

L'outil Docker : principes, objectifs et solutions

Table des matières

I. Contexte	3
II. Introduction à Docker	3
A. Qu'est-ce que Docker ?	3
B. Pourquoi utiliser Docker ?	4
C. Exercice : Quiz	5
III. Docker en pratique	6
A. Création d'une image Docker	6
B. Gestion des conteneurs	8
C. Les réseaux	9
D. Les volumes	9
E. Exercice	10
IV. Docker Compose et Docker Swarm	10
A. Docker Compose	10
B. Docker Swarm	12
C. Exercice : Quiz	13
V. Essentiel	14
VI. Auto-évaluation	15
A. Exercice	15
B. Test	15
Solutions des exercices	17

I. Contexte

Durée (en minutes) : 60 minutes

Pré requis :

- Connaissance de base de la virtualisation
- Expérience de développement ou d'administration système souhaitable

Contexte

Dans ce cours, vous apprendrez les principes, les objectifs et les solutions de l'outil Docker. Vous découvrirez comment Docker peut vous aider à déployer et à gérer des applications dans des conteneurs logiciels, ce qui permet de minimiser les conflits et les dépendances, de rendre les applications plus portables et de faciliter la gestion et le déploiement des applications. Que vous soyez développeur, administrateur système ou responsable informatique, ce cours vous permettra d'acquérir les compétences nécessaires pour utiliser Docker dans vos projets et vos environnements de production. Vous apprendrez à créer des images Docker, à gérer des conteneurs, à utiliser des réseaux et des volumes, et à déployer des applications multiconteneurs à l'aide de Docker Compose et Docker Swarm.

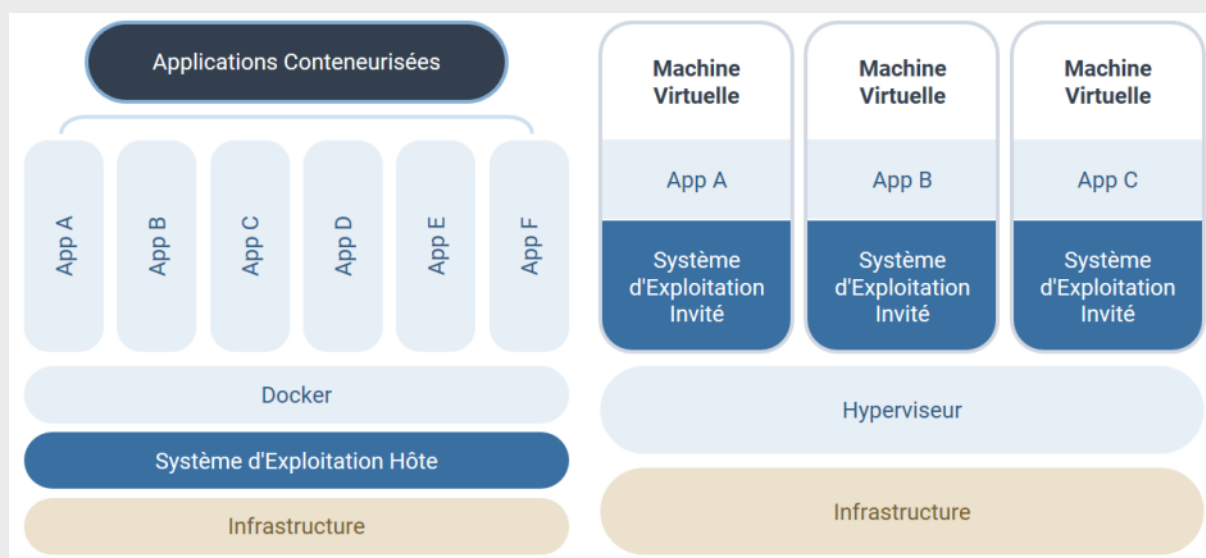
II. Introduction à Docker

A. Qu'est-ce que Docker ?

Docker est une plateforme open source qui automatise le déploiement des applications dans des conteneurs logiciels. Ces conteneurs sont isolés les uns des autres et peuvent s'exécuter sur n'importe quel serveur, ce qui facilite le déploiement et la gestion des applications.

Exemple

Mais qu'est-ce qu'un conteneur logiciel ? Eh bien, pensez à un conteneur physique, comme une boîte dans laquelle vous pouvez mettre différents objets. De la même manière, un conteneur logiciel est une boîte virtuelle dans laquelle vous pouvez mettre tous les éléments nécessaires à votre application, tels que le code, les bibliothèques, les dépendances, etc.



Fondamental

Docker n'est pas une virtualisation complète comme VirtualBox ou VMware, mais plutôt une virtualisation à l'échelle du système d'exploitation, ce qui le rend plus léger et plus rapide. En d'autres termes, vous pouvez exécuter plusieurs conteneurs Docker sur un même serveur, et chacun d'entre eux sera isolé des autres, ce qui garantit une meilleure sécurité et une plus grande flexibilité.

Exemple

Imaginez que vous développiez une application web qui nécessite une base de données spécifique et un serveur web. Au lieu d'installer ces logiciels directement sur votre système, vous pouvez utiliser Docker pour créer des conteneurs isolés pour chaque composant, ce qui facilite la gestion et le déploiement de votre application. Vous pouvez avoir un conteneur pour la base de données, un autre pour le serveur web, et éventuellement d'autres conteneurs pour d'autres services dont votre application a besoin. Cela facilite grandement la gestion et le déploiement de votre application, car vous pouvez déplacer les conteneurs d'un serveur à un autre sans avoir à vous soucier des dépendances ou des configurations spécifiques à chaque composant.

Définition Une image docker

Lorsque vous créez un conteneur, on parle de la création d'une image Docker. Une image Docker est un modèle de base pour créer et exécuter des conteneurs Docker. Elle contient tous les fichiers, les dépendances et la configuration nécessaires pour exécuter une application spécifique. Elle est créée à partir d'un Dockerfile, qui est un fichier texte contenant les instructions nécessaires pour construire l'image.

B. Pourquoi utiliser Docker ?

Fondamental

Docker offre plusieurs avantages pour les développeurs et les équipes informatiques, notamment :

- Isolation des applications
- Portabilité
- Optimisation des ressources
- Uniformisation de la stack technique : déploiement et scalabilité

L'isolation

L'un des principaux avantages de Docker est l'isolation des applications dans des conteneurs. Chaque application est exécutée dans son propre conteneur, ce qui garantit qu'elle est isolée des autres applications et de l'environnement sous-jacent. Cette isolation permet de minimiser les conflits et les dépendances entre les différentes applications, ce qui facilite le déploiement et la gestion des applications.

La portabilité

La portabilité est un autre avantage majeur de Docker. Les conteneurs Docker peuvent être créés une fois et exécutés sur n'importe quel système ou environnement qui prend en charge Docker. Cela signifie que vous pouvez développer et tester votre application localement, puis la déployer sur des serveurs de production ou des environnements de cloud computing sans avoir besoin de modifier le code ou de gérer les dépendances spécifiques à chaque environnement. Cela offre une flexibilité et une facilité de déploiement considérables pour les développeurs.

Optimisation des ressources

Docker permet de réduire la taille des images et des conteneurs par rapport aux machines virtuelles traditionnelles. Comme dit précédemment, les conteneurs Docker partagent le même noyau du système d'exploitation sous-jacent, ce qui signifie qu'ils n'ont pas besoin de démarrer un système d'exploitation complet pour chaque conteneur. Cela se traduit par des conteneurs plus légers et plus rapides à démarrer, ce qui est essentiel pour les déploiements rapides et l'évolutivité des applications.

Déploiement et scalabilité

Docker accélère également le déploiement et la mise à l'échelle des applications. Les conteneurs Docker peuvent être déployés en quelques secondes, ce qui permet une mise en production rapide des applications. De plus, Docker prend en charge l'orchestration des conteneurs, ce qui signifie que vous pouvez facilement mettre à l'échelle vos applications en ajoutant ou en supprimant des conteneurs en fonction de la demande. Cela facilite la gestion des charges de travail et permet une évolutivité horizontale efficace.

Utilisation et partage

En raison de tous ces avantages, Docker est largement utilisé dans les environnements de développement, de test et de production. Il est particulièrement utile pour les déploiements en microservices, où les applications sont décomposées en petits services indépendants, ainsi que pour les environnements de cloud computing, où l'évolutivité et la portabilité sont essentielles. De plus, Docker permet d'enregistrer vos conteneurs sur Docker Hub. Il s'agit d'une plateforme cloud qui permet aux utilisateurs de stocker, de partager et de distribuer des images Docker, ainsi que de gérer des build, des tests et des déploiements automatisés.

C. Exercice : Quiz

[solution n°1 p.19]

Question 1

Qu'est-ce que Docker ?

- ☐ Un langage de programmation
- ☐ Un système d'exploitation
- ☐ Une plateforme de virtualisation légère
- ☐ Un framework de développement

Question 2

Quel est le principal avantage de Docker ?

- ☐ Isoler les applications dans des conteneurs
- ☐ Développer des applications mobiles
- ☐ Stocker et transporter des données
- ☐ Communiquer dans les réseaux informatiques

Question 3

Quel élément est essentiel pour exécuter des conteneurs Docker ?

- ☐ Une machine virtuelle
- ☐ Un serveur web
- ☐ Un navigateur Internet
- ☐ Un système d'exploitation

Question 4

Quelle est la différence entre une image Docker et un conteneur Docker ?

- ☐ Une image est une version stable, tandis qu'un conteneur est en cours de développement
- ☐ Une image est un fichier exécutable, tandis qu'un conteneur est une instance en cours d'exécution
- ☐ Une image est un conteneur vide, tandis qu'un conteneur est une image préconfigurée
- ☐ Il n'y a pas de différence, les termes sont interchangeables

Question 5

Comment peut-on partager une image Docker avec d'autres utilisateurs ?

- ☐ En l'envoyant par e-mail
- ☐ En l'uploadant sur un serveur FTP
- ☐ En la publiant sur Docker Hub
- ☐ En la copiant sur une clé USB

III. Docker en pratique

A. Création d'une image Docker

Le dockerfile

Le Dockerfile est utilisé comme modèle pour générer une image Docker reproductible et portable. Concrètement, il spécifie l'image de base à utiliser, les commandes pour installer les dépendances, copier les fichiers de l'application et configurer l'environnement. Il permet aux développeurs de définir de manière déclarative les éléments nécessaires à l'exécution de leur application dans un conteneur. Le Dockerfile peut être versionné et partagé avec d'autres membres de l'équipe, facilitant ainsi la collaboration et la gestion des configurations.

Une fois que l'image est créée, elle peut être utilisée pour exécuter des conteneurs Docker. Chaque conteneur est une instance exécutable d'une image. Les conteneurs sont isolés les uns des autres et du système hôte, ce qui permet d'exécuter plusieurs instances d'une même image sur un même système. Les images Docker sont stockées dans des registres, tels que Docker Hub, qui permettent de partager et de distribuer les images. Les développeurs peuvent également créer et utiliser des registres privés pour stocker leurs propres images.

Attention

Docker peut être utilisé à l'aide de lignes de commande et d'une interface graphique (GUI) appelée Docker Desktop. Les commandes vous seront expliquées dans le cours, tandis que vous aurez l'occasion de voir Docker Desktop en action grâce aux screencasts intégrés au cours.

Méthode

Pour commencer avec Docker et créer votre première image docker, voici les étapes à suivre :

1. Installation de Docker :

- Rendez-vous sur le site officiel de Docker Desktop¹ et téléchargez Docker Desktop pour votre système d'exploitation.
- Suivez les instructions d'installation pour installer Docker sur votre ordinateur.

2. Vérification de l'installation :

Une fois l'installation terminée, ouvrez une fenêtre de terminal (par exemple, PowerShell sur Windows ou Terminal sur macOS/Linux).

Assurez-vous que Docker Desktop soit ouvert et que le Docker Engine soit en mode running (en bas à gauche de votre Docker Desktop).

Tapez la commande suivante pour vérifier que Docker est bien installé et fonctionne correctement :

```
1 docker version
```

Vous devriez voir les informations sur la version de Docker et du client Docker s'afficher.

3. Création d'un Dockerfile :

Dans un répertoire vide, créez un fichier nommé « *Dockerfile* » (sans extension).

Ouvrez le fichier Dockerfile dans un éditeur de texte et ajoutez les lignes suivantes :

```
1 FROM nginx:latest
2
3 COPY ./index.html /usr/share/nginx/html/index.html
```

Dans cet exemple, nous créons un conteneur basé sur l'image du serveur web Nginx et nous copions un fichier index.html dans le conteneur.

Cela veut dire 2 choses : il nous faut une image nginx et un fichier index.html ! Créons d'abord le fichier index.html (dans le même répertoire que le Dockerfile) :

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Webserver with Nginx</title>
6   </head>
7   <body>
8     <h2>Hello Docker World !</h2>
9   </body>
10 </html>
```

Pour obtenir l'image de nginx, utiliser la commande suivante :

```
1 docker pull nginx
```

4. Création de l'image du conteneur :

Dans le terminal, naviguez jusqu'au répertoire contenant le Dockerfile que vous avez créé.

Exécutez la commande suivante pour construire l'image du conteneur :

```
1 docker build -t test:1 .
```

Cette commande utilise le Dockerfile dans le répertoire actuel pour construire une image nommée "test" et le ":1" permet d'ajouter un tag. Le système de tag est pratique pour versionner vos images. Le point à la fin de la commande indique que le contexte de construction est le répertoire actuel.

¹ <https://www.docker.com/products/docker-desktop/>

5. Exécution du conteneur :

Une fois l'image du conteneur créée, vous pouvez l'exécuter avec la commande suivante :

```
1 docker run -p 8080:80 test:1
```

Cette commande exécute le conteneur à partir de l'image "test:1" et mappe le port 8080 de l'hôte sur le port 80 du conteneur.

6. Accès à l'application :

- Ouvrez un navigateur web et accédez à l'URL suivante : <http://localhost:8080>.
- Vous devriez voir le contenu de votre fichier index.html s'afficher, ce qui confirme que votre application fonctionne dans le conteneur.

B. Gestion des conteneurs

Une fois que vous avez créé une image Docker, vous pouvez l'utiliser pour créer des conteneurs. Les conteneurs sont des instances exécutables d'une image qui peuvent être démarrées, arrêtées, supprimées et gérées à l'aide de commandes Docker.

Méthode

Démarrer un conteneur tel que montré dans la partie précédente.

Accessoirement, pour arrêter un conteneur en cours d'exécution, vous pouvez utiliser la commande docker stop suivie de l'ID ou du nom du conteneur. Par exemple, la commande suivante arrête un conteneur avec l'ID "abcdef123456" :

```
1 docker stop abcdef123456
```

Suppression d'un conteneur : utiliser la commande docker rm.

```
1 docker rm abcdef123456
```

Pour afficher la liste des conteneurs en cours d'exécution sur votre système, vous pouvez utiliser la commande docker ps. Par défaut, cette commande affiche uniquement les conteneurs en cours d'exécution. Si vous souhaitez afficher tous les conteneurs (y compris ceux qui sont arrêtés), vous pouvez utiliser l'option -a. Par exemple, la commande suivante affiche tous les conteneurs sur votre système :

```
1 docker ps -a
```

Complément

Voici également quelques commandes qui vous seront utiles pour monitorer l'utilisation de vos conteneurs.

Vous pouvez spécifier les limites de ressources lors du démarrage d'un conteneur à l'aide des options -m (mémoire) et --cpus (CPU). Par exemple, la commande suivante limite un conteneur à 2 Go de mémoire et à 2 cœurs de CPU :

```
1 docker run -d -m 2g --cpus 2 mycontainer
```

La commande docker stats permet d'afficher les statistiques en temps réel des conteneurs en cours d'exécution, telles que l'utilisation de la CPU, de la mémoire et du réseau.

```
1 docker stats
```

Docker permet de récupérer les logs des conteneurs en utilisant la commande docker logs. Par exemple, la commande suivante affiche les logs du conteneur avec l'ID "abcdef123456" :

```
1 docker logs abcdef123456
```


Pour exécuter une commande à l'intérieur du conteneur avec l'ID "abcdef123456" :

```
1 docker exec abcdef123456 <commande>
```

C. Les réseaux

Docker permet de configurer et de gérer les réseaux pour les conteneurs. Par défaut, Docker crée un réseau appelé « *bridge* » qui permet aux conteneurs de communiquer entre eux et avec l'hôte. Cependant, vous pouvez également créer des réseaux personnalisés pour isoler les conteneurs et contrôler leur connectivité.

Méthode

Vous pouvez créer un réseau personnalisé en utilisant la commande `docker network create`. La commande suivante crée un réseau nommé « *mynetwork* » :

```
1 docker network create mynetwork
```

Vous pouvez spécifier le réseau auquel un conteneur doit être connecté lors de son démarrage en utilisant l'option `--network`.

```
1 docker run --network mynetwork mycontainer
```

Si vous avez plusieurs réseaux et que vous souhaitez permettre la communication entre des conteneurs sur différents réseaux, vous pouvez utiliser le mécanisme des alias de réseau. Vous pouvez attribuer des alias de réseau à un conteneur lors de son démarrage à l'aide de l'option `--network-alias`.

```
1 docker run
2 --network mynetwork --network-alias web mycontainer
```

Complément

Les alias de réseau dans Docker sont utilisés pour permettre la communication entre des conteneurs sur différents réseaux. Ils permettent de créer des noms alternatifs pour les adresses IP des conteneurs, facilitant ainsi la connectivité entre eux.

Lorsque vous connectez un conteneur à un réseau, Docker attribue automatiquement une adresse IP au conteneur dans ce réseau. Cependant, lorsque vous avez plusieurs réseaux et que vous souhaitez que des conteneurs sur des réseaux différents puissent communiquer entre eux, il peut être difficile de connaître les adresses IP attribuées.

C'est là que les alias de réseau entrent en jeu. Vous pouvez utiliser l'option `--network-alias` lors du démarrage d'un conteneur pour attribuer un alias de réseau à ce conteneur. Cet alias sera ensuite utilisé pour référencer le conteneur lors de la communication entre les conteneurs.

D. Les volumes

Définition

Les volumes Docker permettent de stocker et de partager des données entre les conteneurs et l'hôte. Les volumes sont des répertoires montés dans les conteneurs et peuvent être utilisés pour persister les données, partager des fichiers ou configurer des points de montage.

Méthode

Vous pouvez créer un volume en utilisant la commande `docker volume create`.

```
1 docker volume create myvolume
```

Pour monter un volume lors du démarrage d'un conteneur, utilisez l'option `-v`.

```
1 docker run -v myvolume:/data mycontainer
```

Ici, le volume est monté dans le répertoire /data du conteneur.

Si vous avez plusieurs conteneurs qui ont besoin d'accéder aux mêmes données, vous pouvez monter le même volume dans plusieurs conteneurs. Ainsi, les modifications apportées aux données dans un conteneur seront visibles par les autres conteneurs. Par exemple, la commande suivante monte le volume "sharedvolume" dans deux conteneurs :

```
1 docker run -v sharedvolume:/data mycontainer1
2 docker run -v sharedvolume:/data mycontainer2
```

E. Exercice

[solution n°2 p.20]

Cet exercice vise à établir des correspondances entre différents concepts et leurs applications spécifiques dans l'utilisation de Docker. Vous devez associer chaque concept de la colonne de gauche avec sa description ou son application la plus appropriée dans la colonne de droite.

Image de départ utilisée comme point de départ pour construire une nouvelle image.

Commandes spécifiées dans le Dockerfile pour construire l'image.

Répertoire contenant les fichiers nécessaires pour la construction de l'image.

Instruction pour exécuter des commandes lors de la construction de l'image.

Fichier de configuration pour décrire les étapes de construction d'une image Docker.

Commande utilisée pour construire une image Docker en utilisant un Dockerfile.

Instruction pour copier des fichiers du build context vers l'image.

Dockerfile

Base image

Instructions

Build context

COPY

RUN

docker build

IV. Docker Compose et Docker Swarm

A. Docker Compose

Docker Compose est un outil qui permet de définir et de gérer des applications multiconteneurs. Il utilise un fichier YAML pour décrire les services, les réseaux et les volumes nécessaires à votre application. Docker Compose simplifie le processus de démarrage et de gestion de plusieurs conteneurs en une seule commande.

Rappel

Un fichier YAML est un format de données lisible par l'homme qui est couramment utilisé pour représenter des structures de données de manière formalisée. YAML est souvent utilisé pour configurer des applications, définir des modèles de données ou échanger des informations entre différents systèmes.

Fichier de configuration

Vous utilisez un fichier YAML appelé docker-compose.yml pour définir votre application et ses services associés. Dans ce fichier, vous pouvez spécifier les conteneurs, les réseaux, les volumes, les variables d'environnement et d'autres paramètres nécessaires à votre application.

Services

Chaque service dans Docker Compose correspond à un conteneur. Vous pouvez définir les détails du conteneur, tels que l'image Docker à utiliser, les ports à exposer, les variables d'environnement, les volumes à monter, etc. Chaque service peut également dépendre d'autres services, ce qui permet de gérer facilement les relations entre les conteneurs.

Réseaux et volumes

Docker Compose permet de définir et de gérer les réseaux et les volumes nécessaires à votre application. Vous pouvez spécifier les réseaux personnalisés à utiliser pour les services, et définir les volumes nécessaires pour le stockage des données.

Exemple Un fichier docker-compose.yml

```

1 version: '3.8' # Version de Docker Compose à utiliser
2
3 services: # Définition des services (conteneurs)
4
5   web: # Nom du service
6     build: . # Chemin vers le Dockerfile pour construire l'image du conteneur
7     ports:
8       - 80:80 # Mappage du port 80 du conteneur sur le port 80 de l'hôte
9     volumes:
10      - ./app:/app # Montage du répertoire local './app' dans le répertoire '/app' du
      conteneur
11     depends_on:
12       - mysql
13     networks:
14       - default
15
16     environment:
17       - DEBUG=true # Définition de la variable d'environnement DEBUG=true
18
19   db: # Nom du service
20     image: mysql:5.7 # Utilisation d'une image Docker existante pour le conteneur
21     volumes:
22       - db_data:/var/lib/mysql
23     ports:
24       - "3306:3306"
25     networks:
26       - default
27     environment:
28       - MYSQL_ROOT_PASSWORD=secret # Définition de la variable d'environnement
      MYSQL_ROOT_PASSWORD avec la valeur 'secret'
29       - MYSQL_DATABASE=mydb # Définition de la variable d'environnement MYSQL_DATABASE avec
      la valeur 'mydb'
30
31     networks:
32       default:
33         external:
34           name: my-network

```

Ce fichier Docker Compose définit deux services (conteneurs) : "web" et "db". Le service "web" est construit à partir d'un Dockerfile local, expose le port 80 et monte le répertoire local "./app" dans le répertoire "/app" du conteneur. Le service "db" utilise l'image Docker "mysql:5.7" et définit des variables d'environnement pour configurer la base de données MySQL.

Les lignes de commandes

- `docker-compose up` : démarre les conteneurs définis dans le fichier `docker-compose.yml`. Il est possible d'ajouter l'option `-d` pour lancer les conteneurs en mode détaché (en processus d'arrière-plan). Notez que sur linux (depuis la V2.23), la ligne de commande est maintenant `docker compose` (pas de `up` ni de tiret entre `docker` et `compose`).
- `docker-compose down` : arrête et supprime les conteneurs, les réseaux et les volumes associés à votre application.
- `docker-compose start` et `docker-compose stop` : démarre et arrête les conteneurs sans les supprimer.
- `docker-compose restart` : redémarre les conteneurs de votre application.

Variables d'environnement

Docker Compose permet de définir des variables d'environnement pour vos services. Cela vous permet de configurer dynamiquement les paramètres de votre application, tels que les connexions à la base de données ou les clés d'API. Vous pouvez définir ces variables d'environnement dans le fichier `docker-compose.yml` ou dans un fichier `.env`.

Docker Compose simplifie considérablement le déploiement et la gestion d'applications multiconteneurs en automatisant de nombreuses tâches fastidieuses. Il vous permet de décrire votre architecture d'application de manière déclarative et de la gérer facilement à l'aide de simples commandes.

B. Docker Swarm

Docker Swarm est une fonctionnalité intégrée à Docker qui permet de créer et de gérer des clusters de conteneurs. Il permet de déployer et de gérer des applications sur un groupe de machines Docker, en fournissant une haute disponibilité, une scalabilité et une gestion des services améliorées. Docker Swarm permet également la gestion des secrets (informations confidentielles et/ou sensibles) entre conteneurs ainsi qu'un processus de développement continu appelé `rolling upgrade`.

Cluster de conteneurs

Docker Swarm permet de regrouper plusieurs machines (appelées des nœuds) pour former un cluster de conteneurs. Ces machines peuvent être physiques ou virtuelles, et elles exécutent le démon Docker pour gérer les conteneurs.

Définition

Le démon Docker, également connu sous le nom de Docker Engine, est le composant principal de Docker. C'est un processus qui s'exécute en arrière-plan sur une machine hôte et gère les opérations liées aux conteneurs Docker.

Tout est question d'équilibrage

Avec Docker Swarm, vous pouvez définir et gérer des services, qui sont des groupes de conteneurs qui effectuent une tâche spécifique. Chaque service peut être configuré avec un nombre spécifique de réplicas, ce qui permet d'équilibrer la charge et d'assurer une haute disponibilité.

Docker Swarm offre une fonctionnalité d'orchestration des conteneurs, ce qui signifie qu'il peut répartir automatiquement les conteneurs entre les nœuds du cluster en fonction de la charge et des ressources disponibles. Il garantit également que les services sont toujours en cours d'exécution, même en cas de défaillance d'un nœud.

Tout est fait pour équilibrer la charge de travail sur votre cluster de façon automatique. Les requêtes vont donc être redirigées entre les conteneurs d'un service pour assurer une distribution équitable de la charge, et ce sur tous les nœuds du cluster.

En résumé, Docker Swarm simplifie la gestion des applications déployées sur des clusters de conteneurs. Il offre une interface unifiée pour déployer, mettre à jour et gérer des services à grande échelle, tout en garantissant une haute disponibilité et une résilience accrue.

Conseil

Docker Compose et Docker Swarm sont des outils puissants pour gérer des applications conteneurisées, mais ils peuvent être complexes à utiliser. Il est fortement conseillé de se faire accompagner par quelqu'un de plus expérimenté lors des premiers déploiements.

C. Exercice : Quiz

[solution n°3 p.21]

Question 1

Qu'est-ce que Docker Compose ?

- ☐ Une plateforme de virtualisation légère
- ☐ Un outil de gestion de conteneurs Docker
- ☐ Un service de déploiement continu
- ☐ Une infrastructure en tant que service (IaaS)

Question 2

Quel est le rôle principal de Docker Compose ?

- ☐ Créer des images Docker
- ☐ Gérer des conteneurs individuellement
- ☐ Orchestrer et déployer des applications multiconteneurs
- ☐ Surveiller les performances des conteneurs

Question 3

Comment est défini un service dans un fichier de configuration Docker Compose ?

- ☐ Par le biais d'une instruction "service"
- ☐ En utilisant une balise <service></service>
- ☐ En spécifiant le nom du service en tant que clé
- ☐ En utilisant une annotation @service au-dessus du code

Question 4

Comment démarrer un ensemble de services Docker Compose ?

- ☐ Avec la commande "docker-compose up"
- ☐ En utilisant la commande "docker-compose start"
- ☐ En exécutant le script "start.sh" dans le répertoire du projet
- ☐ En utilisant l'interface graphique de Docker Desktop

Question 5

Quelle commande permet de lister les services en cours d'exécution avec Docker Compose ?

- ☐ docker-compose list
- ☐ docker-compose ps
- ☐ docker ps
- ☐ docker-compose services

Question 6

Comment arrêter et supprimer tous les conteneurs associés à un projet Docker Compose ?

- ☐ docker-compose delete
- ☐ docker-compose stop
- ☐ docker-compose down
- ☐ docker stop-all

Question 7

Quelle est la différence entre Docker et Docker Compose ?

- ☐ Docker est une plateforme de virtualisation légère, tandis que Docker Compose est un outil de gestion des conteneurs Docker
- ☐ Docker est utilisé pour gérer des conteneurs individuellement, tandis que Docker Compose est utilisé pour orchestrer et déployer des applications multiconteneurs
- ☐ Docker est utilisé pour le développement, tandis que Docker Compose est utilisé pour la production
- ☐ Docker et Docker Compose sont deux termes interchangeables, ils désignent la même chose

V. Essentiel

Dans ce cours, vous avez découvert les principes, les objectifs et les solutions de l'outil Docker. Vous avez appris comment Docker peut vous aider à déployer et à gérer des applications dans des conteneurs logiciels, ce qui permet de minimiser les conflits et les dépendances, de rendre les applications plus portables et de faciliter la gestion et le déploiement des applications. Vous êtes également en mesure de déployer et de gérer des applications multiconteneurs à l'aide de Docker Compose. Au final, cela vous permettra de gagner du temps, de minimiser les risques et d'améliorer la qualité de vos applications. Vous êtes également équipé pour continuer à apprendre et à explorer les fonctionnalités avancées de Docker et Docker Compose, telles que Docker Hub, Docker Trusted Registry et Docker Enterprise, qui peuvent vous aider à gérer et à déployer des applications à grande échelle dans des environnements de production plus complexes.

VI. Auto-évaluation

A. Exercice

Déploiement d'une application web avec Docker-compose

Vous avez été chargé de déployer une application web composée de deux services : un service frontend utilisant Nginx et un service backend utilisant Node.js. Vous devez utiliser Docker et Docker-compose pour faciliter le déploiement de l'application.

Question

[solution n°4 p.23]

1. Le service frontend doit utiliser une image Nginx officielle à partir de Docker Hub.
2. Le service backend doit utiliser une image Node.js officielle à partir de Docker Hub.
3. Les deux services doivent être connectés à un réseau Docker dénommé « *my-network* ».
4. Le service frontend doit être accessible depuis le port 80 de la machine hôte.
5. Le service backend doit utiliser un volume pour persister les données de la base de données.

Indice :

- Sélection des données à afficher (images, textes ou autre, etc.).
- Création du docker-compose.
- Vérifiez que le front-end fonctionne.
- Vérifiez que les données sont bien présentes dans le conteneur backend.

B. Test

Exercice 1 : Quiz

[solution n°5 p.23]

Question 1

Quelle commande permet de créer une nouvelle image Docker à partir d'un Dockerfile ?

- ☐ docker build
- ☐ docker create
- ☐ docker run
- ☐ docker compose

Question 2

Quelle commande permet de démarrer un conteneur Docker en spécifiant un nom personnalisé pour ce conteneur ?

- ☐ docker create --name <nom_conteneur> <image>
- ☐ docker start --name <nom_conteneur> <image>
- ☐ docker run --name <nom_conteneur> <image>
- ☐ docker launch --name <nom_conteneur> <image>

Question 3

Comment exécuter une commande à l'intérieur d'un conteneur Docker en cours d'exécution ?

- ☐ docker exec <nom_conteneur> <commande>
- ☐ docker run <nom_conteneur> <commande>
- ☐ docker start <nom_conteneur> <commande>
- ☐ docker exec <commande> <nom_conteneur>

Question 4

Comment lier un conteneur à un réseau spécifique dans Docker-compose ?

- ☐ En spécifiant le nom du réseau dans la section "networks" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker network connect" après avoir démarré le conteneur
- ☐ En spécifiant le nom du réseau dans la section "links" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker network create" avant de démarrer le conteneur

Question 5

Comment lier un volume à un conteneur dans Docker-compose ?

- ☐ En utilisant la commande "docker volume create" avant de démarrer le conteneur
- ☐ En spécifiant le nom du volume dans la section "volumes" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker volume attach" après avoir démarré le conteneur
- ☐ En spécifiant le nom du volume dans la section "links" du service dans le fichier Docker-compose

Question 6

Quelle est la commande pour supprimer un conteneur Docker en utilisant Docker-compose ?

- ☐ docker-compose delete <nom_conteneur>
- ☐ docker-compose stop <nom_conteneur>
- ☐ docker-compose down <nom_conteneur>
- ☐ docker-compose remove <nom_conteneur>

Question 7

Comment exposer un port spécifique d'un conteneur dans Docker-compose ?

- ☐ En utilisant la commande "docker expose" avant de démarrer le conteneur
- ☐ En spécifiant le port à exposer dans la section "ports" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker port" après avoir démarré le conteneur
- ☐ En spécifiant le port à exposer dans la section "links" du service dans le fichier Docker-compose

Question 8

Comment spécifier les variables d'environnement dans Docker-compose ?

- ☐ En utilisant l'instruction "env" dans le fichier Docker-compose
- ☐ En spécifiant les variables d'environnement dans la section "links" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker env" avant de démarrer le conteneur
- ☐ En spécifiant les variables d'environnement dans la section "environment" du service dans le fichier Docker-compose

Question 9

Comment démarrer tous les conteneurs associés à un projet Docker-compose ?

- ☐ docker-compose restart
- ☐ docker-compose up
- ☐ docker-compose down
- ☐ docker-compose stop

Question 10


Comment vérifier les journaux (logs) d'un conteneur Docker en utilisant Docker ?

- ☐ docker-compose logs <nom_conteneur>
- ☐ docker logs <nom_conteneur>
- ☐ docker-compose inspect <nom_conteneur>
- ☐ docker inspect <nom_conteneur>

Solutions des exercices


Exercice p. 5 Solution n°1**Question 1**

Qu'est-ce que Docker ?

- ☐ Un langage de programmation
- ☐ Un système d'exploitation
- ☒ Une plateforme de virtualisation légère
- ☐ Un framework de développement
-  Docker est une plateforme de virtualisation légère qui permet d'isoler des applications dans des conteneurs.


Question 2

Quel est le principal avantage de Docker ?

- ☒ Isoler les applications dans des conteneurs
- ☐ Développer des applications mobiles
- ☐ Stocker et transporter des données
- ☐ Communiquer dans les réseaux informatiques
-  Le principal avantage de Docker est de permettre l'isolation des applications dans des conteneurs. Cela signifie que chaque application est exécutée de manière indépendante, avec ses propres dépendances et configurations, sans interférer avec d'autres applications ou le système d'exploitation hôte.

Question 3

Quel élément est essentiel pour exécuter des conteneurs Docker ?

- ☐ Une machine virtuelle
- ☐ Un serveur web
- ☐ Un navigateur Internet
- ☒ Un système d'exploitation
-  Pour exécuter des conteneurs Docker, il est essentiel d'avoir un système d'exploitation compatible, tel que Linux, Windows ou MacOS. Docker utilise les fonctionnalités du système d'exploitation pour isoler les conteneurs et les exécuter de manière efficace.

Question 4

Quelle est la différence entre une image Docker et un conteneur Docker ?

- ☐ Une image est une version stable, tandis qu'un conteneur est en cours de développement
- ☒ Une image est un fichier exécutable, tandis qu'un conteneur est une instance en cours d'exécution
- ☐ Une image est un conteneur vide, tandis qu'un conteneur est une image préconfigurée
- ☐ Il n'y a pas de différence, les termes sont interchangeables

- Q Une image Docker est un fichier statique qui contient tous les éléments nécessaires pour exécuter une application, y compris le code, les dépendances et les configurations. Un conteneur Docker, quant à lui, est une instance en cours d'exécution d'une image. Plusieurs conteneurs peuvent être créés à partir de la même image.

Question 5

Comment peut-on partager une image Docker avec d'autres utilisateurs ?

- ☐ En l'envoyant par e-mail
- ☐ En l'uploadant sur un serveur FTP
- ☒ En la publiant sur Docker Hub
- ☐ En la copiant sur une clé USB

- Q Docker Hub est un registre centralisé où les utilisateurs peuvent publier et partager leurs images Docker. En publiant une image sur Docker Hub, d'autres utilisateurs peuvent y accéder et la télécharger pour l'utiliser dans leurs propres environnements Docker. Cela facilite la collaboration et le partage d'applications basées sur Docker.

Exercice p. 10 Solution n°2

Cet exercice vise à établir des correspondances entre différents concepts et leurs applications spécifiques dans l'utilisation de Docker. Vous devez associer chaque concept de la colonne de gauche avec sa description ou son application la plus appropriée dans la colonne de droite.

Dockerfile

Fichier de configuration pour décrire les étapes de construction d'une image Docker.

Base image

Image de départ utilisée comme point de départ pour construire une nouvelle image.

Instructions

Commandes spécifiées dans le Dockerfile pour construire l'image.

Build context

Répertoire contenant les fichiers nécessaires pour la construction de l'image.

COPY

Instruction pour copier des fichiers du build context vers l'image.

RUN

Instruction pour exécuter des commandes lors de la construction de l'image.

docker build

Commande utilisée pour construire une image Docker en utilisant un Dockerfile.

- Q **Dockerfile - Fichier de configuration pour décrire les étapes de construction d'une image Docker.**

Un Dockerfile est un fichier de configuration utilisé pour décrire les étapes nécessaires à la construction d'une image Docker. Il contient des instructions qui spécifient les dépendances, les fichiers sources, les commandes à exécuter, etc.

Base image - Image de départ utilisée comme point de départ pour construire une nouvelle image.

Une base image est une image existante utilisée comme point de départ pour construire une nouvelle image Docker. Elle peut contenir un système d'exploitation, des bibliothèques, des dépendances, etc., qui serviront de base pour l'image que vous créez.

Instructions - Commandes spécifiées dans le Dockerfile pour construire l'image.

Les instructions sont les commandes spécifiées dans le Dockerfile qui indiquent à Docker comment construire l'image. Par exemple, les instructions peuvent inclure la copie de fichiers, l'exécution de commandes, la configuration des variables d'environnement, etc.

Build context - Répertoire contenant les fichiers nécessaires pour la construction de l'image.

Le build context est le répertoire qui contient tous les fichiers nécessaires à la construction de l'image Docker. Lorsque vous exécutez la commande docker build, Docker utilise le contenu de ce répertoire pour construire l'image.

COPY - Instruction pour copier des fichiers du build context vers l'image.

L'instruction COPY dans le Dockerfile permet de copier des fichiers du build context (répertoire local) vers l'image Docker en cours de construction. Cela vous permet d'inclure des fichiers sources, des scripts ou tout autre élément nécessaire à votre application dans l'image.

RUN - Instruction pour exécuter des commandes lors de la construction de l'image.

L'instruction RUN dans le Dockerfile permet d'exécuter des commandes spécifiées lors de la construction de l'image. Par exemple, vous pouvez utiliser cette instruction pour installer des paquets, exécuter des scripts ou effectuer d'autres tâches nécessaires à la préparation de l'image.

docker build - Commande utilisée pour construire une image Docker en utilisant un Dockerfile.


La commande docker build est utilisée pour construire une image Docker en utilisant un Dockerfile. Cette commande est exécutée dans le terminal et spécifie le chemin vers le répertoire contenant le Dockerfile et les fichiers nécessaires à la construction de l'image.

Exercice p. 13 Solution n°3

Question 1

Qu'est-ce que Docker Compose ?


- ☐ Une plateforme de virtualisation légère
- ☒ Un outil de gestion de conteneurs Docker
- ☐ Un service de déploiement continu
- ☐ Une infrastructure en tant que service (IaaS)

 Docker Compose est un outil de gestion de conteneurs Docker qui permet de définir et d'orchestrer des applications multiconteneurs.

Question 2


Quel est le rôle principal de Docker Compose ?

- ☐ Créer des images Docker
- ☐ Gérer des conteneurs individuellement
- ☒ Orchestrer et déployer des applications multiconteneurs
- ☐ Surveiller les performances des conteneurs

 Le rôle principal de Docker Compose est d'orchestrer et de déployer des applications qui nécessitent plusieurs conteneurs Docker interagissant les uns avec les autres.


Question 3

Comment est défini un service dans un fichier de configuration Docker Compose ?

- ☐ Par le biais d'une instruction "service"
- ☐ En utilisant une balise <service></service>
- ☒ En spécifiant le nom du service en tant que clé
- ☐ En utilisant une annotation @service au-dessus du code
-  Dans un fichier de configuration Docker Compose, un service est défini en spécifiant le nom du service en tant que clé. Les autres propriétés, telles que l'image à utiliser, les ports exposés, etc., sont définies comme des sous-clés de ce dictionnaire.


Question 4

Comment démarrer un ensemble de services Docker Compose ?

- ☒ Avec la commande "docker-compose up"
- ☐ En utilisant la commande "docker-compose start"
- ☐ En exécutant le script "start.sh" dans le répertoire du projet
- ☐ En utilisant l'interface graphique de Docker Desktop
-  Pour démarrer un ensemble de services Docker Compose, vous pouvez utiliser la commande "docker up" ou "docker-compose up" dans le répertoire contenant le fichier de configuration Docker Compose.


Question 5

Quelle commande permet de lister les services en cours d'exécution avec Docker Compose ?

- ☐ docker-compose list
- ☒ docker-compose ps
- ☐ docker ps
- ☐ docker-compose services
-  La commande "docker-compose ps" permet de lister les services en cours d'exécution pour un projet Docker Compose spécifique.

Question 6

Comment arrêter et supprimer tous les conteneurs associés à un projet Docker Compose ?

- ☐ docker-compose delete
- ☐ docker-compose stop
- ☒ docker-compose down
- ☐ docker stop-all
-  La commande "docker-compose down" arrête et supprime tous les conteneurs associés à un projet Docker Compose spécifique.

Question 7

Quelle est la différence entre Docker et Docker Compose ?

- ☐ Docker est une plateforme de virtualisation légère, tandis que Docker Compose est un outil de gestion des conteneurs Docker

- ⦿ Docker est utilisé pour gérer des conteneurs individuellement, tandis que Docker Compose est utilisé pour orchestrer et déployer des applications multiconteneurs
- Docker est utilisé pour le développement, tandis que Docker Compose est utilisé pour la production
- Docker et Docker Compose sont deux termes interchangeables, ils désignent la même chose
- 🔍 Docker est un outil pour créer, gérer et exécuter des conteneurs individuels, tandis que Docker Compose est un outil spécifiquement conçu pour orchestrer et gérer des applications multiconteneurs, en utilisant un fichier de configuration pour définir la relation et la configuration entre les conteneurs.

p. 15 Solution n°4

Créez un fichier docker-compose.yml avec le contenu suivant :

```
1 version: '3'
2 services:
3   frontend:
4     image: nginx:latest
5     ports:
6       - 80:80
7     networks:
8       - my-network
9   backend:
10    image: node:latest
11    volumes:
12      - backend-data:/app/data
13    networks:
14      - my-network
15 networks:
16   my-network:
17 volumes:
18   backend-data:
```

Ouvrez un terminal et placez-vous dans le répertoire contenant le fichier docker-compose.yml.

Exécutez la commande `docker-compose up -d` pour démarrer les services en arrière-plan.

Vérifiez que les services sont en cours d'exécution en utilisant la commande `docker ps`.

Accédez à l'application frontend en ouvrant un navigateur et en visitant `http://localhost:80`.

Assurez-vous que le service frontend est accessible et fonctionne correctement.

Accédez au conteneur du service backend en utilisant la commande `docker exec -it <nom_du_conteneur_backend> sh`.


À l'intérieur du conteneur, naviguez vers le répertoire `/app/data` pour vérifier que les données de la base de données sont bien présentes.

Exercice p. 15 Solution n°5

Question 1

Quelle commande permet de créer une nouvelle image Docker à partir d'un Dockerfile ?


- ☒ docker build
- ☐ docker create
- ☐ docker run
- ☐ docker compose

 La commande "docker build" permet de créer une nouvelle image Docker en utilisant les instructions spécifiées dans un fichier Dockerfile.

Question 2

Quelle commande permet de démarrer un conteneur Docker en spécifiant un nom personnalisé pour ce conteneur ?


- ☐ docker create --name <nom_conteneur> <image>
- ☐ docker start --name <nom_conteneur> <image>
- ☒ docker run --name <nom_conteneur> <image>
- ☐ docker launch --name <nom_conteneur> <image>

 En utilisant la commande "docker run" avec l'option "--name", vous pouvez spécifier un nom personnalisé pour le conteneur Docker lors de son démarrage.

Question 3

Comment exécuter une commande à l'intérieur d'un conteneur Docker en cours d'exécution ?


- ☒ docker exec <nom_conteneur> <commande>
- ☐ docker run <nom_conteneur> <commande>
- ☐ docker start <nom_conteneur> <commande>
- ☐ docker exec <commande> <nom_conteneur>

 La commande "docker exec" permet d'exécuter une commande à l'intérieur d'un conteneur Docker en cours d'exécution. Vous devez spécifier le nom du conteneur et la commande à exécuter.

Question 4


Comment lier un conteneur à un réseau spécifique dans Docker-compose ?

- ☒ En spécifiant le nom du réseau dans la section "networks" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker network connect" après avoir démarré le conteneur
- ☐ En spécifiant le nom du réseau dans la section "links" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker network create" avant de démarrer le conteneur

 Dans le fichier Docker-compose, vous pouvez spécifier le nom du réseau dans la section "networks" du service auquel vous souhaitez lier le conteneur.


Question 5

Comment lier un volume à un conteneur dans Docker-compose ?

- ☐ En utilisant la commande "docker volume create" avant de démarrer le conteneur
- ☒ En spécifiant le nom du volume dans la section "volumes" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker volume attach" après avoir démarré le conteneur
- ☐ En spécifiant le nom du volume dans la section "links" du service dans le fichier Docker-compose
-  Dans le fichier Docker-compose, vous pouvez spécifier le nom du volume dans la section "volumes" du service auquel vous souhaitez lier le volume.


Question 6

Quelle est la commande pour supprimer un conteneur Docker en utilisant Docker-compose ?

- ☐ docker-compose delete <nom_conteneur>
- ☐ docker-compose stop <nom_conteneur>
- ☐ docker-compose down <nom_conteneur>
- ☒ docker-compose remove <nom_conteneur>
-  La commande "docker-compose remove" permet de supprimer un conteneur Docker spécifié dans le fichier Docker-compose et tous ses services associés.


Question 7

Comment exposer un port spécifique d'un conteneur dans Docker-compose ?

- ☐ En utilisant la commande "docker expose" avant de démarrer le conteneur
- ☒ En spécifiant le port à exposer dans la section "ports" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker port" après avoir démarré le conteneur
- ☐ En spécifiant le port à exposer dans la section "links" du service dans le fichier Docker-compose
-  Dans le fichier Docker-compose, vous pouvez spécifier le port à exposer dans la section "ports" du service auquel vous souhaitez exposer le port.


Question 8

Comment spécifier les variables d'environnement dans Docker-compose ?

- ☐ En utilisant l'instruction "env" dans le fichier Docker-compose
- ☐ En spécifiant les variables d'environnement dans la section "links" du service dans le fichier Docker-compose
- ☐ En utilisant la commande "docker env" avant de démarrer le conteneur
- ☒ En spécifiant les variables d'environnement dans la section "environment" du service dans le fichier Docker-compose
-  Dans le fichier Docker-compose, vous pouvez spécifier les variables d'environnement dans la section "environment" du service auquel vous souhaitez définir les variables.


Question 9

Comment démarrer tous les conteneurs associés à un projet Docker-compose ?

- ☐ docker-compose restart
- ☒ docker-compose up
- ☐ docker-compose down
- ☐ docker-compose stop
-  La commande "docker-compose up" permet de démarrer tous les conteneurs associés à un projet Docker-compose.

Question 10

Comment vérifier les journaux (logs) d'un conteneur Docker en utilisant Docker ?

- ☐ docker-compose logs <nom_conteneur>
- ☒ docker logs <nom_conteneur>
- ☐ docker-compose inspect <nom_conteneur>
- ☐ docker inspect <nom_conteneur>
-  La commande "docker logs" permet de vérifier les journaux (logs) d'un conteneur Docker spécifié. Vous devez spécifier le nom du conteneur pour lequel vous souhaitez afficher les journaux.