**Docker Résumé**

Installation Docker Windows: Avent d’installer Docker descktop installer WSL et pansée a activer Hyper V entée dans PowerShell « wsl –install », avec la commande « wsl -l -v » vérifier la version qu’elle soit bien sur 2.

Pour changer la version de WSL entée « wsl --set-version <Nom\_de\_la\_distribution> 2 » exemple « wsl --set-version Ubuntu-20.04 2 »

Container et ces couches :

Un containeur contrairement à la virtualisation qui a juste une couche d’abstraction les containeur utilise le noyau du système d’exploitation de l’hôte. Un containeur est comme une enveloppe fournissant l’essentielle a l’exécution de programme informatique.

ContaineurD : ContaineurD et un Daemon qui s’occupe de la vie des containeur de la création a la suppression du containeur.

Les Images : Une image [Docker](https://www.techtarget.com/searchitoperations/definition/Docker) est un fichier utilisé pour exécuter du code dans un conteneur Docker. Les images Docker agissent comme un ensemble d'instructions pour créer un [conteneur](https://www.techtarget.com/searchitoperations/definition/container-containerization-or-container-based-virtualization) Docker , comme un modèle. Les images Docker servent également de point de départ lors de l'utilisation de Docker. Une image est comparable à un instantané dans les environnements de machines virtuelles (VM).

Dockerfile : « Dockerfile » est un document texte qui contient toutes les commandes qu'un utilisateur peut appeler sur la ligne de commande pour assembler une image.

Les commandes de base : docker images affiche les image sur le PC

docker run -d -p 80:80 **docker/Image\_name** -v **mon\_volum**

docker rm -f ID **Force la supprimé le container**

Docker build . **Permet de crée un container avec un Dockerfile**

Docker ps **Permet d’afficher les service docker en cours**

Docker stop ID **stop l’exécution d’un container**

Docker tag ID **permet d’attribuer un tag**

Docker push **permet de mettre le container sur le cloud**

Docker pull nom\_imag **permet de télécharger une image**

docker network create Nom\_reseau **crée un réseau**

docker login -u USER-NAME **pour se connecter au cloud**

docker tag getting-started YOUR-USER-NAME/getting-started:v2

docker exec ID LINUX\_COMMAND **permet d’exécuter une commande linux dans le container**

docker volume create NOM\_VOLUME **permet de crée un volume pour conserver les données**

Docker build : exécuter tous seul cherche un fichier nommer dockerfile il est possible de rajouter une URL GitHub il est possible d’ajouter une option -t qui permet de rajouter un tag a l’image pour terminer la commande ajouter un « . » pour indiquer que le fichier dockerfile se trouve dans le répertoire actuelle. Après avoir créé un container avec la ou les images nécessaires on pourra la démarrée avec docker run et le nom du tague crée précédemment.

|  |  |
| --- | --- |
| **Nom** | **Description** |
| **Named Volume** | Placé près du Docker Engine, à l’intérieur de l’espace réservé localement à Docker |
| **Bind Mounts** | «Partage» un répertoire entre le containeur et l’hôte local |
| **SFTP Volumes** | Synchronise un répertoire avec un serveur de fichier SFTP |
| **NetApp Volumes** | Synchronise un répertoire avec un Cloud NetApp |
| **S3 Volumes** | Synchronise un répertoire avec un Cloud AWS |
| **Azure Volumes** | Synchronise un répertoire avec un Cloud Microsoft |
| Et il y en a pleins d’autres **…** | |

Utiliser un volume : Un volume permet de conserver les données car un containeur et jetable les données seront supprimées. Pour éviter cela on utilise un volume pour crée un volume on utilise la commande docker create NOM\_VOLUME et pour l’intégrée a un containeur docker run -dp 3000:3000 -v NOM\_VOLUME:REPERTOIRE\_INTERNE

Docker network : docker network creat todo-app pour créer un nouveau Network

Docker network ls pour voir tous les Network

Docker network inspect todo-app pour inspecter un Network (et voir quels conteneurs les utilisent – super utile !)

Docker network rm todo-app pour supprimer un Network

**Pour connecter des conteneurs au network.**



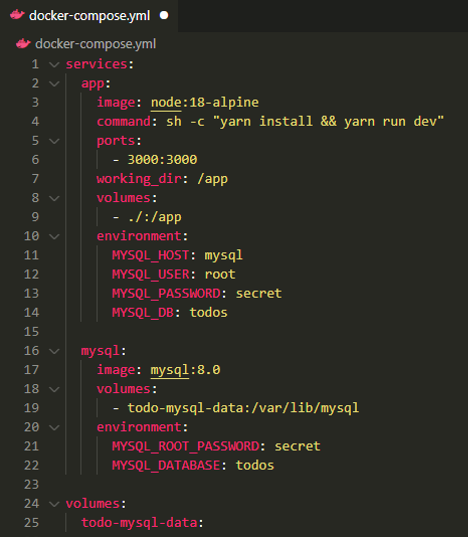


Micro service : le fait d’avoir plusieurs conteneurs permet d’avoir un service pour chaque tache et lors de mise a jour par exemple tous l’infrastructure n’est pas hors service. Les micro services vise à séparer les composants d’une application en plusieurs services qui communiquent. C’est application utilise des technologies asynchrones. Comme REST.

REST : les service REST sont fourni via des requête http (GET, POST, PUT, PATCH, DELETE) se trouvent dans une URL.

Docker compose : Permet de gérée mieux des applications multi-conteneurs et utilise un fichier **docker-compose.YML**

Pour exécuter la commande il fau se rendre dans le répertoire puis taper docker compose up -d attention au nom du conteneur il ne peu pas être identique a l’IMAGE. Pour désactiver le compose entrée docker compose down



Les containeurs à une large échelle

Les containeurs peuvent être utilisée dans les environnement productif et non seulement comme technologie de test. Les containeurs permettent une grande scalabilité des services. Effectivement ci un des containeurs cesse de fonctionner son arrêt n’impactera pas ou peu les autres containeurs selon le contenu de ce dernier.

Affin d’assurée une utilisation de la containerisation dans un environnement de production certains points sont obligatoires.

1. Une scalabilité irréprochable
2. Une répartition de charge sur plusieurs nœuds
3. Une méthode de provisionnement efficace

Orchestration

Pour utiliser et mettre en place des containeurs pour la production et être sûre de remplir tous les points mentionnés précédemment et gérée les conteneurs on utilise des « Orchestrateurs ».

L’orchestrateur s’occupe de tout ce qui concerne le déploiement de configuration, la répartition de charge et le cycle de vie des applications composées de microservices

L’orchestrateur natif de Docker utilise deux outils :

1. Docker Swarm « Pour les nœuds et le cycle de vie »

Avantage de Swarm : Simple à utiliser.

Défaut : Moins complait que ses concurrents

1. Docker Compose « Pour provisionner les containeur »

Outils cloud d’orchestration :

1. Amazon EKS (Elastic Kubernetes Service)
2. Microsoft Azure ACI (Azure Container Instances)
3. Google Cloud Run

On prédit la disparitions de Swarm en 2022 mais Docker Swarm restera Open Source.

Plateformes d’orchestration

Une image contenant texte

Description générée automatiquement

Open Shift est une plateforme qui utilise d’autres technologies, notamment Docker et K8s

Le cœur d’OpenShift est Open Source (License Apache), mais tous les reste est vendu sous licence commerciale

Alternative plus ouverte :



Kubernetes

Kubernetes et un orchestrateur de conteneur complexe crée par google et rendue Open Source en 2015. Kubernetes est aujourd’hui le leader et éclipse presque Docker dans ce domaine.

Dans un service Kubernetes composé de 1 ou plusieurs microservices « conteneurs » est appelé un « Pods »

Un « Pods » est déployé sur un nœud Kubernetes qui garantit que les containeurs qui le composent n’aient pas de problèmes d’IP (et d’ouverture de ports), de volumes et de puissance requises.

CNCF

La Cloud Native Computing Fondation a été créé en 2015 par la Linux Fondation afin de « cadrer » le développement de certains outils Open Source lié au Cloud

Le premier outil qui a rejoint la CNCF est Kubernetes

Quand un grand acteur du numérique développe un outil maison pour un besoin spécifique, mais qu’il n’a pas envie de faire beaucoup de suivi dessus, il le donne à la CNCF

Lien :

<https://www.techtarget.com/searchitoperations/definition/Docker-image#:~:text=A%20Docker%20image%20is%20a,virtual%20machine%20(VM)%20environments>.

<https://docs.docker.com/engine/reference/commandline/build/>

<https://learn.microsoft.com/fr-fr/windows/wsl/install>

Lien CNCF :

<https://www.cncf.io/projects/>

<https://www.cncf.io/about/members/>