

PRÉSENTATION DE DOCKER

QU'EST-CE QUE DOCKER

DÉFINITION

Docker est une plateforme **open source** qui permet de créer, déployer et gérer des applications et des services dans des **containers**. Un container est une unité logicielle légère comprenant tout ce dont une application à besoin pour fonctionner.

AVANTAGES

- **Portabilité** : fonctionnement uniforme sur différentes plateformes
- **Isolation** : séparation des applications et limitation des dépendances
- **Légèreté** : pas besoin d'allouer une machine virtuelle complète

CAS D'UTILISATION

- **Déploiement simplifié** d'applications web
- **Environnements de développement isolés**
- **Intégration continue** et **déploiement continu**
- **Tests**

ARCHITECTURE DE DOCKER

DOCKER ENGINE

Le **Docker Engine** est le cœur et le moteur de Docker. Il gère la **création**, l'**exécution** et la **gestion des containers**.

DOCKER DAEMON

Le **Docker Daemon** (ou dockerd) est un processus qui gère les **images** et les **containers**, et communique avec d'autres daemons pour gérer les services Docker.

DOCKER CLIENT

Le **Docker Client** est l'**interface en ligne de commande (CLI)** par laquelle un utilisateur peut interagir avec le **Docker Daemon**.

INSTALLATION DE DOCKER

PRÉREQUIS

SYSTÈMES D'EXPLOITATION COMPATIBLES

| Système d'exploitation | Compatibilité |
|------------------------|---------------|
| Windows | Oui |
| Mac | Oui |
| Linux | Oui |

CONDITIONS MATÉRIELLES

- **Processeur 64 bits**
- **4 Go de RAM minimum** (8 Go recommandé)
- **Espace disque** suffisant pour les images et les conteneurs

INSTALLATION SUR WINDOWS

TÉLÉCHARGEMENT DE DOCKER DESKTOP

1. Accédez au **site officiel** de Docker : <https://www.docker.com>
2. Téléchargez **Docker Desktop** pour Windows
3. Exécutez le **fichier d'installation** téléchargé

CONFIGURATION

1. Suivez les instructions de l'**assistant d'installation**
2. **Redémarrez** votre ordinateur si nécessaire
3. Vérifiez que Docker est bien installé en ouvrant une invite de commande et en tapant `docker --version`

INSTALLATION SUR MAC

TÉLÉCHARGEMENT DE DOCKER DESKTOP

1. Accédez au site officiel de Docker : <https://www.docker.com>
2. Téléchargez Docker Desktop pour **Mac**
3. Ouvrez le fichier d'installation téléchargé

CONFIGURATION

1. Suivez les instructions de l'**assistant d'installation**
2. Redémarrez votre ordinateur si nécessaire
3. Vérifiez que **Docker** est bien installé en ouvrant un terminal et en tapant docker --version

INSTALLATION SUR LINUX (UBUNTU, DEBIAN, CENTOS)

DOCKER POUR DÉBUTANTS

UTILISATION DE BASE DE DOCKER

LIGNE DE COMMANDE DOCKER

COMMANDES GÉNÉRALES

| Commande | Description |
|-----------------------|--|
| docker version | Affiche la version de Docker installée |
| docker info | Affiche des informations sur Docker |
| docker help | Liste les commandes disponibles |

COMMANDES LIÉES AUX IMAGES

| Commande | Description |
|---------------|--|
| docker pull | Télécharge une image depuis Docker Hub |
| docker build | Construit une image à partir d'un Dockerfile |
| docker images | Liste les images disponibles localement |
| docker rmi | Supprime une ou plusieurs images |

COMMANDES LIÉES AUX CONTAINERS

| Commande | Description |
|--------------|---|
| docker run | Crée et démarre un nouveau container |
| docker ps | Liste les containers en cours d'exécution |
| docker start | Démarre un container existant |
| docker stop | Arrête un container en cours d'exécution |
| docker rm | Supprime un ou plusieurs containers |
| docker logs | Affiche les logs d'un container |
| docker exec | Exécute une commande dans un container |

DOCKER COMPOSE

PRÉSENTATION

Docker Compose est un outil permettant de définir et gérer plusieurs **containers** au sein d'une même application à l'aide d'un fichier de configuration.

INSTALLATION

Pour installer **Docker Compose**, suivez les instructions de la [documentation officielle](#).

USAGE DE BASE

- docker-compose up : démarre l'ensemble des services définis dans le fichier docker-compose.yml
- docker-compose down : arrête et supprime les **containers, réseaux** et **volumes** définis dans le fichier docker-compose.yml

DOCKERFILES

INTRODUCTION AUX DOCKERFILES

DÉFINITION

Un **Dockerfile** est un fichier texte contenant les **instructions** pour créer une **image Docker**.

| Instruction | Description |
|-------------|--|
| FROM | Définit l'image de base sur laquelle on se base |
| RUN | Exécute une commande lors de la construction de l'image |
| CMD | Spécifie la commande par défaut à lancer lors du démarrage du conteneur |
| COPY / ADD | Copie des fichiers de l'hôte vers l'image |
| WORKDIR | Définit le répertoire de travail dans le conteneur |
| ENV | Définit les variables d'environnement |
| EXPOSE | Informe Docker que l'image accepte les connexions sur un port spécifique |

STRUCTURE

Un **Dockerfile** est composé de plusieurs **instructions** suivies de leurs **arguments**.

| Instruction | Description |
|-------------|---|
| FROM | Image de base à utiliser |
| RUN | Exécuter une commande |
| CMD | Commande à exécuter au démarrage du container |
| COPY | Copier des fichiers du hôte vers l'image |
| ADD | Ajouter des fichiers avec une source URL |

INSTRUCTIONS DE BASE

| Instruction | Description |
|----------------|---|
| FROM | Spécifie l'image de base à partir de laquelle construire |
| RUN | Exécute une commande et enregistre le résultat dans l'image |
| ADD/COPY | Copie des fichiers ou des répertoires de l'hôte vers l'image |
| CMD/ENTRYPOINT | Définit la commande à exécuter lors du démarrage du conteneur |
| ENV | Définit les variables d'environnement |
| EXPOSE | Indique les ports réseau sur lesquels le conteneur doit écouter |

FROM

La première instruction d'un **Dockerfile** est généralement FROM.

```
FROM ubuntu:18.04
```

RUN

RUN permet d'exécuter des **commandes** lors de la **construction** de l'image.

```
RUN apt-get update && apt-get install -y nginx
```



ADD / COPY

ADD et COPY copient des fichiers ou des répertoires depuis l'hôte vers l'image.

```
COPY index.html /var/www/html/
```

Note : ADD a une fonctionnalité supplémentaire d'extraire automatiquement les fichiers compressés tandis que COPY est préféré pour des usages simples. Expliquez aux stagiaires les différences et quand utiliser chaque commande.

CMD / ENTRYPOINT

CMD et ENTRYPOINT définissent la commande à exécuter lors du **démarrage du conteneur**.

```
CMD ["nginx", "-g", "daemon off;"]
```

ENV

ENV définit une **variable d'environnement**.

```
ENV DEBIAN_FRONTEND="noninteractive"
```

EXPOSE

EXPOSE indique les **ports réseau** sur lesquels le **conteneur** doit écouter.

EXPOSE 80

CRÉATION D'UN DOCKERFILE SIMPLE

Voici un exemple de Dockerfile pour un serveur **nginx** :

```
FROM ubuntu:18.04
ENV DEBIAN_FRONTEND="noninteractive"
RUN apt-get update && apt-get install -y nginx
COPY index.html /var/www/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

DOCKER HUB

PRÉSENTATION DU DOCKER HUB

Docker Hub est un service cloud permettant de partager et de gérer des **images Docker**. Il simplifie le workflow de développement en fournissant des fonctionnalités pour la **gestion centralisée** des images Docker.

DÉFINITION

Docker Hub est :

- Un registre public pour les **images Docker**
- Un service en ligne proposé par **Docker Inc.**
- Une plateforme pour partager et gérer les **images Docker**

FONCTIONNALITÉS

- **Hébergement gratuit** d'images publiques
- Hébergement **privé payant**
- Intégrations avec **GitHub** et **Bitbucket**
- **Automatisation des builds**

CRÉATION D'UN COMPTE

1. Aller sur le [site](#)
2. Cliquer sur '**Sign Up**'
3. Remplir les **informations** demandées
4. Valider l'**adresse email**

PUSH ET PULL D'IMAGES

PUBLIER UNE IMAGE

1. **Authentification**: docker login
2. **Taguer l'image**: docker tag my-image username/my-image:tag
3. **Pousser l'image**: docker push username/my-image:tag

RÉCUPÉRER UNE IMAGE

1. **Rechercher l'image** : docker search image-name
2. **Télécharger l'image** : docker pull image-name
3. **Exécuter un container à partir de l'image** : docker run -it image-name

MEILLEURES PRATIQUES

ASTUCES POUR OPTIMISER LES IMAGES

- Utiliser des **images de base légères** (ex: Alpine)
- **Chainer les commandes RUN** pour limiter le nombre de couches
- **Nettoyer les fichiers temporaires** après utilisation
- Eviter l'inclusion de **fichiers inutiles** (utiliser `.dockerignore`)

GESTION DES VOLUMES ET DES DONNÉES PERSISTANTES

- Distinguer les **données persistantes** et les **données temporaires**
- Utiliser les **volumes Docker** pour stocker les données persistantes
- Monter les volumes sur les **conteneurs** lors de leur création

SÉCURITÉ

CONTRÔLE DES ACCÈS

- Contrôler l'accès aux **conteneurs** et aux **images**
- Utiliser les **autorisations minimales** nécessaires

UTILISATION DE L'UTILISATEUR NON-ROOT

- **Ne pas** utiliser l'utilisateur **root** pour exécuter des applications à l'intérieur du conteneur
- Utiliser un utilisateur avec des **permissions limitées**

NETWORKING

RÉSEAU PAR DÉFAUT

- Les conteneurs sont connectés au **réseau par défaut** bridge
- Les conteneurs peuvent communiquer entre eux via leurs **adresses IP**

RÉSEAU PERSONNALISÉ

- **Créer** des réseaux **personnalisés** pour isoler les applications et améliorer la **sécurité**
- Utiliser des noms de **conteneurs** pour se connecter au lieu des **adresses IP**

EXEMPLES D'UTILISATION DE DOCKER

DÉPLOIEMENT D'UNE APPLICATION WEB

- Créer une **image** avec l'application et ses dépendances
- Utiliser **Docker Compose** pour gérer les services (frontend, backend, base de données)
- Exposer les **ports** nécessaires pour accéder à l'application
- Utiliser les **volumes** pour gérer les données persistantes

UTILISATION DE DOCKER POUR L'INTÉGRATION CONTINUE

- Créer des images pour les différentes étapes du pipeline (**build, test, déploiement**)
- Garantir la **consistance** des environnements d'exécution
- Faciliter la **collaboration** entre les développeurs et les opérations

ISOLATION D'ENVIRONNEMENTS DE DÉVELOPPEMENT

- Utiliser **Docker** pour créer des environnements de développement **isolés**
- Faciliter la reproduction des bugs en utilisant des **images spécifiques**
- Améliorer la **productivité** en permettant aux **développeurs** de se concentrer sur le code

DOCKER POUR DÉBUTANTS - EXEMPLES D'UTILISATION DE DOCKER

EXEMPLES D'UTILISATION DE DOCKER

EXEMPLE 1: DÉPLOIEMENT D'UNE APPLICATION WEB

1. Écrire un **Dockerfile** décrivant l'application
2. Construire une **image Docker** à partir du Dockerfile
3. Exécuter un **container** à partir de cette image
4. Exposer le **port** pour accéder à l'application web

EXEMPLE 2: UTILISATION DE DOCKER POUR L'INTÉGRATION CONTINUE

1. Automatiser le processus de build
2. Package l'application dans une **image Docker**
3. Exécuter des **tests** sur l'image
4. Pousser l'image sur un **registry** si les tests réussissent

EXEMPLE 3: ISOLATION D'ENVIRONNEMENTS DE DÉVELOPPEMENT

1. Créer un **Dockerfile** pour chaque environnement de développement
2. Construire des **images Docker** pour chaque environnement
3. Exécuter des **containers** pour chaque environnement
4. Permet aux développeurs de travailler indépendamment et dans des environnements **isolés**.

C'EST TOUT POUR CETTE
SECTION! VOUS ÊTES
MAINTENANT
FAMILIARISÉ AVEC
PLUSIEURS EXEMPLES
D'UTILISATION DE

