**Gérer ses images Docker avec Docker Compose**

* **Question 1**

**Docker-compose sert à :**

* + Orchestrer plusieurs conteneurs
  + Générer plusieurs fichiers Dockerfile
  + Installer la dernière version de Docker

*Docker-compose sert à lancer plusieurs conteneurs en parallèle ou l’un après l'autre, de manière automatique.*

* **Question 2**

**À quoi sert le paramètre --tail de la commande docker-compose logs --tail ?**

* + Il permet d’afficher les derniers logs du conteneur
  + Il permet d’afficher les logs de manière continue
  + Il permet de cacher les logs du conteneur

*--tail est une commande qui est beaucoup utilisée dans les librairies et logiciels et nous permet de récupérer les derniers logs du conteneur.  
Afin de suivre les logs de manière continue on utilise l'option -f*

* **Question 3**

**Quelle commande va nous permettre de faire tourner nos conteneurs en arrière-plan ?**

* + docker-compose exec -e
  + docker-compose ps --services
  + docker-compose up -d

*docker-compose up -d ou --detach est une option qui va faire tourner les conteneurs de façon détachée, en arrière-plan dans le terminal.*

*docker-compose exec va exécuter une commande à l’intérieur du conteneur.*

*docker compose ps va permettre de lister les conteneurs qui tournent sur la machine.*

* **Question 4**

**Quelle est la première ligne que l’on spécifie lorsque l’on écrit un fichier docker-compose.yml ?**

* + Les images du docker-compose
  + Les ports de notre conteneur
  + La version de docker-compose

*La première ligne à spécifier dans votre fichier docker-compose est la version du format de fichier.*

*Actuellement, il existe trois versions du format de fichier Compose :*

* + *Version 1, le format hérité non recommandé.*
  + *Version 2.x. Spécifiée avec l'entrée de version: '2' ou "version: '2.1', etc.*
  + *La version 3.x, la version la plus récente et recommandée. Spécifiée avec la version: '3' ou version: '3.1' , etc.*

*Les images et les ports de conteneurs sont spécifiés dans la section Service du docker-compose.*

* **Question 5**

**Dans le début de ce fichier docker-compose, on a deux chiffres pour les ports de notre application.**

**À quoi correspondent-ils ?**

version: "3.7"

services:

db:

image: postgres

ports:

- "8080:5432"

* + Au port externe et au port interne de notre service.
  + Au port de l’ensemble de nos conteneurs.
  + Au premier port de notre premier conteneur et au deuxième port de notre deuxième conteneur.

*Il est important de noter la distinction entre le port externe (port host) et le port interne (port conteneur).*

*Dans l'exemple ci-dessus, pour le service db, le port externe est 8080 et le port de conteneur interne est 5432.*

*Dans le conteneur web, votre chaîne de connexion à la db ressemblerait à des adresses “postgres://db:5432” et, depuis la machine hôte, la chaîne de connexion ressemblerait à postgres://{docker\_ip}: 8080.*

* **Question 6**

**Dans le docker-compose, on ne peut utiliser que des images en local :**

* + Vrai
  + Faux

*Dans le docker-compose, les images qui ne sont pas trouvées localement seront automatiquement téléchargées depuis le docker hub.*

* **Question 7**

**On a une image qui est enregistrée sur Docker Hub sous le nom dockerfacile/app. Cette image ne fonctionne qu’avec une base de données redis.**

**Écrivez un fichier docker-compose avec cette image et un service Redis qui aura pour base l’image de la base de données redis.**

**Quel est le premier log que notre image dockerfacile/app renvoie ?**

* + Docker-compose c’est pratique !
  + Les conteneurs tournent grâce à docker-compose !
  + Bienvenue dans ce tutoriel !

*Il faut écrire le fichier docker-compose avec deux services de cette façon et faire tourner notre docker-compose avec la commande docker-compose up.*

*Voici le déroulé des commandes à entrer dans le fichier docker-compose pour récupérer la solution :*

*version: "3"*

*services:*

*redis:*

*image: redis*

*app:*

*depends\_on:*

*- redis*

*image: dockerfacile/app*

*ports:*

*- 5000:5000*

* **Question 8**

**À partir de votre docker-compose précédemment écrit, ajoutez la variable d'environnement starwars=1 dans votre service app.**

**Quel est, maintenant, le premier log de notre service ?**

* + Je suis ton père.
  + Des machines qui créent des machines !
  + Que la force soit avec toi !

*Il est nécessaire de rajouter l’argument environment dans notre docker-compose.*

*Voici le déroulé des commandes à entrer dans le fichier docker-compose pour récupérer la solution :*

*version: "3"*

*services:*

*redis:*

*image: redis*

*app:*

*depends\_on:*

*- redis*

*image: tuto*

*ports:*

*- 5000:5000*

*environment:*

*- starwars=1*

* **Question 9**

**Voici un fichier dockerfile :**

FROM openjdk:11

CMD ["java", "-version"]

**Je souhaite obtenir ce résultat sur mon terminal :**

openjdk version "11.0.13" 2021-10-19

OpenJDK Runtime Environment 18.9 (build 11.0.13+8)

OpenJDK 64-Bit Server VM 18.9 (build 11.0.13+8, mixed mode, sharing)

**Quel(les) commande(s) Docker dois-je lancer ?**

*Attention, plusieurs réponses sont possibles.*

* + docker run -it  nom\_de\_l’image
  + docker exec nom\_de\_l’image
  + docker run nom\_de\_l’image

*Les deux commandes qui vont me permettre d'obtenir ce résultat sont :*

* + *docker run -it  qui va exécuter notre image en mode interactif, on aura donc la version qui va s’afficher.*
  + *docker run nom\_de\_l’image  qui permet de faire tourner notre conteneur à partir de l’image.*

*En revanche,  docker exec nom\_de\_l’image  permet d’envoyer une commande à un conteneur déjà lancé.*

* **Question 10**

**Je lance le conteneur d'une image absente localement mais existante sur le docker hub.**

**Quel message va s'afficher ?**

* + Unable to find image nom\_de\_l’image:latest' locally
  + Run failed : the image nom\_de\_limage wasn’t found locally
  + Error response from daemon: pull access denied for nom\_de\_l’image

*Afin d’obtenir la réponse à cette question, il suffit de taper dans le terminal la commande docker run avec le nom d’une image non disponible localement mais disponible sur docker hub.*

*Par exemple:*

*docker run node*