Formation Docker

Ihab ABADI / UTOPIOS

SOMMAIRE

Partie 1

1- Introduction docker / virtualisation
2- Docker / Isolation
3- Avantages conteneurisation
4- Rappel linux et définition
5- Fonctionnement docker
6- Installation docker
7- Image docker
8- Conteneur Docker
9- Volume docker
10- Dockerfile
11- Docker compose

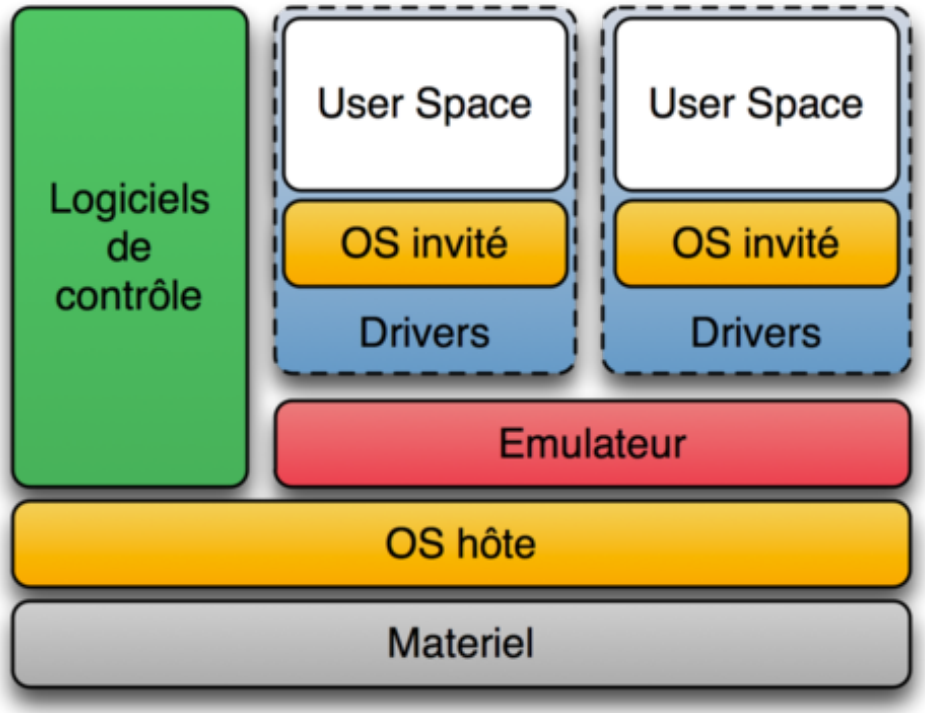
SOMMAIRE

Partie 2

1- Utilisation d’un Docker registry
2- Débogage des conteneurs
3- Utilisation de Docker machine
4- Utilisation de Docker Swarm

INTRODUCTION DOCKER- Virtualisation

- La virtualisation est la capacité de faire tourner un, ou plusieurs serveur virtuel sur une seul machine physique grâce à un hyperviseur.
- L'hyperviseur permet d'émuler les différentes ressources matérielles d'un serveur physique et permet à des machines virtuelles de les partager.
- Une machine virtuelle possède ses propres ressources matérielles et son propre système d'exploitation



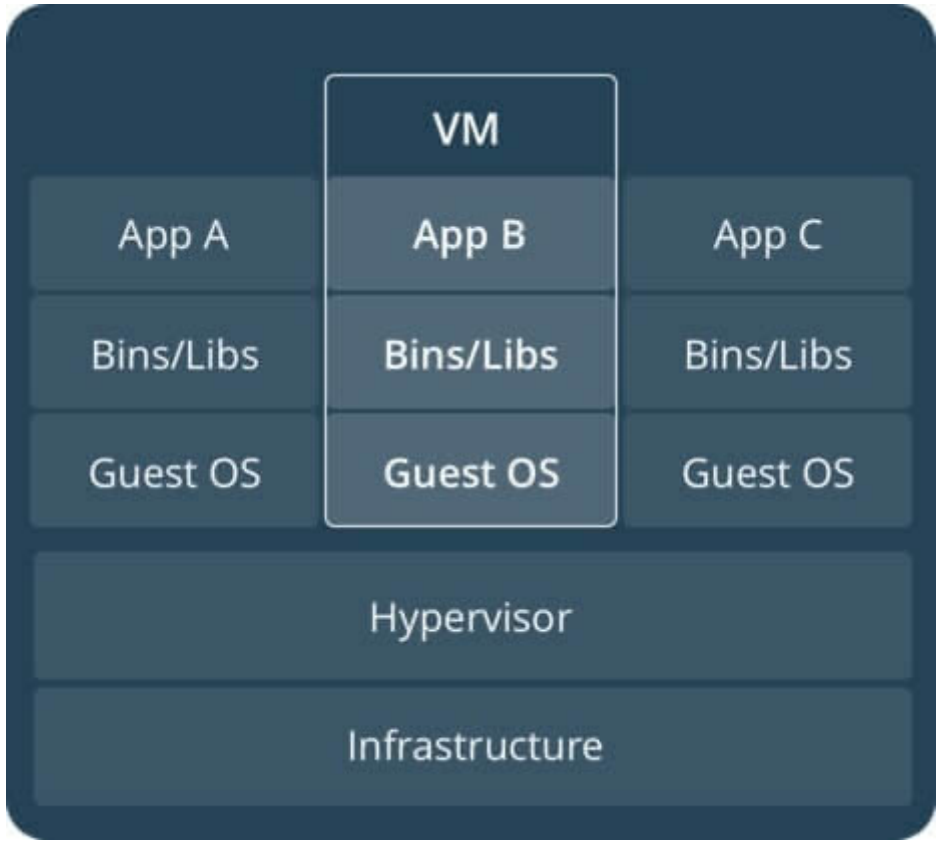
INTRODUCTION DOCKER- Virtualisation

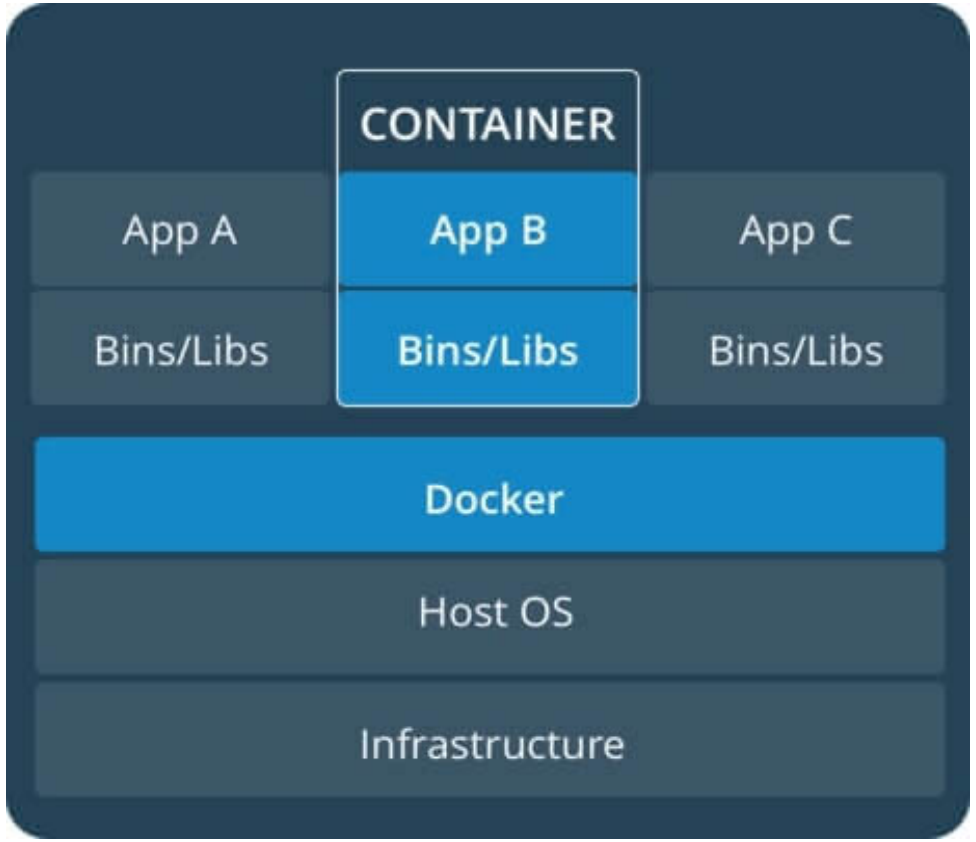
- **La virtualisation apporte des avantages :**
  - Ressources adaptées aux besoins de l'application.
  - Faciliter de manipulation des VMs (Sauvegarde, bascule,...)
  - Réduction des dépenses et réduction d'équipements nécessaires
  - Faciliter dans l'administration
- **La virtualisation a des inconvénients également :**
  - Réduction des performances
  - Multiplication des couches OS
- **Conclusion**
  - La virtualisation est une technologie avec beaucoup d'avantages mais avec certain inconvénients bloquant, d'où la nécessité de pousser vers une nouvelle technologie qui répond à ces inconvénients

DOCKER – Isolation

- **Isolation en VM:**
  - Se fait au niveau matérielle
  - Accès virtuel aux ressources via l'hôte à l'aide de l'hyperviseur
- **Isolation Dans la conteneurisation:**
  - Se fait au niveau du système d'exploitation
  - Exécution native sur linux et partage du noyau hôte avec les conteneurs
- **Conclusion**
  - La virtualisation est une technologie avec beaucoup d'avantages mais avec certain inconvénients bloquant, d'où la nécessité de pousser vers une nouvelle technologie qui répond à ces inconvénients

DOCKER – Isolation





DOCKER – Avantages

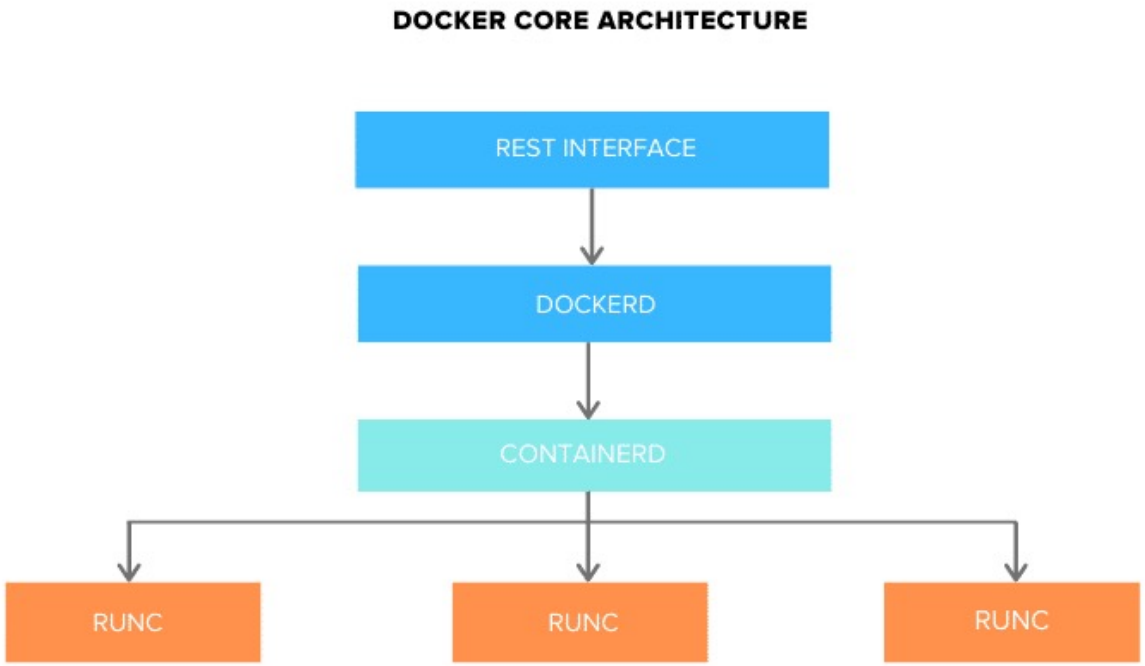
- **Avantages :**
  - Flexibilité quelque soit la complexité de l'application
  - Légèreté grâce au partage du noyau du hôte
  - Scalabilité ou Extensibilité
  - …

DOCKER – Rappel Linux et définition

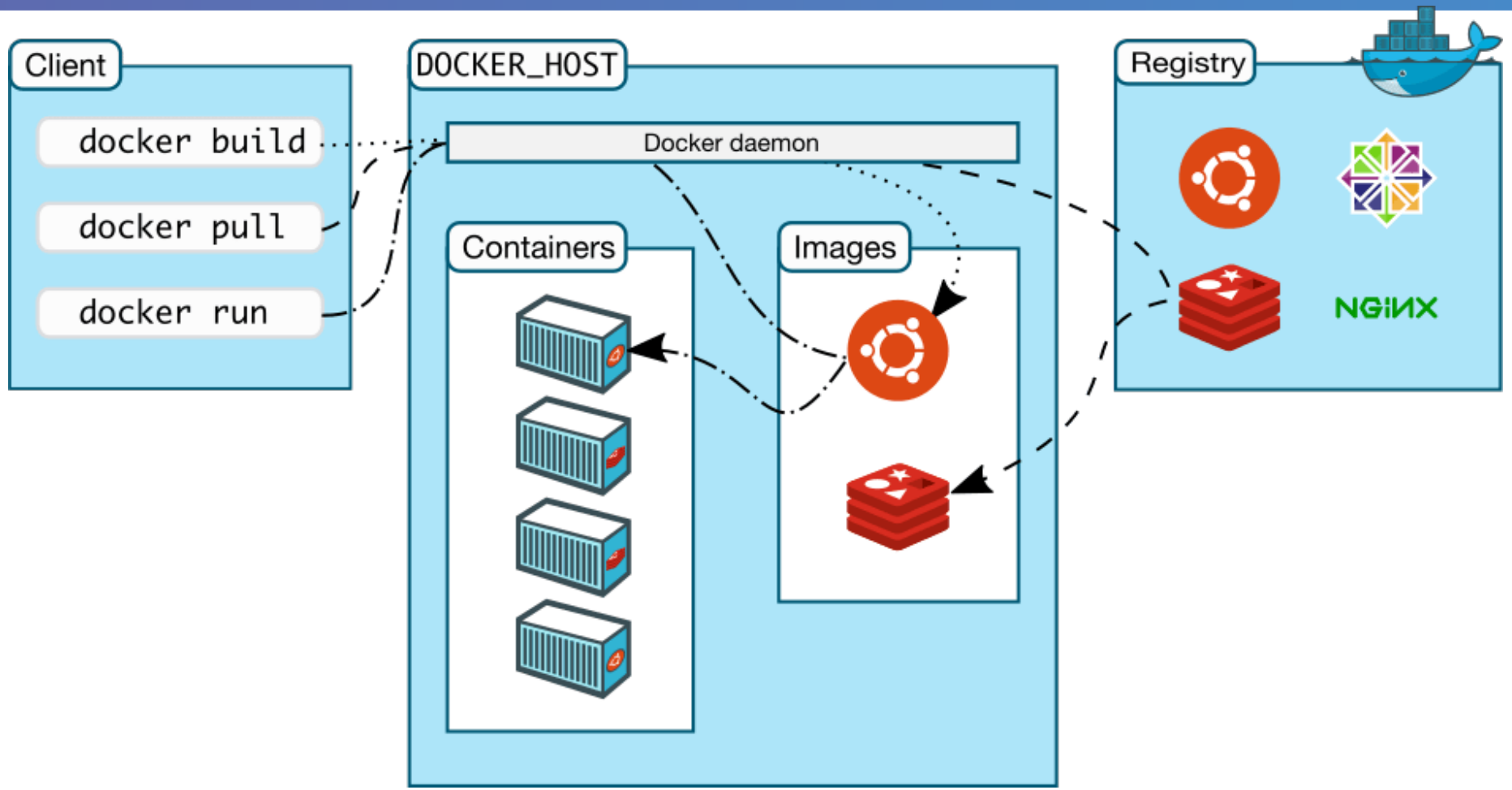
- Un conteneurs et un processus linux isolé à l'aide des namespaces et cgroups
- Les namespaces linux est un mécanisme qui permet de limiter l'accès d'un processus
- Les cgroups linux est un mécanisme qui permet de limiter l'accès au ressources d'un processus
- Quelques exemple de namespaces:
  - Namespace PID
  - Namespace IPC
  - Namespace USER
- Quelques exemple de cgroups
  - Cgroup cpuset
  - Cgroup cpu
  - Cgroup memory

DOCKER – définition

- Docker est une technologie qui permet la conteneurisation sur Linux
- Docker démocratise l'utilisation des conteneur
- Docker est composé de :
- Docker engine
- Docker-containerd
- Docker-runc



DOCKER – Fonctionnement



DOCKER – Installation

- Docker peut être installer sur toute type de distribution
- Windows
- Linux
- MacOS
- https://docs.docker.com/get-docker/

DOCKER – Image

- Sur Docker, un conteneur est lancé à partir d'une image
- Une image est un package qui inclut les fonctionnalités nécessaires à l'exécution de notre processus
- Le contenu de l'image est :
  - Le code
  - Les variables d'environnement
  - Les fichiers de configuration
  - Les bibliothèques
- On peut créer une image de plusieurs façons
  - A partir d'un conteneur
  - A partir d'un fichier DockerFile
- Docker fournit un CLI pour gérer les images sur notre machine

DOCKER – Commande pour Image

- Pour lister les images sur la machines : docker images
- Pour rechercher une image sur le hub docker : docker search
- Pour télécharger une image à partir du hub : docker pull
- Pour exécuter une image : docker run
  - Chaque image a un nom unique
  - Chaque image peut avoir une description
  - Chaque image a plusieurs tags
- Pour supprimer une image : docker rmi nom_image

DOCKER – Conteneur

- Un conteneur est la représentation en mémoire lors de l'exécution d'une image
- Pour créer un conteneur, on utilise la commande docker run [OPTIONS] <nom_image>
- Lors de la création d'un conteneur, on peut spécifier une ou plusieurs options
- Quelques options :
- -t : Allouer un pseudo TTY
- -i : Garder un STDIN ouvert
- -d: Exécuter le conteneur en arrière-plan
- -p :exposer un ou plusieurs ports
- --name : donner un nom au container

DOCKER – Conteneur

- Pour afficher la liste des conteneurs docker container ls ou docker ps
- On peut utiliser l'option –a pour afficher la totalité des conteneurs quelque soit le statut
- Chaque conteneur a :
  - Container ID : id du conteneur
  - IMAGE : l'image utilisée pour créer le conteneur
  - COMMAND : dernière commande lancée lors de l'exécution de l'image
  - CREATED : date de création du conteneur
  - STATUS : statut du conteneur
  - PORTS : les ports utilisés par le conteneurs
  - NAME : nom du conteneur

DOCKER – Conteneur

- Un conteneur passe par plusieurs états
  - created : conteneur créé mais non démarré (cet état est possible avec la commande docker create)
  - restarting : conteneur en cours de redémarrage
  - running : conteneur en cours d'exécution
  - paused : conteneur stoppé manuellement (cet état est possible avec la commande docker pause)
  - exited : conteneur qui a été exécuté puis terminé
  - dead : conteneur que le service docker n'a pas réussi à arrêter correctement (généralement en raison d'un périphérique occupé ou d'une ressource utilisée par le conteneur)

DOCKER – Conteneur

- Pour supprimer un conteneur :
- Docker rm <container_id> ou <container_name>
- Pour exécuter une commande dans un conteneur
- Docker exec <OPTIONS> <container_id ou container_name> command
- Pour convertir un conteneur en Image
- Docker commit <container_id ou container_name> <image_name>

DOCKER – Volumes

- Les données dans un conteneur sont éphémères.
- Pour sauvegarder les données d'un conteneur, on peut le convertir à chaque fois en image (mauvaise pratique)
- La deuxième solution, consiste à utiliser les volumes

DOCKER – Volumes (Système de fichier docker)

- Docker utilise un système de fichier en calques
- Chaque image se compose de pile de calques en lecture seule
- A la création d'un conteneur un calque est ajouté dans le haut de pile en lecture-écriture
- Lors d'une modification de fichier, Docker crée une copie depuis les couches en lecture seule vers le layer en lecture-écriture.
- Lors d'une création de fichier, Docker crée le fichier que sur le layer en lecture-écriture, et ne touche pas au layer en lecture seule.
- Lors d'une suppression de fichier, Docker supprime le fichier que sur le layer en lecture-écriture, et s'il est déjà existant dans le layer en lecture seule alors il le garde.
- Les données dans le layer de base sont en lecture seule, elles sont donc protégées et intactes par toutes modifications, seul le layer en lecture-écriture est impacté lors de modifications de données.
- Lorsqu'un conteneur est supprimé, le layer en lecture-écriture est supprimé avec. Cela signifie que toutes les modifications apportées après le lancement du conteneur auront disparus avec.

DOCKER – Volumes (Volume non lié)

- Docker permet la création de volume en dehors de tout conteneur
- Pour créer un volume, on utilise la commande
- Docker volume create <nom_volume>
- Pour lister des volumes
- Docker volume ls
- Pour supprimer un volume
- Docker volume rm <nom_volume>
- Pour créer un conteneur avec volume, on utilise l'option v
- Docker run –v <nom_volume>:<dossier_conteneur>

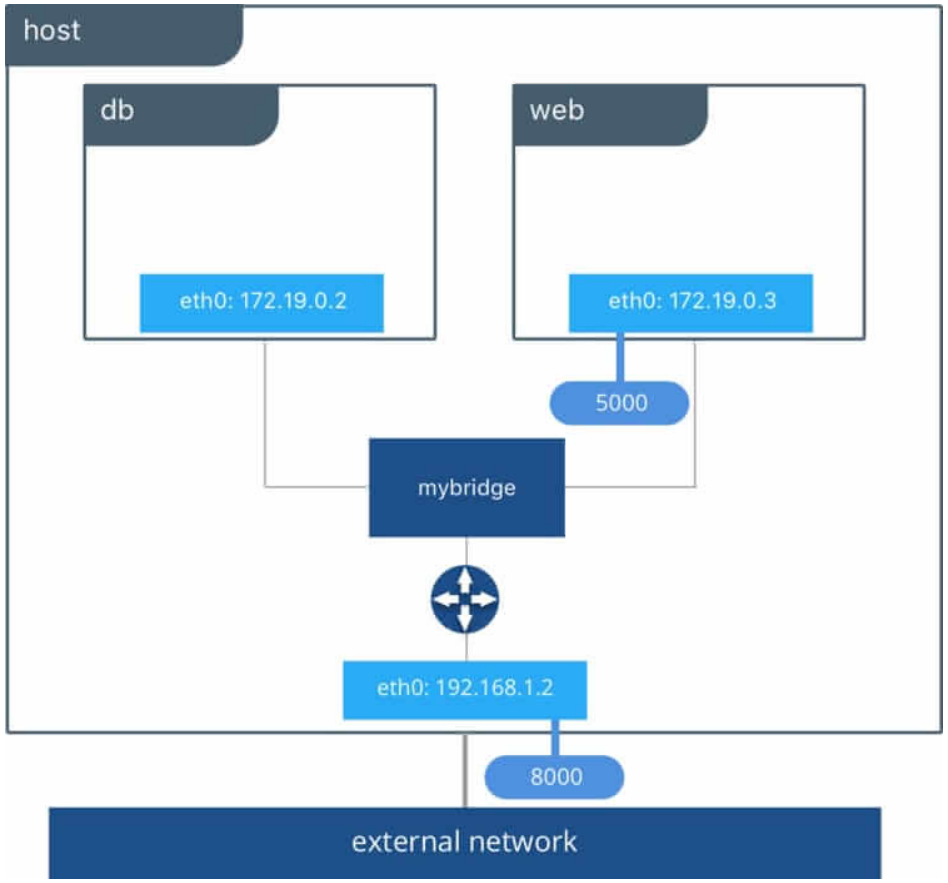
DOCKER – Volumes (Volume lié)

- On peut également monter un volume local sur un conteneur
- Docker run –v <volume_local>:<volume_conteneur>

DOCKER – Gestion de réseau

- Le réseau docker utilise des drivers réseau
- Par défaut docker crée un driver Bridge
- Chaque nouveau conteneur est automatiquement connecté à ce réseau
- Les conteneurs qui utilisent ce driver, ne peuvent communiquer qu'entre eux.
- Il ne sont pas accessibles de l'extérieur

DOCKER – Gestion de réseau



DOCKER – Gestion de réseau

- Il existe d'autre type de driver
- Driver none
- Driver host
- Driver overlay
- Driver macvlan

DOCKER – Gestion de réseau

- Pour créer un réseau docker
- docker network create --driver <DRIVER TYPE> <NETWORK NAME>
- Pour lister les réseaux docker
- Docker network ls
- Pour inspecter un réseau
- Docker network inspect <network_name>
- Un conteneur docker est par défaut connecté à un réseau de type Bridge
- Pour connecter un conteneur au moment de la création, on utilise l'option --network
- Pour connecter un conteneur déjà créer, on utilise la fonction connect de docker network
- Pour déconnecter un conteneur déjà créer, on utilise la fonction disconnect de docker network
- Des conteneurs dans le même réseau peuvent communiquer entre eux
- Les conteneurs dans un bridge ne peuvent pas communiquer avec l'extérieur qu'à travers des ports exposés vers l'hôte
- On peut exposer des ports uniquement au démarrage d'un conteneur avec l'option p
- Exemple
- Docker run –p <port hot>:<port container> --name=<nom_du_container> <image_container>

DOCKER – Dockerfile

- Dockerfile est fichier qui permet de créer une image docker à l'aide de Docker DSL
- DSL pour dockerfile contient une multitude d'instruction
- FROM : Définit l'image de base qui sera utilisée par les instructions suivantes.
- LABEL : Ajoute des métadonnées à l'image avec un système de clés-valeurs, permet par exemple d'indiquer à l'utilisateur l'auteur du Dockerfile.
- ARG : Variables temporaires qu'on peut utiliser dans un Dockerfile.
- ENV : Variables d'environnements utilisables dans votre Dockerfile et conteneur.
- RUN : Exécute des commandes Linux ou Windows lors de la création de l'image. Chaque instruction RUN va créer une couche en cache qui sera réutilisée dans le cas de modification ultérieure du Dockerfile.

DOCKER – Dockerfile

- COPY : Permet de copier des fichiers depuis notre machine locale vers le conteneur Docker.
- ADD : Même chose que COPY mais prend en charge des liens ou des archives (si le format est reconnu, alors il sera décompressé à la volée).
- ENTRYPOINT : comme son nom l'indique, c'est le point d'entrée de votre conteneur, en d'autres termes, c'est la commande qui sera toujours exécutée au démarrage du conteneur. Il prend la forme de tableau JSON (ex : CMD ["cmd1","cmd1"]) ou de texte.
- CMD : Spécifie les arguments qui seront envoyés au ENTRYPOINT, (on peut aussi l'utiliser pour lancer des commandes par défaut lors du démarrage d'un conteneur). Si il est utilisé pour fournir des arguments par défaut pour l'instruction ENTRYPOINT, alors les instructions CMD et ENTRYPOINT doivent être spécifiées au format de tableau JSON.
- WORKDIR : Définit le répertoire de travail qui sera utilisé pour le lancement des commandes CMD et/ou ENTRYPOINT et ça sera aussi le dossier courant lors du démarrage du conteneur.
- EXPOSE : Expose un port.
- VOLUMES : Crée un point de montage qui permettra de persister les données.
- USER : Désigne quel est l'utilisateur qui lancera les prochaines instructions RUN, CMD ou ENTRYPOINT (par défaut c'est l'utilisateur root).

DOCKER – Dockerfile COPY et ADD

- Ils permettent tous les deux de copier un fichier/dossier local vers un conteneur.
- ADD autorise les sources sous forme d'url et si jamais la source est une archive dans un format de compression reconnu (ex : zip, tar.gz, etc ...), alors elle sera décompressée automatiquement vers votre cible.
- Dans les best-practices de docker, ils recommandent d’utiliser l'instruction COPY quand les fonctionnalités du ADD ne sont pas requises.

DOCKER – Dockerfile ENV et ARG

- Ils permettent tous les deux de stocker une valeur
- ARG permet de stocker en tant que variable temporaire, utilisable qu'au niveau de votre Dockerfile.
- ENV permet de stocker une variable d'environnements accessible depuis le Dockerfile et votre conteneur.

DOCKER – Dockerfile RUN, CMD, ENTRYPOINT

- L'instruction RUN est exécutée pendant la construction de votre image, elle est souvent utilisée pour installer des packages logiciels qui formeront les différentes couches de votre image.
- L'instruction ENTRYPOINT est exécutée pendant le lancement de votre conteneur et permet de configurer un conteneur qui s'exécutera en tant qu'exécutable.
- L'instruction CMD est aussi exécutée pendant le lancement de votre conteneur, elle définit les commandes et/ou les paramètres de l'instruction ENTRYPOINT par défaut, et qui peuvent être surchargées à la fin de la commande docker run.

DOCKER – Docker compose

- Docker Compose est un outil permettant de **définir le comportement de vos conteneurs** et d'**exécuter des applications Docker à conteneurs multiples**.
- La config se fait à partir d'un fichier YAML, et ensuite, avec une seule commande, vous **créez et démarrez tous vos conteneurs de votre configuration**.
- Installation de docker compose

DOCKER – Docker compose

- Pour utiliser docker compose, il faut créer un fichier docker-compose.yml
- Docker compose lance les conteneurs en tant que services
- Dans docker-compose.yml on doit :
- Indiquer la version (Il existe plusieurs version de docker-compose, les versions à partir du 3 sont compatible avec docker swarm)
- Indiquer les services (les services sont nos conteneurs)
- Chaque service est défini par son nom
- Chaque service peut avoir les options suivantes:
- Image
- Container_name
- Restart
- Volumes
- Environment
- Ports

DOCKER – Docker compose

- Docker compose fournit un cli pour démarrer les services et les arrêter
- Commande pour démarrer les services à partir d'un docker compose
- Docker-compose up
- Commande pour arrêter les services
- Docker-compose down
- On peut ajouter –d au commande de démarrage de services pour les démarrer en arrière plan

DOCKER – Docker compose

- Première exemple de docker compose
- Démarrer deux services
- Premier service pour démarrer un serveur web
- Deuxième service pour démarrer une base de données mysql

DOCKER – Docker compose

- Docker compose permet de générer des services directement à partir de dockerfile à l'aide de l'option build
- Dans l'option build, il faut indiquer le chemin du dockerfile à l'utiliser pour créer le conteneur

DOCKER – registry

- Docker Registry est une application open source qui permet de mettre en place un serveur de distribution d'image Docker
- L'utilisation des Docker registry est recommandé dans les cas suivants :
- Contrôler la sécurité d'accès au images Docker
- Contrôler l'utilisation des images dans nos pipelines d'intégration et de déploiement

DOCKER – Création

- Pour créer un Docker Registry :
- On peut installer directement docker-registry
- On peut utiliser l'image Docker registry pour déployer un Docker registry

DOCKER – Registry Déploiement d'image

- Pour déployer une image sur un Docker-registry privé
- Il faut créer un tag pour identifier l'image
- Le tag doit être sous le format suivant :
- <SERVER NAME REGISTRY>:<PORT SERVER REGISTRY>/<IMAGE_NAME>
- Le déploiement se fait à l'aide de la commande docker push
- La récupération se fait à l'aide de la commande docker pull
- Dans le cadre, de l'utilisation de l'image registry, on peut accéder au images directement par API rest des ressources

DOCKER – Sécurisation d'un Docker Registry

- Pour sécuriser un Docker Registry, Il faut sécuriser:

1- Les échanges avec le serveur

2- L'authentification des utilisateurs

3- Les données (dans le cadre d'utilisation d'une image Docker registry)

DOCKER – Sécurisation d'un Docker Registry

Pour sécuriser les échanges avec le serveur:

- On peut utiliser un certificat de chiffrement auto-signé
- Démo

DOCKER – Sécurisation d'un Docker Registry

Pour sécuriser l'authentification avec le serveur:

- On peut utiliser une authentification basique à l'aide htpasswd
- Démo

DOCKER – Débogage des conteneurs

Pour déboguer un conteneurs, Docker nous met à disposition, une multitude de fonctionnalités pour analyser le comportement de nos conteneurs.
On peut trouver :

- Docker stats : cette fonctionnalité nous fournit un rapport d'analyse sur l'utilisation des ressources par le conteneur
- La commande docker stats permet d'utiliser l'option ' format' pour un meilleur export des resultats
- Docker inspect : cette fonctionnalité nous fournit des informations détaillées sur le conteneurs sous format json, elle accepte également l'option 'format'
- Docker logs : cette fonctionnalité nous fournit la commande en cours d'exécution sur notre conteneurs
- Docker history: cette fonctionnalité nous fournit des informations sur l'historique de la construction de l'image

DOCKER – Débogage des conteneurs

Exercice:

Réaliser un script qui permet, à l'aide des fonctionnalités de débogage, d'extraire la liste :

- Des adresses IP de chaque conteneurs.
- Les adresses MAC
- Les ports utilisés
- Les adresses GATEWAY

DOCKER – Débogage des conteneurs

Exercice:

Réaliser un script qui permet, à l'aide des fonctionnalités de débogage, d'extraire la liste :

- Des adresses IP de chaque conteneurs.
- Les adresses MAC
- Les ports utilisés
- Les adresses GATEWAY