# Chapitre 2 : Conteneurisation avec Docker

| | |
|---|---|
| Owner | louis reynouard |
| Tags | |
| Created time | @February 14, 2024 3:32 PM |



La conteneurisation est une technologie clé dans le développement et le déploiement modernes d'applications, offrant une flexibilité, une portabilité et une efficacité accrues. Docker, en tant que plateforme de conteneurisation la plus populaire, joue un rôle central dans l'écosystème DevOps. Ce chapitre explore les concepts de base de Docker, la différence entre les conteneurs et les machines virtuelles, et vous guide à travers la création et la gestion des images et conteneurs Docker.

# Un peu de théorie

## Conteneurs vs Machines Virtuelles

Imaginez que vous vivez dans un grand immeuble avec plusieurs appartements. Chaque appartement a ses propres installations : cuisine, salle de bain, chauffage, etc. Nous sommes ici dans le cas de **machines virtuelles**.

Ces différentes installations sont comme le **système d'exploitation invité** dans une machine virtuelle, prenant beaucoup de place et nécessitant beaucoup de ressources pour chaque appartement (au delà de dupliquer l’ensemble de ces installations, il est aussi primordiale de vérifier la compatibilité de chacun des éléments. Imaginez arriver avec votre nouveau four de 60 cm pour un emplacement de four de seulement 50cm...)

Maintenant, pensez à un modèle différent, où au lieu d'avoir toutes ces installations dans chaque appartement, les résidents partagent des installations

communes (cuisine, salle de bain etc..). C'est le concept des **conteneurs**.

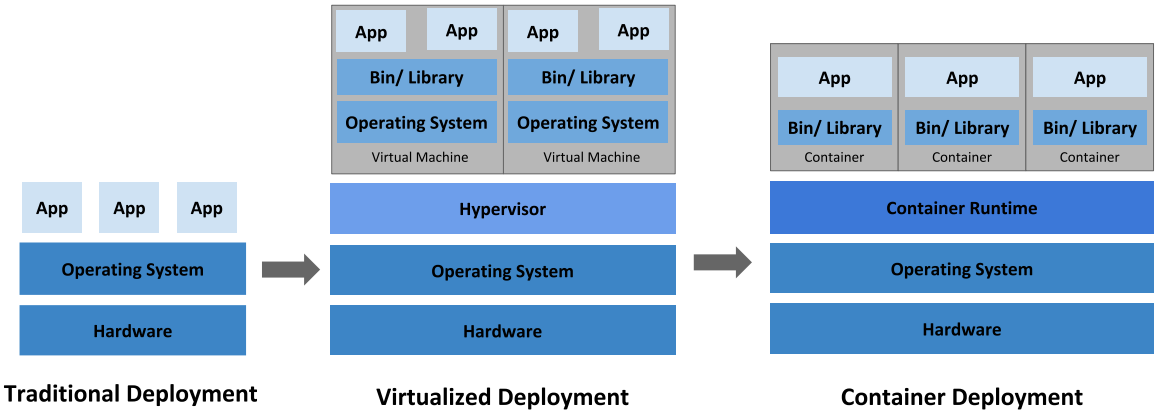
Ils partagent le même système (l'immeuble), rendant chaque espace de vie (**conteneur**) beaucoup plus léger et nécessitant moins de ressources pour fonctionner. C'est parce qu'ils partagent des ressources communes (**système d'exploitation hôte**), au lieu d'avoir leur propre copie de tout.

Cette analogie nous aide à comprendre pourquoi les conteneurs sont plus légers, démarrent plus rapidement et utilisent moins de ressources que les machines virtuelles.

## Quelles différences techniques ?

Entrons un peu plus précisément dans les différences de concept technique:

- **Machines Virtuelles (VMs)** :
    - Chaque VM fonctionne avec un système d'exploitation complet, incluant le noyau, ce qui consomme une quantité significative de ressources système.
    - Les VMs s'exécutent sur un hyperviseur (Ex: VMware ou VirtualBox), qui gère ces machines virtuelles indépendantes sur l'hôte physique. Cette isolation complète garantit la sécurité et la flexibilité, mais au prix de l'overhead de performance et d'espace disque.
- **Conteneurs** :
    - les conteneurs partagent le même noyau du système d'exploitation hôte mais isolent les processus de l'application dans des espaces utilisateurs séparés.
    - Cela est rendu possible grâce à des fonctionnalités Linux comme cgroups et namespaces. Cgroups limite et isole les ressources utilisées par un processus (comme la mémoire, CPU, etc.), tandis que namespaces isole les processus afin qu'ils aient leur propre vue de l'environnement système (fichiers, réseau, utilisateurs, etc.).
    - Cette architecture rend les conteneurs extrêmement légers et rapides à démarrer par rapport aux VMs.



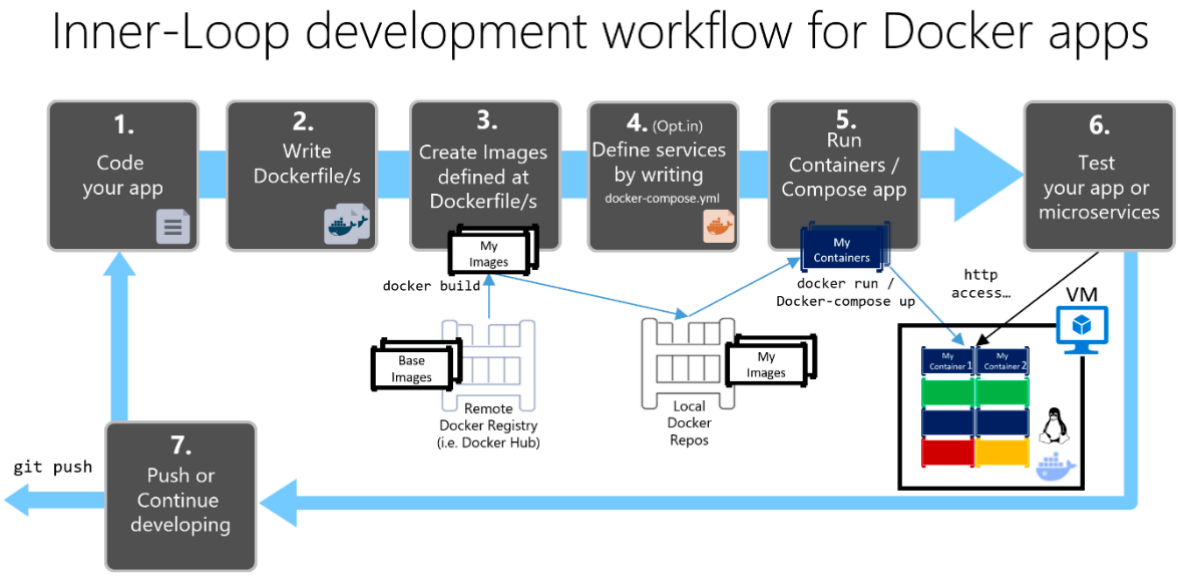

# Détail de l'architecture de Docker

Dans la suite de ce cours nous utiliserons les technologies docker. Entrons un peu plus dans le détail de ce nouvel environnement:

- **Daemon Docker** : Le cœur de l'architecture Docker est le daemon Docker (`dockerd`), qui s'exécute en tant que service de fond sur l'hôte. Les utilisateurs interagissent avec le daemon via le client Docker (`docker`), en utilisant l'interface de ligne de commande (CLI) ou via des applications utilisant l'API Docker.  Ce daemon est responsable de la gestion des objets Docker, tels que:
    - **Images Docker** : Une image Docker est un modèle immuable qui contient l'application et son environnement d'exécution. Les images sont construites à partir d'un fichier de configuration nommé `Dockerfile`, qui définit toutes les étapes nécessaires pour créer l'image. Ces étapes peuvent inclure l'installation de logiciels, la copie de fichiers dans l'image, et la définition de commandes à exécuter au démarrage du conteneur.
    - **Conteneurs Docker** : Lorsque vous lancez un conteneur à partir d'une image, Docker utilise l'image comme système de fichiers en lecture seule et ajoute une nouvelle couche en écriture par-dessus pour les modifications effectuées pendant l'exécution du conteneur. Cette séparation permet aux images de rester immuables tout en permettant aux conteneurs d'être modifiables et persistants le temps de leur exécution.
    - **Registres d'Images** : Les images Docker peuvent être stockées et partagées via des registres tels que Docker Hub, permettant aux développeurs de télécharger et de partager facilement des images à

travers des environnements de développement, de test et de production.

Ci dessous un diagramme de flux montrant les grandes étapes d'un projet Docker:




# TP: Création de l'Application Web et Empaquetage dans une Image Docker

👉 Finissez la partie 2 du TP