TD PART 01

* Setting up your computer

\_ Installation de Docker, puis on teste pour voir si c’est dans la machine

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Running your first container

\_ exécution du conteneur Alpine Linux (une distribution Linux légère)

La commande pull récupère l'image alpine du registre Docker et l'enregistre dans notre système.

On peut utiliser la commande docker images pour voir une liste de toutes les images de votre système

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Docker run

On va maintenant exécuter un conteneur Docker basé sur cette image.

Une image contenant texte, capture d’écran, menu, Police

Description générée automatiquement



\_ Pour éviter de sortir du shell avec :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

La commande docker ps nous montre tous les conteneurs en cours d'exécution.

docker ps -a

Ce que vous voyez ci-dessous est une liste de tous les conteneurs que vous avez exécutés. Notez que la colonne STATUS indique que ces conteneurs ont été fermés il y a quelques minutes

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour exécuter plusieurs commandes dans un conteneur. On peut taper :

docker run -it alpine /bin/sh

L'exécution de la commande run avec les indicateurs -it nous attache à un terminal interactif dans le conteneur.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Webapps with Dockers

Déployer des applications web avec Docker

* Run a static website in a container

L'image que vous allez utiliser est un site Web d'une seule page déjà créé pour cette démo et disponible sur le Docker Store sous le nom dockersamples/static-site.

On la télécharger et on l’ exécute directement en une seule fois à l'aide de Docker Run comme suit.



Étant donné que l'image n'existe pas sur notre hôte Docker, le démon Docker la récupère d'abord dans le registre, puis l'exécute en tant que conteneur.

On a stoppé le conteneur qui était actif et on lance la commande docker ps pour voir ce qui est resté en cours d’execution



On a stoppé puis supprimé le containeur avec les commandes suivantes ainsi que de son CONTAINER ID

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Maintenant, lançons un conteneur en mode détaché



Dans la commande ci-dessus :

-d créera un conteneur avec le processus détaché de notre terminal

-P publiera tous les ports de conteneur exposés sur des ports aléatoires sur l'hôte Docker

-e est la façon dont vous transmettez les variables d'environnement au conteneur.

--name vous permet de spécifier un nom de conteneur

AUTEUR est le nom de la variable d'environnement et Votre nom est la valeur que vous pouvez transmettre.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Avec cette commande on peut voir la liste des ports utilisés

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

On lance la commande en utilisant le port 80 pour pouvoir le site

* On va passer à la création de notre propre image

Mais on va d’abord stopper et supprimer le précédent conteneur

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Docker Images

On va créer notre propre image, et utiliser cette image pour exécuter une application localement et enfin, transférerez certaines de nos propres images vers Docker Cloud.

\_ Images locales présentes sur ma machine

Une image contenant texte, capture d’écran, Police, noir

Description générée automatiquement

* Create a Python Flask app that displays random cat pix.

On va commencez par créer un répertoire appelé flask-app où nous créerons les fichiers suivants : app.py

exigences.txt

modèles/index.html

Fichier Docker

* Création du repertoire flask-app

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* On passe à la création des fichiers

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Write a Dockerfile

Nous allons construire notre propre image Python basée sur Alpine. Nous ferons cela en utilisant un Dockerfile.

Un Dockerfile est un fichier texte qui contient une liste de commandes que le démon Docker appelle lors de la création d'une image. Le Dockerfile contient toutes les informations dont Docker a besoin pour exécuter l'application : une image Docker de base à partir de laquelle exécuter, l'emplacement du code de votre projet, ses dépendances et les commandes à exécuter au démarrage.

* Docker build

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Docker run

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

L’image est ouverte

Une image contenant capture d’écran, texte, Logiciel multimédia, logiciel

Description générée automatiquement