keYce Informatique

CONTENEURISATION(DOCKER)

Code : B2COM0207 **Durée** : 20H

Ing. BOGNI-DANCHI T.

OBJECTIFS PEDAGOGIQUES

Ce cours a pour objectif d'introduire les concepts de conteneurisation en utilisant Docker. Les étudiants apprendront à déployer et gérer des applications à l'aide de conteneurs Docker, en optimisant les ressources et la portabilité des applications dans des environnements multi-clouds.

À l'issue de ce cours, l'étudiant doit être capable de :

- ✓ Comprendre les concepts fondamentaux de la conteneurisation et de Docker.
- ✓ Déployer et gérer des conteneurs Docker pour exécuter des applications.
- ✓ Créer et orchestrer des services multi-conteneurs à l'aide de Docker Compose.
- ✓ Optimiser la gestion des ressources et de la sécurité dans un environnement Docker.

PLAN

- **Module 1 : Introduction à la conteneurisation et à Docker**
- Concepts de base de la conteneurisation
- Introduction à Docker : installation et configuration
- Conteneurisation Vs Virtualisation
- Les composants de Docker

- **Module 2 : Les bases de docker(conteneurs, images, Dockerfile)**
- Les conteneurs
- Le Dockerfile
- Les commandes Docker de bases
- Le registres Docker

TP:

- Installation de Docker
- Création de Dockerfile
- Construction et gestion d'images
- Docker Compose
- Déploiement d'applications conteneurisées

MODULE2: LES BASES DE DOCKER(CONTENEURS, IMAGES, DOCKERFILE)

1-Dockerfile et les Commandes Docker

- **Le Dockerfile**

Un Dockerfile est un fichier texte qui contient un ensemble d'instructions permettant de construire une image Docker. C'est une sorte de "recette" pour créer un environnement logiciel standardisé et reproductible.

Instructions de Base

1.FROM

- •Définit l'image de base
- •Première instruction obligatoire
- •Exemple : *FROM ubuntu:20.04*

2.RUN

- •Exécute des commandes pendant la construction de l'image
- •Peut installer des packages, créer des répertoires
- •Exemple : *RUN apt-get update && apt-get install -y python3*

3.WORKDIR

- •Définit le répertoire de travail pour les instructions suivantes
- •Équivalent à cd dans un terminal
- •Exemple : *WORKDIR /app*

4.COPY

- •Copie des fichiers depuis l'hôte vers l'image
- •Exemple : *COPY . /app*

5.ADD

- •Similaire à COPY mais avec des fonctionnalités supplémentaires
- •Peut télécharger des fichiers distants
- •Décompresse automatiquement des archives

6.ENV

- •Définit des variables d'environnement
- •Exemple : *ENV PORT=8080*

7.EXPOSE

- •Indique les ports sur lesquels le conteneur écoutera
- •Exemple : *EXPOSE 80*

8.CMD

- •Définit la commande par défaut lors du lancement du conteneur
- •Une seule instruction CMD est autorisée
- •Exemple : *CMD ["python3", "app.py"]*

9.ENTRYPOINT

- •Configure le conteneur pour s'exécuter comme un exécutable
- •Peut être combiné avec CMD

MODULE 2: LES BASES DE DOCKER(CONTENEURS, IMAGES, DOCKERFILE)

1- Dockerfile et les Commandes Docker

- Les Commandes de base de Docker

Téléchargement et Recherche

- docker pull <image> : Télécharger une image
- docker search <terme> : Rechercher une image sur Docker Hub

Gestion des Images

- docker images : Lister les images locales
- docker rmi <image> : Supprimer une image
- docker build -t <nom>:<tag> . : Construire une image à partir d'un Dockerfile
- docker tag <image> <nouveau_nom> : Renommer/Étiqueter une image

Commandes de Gestion des Conteneurs

Création et Exécution

- docker run <options> <image> : Créer et démarrer un conteneur
  - -d : Mode détaché
  - -it : Mode interactif
  - --name : Nommer le conteneur
  - -p : Mapper les ports
  - -v : Monter des volumes

Listing et Inspection

- docker ps : Lister les conteneurs en cours d'exécution
- docker ps -a : Lister tous les conteneurs
- docker inspect <conteneur> : Détails d'un conteneur

Contrôle des Conteneurs

- docker start <conteneur> : Démarrer un conteneur arrêté
- docker stop <conteneur> : Arrêter un conteneur
- docker restart <conteneur> : Redémarrer un conteneur
- docker rm <conteneur> : Supprimer un conteneur
- docker pause <conteneur> : Mettre en pause un conteneur
- docker unpause <conteneur> : Reprendre un conteneur mis en pause

Interaction

- docker exec -it <conteneur> <commande> : Exécuter une commande dans un conteneur en cours
- docker logs <conteneur> : Afficher les logs
- docker attach <conteneur> : Se connecter au processus principal

Par Thomas BOGNI-DANCHI

MODULE 2: LES BASES DE DOCKER(CONTENEURS, IMAGES, DOCKERFILE)

1-Dockerfile et les Commandes Docker

- **Les Commandes de base de Docker**

Commandes Réseau

- docker network ls : Lister les réseaux
- docker network create <nom> : Créer un réseau
- docker network connect <réseau> <conteneur> : Connecter un conteneur à un réseau

Commandes de Volume

- docker volume create <nom> : Créer un volume
- docker volume ls : Lister les volumes
- docker volume rm <nom> : Supprimer un volume

Commandes de Nettoyage

- docker system prune : Supprimer les ressources inutilisées
- docker image prune : Supprimer les images non utilisées
- docker container prune : Supprimer les conteneurs arrêtés

2- Les images Docker

Les **images Docker** sont les éléments de base qui permettent de créer et de lancer des conteneurs Docker. Une image peut être considérée comme un **modèle prêt à l'emploi** qui contient tout le nécessaire pour exécuter une application, notamment :

- Le système d'exploitation de base (souvent une version minimale comme Alpine ou Ubuntu).
- Les dépendances logicielles.
- Les fichiers et configurations de l'application.

Cycle de vie des images

- **Construction :** Les images sont créées à l'aide d'un fichier de configuration appelé **Dockerfile** , qui contient des instructions pour assembler l'image.
- **Stockage :** Les images sont sauvegardées dans des registres (publics ou privés), comme **Docker Hub** ou des registres internes.
- **Exécution :** Les conteneurs sont créés à partir d'une image grâce à la commande *docker run*.

2- Les images Docker

Caractéristiques des images Docker

1. Format en couches :

Les images sont construites sous forme de **couches superposées** , en utilisant un système de fichiers Union. Chaque instruction dans le Dockerfile ajoute une couche. Par exemple :

- Ajouter un fichier.
- Installer une dépendance.
- Configurer un environnement.

Ces couches permettent de :

- **Réutiliser les éléments existants** pour réduire la taille et accélérer la construction.
- **Partager des canapés communes** entre plusieurs images.

2. Portabilité :

Une fois créée, une image est indépendante de la plateforme sous-jacente, ce qui signifie qu'elle peut être exécutée sur n'importe quel hôte Docker (Linux, Windows, macOS).

3. Source de conteneurs :

Les conteneurs Docker sont des instances en cours d'exécution d'images. Une seule image peut servir à créer plusieurs conteneurs.

Création d'images avec un Dockerfile

```
# Utilisation de l'image de base Ubuntu
FROM ubuntu:20.04
# Installation d'Apache
RUN apt-get update && apt-get install -y apache2
# Ajout d'un fichier HTML
 COPY index.html /var/www/html/index.html
# Exposition du port 80
EXPOSE 80
# Commande à exécuter au démarrage
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

3- Les registres Docker

- **Définition**

Un **registre Docker** est un dépôt (repository) où les **images Docker** sont stockées, partagées et gérées. Les registres servent de point central pour :

- Sauvegarder vos images Docker après leur création.
- Télécharger des images existantes pour créer des conteneurs.
- Partager des images avec d'autres utilisateurs ou équipes.

Les registres permettent d'organiser et de distribuer les images en garantissant leur versionnement et leur disponibilité.

- **Types de registres Docker**

1.Registres publics :

•Le **Docker Hub** est le registre public officiel tenu par Docker. Il contient des milliers d'images publiques, comme Nginx, MySQL, Redis, etc.

•Ces registres sont accessibles à tous et permettent le partage de projets open source.

•Exemple d'utilisation :

frapper

docker pull nginx # Télécharger l'image Nginx depuis Docker Hub docker push mon_image # Pousser une image vers un registre public

3- Les registres Docker

- **Types de registres Docker**

2.Registres privés :

•Les registres privés sont hébergés en interne ou sur une infrastructure cloud pour une utilisation limitée à une organisation ou à un projet.
•Ils offrent plus de contrôle et de sécurité pour les images sensibles.
•Exemple :
•Registre de conteneurs élastiques AWS (ECR)
•Registre de conteneurs Google (GCR)
•Registre de conteneurs Azure

Docker Hub : Le registre public officiel

•Docker Hub propose :
•**Images officielles :** Des images vérifiées et exécutées par Docker ou des organisations partenaires. (Ex. : nginx, mysql, redis)
•**Dépôts communautaires :** Des images partagées par des utilisateurs.
•**Gestion des dépôts privés :** Vous pouvez stocker vos images dans des dépôts privés, limités au nombre de dépôts gratuits pour un compte Docker gratuit.

3- Les registres Docker

- Commandes des registres :

- **Se connecter à un registre :** docker login REGISTRY_URL
- **Télécharger une image :** docker pull REGISTRY_URL/IMAGE_NAME
- **Pousser une image :** docker push REGISTRY_URL/IMAGE_NAME
- **Lister les tags disponibles pour une image :** curl REGISTRY_URL/v2/IMAGE_NAME/tags/list

Création d'un registre privé

Si vous souhaitez héberger un registre privé, Docker propose un conteneur pré-construit pour cela.

1. **Lancer un registre privé avec Docker :**

   docker run -d -p 5000:5000 --name registry registry:2

   Cela lance un registre local accessible via localhost:5000.

2. **Pousser une image vers le registre privé :**

   docker tag mon_image localhost:5000/mon_image

   docker push localhost:5000/mon_image

3. **Télécharger une image depuis le registre privé :**

   docker pull localhost:5000/mon_image

II-5- Docker compose

Docker Compose est un outil permettant de définir et de gérer plusieurs conteneurs Docker comme une seule application. Grâce à un fichier de configuration YAML, vous pouvez décrire les services (conteneurs), leurs réseaux, leurs volumes et leurs dépendances, puis les gérer avec des commandes simples.

Pourquoi utiliser Docker Compose ?

- **Faciliter la gestion multi-conteneurs :** Parfait pour les applications complexes avec plusieurs services (exemple : une API, une base de données, un frontend).
- **Automatisation :** Permet de démarrer, arrêter et configurer tous les conteneurs d'une application avec une seule commande.
- **Portabilité :** Avec un fichier docker-compose.yml, toute l'application est portable et facile à déployer.

- Structure de Docker Compose

MODULE2: LES BASES DE DOCKER(CONTENEURS, IMAGES, DOCKERFILE)

II-4- TP

- TP 2.1 Docker

Installation de Docker, Docker compose

- TP 2 Image, Conteneur et Registre Docker

Création du Dockerfile

Création de l'image d'une application

Création d'un conteneur pour exécuté l'application

Sauvegarde de l'image sur Docker hub