Docker

Bachelor 3ème année

Thibault Vinchent, avec l'aide de la chaine Youtube « xavki »

Thibault Vinchent

Formateur permanent EPSI

thibault.vinchent@competences-developpement.com

Toujours à votre disposition pour des compléments de cours, suivi de projet.

- Formateur en conception d'applications depuis 2015 (écoles d'ingénieur, universités, instituts)

- Ingénieur développement depuis 2010 (Sopra, Toyota, Eurotunnel etc.).

Illustration : Exemples de sites créés…
… dont certains *toujours en fonctionnement…*

Docker

Programme de ces 3 demi-journée

- **Introduction :** histoire, utilité, concepts, infrastructure

1. Premiers pas & installation, mise en situation via ateliers basiques
2. Utilisation avancée, édition d’un environnement docker
3. Volumes et build, publication sur docker hub, notions d’orchestration de conteneurs

QCM

Docker

Introduction

Présent dans de très nombreuses entreprises informatiques

Utile pour les :

- Développeurs
- Devops
- Sysadmin

Docker

Introduction : histoire

2013 : première release

Créé par Solomon Hykes

Qui est parti de dotcloud

Dispose :

- d’un repo github

- d’un site

Docker

Introduction : objectifs

Simplifier les déploiements

Changer le mode de livrables

Faciliter les dépendances

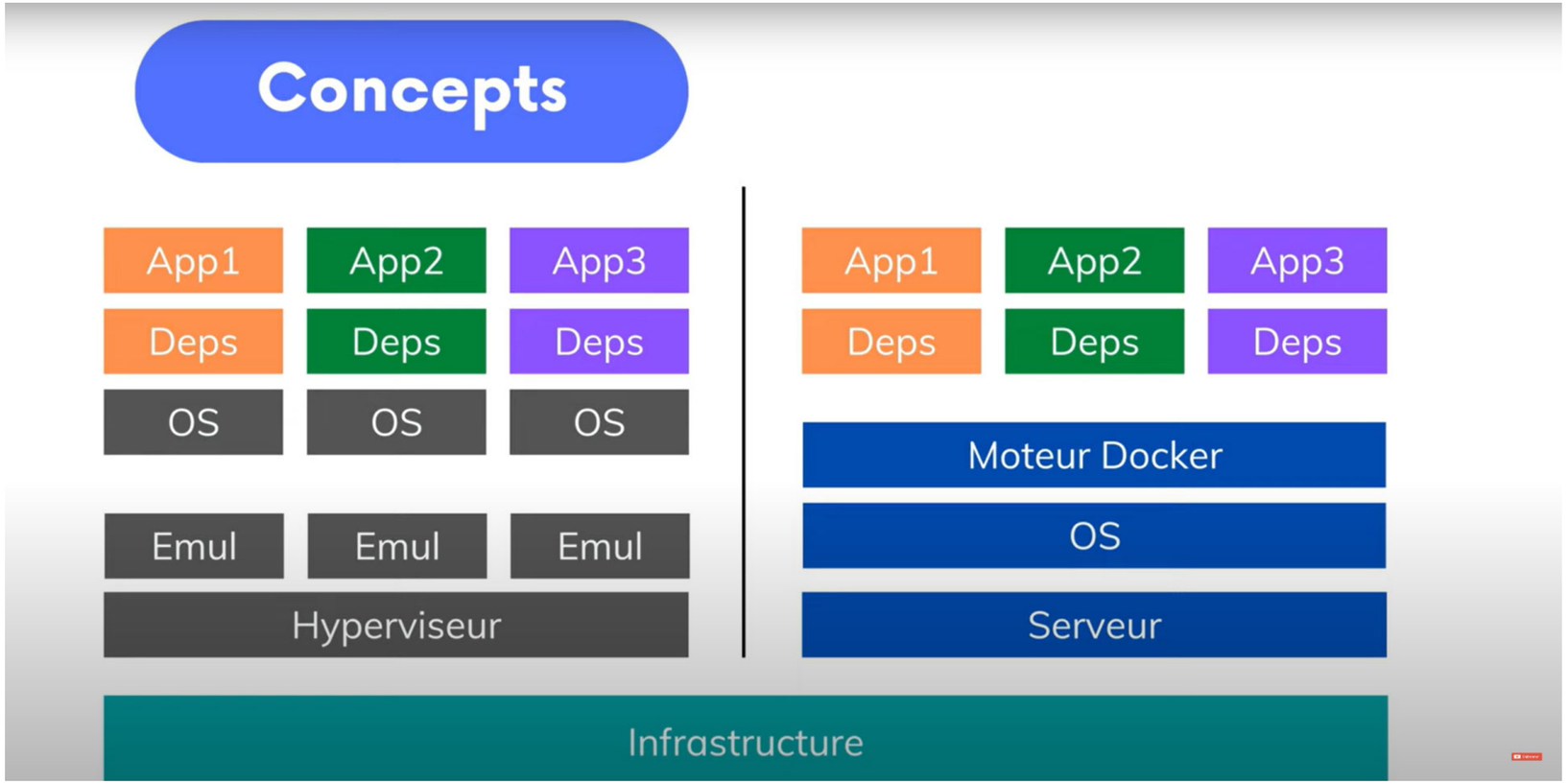
Docker

Introduction : concepts

Conteneur (ou process isolé)

Images (enveloppe contenant le code et ses dépendances)

Différence d'infrastructure virtualisation / docker



1. Premiers pas & installation

Docker

1. Installation

Suppression des anciennes versions de docker

Télécharger sur le site https://www.docker.com/

Docker

1. Premiers pas : distinguer images et containers

Les containers :

- Activation d’une image
- Contenant des apps
- Isolés (niveau sécurité ET ressource)

Les images :

- Un package, une enveloppe
- Inactive
- Stockée à distance (sur docker Hub par ex) et rapatriée localement

Docker

Atelier #1 : déployer une application dockerisée

- Installer docker
- Récupérer (git clone) le repo suivant : https://github.com/tvinchent/docker.git
- Lancer la commande : docker compose up

Attention : si vous avez une erreur invoquant des autorisations, c'est probablement que vous n'avez pas cliqué dans le mail de confirmation d'inscription à Docker

Vous pourriez également avoir une erreur « no matching manifest linux » : il faut alors ajouter platform: linux/amd64 dans le service mysql

Docker

Atelier #2 : dockerisation d'une application de votre choix

- Atelier : dockerisation d'une application
- Créer une application avec les technos de votre choix (PHP avec MySQL, Python ou Java avec PostgreSQL, NodeJS avec MongoDB, autre..) qui affiche le contenu d'une BDD
- De quelles images docker va t'on avoir besoin ?
- Créer un docker-compose.yml, Dockerfile et init.sql

2. Utilisation avancée

Docker

Rappels et corrections

- Rappels sur la notion d’image et de conteneur
- Rappel du déploiement d’une appli dockerisée
- Wooclap
- Démarrage et installation des mise à jour Docker desktop
- Correction de l’exercice de dockerisation d’une appli :
  - Passage en revue du Dockerfile et docker-compose dans le cadre de ces 2 premiers ateliers

Docker

Création d'un dockerfile

Objectif : image reproductible, minimale, rapide à builder.

1. Base
2. Dépendances système
3. Dépendances applicatives
4. Code
5. Configuration runtime
6. Commande de démarrage

Docker

Utilité du Dockerfile et du docker-compose

Dockerfile : Il est utilisé pour définir l'image Docker d'une application. Il contient les instructions pour installer les dépendances, configurer l'environnement (comme PHP, Apache dans notre cas), copier le code source, installer les extensions, etc. Il construit l'environnement nécessaire pour exécuter l’application à l'intérieur d'un conteneur.

docker-compose.yml : Ce fichier permet de gérer plusieurs conteneurs qui travaillent ensemble (comme une application web, une base de données, et PhpMyAdmin). Il définit et orchestre les services, les réseaux, les volumes, et simplifie le lancement de l'ensemble de l'application en une seule commande (docker-compose up).

Docker

Le petit souci avec docker..

Utilisation intempestive des ressources de l'ordinateur..

D'où l'importance de terminer les processus inutilisés :

- Soit dans le docker desktop, en vérifiant qu'il n'y a plus rien également dans le systray*
- Soit avec la commande *docker system prune*

Limiter la ram. Dans le docker-compose :

mem_limit: 512m

mem_reservation: 256m

Docker

Atelier #3 : ajout de PhpMyAdmin

- En partant de l'atelier 1 :
  - Ajoutez le service PhpMyAdmin (PMA)
  - Validez en ajoutant une entrée dans la table

3. Les volumes

Docker - volumes

Programme d’aujourd’hui

- Rappels sur la notion d’image et de conteneur
- Volumes partie théorique :
  - Utilité
  - Mise en place
  - Les 3 types de volumes
  - Les utilisateurs liés aux volumes
- QCM
- Atelier 4 python
- Prochaine fois : mise en pratique des volumes

Docker - volumes

Rappels sur la notion d'image et de conteneur

- Image : récupéré sur des registry (docker hub), inactif
- Container : image activée pour les besoins de notre appli, 1 ou plusieurs

Docker - volumes

Utilité

- Persister les données (database, variables d'environnement etc.) de l'application.
- En les plaçant à l'extérieur des containers, et les rendant accessible à l'ensemble des containers.
- Possible de donner des niveaux de permission (R, RW etc.).
- Possible dès lors de faire des backups.
- Local ou distant.

Docker - volumes

Mise en place

- Création d’un volume
- Utilisation du volume en l’attachant à un container,
- Modification d’un fichier d’un volume commun à deux containers.
- Pour résumer, un volume est un répertoire d’un container docker qui va être monté sur un répertoire de notre host docker

Docker - volumes

Les 3 types de volumes

- Bind Mount : montage d’un répertoire dans le container. Comportement : surcharge des données du container à partir des données source sur notre volume.
- Volumes docker : création d’un volume et montage d’un répertoire dans ce volume (ici cantonné à var/lib/docker, le répertoire des volumes). Comportement : inverse du bind mount car ce sont les données du container qui surchargent les données du volume.
- TMPFS : espace de travail temporaire : pas de persistance de donnée mais espace de travail en mémoire qui sera stoppé avec le container.

Docker - volumes

Les utilisateurs liés aux volumes

- Important pour la sécurité.
- A l'image des processus dans les containers (à l'intérieur) qui sont référencés dans les volumes (à l'extérieur des containers), les utilisateurs des containers sont référencés dans les volumes
- Ce sont les ID qui vont être référencés et non les noms des utilisateurs
- Il faut donner les droits pour pouvoir RW les volumes
- L'utilisateur par défaut est root

Docker - volumes

QCM

- https://app.wooclap.com/DOCKERVOL

Docker - volumes

Atelier 4 python

- Correction sur https://github.com/tvinchent/docker-atelier-4
- Vous pouvez vous aider de la correction détaillée de l'atelier 3

4. Les builds

Docker - build

Programme d'aujourd'hui

- Rappels (image, conteneur et volumes, fichiers etc.).
- Build et publication d'une application sur Docker Hub
  - Commandes d'un dockerfile
  - Build de l'application
  - CLI : pull run push
  - Création d'une image sur un registre distant
- Modalités du TP

Docker - build

Rappels

- Conteneur : contenu actif créé sur la base d'une image
- Image : contenu inactif récupéré du registre
- Volumes : espace de persistance des données
- Fichiers :
  - docker-compose.yml : gestion et chargement des services et de leurs conteneurs, ports, volumes associés.
  - Dockerfile : définition de l'image d'une application pour l'exécuter à l'intérieur d'un conteneur.

Docker - build

Build et publication d’une application sur Docker Hub

- Commandes d’un dockerfile
- Build de l’application
- CLI : pull run push
- Création d’une image sur un registre distant

Docker - build

Build et publication d’une application sur Docker Hub

Commandes d’un dockerfile 1/5:

- Le Dockerfile est un fichier texte qui contient une série d'instructions pour construire une image Docker personnalisée.
- FROM
  - Description : Spécifie l'image de base à utiliser pour la construction de l'image.
  - Syntaxe : FROM NOM_IMAGE[:TAG]
  - Exemple : FROM ubuntu:20.04 : Utilise l'image Ubuntu 20.04 comme base.

Docker - build

Build et publication d’une application sur Docker Hub

Commandes d’un dockerfile 2/5:

- RUN
  - Description : Exécute une commande pendant la construction de l'image.
  - Syntaxe : RUN commande
  - Exemples :
    - RUN apt-get update && apt-get install -y python3 : Met à jour les paquets et installe Python 3.
    - RUN mkdir /app : Crée un répertoire nommé /app.

Docker - build

Build et publication d’une application sur Docker Hub

Commandes d’un dockerfile 3/5:

- CMD
  - Description : Spécifie la commande par défaut à exécuter lorsque le conteneur est lancé. Il ne peut y avoir qu'une seule instruction CMD par Dockerfile.
  - Syntaxe : CMD ["exécutable", "param1", "param2"]
  - Exemple :
    - CMD ["python3", "app.py"] : Exécute python3 app.py lorsque le conteneur démarre.

Docker - build

Build et publication d’une application sur Docker Hub

Commandes d’un dockerfile 4/5:

- COPY
  - Description : Copie des fichiers ou des répertoires depuis le contexte de construction vers le système de fichiers de l'image.
  - Syntaxe : COPY source destination
  - Exemple :
    - COPY . /app : Copie tous les fichiers du répertoire courant vers /app dans l'image.

Docker - build

Build et publication d’une application sur Docker Hub

Commandes d’un dockerfile 5/5:

- EXPOSE
  - Description : Informe Docker que le conteneur écoute sur les ports réseau spécifiés lors de l'exécution.
  - Syntaxe : EXPOSE port1 [port2 ...]
  - Exemple :
    - EXPOSE 80 : Le conteneur écoutera sur le port 80.

Docker - build

Build et publication d’une application sur Docker Hub

- Build de l’application :
  - docker build -t <votre-utilisateur-docker>/<nom-image>:<tag> .
  - Exemple :
    - docker build -t tvinchentepsi/atelier-docker-1:1.0 .
    - Attention à ne pas oublier le . À la fin !

Docker - build

Build et publication d'une application sur Docker Hub

- CLI : pull run push
  - Pull
    - Syntaxe : docker pull [OPTIONS] NOM_IMAGE[:TAG|@DIGEST]
    - Exemple : docker pull mysql:5.7
  - Run
    - Syntaxe : docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
    - Exemples :docker run -d --name mon-nginx -p 8080:80 nginx
- Création d'une image sur un registre distant

Docker - build

Build et publication d'une application sur Docker Hub

- CLI : pull run push
- Push : Création d'une image sur un registre distant
  - Syntaxe : docker push NOM_IMAGE[:TAG]
  - Exemples :
    - docker push tvinchentepsi/atelier-docker-1:1.0

Docker - build

Atelier / TP

- Travail en binôme
- Dockerisation d’une application de votre choix en solo
- Dépot de l'image de votre appli sur Docker Hub
- Récupération et déploiement par le binôme
  - Ouverture : Déploiement de l'ensemble des projets de la promo via Docker Swarm ?

Docker

Programme

- Fin de l’exercice de publication et déploiement d’une image docker hub
- Quelques bonnes pratiques
- Wooclap

Docker

Quelques bonnes pratiques

- Ne pas utiliser le @latest
- Vérifier la source FROM d’une image
- Utiliser le .dockerignore
- Attention aux données sensibles
  - Informations importantes de l’entreprise
  - Clefs de logiciels
  - Clefs API
  - Accès BDD etc.

Docker

Introduction à l'orchestration avancée

- Limites de Docker seul
- Présentation de Docker Swarm
- Introduction à Kubernetes
- Cas d'utilisation et comparaisons

Docker

Ressources et perspectives

- Certifications Docker :
  - **Docker Certified Associate (DCA) :** Certification officielle pour valider vos compétences. Couvre les fondamentaux de Docker, la sécurité, la mise en réseau, etc.
  - **Préparation :**
    - **Cours en ligne :** Plateformes comme Udemy, Coursera, ou Pluralsight proposent des formations.
    - **Examens blancs :** Pour vous entraîner dans les conditions de l'examen.

Docker

Tendances actuelles et futures des conteneurs

- Microservices : Les architectures microservices continuent de gagner en popularité.
- Serverless : Combinaison de conteneurs et de fonctions serverless pour une plus grande flexibilité.
- Edge Computing : Déploiement de conteneurs à la périphérie du réseau pour réduire la latence.
- Standardisation : Initiatives comme l'Open Container Initiative (OCI) pour standardiser les formats d'images et de runtime.
- Sécurité : Accent accru sur la sécurité des conteneurs et des chaînes d'approvisionnement logicielle (supply chain security).

Docker

Conseils pour aller plus loin

- Pratique régulière : Rien ne remplace l'expérience pratique. Créez des projets personnels pour appliquer vos connaissances.
- Contribuer à des projets open-source : Participez à la communauté en contribuant à des projets liés à Docker.
- Rester informé :
  - Blogs et newsletters : Suivez les blogs officiels et inscrivez-vous à des newsletters.
  - Conférences et meetups : Participez à des événements pour réseauter et apprendre des experts.
- Explorer d'autres outils :
  - Podman, Buildah : Alternatives à Docker pour la gestion des conteneurs et la construction d'images.
  - Ansible, Terraform : Pour l'automatisation et l'infrastructure as code.