Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

DOCKER COURS

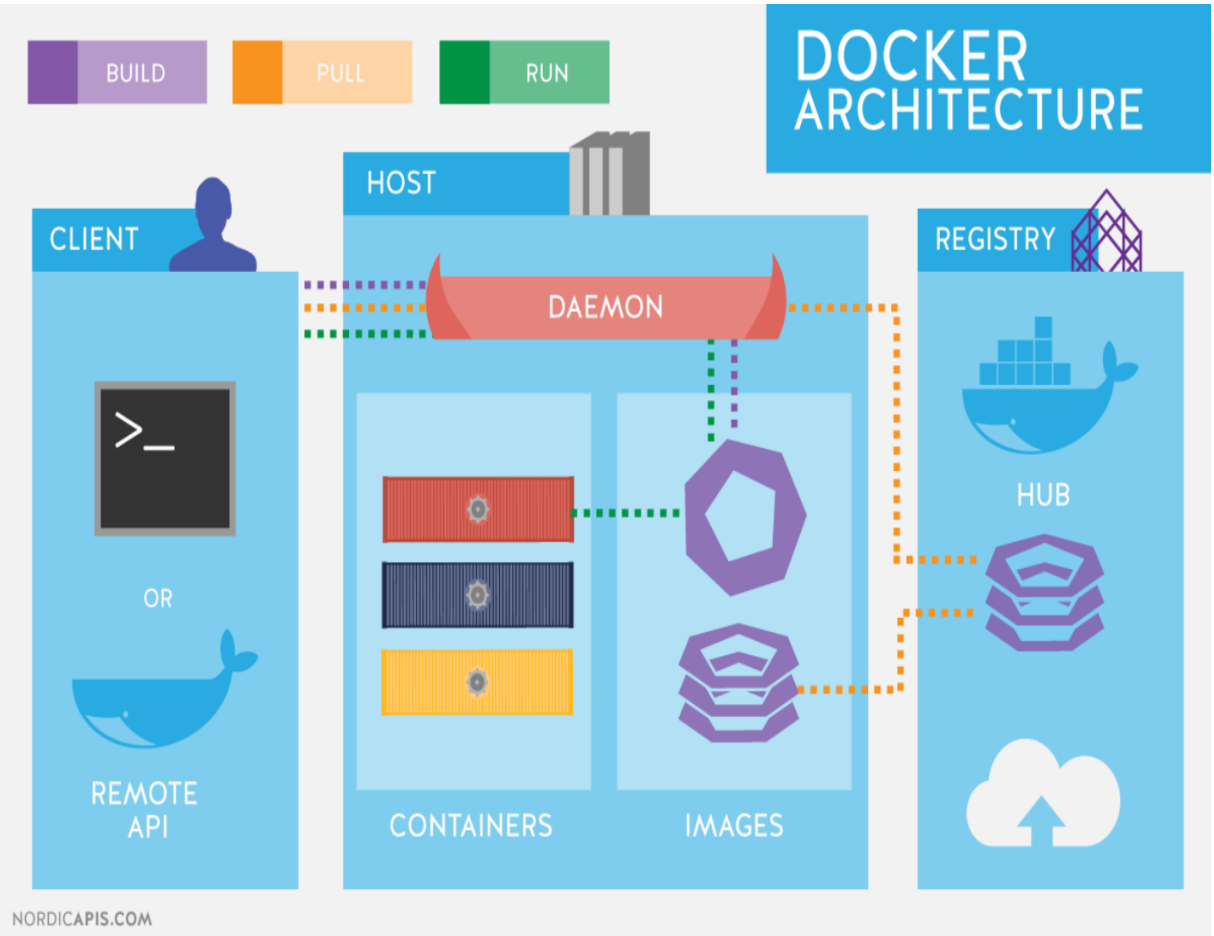
Docker est la plateforme de conteneurisation la plus utilisée. Les conteneurs et les microservices sont de plus en plus utilisés pour le développement et le déploiement des applications. C'est ce qu'on appelle le développement " cloud-native ". Dans ce contexte, Docker est devenue une solution massivement exploitée en entreprise.

C'est quoi un conteneur ?

Avant de découvrir Docker, vous devez comprendre ce qu'est un conteneur. Il s'agit d'un environnement d'exécution léger, et d'une alternative aux méthodes de virtualisation traditionnelles basées sur les machines virtuelles. L'une des pratiques clés du développement de logiciel moderne est d'isoler les applications déployées sur un même hôte ou sur un même cluster. Ceci permet d'éviter qu'elles interfèrent.

Pour exécuter les applications, il est toutefois nécessaire d'exploiter des packages, des bibliothèques et divers composants logiciels. Pour exploiter ces ressources tout en isolant une application, on utilise depuis longtemps les machines virtuelles. Celles-ci permettent de séparer les applications entre elles sur un même système, et de réduire les conflits entre les composants logiciels et la compétition pour les ressources. Cependant, une alternative a vu le jour : les conteneurs.

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT



La plateforme Docker repose sur plusieurs technologies et composants. Voici les principaux éléments.

Docker Engine

Le Docker Engine est l'application à installer sur la machine hôte pour créer, exécuter et gérer des conteneurs Docker. Comme son nom l'indique, il s'agit du moteur du système Docker. C'est ce moteur qui regroupe et relie les différents composants entre eux. C'est la technologie client-serveur permettant de créer et d'exécuter les conteneurs, et le terme Docker est souvent employé pour désigner Docker Engine.

On distingue le Docker Engine Enterprise et le Docker Engine Community. La Docker Community Edition est la version originale, proposée en open source gratuitement. La version Enterprise, lancée en 2017, ajoute des fonctionnalités de gestion comme le contrôle de cluster et la gestion d'image ou la détection de vulnérabilité. Elle est tarifée à 1500 $ par noeud et par an.

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com

Docker Daemon

Le Docker Daemon traite les requêtes API afin de gérer les différents aspects de l'installation tels que les images, les conteneurs ou les volumes de stockage.

Docker Client

Le client Docker est la principale interface permettant de communiquer avec le système Docker. Il reçoit les commandes par le biais de l'interface de ligne de commande et les transmet au Docker Daemon.

Dockerfile

Chaque conteneur Docker débute avec un " Dockerfile ". Il s'agit d'un fichier texte rédigé dans une syntaxe compréhensible, comportant les instructions de création d'une image Docker. Un Dockerfile précise le système d'exploitation sur lequel sera basé le conteneur, et les langages, variables environnementales, emplacements de fichiers, ports réseaux et autres composants requis.

Les images Docker

Une image Docker est un modèle en lecture seule, utiliser pour créer des conteneurs Docker. Elle est composée de plusieurs couches empaquetant toutes les installations, dépendances, bibliothèques, processus et codes d'application nécessaires pour un environnement de conteneur pleinement opérationnel.

Après avoir écrit le Dockerfile, on invoque l'utilitaire " build " pour créer une image basée sur ce fichier. Cette image se présente comme un fichier portable indiquant quels composants logiciels le conteneur exécutera et de quelle façon.

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

Les conteneurs Docker

Un conteneur Docker ou Docker Container est une instance d'image Docker exécutée sur un microservice individuel ou un stack d'application complet. En lançant un conteneur, on ajoute une couche inscriptible sur l'image. Ceci permet de stocker tous les changements apportés au conteneur durant le runtime.

Docker run

L'utilitaire "run" de Docker est la commande permettant de lancer un conteneur. Chaque conteneur est une instance d'une image. Les conteneurs sont conçus pour être temporaires, mais peuvent être arrêtés et redémarrés dans le même état. Plusieurs instances d'une même image peuvent être exécutées simultanément.

Le registre Docker

Le registre Docker est un système de catalogage permettant l'hébergement et le " push and pull " des images Docker. Il est possible d'utiliser votre propre registre local, ou l'un des nombreux services de registre hébergés par des tiers comme Red Hat Quay, Amazon ECR, Google Container Registry.

Le Docker Hub est le registre officiel de Docker. Il s'agit d'un répertoire SaaS permettant de gérer et de partager les conteneurs. On peut y trouver des images Docker de projets open source ou de vendeurs logiciels. Il est possible de télécharger ces images et de partager les vôtres.

Un registre Docker organise les images dans différents répertoires de stockage. Chacun d'entre eux contient différentes versions d'une image Docker partageant le même nom d'image.

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

DOCKERFILE

Un Dockerfile est un script texte contenant une série d'instructions pour assembler une image Docker. Voici la structure générale d'un Dockerfile, ainsi que des explications sur chaque instruction clé.

Exemple de dockerfile :

```
# Utilisation d'une image de base
FROM ubuntu:20.04

# Définition du mainteneur (optionnelle)
LABEL maintainer="votre.email@example.com"

# Mise à jour des paquets et installation de dépendances
RUN apt-get update && apt-get install -y \
    curl \
    vim \
    git

# Création d'un répertoire de travail
WORKDIR /app

# Copier les fichiers locaux dans l'image
COPY . /app

# Installation de dépendances spécifiques à l'application
RUN npm install

# Exposer un port pour l'application
EXPOSE 3000

# Définir une variable d'environnement
ENV NODE_ENV=production

# Commande exécutée au démarrage du conteneur
CMD ["npm", "start"]
```

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

Explication des instructions principales :

FROM : Définit l'image de base à partir de laquelle l'image sera construite. Par exemple, ubuntu:20.04 ou node:14.

LABEL : Ajoute des métadonnées à l'image, comme l'information du mainteneur.

RUN : Exécute une commande à l'intérieur de l'image lors de sa création. Il est souvent utilisé pour installer des paquets ou exécuter des scripts.

WORKDIR : Définit le répertoire de travail dans lequel les commandes suivantes seront exécutées.

COPY : Copie des fichiers depuis votre système local vers l'image Docker.

EXPOSE : Indique quel port sera exposé lorsque le conteneur sera en cours d'exécution.

ENV : Définit des variables d'environnement pour le conteneur.

CMD : Spécifie la commande par défaut à exécuter lorsque **le conteneur démarre**.

Utilisation du Dockerfile :

Pour créer une image Docker à partir de ce Dockerfile, exécutez la commande suivante dans le répertoire où se trouve le Dockerfile :

```
docker build -t nom-image .
```

Ensuite, vous pouvez exécuter un conteneur basé sur cette image avec :

```
docker run -d -p 3000:3000 nom-image
```


Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

Commandes impératives

Les commandes Docker "impératives" font référence à celles qui sont exécutées directement dans la ligne de commande pour interagir avec Docker. Voici un récapitulatif des commandes Docker les plus couramment utilisées, classées par catégorie :

1. Gestion des Images

docker pull <image>

Télécharge une image depuis Docker Hub ou un registre privé.

docker pull nginx:latest

docker build -t <nom_image>:<tag> .

Construit une image à partir d'un Dockerfile dans le répertoire courant.

docker build -t mon_app:1.0 .

docker images

Liste toutes les images locales.

docker images

docker rmi <image>

Supprime une image.

docker rmi nginx

2. Gestion des Conteneurs

docker run <options> <image>

Lance un nouveau conteneur à partir d'une image. Les options les plus courantes incluent :

- -d : Exécuter en arrière-plan (détaché)
- -p : Mapper un port local vers un port du conteneur
- --name : Donner un nom au conteneur

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com

Exemple :

docker run -d -p 8080:80 --name mon_conteneur nginx

docker ps

Affiche les conteneurs en cours d'exécution.

docker ps

Pour voir les conteneurs arrêtés aussi, utilisez -a :

docker ps -a

docker stop <nom_conteneur>

Arrête un conteneur.

docker stop mon_conteneur

docker start <nom_conteneur>

Redémarre un conteneur déjà existant.

docker start mon_conteneur

docker rm <nom_conteneur>

Supprime un conteneur arrêté.

docker rm mon_conteneur

3. Gestion des Volumes

docker volume create <nom_volume>

Crée un volume Docker.

docker volume create mon_volume

docker volume ls

Liste tous les volumes Docker.

docker volume ls

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com

docker volume rm <nom_volume>

Supprime un volume.

docker volume rm mon_volume

docker run -v <nom_volume>:<chemin_dans_conteneur> <image>

Monte un volume dans un conteneur.

Exemple :

docker run -d -v mon_volume:/data busybox

4. Gestion des Réseaux

docker network create <nom_reseau>

Crée un réseau Docker.

docker network create mon_reseau

docker network ls

Liste tous les réseaux Docker.

docker network ls

docker network connect <nom_reseau> <nom_conteneur>

Connecte un conteneur à un réseau Docker.

docker network connect mon_reseau mon_conteneur

docker network disconnect <nom_reseau> <nom_conteneur>

Déconnecte un conteneur d'un réseau.

docker network disconnect mon_reseau mon_conteneur

5. Exploration des Conteneurs

docker exec -it <nom_conteneur> <commande>

Exécute une commande à l'intérieur d'un conteneur en cours d'exécution. Souvent utilisé pour démarrer un shell interactif (bash, sh).

Exemple pour obtenir un shell dans un conteneur :

docker exec -it mon_conteneur bash

docker logs <nom_conteneur>

Affiche les logs d'un conteneur.

docker logs mon_conteneur

docker inspect <nom_conteneur>

Retourne les détails JSON du conteneur ou d'une image.

docker inspect mon_conteneur

6. Autres Commandes Utiles

docker-compose up

Lance des services définis dans un fichier docker-compose.yml.

docker-compose up

docker stats

Affiche l'utilisation des ressources (CPU, mémoire, etc.) par les conteneurs.

docker stats

Combinaison de commandes

Voici quelques exemples de workflows :

Exécuter un conteneur en arrière-plan et supprimer automatiquement le conteneur une fois terminé :

docker run --rm -d -p 8080:80 nginx

Supprimer tous les conteneurs arrêtés :

docker rm $(docker ps -a -q)

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

DOCKER COMPOSE

Docker Compose est un outil qui vous permet de définir et de gérer des applications multi-conteneurs Docker à l'aide d'un fichier YAML (docker-compose.yml). Ce fichier décrit les services, les réseaux et les volumes nécessaires à votre application. Voici une vue d'ensemble de son utilisation.

Structure de base d'un fichier docker-compose.yml

Voici un exemple de fichier docker-compose.yml pour une application simple avec un service web et une base de données.

```yaml
version: '3.8'  # Version de la syntaxe Docker Compose

services:  # Définition des services de l'application
  web:
    image: nginx:latest  # Image du service
    ports:
      - "8080:80"  # Mappe le port 8080 sur l'hôte au port 80 dans le conteneur
    volumes:
      - ./html:/usr/share/nginx/html  # Monte un volume pour les fichiers HTML

  db:
    image: mysql:5.7  # Image de la base de données
    environment:
      MYSQL_ROOT_PASSWORD: example  # Variable d'environnement pour le mot de passe root
      MYSQL_DATABASE: mydatabase  # Crée une base de données par défaut
    volumes:
      - db_data:/var/lib/mysql  # Persiste les données de la base de données

volumes:  # Définition des volumes
  db_data:
```

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.

rambeloson.boni9@gmail.com

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

Voici une explication détaillée de chaque ligne de votre fichier docker-compose.yml :

version: '3.8' # Version de la syntaxe Docker Compose

version: '3.8' : Cela spécifie la version de la syntaxe de Docker Compose que vous utilisez. La version 3.8 est compatible avec les fonctionnalités récentes de Docker.

#################

services: # Définition des services de l'application

services: : Cette section définit les différents services qui composent votre application. Chaque service correspond à un conteneur Docker.

Service web

web: : Nom du service, ici nommé web, qui représentera un conteneur Nginx.

#################

image: nginx:latest # Image du service

image: nginx:latest : Indique l'image Docker à utiliser pour ce service. Ici, cela signifie que le service web utilisera la dernière version de

l'image officielle Nginx disponible sur Docker Hub.

#################

ports:

- "8080:80" # Mappe le port 8080 sur l'hôte au port 80 dans le conteneur

ports: : Cette section spécifie les ports à exposer. Le port 8080 sur l'hôte (votre machine) sera mappé au port 80 du conteneur Nginx. Ainsi, vous pourrez accéder à votre serveur web via http://localhost:8080.

#################

volumes:

- ./html:/usr/share/nginx/html # Monte un volume pour les fichiers HTML

volumes: : Cette section permet de monter des volumes, c'est-à-dire de lier un répertoire de votre système de fichiers à un répertoire dans le conteneur. Ici, le répertoire local ./html sera

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com

monté dans /usr/share/nginx/html à l'intérieur du conteneur. Cela signifie que tous les fichiers HTML présents dans le répertoire ./html de votre machine seront accessibles par Nginx.

#################

Service db

db: : Nom du service, ici nommé db, qui représentera un conteneur MySQL.

#################

image: mysql:5.7 # Image de la base de données

image: mysql:5.7 : Indique l'image Docker à utiliser pour ce service, ici la version 5.7 de MySQL. Cela tirera l'image officielle de MySQL depuis Docker Hub.

#################

environment:

MYSQL_ROOT_PASSWORD: example # Variable d'environnement pour le mot de passe root

environment: : Cette section définit les variables d'environnement qui seront passées au conteneur lors de son démarrage. Ici, MYSQL_ROOT_PASSWORD définit le mot de passe de l'utilisateur root pour la base de données MySQL. Dans cet exemple, il est fixé à example.

MYSQL_DATABASE: mydatabase # Crée une base de données par défaut

MYSQL_DATABASE: mydatabase : Cette variable d'environnement crée une base de données nommée mydatabase lors du démarrage du conteneur.

#################

volumes:

- db_data:/var/lib/mysql # Persiste les données de la base de données

volumes: : De manière similaire au service web, cette section permet de persister les données de la base de données MySQL. Le volume db_data sera monté dans /var/lib/mysql, qui est le répertoire où MySQL stocke ses fichiers de données. Cela garantit que les données de la base de données ne seront pas perdues lorsque le conteneur est arrêté ou supprimé.

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

##################

Volumes

volumes: # Définition des volumes

 db_data:

volumes: : Cette section définit les volumes utilisés par les services. Ici, db_data: définit un volume nommé db_data qui sera utilisé pour stocker les données de MySQL. En définissant le volume ici, Docker s'assure que le volume est géré et persistant en dehors du cycle de vie du conteneur.

Instructions de base pour Docker Compose

Lancer les services

Utilisez la commande suivante pour démarrer tous les services définis dans le fichier docker-compose.yml.

docker-compose up

Pour exécuter les services en arrière-plan (mode détaché), ajoutez l'option -d :

docker-compose up -d

Arrêter les services

Pour arrêter les services en cours d'exécution :

docker-compose down

Cette commande arrête et supprime tous les conteneurs définis dans le fichier docker-compose.yml.

Afficher les logs

Pour afficher les logs de tous les services :

docker-compose logs

Pour afficher les logs d'un service spécifique :

docker-compose logs web

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com

Résumé par Mr Bonitah RAMBELOSON
Consultant DevOps/SysOps | Ingénieur de production IT

Redémarrer les services

Pour redémarrer tous les services :

docker-compose restart

Exécuter une commande dans un service

Si vous souhaitez exécuter une commande dans un conteneur d'un service, utilisez exec :

docker-compose exec web bash

Mon objectif est de fournir des solutions d'infrastructure et de déploiement fiables, évolutives et sécurisées pour aider les entreprises à atteindre leurs objectifs commerciaux.
rambeloson.boni9@gmail.com