semestre 4.

- Une bonne compréhension des conteneurs vous permettra d'aborder sereinement les travaux que nous effectuerons sur Kubernetes tout au long de ce module.

- Le CM du module R4.08 est disponible en téléchargement sur ce lien.

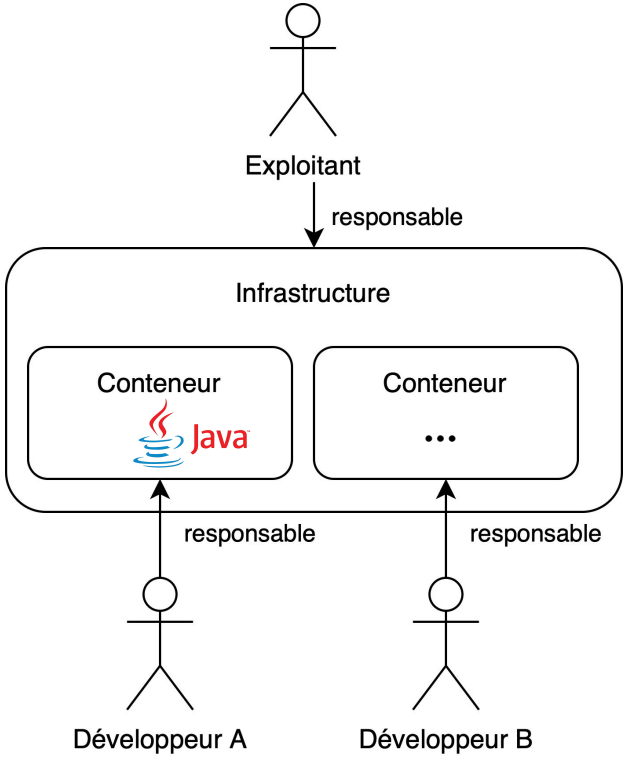
1. Rappels sur les conteneurs

Les conteneurs en informatique

Dans une architecture à base de conteneurs:

- **Le développeur a la responsabilité du contenu du conteneur**: l'applicatif, ses dépendances et sa configuration

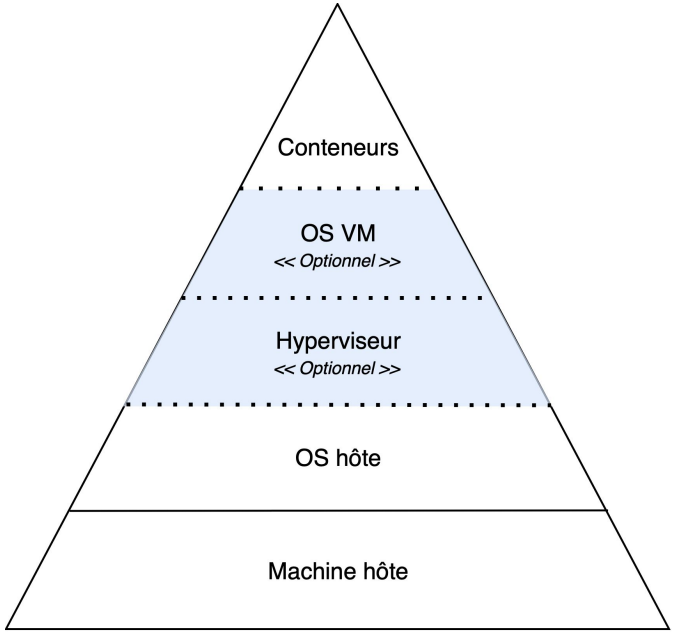
- **L'exploitant a la responsabilité de la gestion du conteneur** au sein de son infrastructure: déploiement, réseau, stockage et allocation de ressources



1. Rappels sur les conteneurs

Lien avec la virtualisation

- La conteneurisation est également qualifiée de **virtualisation au niveau du système d'exploitation**.
- Un conteneur **ne contient pas un système d'exploitation.**
  Il s'appuie sur le noyau de la machine hôte l'accueillant
- La portabilité d'un conteneur **est limitée à des *OS* de la même famille**.
- Machines virtuelles et conteneurs ne sont pas incompatibles.
  Il est possible d'exécuter des conteneurs dans des VM.



1. Rappels sur les conteneurs

Qu'est ce qu'un conteneur ?

- Un conteneur dispose de **son propre système de fichiers** et permet l'exécution de processus dans **un environnement isolé et contraint**.
- **L'isolation est possible grâce aux espaces de noms**. Il s'agit d'une fonctionnalité du noyau Linux permettant à des ensembles de processus de ne voir qu'un ensemble de ressources.
- **Le contrôle de la consommation des ressources** (cpu, mémoire, réseau…) **est assuré par la librairie CGroups** du noyau Linux.

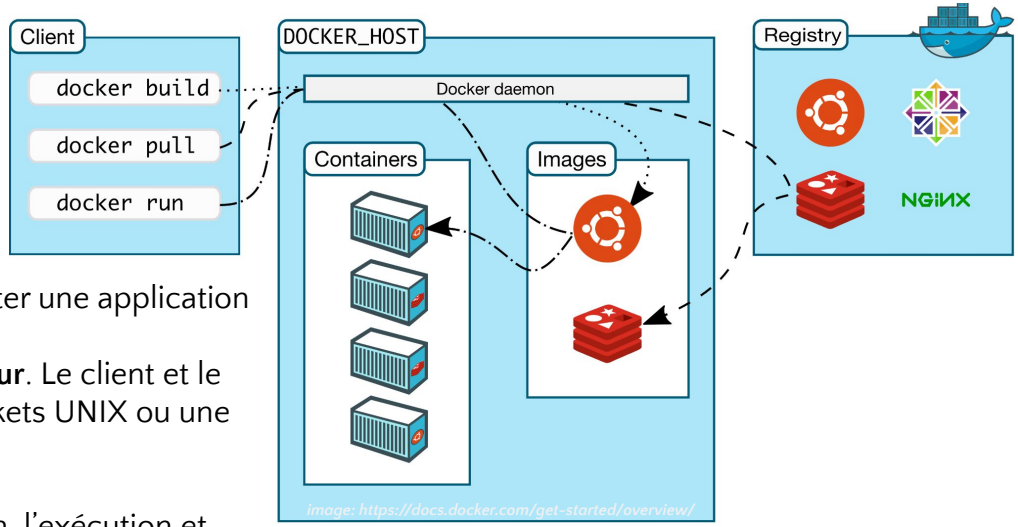
1. Rappels sur les conteneurs

Usages et caractéristiques

- Les conteneurs ont des cas d'utilisation similaires aux machines virtuelles. Toutefois, dans la philosophie qui leur est associée, **les conteneurs ont généralement vocation à être éphémères, et si possible, sans état**.
- De par leur nature, les applications au sein de conteneurs sont **moins consommatrices de ressources** que des VM.
- **Les conteneurs peuvent très facilement migrer sur n'importe quelle machine**, du moment qu'elle dispose d'un OS de la même famille que la machine ayant permis de créer le conteneur.
- Ne devant pas amorcer un OS complet, **les conteneurs sont plus rapides à démarrer et éteindre que des machines virtuelles**. C'est une fonctionnalité intéressante dans les environnements de *cloud computing* afin de pratiquer de la scalabilité à la demande.
- En dehors des applications conteneurisées misent en production, on retrouve fréquemment les conteneurs dans les environnements de développement (instanciation de bdd, *broker*...), de qualification (afin de jouer des tests automatisés ou manuels) et sur les pipelines d'intégration et de déploiement continu.

2. Rappels sur la plateforme Docker

L'architecture de Docker



- Docker est une **plateforme ouverte** pour le développement, la livraison et l'exécution d'applications
- Le conteneur devient l'unité pour distribuer et tester une application
- Docker est basé sur une **architecture client-serveur**. Le client et le démon communiquent via une API REST, des sockets UNIX ou une interface réseau.
- Le démon Docker est en charge de la construction, l'exécution et la distribution des conteneurs.
- Il existe 2 principaux clients:
  - `docker cli` exposant un interface en ligne de commande
  - `docker-compose` permettant de définir et d'exécuter une application constituée d'un ensemble de conteneurs.
- Le registre est un serveur assurant le stockage et la distribution des images.

2. Rappels sur la plateforme Docker

client – l'interface de commandes docker

- Docker dispose d'une interface en ligne de commandes (*CLI*). C'est le moyen privilégié d'interagir avec le démon lorsque l'on travaille sur des conteneurs autonomes ou afin d'administrer les différents objets Docker (conteneurs, images, volumes, réseaux…)

- Les commandes suivent la syntaxe suivante: `docker <<objet>> <<commande>> <<options>>`
  - exemple pour afficher la liste des conteneurs: `docker container ls`

- La ligne de commande Docker est configurable via des variables d'environnement (DOCKER_HOST, DOCKER_CONFIG…)

- Pour l'utilisation, se référer à la documentation en ligne[1] et celle obtenue avec la commande `help`.

[1] *https://docs.docker.com/engine/reference/commandline/cli/*

2. Rappels sur la plateforme Docker

Les objets fondamentaux - Image

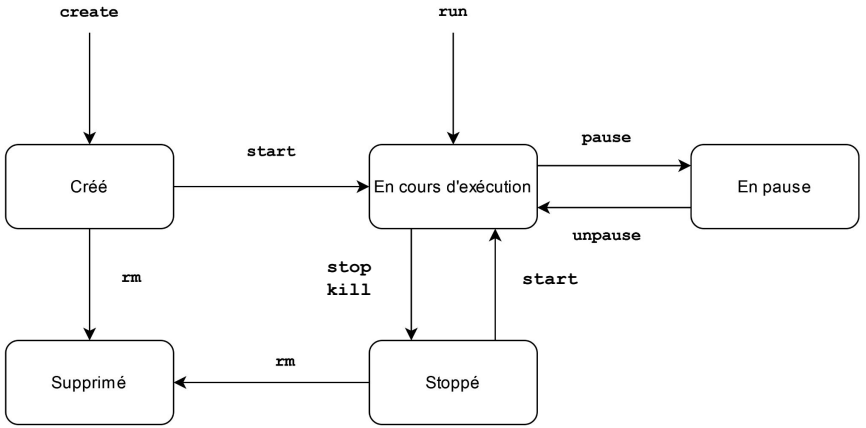
- Une image Docker est **un modèle en lecture seule contenant des instructions** pour créer un conteneur Docker.
- Les instructions d'une image sont de trois natures différentes:
  - **des modifications ordonnées** d'un systèmes de fichiers
  - **des paramètres d'exécution** destinés à être utilisés lors de l'instanciation d'un conteneur
  - **des méta-données** (optionnelles) utiles notamment à des fins de documentation

  .
- **Chaque instruction crée une nouvelle couche** dans l'image.
- **Une image est une méthode** afin de construire un conteneur de manière reproductible mais pas forcément déterministe.
- Vous pouvez construire vos propres images ou utiliser des images publiées par d'autres personnes sur un registre.

2. Rappels sur la plateforme Docker

Les objets fondamentaux - Conteneur

- **Un conteneur Docker est une instance exécutable d'une image Docker**.
- **Toutes les modifications apportées à un conteneur ne seront jamais répercutées sur son image de base**. Elles sont stockées dans la couche "conteneur" accessible en lecture/écriture.
- **Au démarrage d'un conteneur, des instructions préalablement définies dans l'image permettent d'exécuter une ou des commandes** ; qui peuvent, si l'utilisateur le souhaite, être surchargées au moment de l'instanciation du conteneur.
- **Un conteneur possède un cycle de vie** constitué de 5 états : créé, en cours, arrêté, en pause, supprimé



2. Rappels sur la plateforme Docker

Le Registre

- Le registre est un composant essentiel dans l'architecture de Docker. Il s'agit d'un serveur assurant le stockage et la distribution des images.
- L'application en elle-même est open source sous licence Apache 2.0. Si vous souhaitez maîtriser de bout en bout votre chaîne de déploiement, il est possible de déployer votre propre registre.
- Le registre par défaut est Docker Hub. Il appartient à la société Docker et différentes offres de tarifications sont proposées. A ce jour la société Docker propose une offre "Personal" qui convient pour les projets open sources, les petites entreprises, et l'éducation. La limitation principale concerne le nombre d'images (200) que nous pouvons récupérer par tranche de 6 heures.
- En fonction du registre utilisé, il peut être nécessaire de s'authentifier pour déposer ou récupérer des images.
- Pour pointer vers un autre registre que Docker Hub, il faut préfixer votre image par l'adresse de votre registre. Par exemple:
  - **prof:5000**/fr/iut/caen/mapremiereimage:latest

3. Initialisation d'une VM sur Proxmox

- Rendez-vous sur le serveur Proxmox de l'université (https://proxmox-iutc3.unicaen.fr) depuis un navigateur web et authentifiez-vous avec **vos identifiants C3**.

- Créez une VM en clonant depuis le `pve1` le `r509-template-docker-etudiant (128)`
  - Noeud: Sélectionnez le noeud ayant le plus de ressources disponibles
  - Pool de ressources : `INFO`
  - Mode: `Clone intégral`

- Supprimez l'étiquette `ne_pas_supprimer`qui a été placée sur votre VM.

- Démarrez votre VM, sur le bureau exécutez le script `Setup Proxy`et suivez les instructions. Modifiez ensuite votre mot de passe (à l'aide de `passwd`). Mot de passe actuel : `pass`

- Exécutez les commandes suivantes afin d'utiliser docker plus confortablement (ajout de l'utilisateur au groupe docker):
  - sudo groupadd docker
  - sudo usermod -aG docker $USER
  - sudo reboot

- Faites une requête `curl` vers une url sur internet. En cas de non réponse du proxy, exécutez la commande : `sudo dhclient`

4. Se refamiliariser avec le Docker CLI

- Sur cet exercice, vous utiliserez l'image `prof:5000/hello-world`
- Prenez le temps de vous familiariser avec les commandes suivantes : `help`, `pull`, `run`, `inspect`, `create`, `ls`, `start`, `rm`, `prune`
- Réalisez les actions suivantes:
  - Récupérer une image depuis le registre prof
  - Instancier un conteneur
  - Lister les conteneurs en cours d'exécution, les conteneurs arrêtés, les réseaux, les volumes
  - Démarrer un conteneur à l'état arrêté (vous pouvez utilisez `l'option -a` pour attacher la sortie standard)
  - Instancier et démarrer un conteneur en une seule commande
  - Inspecter un conteneur
  - Effacer les conteneurs précédemment créés

5. Rappels sur la plateforme Docker

Le Dockerfile

- Pour construire une image, il existe deux méthodes:
  - Le commit d'un conteneur existant. Cela va sauvegarder les modifications de la couche conteneur dans une nouvelle image. Cette méthode est plutôt conseillée à des fins de débogage.
  - **L'écriture d'un fichier en utilisant un langage dédié (*DSL*). Ce fichier se nomme le Dockerfile.**

```
1 FROM ubuntu:latest
2
3 RUN apt update && apt install -y nano
4
5 CMD ["sh"]
```

Exemple de Dockerfile

- Un Dockerfile est simplement une suite d'instructions. Chaque ligne débute par l'intitulé de l'instruction et chacune va correspondre à une nouvelle couche dans l'image.

- Pour passer du Dockerfile à une image exploitable, il y a une étape de construction (*build*) qui consiste à répéter l'opération suivante:
  - créer un conteneur temporaire à partir de la couche précédente (ou de l'instruction **FROM**)
  - Exécuter l'instruction à l'intérieur de ce conteneur temporaire
  - Sauvegarder les modifications du système de fichier dans une nouvelle couche

5. Rappels sur la plateforme Docker

Point d'entrée et commande

- Vous pouvez spécifier quel programme sera exécuté au démarrage d'un conteneur à l'aide des instructions `ENTRYPOINT` et `CMD`.

- La principale différence c'est que l'instruction `CMD` est ignorée si des paramètres sont passés à l'instanciation d'un conteneur alors qu'ils seront traités comme des arguments avec l'instruction `ENTRYPOINT`.

- Ces instructions peuvent être écrites sous deux formes :
  - `exec`:
    - <INSTRUCTION> [<COMMANDE>, <PARAMETRES>]
    - exemple: `CMD ["echo", "hello-world"]`
  - `shell`:
    - <INSTRUCTION> <COMMANDE> <PARAMETRES>
    - exemple: `ENTRYPOINT echo "hello-world"`

- Avec la forme `shell`, il est possible de faire des substitutions de commandes, d'utiliser des variables d'environnement ou encore de rediriger des entrées sorties. Ce n'est pas possible avec la forme `exec`.

- Il est conseillé de privilégier la forme `exec` car elle permet de prendre en compte des arguments passés en paramètres. Généralement, on utilise `ENTRYPOINT` pour lancer un programme et `CMD` pour spécifier des paramètres par défaut pouvant être surchargés par le client.

6. Exploitation d'un service web avec Docker

- Vos enseignants mettent à votre disposition un service web écrit en java empaqueté dans une archive `jar`:
  - Pour l'exécuter vous aurez besoin d'un environnement d'exécution **Java (JRE) en version 11**
    - Appuyez-vous sur l'image `prof:5000/eclipse-temurin:11-jre`
  - A son démarrage **le serveur écoute les requêtes HTTP sur le port TCP 8080**
  - Ce serveur expose une API HTTP REST dont la documentation est disponible sur ce lien

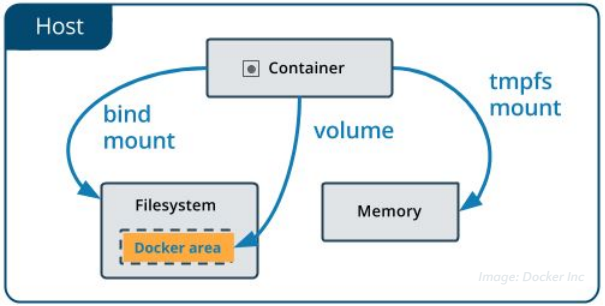
- Il est attendu de déployer cette application dans un conteneur sur votre machine hôte. Voici quelques indications:
  - Télécharger l'archive grâce à la commande : `curl -LJO http://prof/score-jre11.jar`
  - Créez un fichier se nommant `Dockerfile` (Attention à la casse !)
  - Complétez-le **à minima** avec les instructions: `FROM`, `COPY`, `CMD` ou `ENTRYPOINT`
    - Au besoin, appuyez-vous sur le cours ou la documentation du Dockerfile
  - Construisez une image taguée `score:jre11` à partir de votre Dockerfile avec la commande `docker build`
  - Instanciez un conteneur à partir de votre image
  - Vérifier le bon fonctionnement de votre service en augmentant le compteur de 3 points, en l'affichant, puis en le réinitialisant

- Trop facile ? Faites la même chose avec un jar utilisant un JRE 17. Reprenez les instructions précédentes en remplaçant 11 par 17. Puis déployez simultanément sur votre machine hôte les deux versions du service.

7. Rappels sur la plateforme Docker

La gestion du stockage des données

- **Par défaut, la persistance des données s'effectue dans la couche conteneur**. Ce qui induit quelques limitations:
  - Si le conteneur est supprimé, les données sont perdues.
  - La couche conteneur est étroitement liée à l'hôte.
  - L'écriture dans la couche conteneur passe par une couche d'abstraction. Les performances sont donc moindre qu'une écriture directe sur le système de fichier de l'hôte



- Il existe 3 méthodes pour écrire en dehors de la couche conteneur:
  - Volumes: un objet Docker appartenant aux systèmes de fichiers de l'hôte, créé et géré par Docker.
  - Montages liés (*bind mounts*): Similaire aux volumes mais avec des fonctionnalités limités
  - Montages temporaires (*tmpfs mounts*): Persistance en mémoire ne pouvant être partagée entre plusieurs conteneurs. Utile pour enregistrer des états non persistants ou des informations sensibles.

7. Rappels sur la plateforme Docker

Les volumes

- Les volumes sont des objets Docker stockés dans un répertoire sur la machine hôte (sur Linux `/var/lib/docker/volumes`).
- Lorsqu'un volume est monté dans un conteneur, cela monte un répertoire dans ce conteneur.
- Plusieurs conteneurs peuvent monter simultanément le même volume.
- Lors de la suppression d'un conteneur, le volume qui lui est associé n'est pas supprimé.
- Un volume possède un nom défini explicitement ou il est généré aléatoirement par défaut.
- Il est possible d'utiliser des pilotes pour les volumes, par exemple REX-Ray. Ces derniers permettent de stocker les données sur des hôtes distants ou des fournisseurs de services *cloud.*

8. La persistance des données dans un conteneur

- Sur cet exercice, vous utiliserez l'image `prof:5000/busy-box`
- Instanciez un conteneur en mode interactif et attaché, option `-it`, et avec suppression automatique lors d'un `exit`, option `--rm`.
- Créez quelques fichiers dans ce conteneur, puis quittez le avec `exit`.
- Qu'est-il advenu des fichiers ? Pourquoi ?
- Trouver un moyen d'instancier un conteneur et de faire persister ses modifications après la suppression de ce dernier.

9. Rappels sur la plateforme Docker

Les pilotes réseau

- Les conteneurs doivent être conçus pour traiter une tâche spécifique. Pour tirer pleinement parti de leur potentiel, ils doivent coopérer entre eux ou avec des services tiers. Nous allons nous intéresser aux fonctionnalités réseau de Docker.

- Docker gère la partie réseau de manière agnostique, peu importe que les conteneurs proviennent d'hôtes sous des OS Linux, Windows ou les deux.

- Sans plugins tiers, Docker est livré avec 6 pilotes réseau, voici un aperçu des plus utilisés:
  - **bridge**: le pilote par défaut. A privilégier lorsque plusieurs conteneurs sur un même hôte doivent communiquer entre eux.
  - **host**: supprime l'isolation du réseau entre le conteneur et l'hôte.
  - **overlay**: connecte plusieurs démons Docker entre eux. Idéal lorsque plusieurs conteneurs sur des hôtes différents doivent communiquer entre eux.
  - **none**: Désactive les fonctionnalités réseaux (excepté l'adresse de loopback)
  - **macvlan**: permet d'assigner une adresse MAC à un conteneur. Utile pour faire apparaître le conteneur comme un hôte physique sur le réseau.

- Au démarrage de Docker, un réseau de type "bridge" est créé. A moins de spécifier le contraire, tous les conteneurs créés seront rattachés par défaut à ce réseau. Ils pourront communiquer entre eux via leur adresse IP.

9. Rappels sur la plateforme Docker

Le réseau depuis un conteneur

- Le type de réseau utilisé (bridge, overlay, macvlan) est transparent depuis le conteneur.
- Le conteneur ne possède à son niveau qu'une interface réseau dont les principaux éléments sont:
  - une adresse IP
  - une passerelle
  - un table de routage
  - un service DNS
- **Par défaut, aucun port n'est exposé**. Pour rendre accessible le port d'un conteneur à des services en dehors de Docker ou à des conteneurs qui ne sont pas connectés au même réseau ; il est possible de mapper un port de l'hôte sur un port du conteneur.
- Le démon Docker se comporte comme un serveur DHCP en attribuant une IP à chaque conteneur se connectant à un réseau.
- Les conteneurs disposant de la configuration réseau par défaut hérite d'une copie du fichier `/etc/resolv.conf` Alors que les conteneurs avec un réseau spécifique utilisent le serveur DNS embarqué de Docker.

Merci de votre attention

Maxime Lambert

maxime.lambert@unicaen.fr