Rapport - Julian

TP 3 : Docker & VMs

Exercice 1

1)

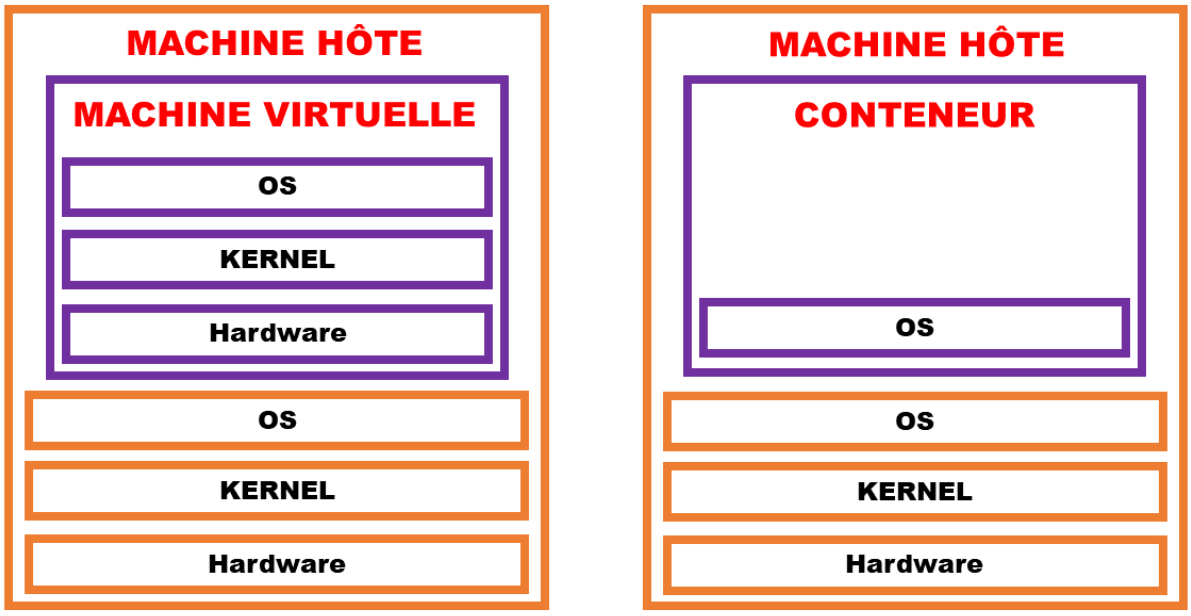
a)

Une machine virtuelle (ou VM) est une virtualisation ou émulation d'un appareil informatique créée par un logiciel d'émulation. Le logiciel d'émulation simule la présence de ressources matérielles et logicielles telles que la mémoire, le processeur, le disque dur, voire le système d'exploitation et les pilotes, permettant d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée.

b)

Les conteneurs fournissent une isolation légère en virtualisant uniquement les couches logicielles au-dessus du système d'exploitation hôte. Ils contiennent à la fois le code de l'application et son environnement (bibliothèques, outils système, etc.)
Tous les conteneurs partagent le même noyau d'OS hôte, ce qui les rend légers et efficaces. Ils peuvent s'exécuter sur n'importe quelle machine sans modification. Le démarrage et l'arrêt des conteneurs sont plus rapides que pour les VM.

c)



d)

Pour les VMs :

- **Avantages :**
    - Isolation complète : Chaque VM fonctionne de manière isolée, ce qui permet d'éviter les conflits entre applications et systèmes d'exploitation. C'est idéal pour tester des logiciels sans risquer d'endommager le système hôte.
    - Gestion des ressources : Il est possible d'allouer des ressources (CPU, mémoire, stockage) aux VM en fonction des besoins. Cela permet d'optimiser l'utilisation des ressources matérielles.
    - Sauvegardes et snapshots : Il est possible de créer des instantanés (snapshots) de l'état d'une VM à un moment donné. Cela facilite la restauration en cas de problème.
- **Inconvénients :**
    - Lourdeur : Il est nécessaire d'avoir un hyperviseur pour gérer les ressources, ce qui peut entraîner une surcharge. De plus, chaque VM inclut un système d'exploitation complet, ce qui augmente la consommation de ressources.
    - Temps de démarrage : Les VM prennent plus de temps à démarrer que les conteneurs. L'émulation matérielle ralentit le processus de démarrage.
    - Consommation de stockage : Chaque VM a besoin d'un espace de stockage dédié pour son système d'exploitation et ses application. Cela peut entraîner une utilisation inefficace de l'espace disque.

Pour les Conteneurs :

- **Avantages :**
    - Accélération du développement : Il est possible de travailler dans un environnement restreint, ce qui favorise la rapidité et l'efficacité. C'est une peu comme la création de bacs à sable pour tester des interactions.
    - Portabilité : Les conteneurs sont cohérents et peuvent s'exécuter sur différents environnements sans modification. Il y a aussi une facilité de déplacement, de copie et de redémarrage.
    - Impact réduit sur les performances : Les conteneurs libèrent rapidement les ressources inutilisées (mémoire, stockage).
- **Inconvénients:**
    - Moins isolés que les VM : Les conteneurs partagent le même noyau d'OS hôte, ce qui peut entraîner des problèmes de sécurité, contrairement au VMs qui offrent une isolation complète avec leurs propres systèmes d'exploitation.
    - Dépendance aux outils d'orchestration : Pour gérer efficacement les conteneurs à grande échelle, il est nécessaire d'avoir des outils d'orchestration tels que Docker. L'apprentissage et la configuration de ces outils peuvent être complexes.

i)

Les VM offrent une isolation complète et sont plus sécurisées, tandis que les conteneurs sont plus légers mais nécessitent une attention particulière pour renforcer leur sécurité.

ii)

Les conteneurs sont plus performants en termes de démarrage, d'utilisation des ressources et de légèreté comparé au VMs.

iii)

Les VM sont recommandées pour les charges de travail lourdes nécessitant une isolation complète, tandis que les conteneurs sont parfaits pour les déploiements légers, portables et rapides.

e)

Pour les VMs :

- Développement et test d'applications : Les VM permettent aux développeurs de créer des environnements isolés pour tester des applications sans affecter l'ordinateur hôte. Ils peuvent installer différents systèmes d'exploitation (Windows, Linux, etc.) sur des VM distinctes pour vérifier la compatibilité.
- Environnements de production : Les VM sont utilisées pour héberger des applications dans des environnements de production. Elles offrent une isolation complète, ce qui est essentiel pour les charges de travail critiques.
- Migration et consolidation : Les VM facilitent la migration d'applications entre différents serveurs physiques. Elles permettent également de consolider plusieurs serveurs virtuels sur un seul matériel physique, optimisant ainsi l'utilisation des ressources.

Pour les Conteneurs :

- Développement et Tests : Lors du développement d'une application, les conteneurs permettent aux développeurs de créer un environnement isolé pour tester leur code. Ils peuvent facilement reproduire l'environnement de production, garantissant ainsi que l'application fonctionne correctement avant le déploiement.
- Déploiement et Scalabilité : Les conteneurs sont idéaux pour le déploiement d'applications dans des environnements cloud ou sur site. Ils encapsulent l'application, ses bibliothèques et ses dépendances, ce qui les rend portables et faciles à déployer.
- Isolation et Sécurité : Les conteneurs isolent les processus et les ressources, ce qui réduit les risques de conflits entre applications. Chaque conteneur a son propre espace de noms, système de fichiers et variables d'environnement.

2)

a)

Faux car une VM à besoin d'émuler un OS, mais aussi le hardware.

b)

Faux car tous les conteneurs partagent le même noyaux.

c)

Vrai, c'est le but des conteneurs.

d)

Faux car il est possible de lancer plusieurs VMs sur une machine physique. C'est ce que fait certain type de serveur.

e)

Faux car les conteneurs utilisent le même noyaux que la machine hôte. Une VM sera plus adaptée.

f)

Faux, les VM permettent d'émuler n'importe quelle architecture quelque soit celle de la machine hôte.

Exercice 2

1)

La commande `qemu-system-x86_64` ouvre une nouvelle fenêtre et donne comme message d'erreur après un certain nombre d'opérations : `No bootable device`.
Cela signifie qu'il n'y a pas de dispositif sur lequel démarrer.

2)

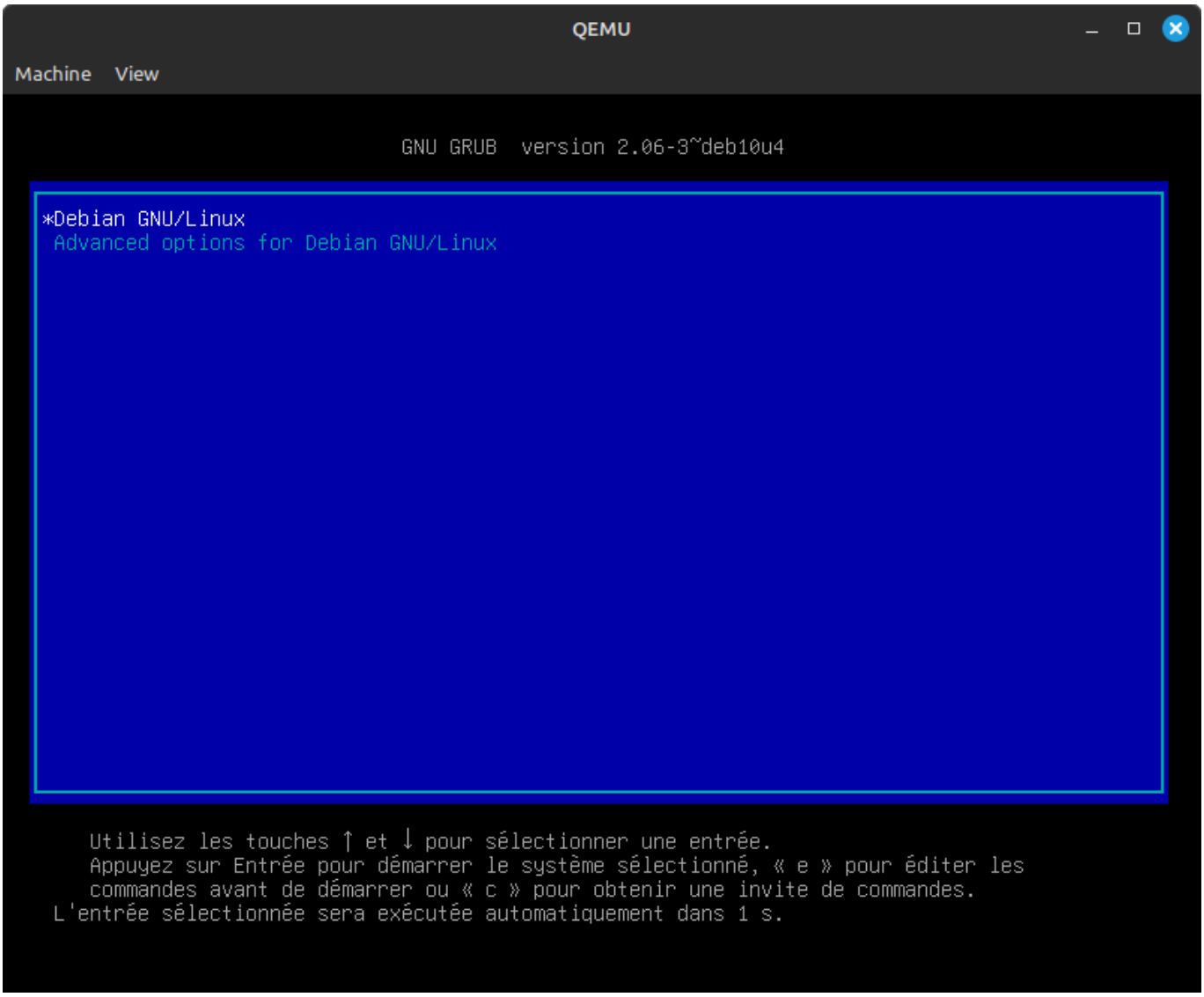
a)

Dans la commande `qemu-system-x86_64 -hda atoi.qcow2 -m 1024 -netdev user,id=eth0,hostfwd=tcp::10022-:22 -device e1000,netdev=eth0` :

- `-hda atoi.qcow2` : Spécifie le disque dur virtuel (ou image disque) à utiliser pour la machine virtuelle.
- `-m 1024` : Définit la quantité de mémoire RAM allouée à la machine virtuelle.
- `-netdev` : Configure un périphérique réseau virtuel.
  - `user` : Utilise un réseau virtuel de type “utilisateur” qui permet à la machine virtuelle d’accéder au réseau de l’hôte.
  - `id=eth0` : Attribue un identifiant au périphérique réseau virtuel, dans ce cas, “eth0”.
  - `hostfwd=tcp::10022-:22` : Configure une redirection de port.
- `-device` : Ajoute un contrôleur réseau virtuel à la machine virtuelle.
  - `e1000` : Utilise le modèle de contrôleur réseau virtuel Intel E1000.
  - `netdev=eth0` : Associe le contrôleur réseau virtuel au périphérique réseau virtuel “eth0”.

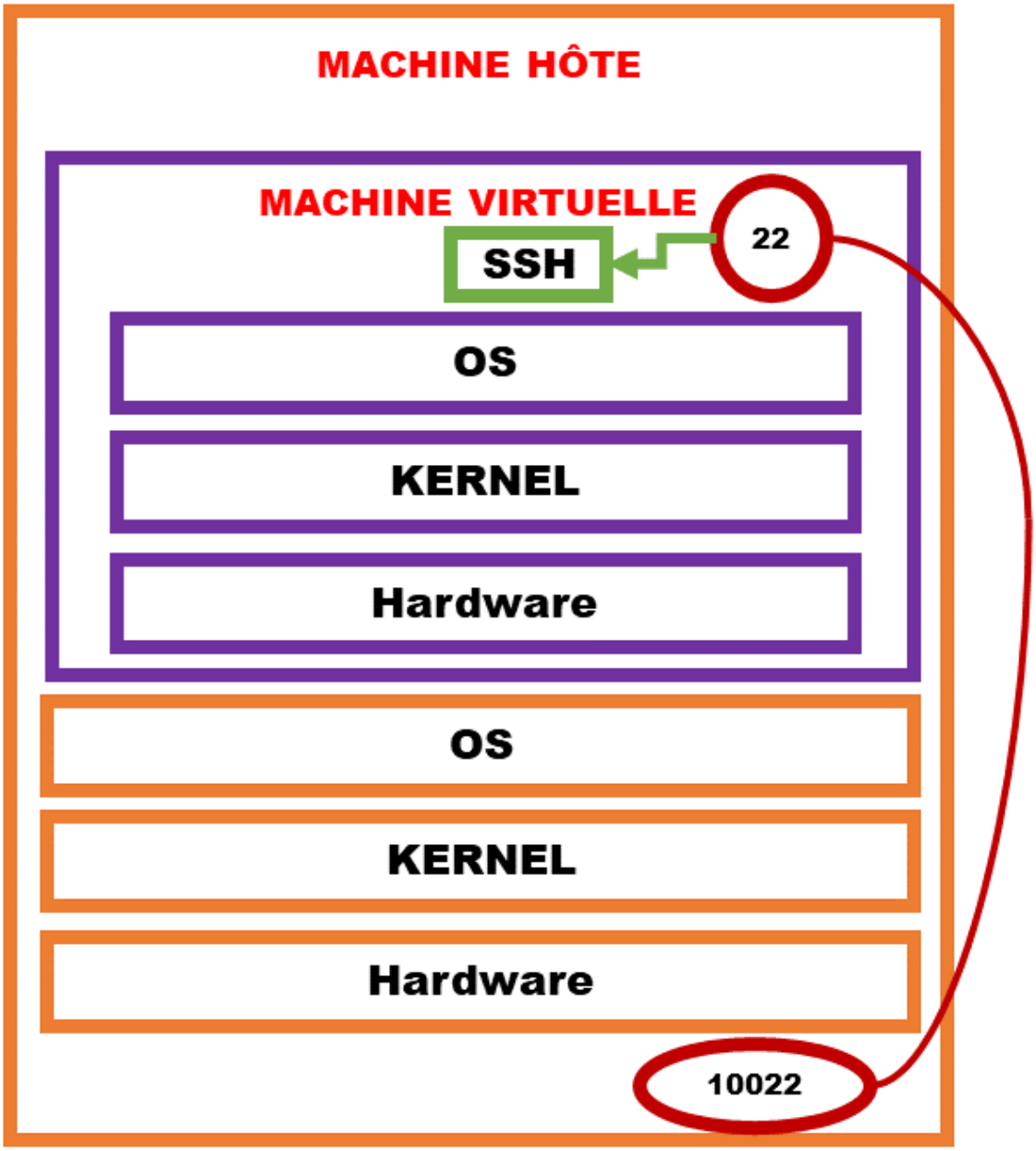
3)

Ce qui s'affiche est le GRUB pour lancer Debian :



4)

a)



b)

Sur la VM, le serveur SSH tourne sur le port 22.

5)

b)

On accède à la VM par le biais de SSH.

Exercice 3

1)

La commande `qemu-img create -f qcow2 mydebian.qcow2 5G` permet de créer une image vide nommé `mydebian.qcow2` de taille `5 Go`.

2)

a)

L'erreur est dû au fait que l'image est vide, il n'y a pas de système d'exploitation d'installé dessus.

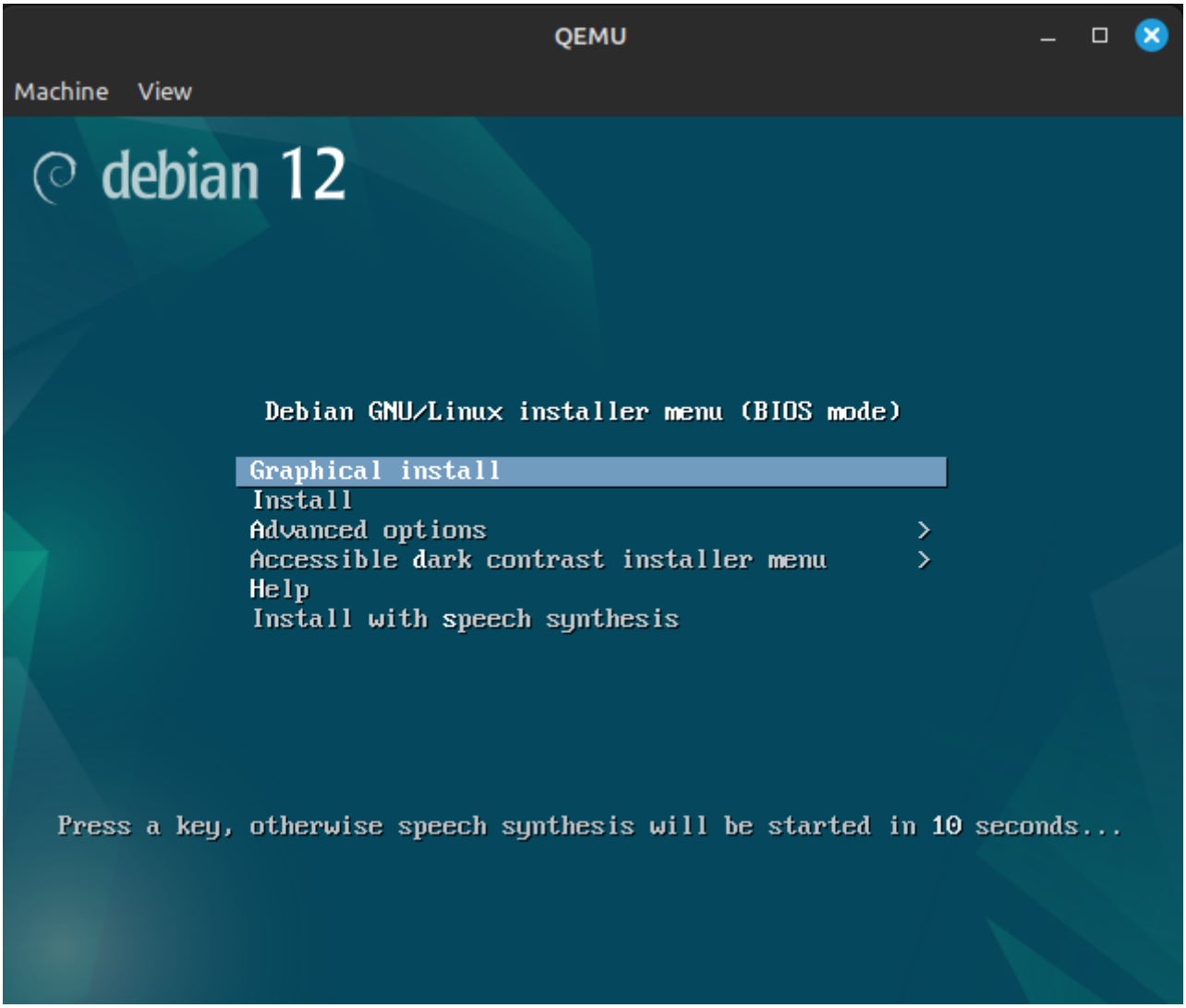
3)

Pour avoir la configuration réseau fonctionnel, il faut rajouter les paramètres suivant :
`-netdev user,id=eth0,hostfwd=tcp::10022-:22 -device e1000,netdev=eth0`

5)

Il faudra rajouter la commande `-cdrom <nom_de_l'image_ISO>` pour insérer un CD, et `-boot d` pour lancer le CD directement après le démarrage de la VM. Pour lancer la VM et installer le CD sera :
`qemu-system-x86_64 -m 1024 -hda ./mydebian.qcow2 -cdrom ./debian-12.5.0-amd64-netinst.iso -boot d`

6)



Exercice 4

1)

a)

Dans le conteneur, on est l'utilisateur `root`.

b)

Le système de fichier contient les éléments suivant :
`.dockerenv`, `bin`, `boot`, `dev`, `etc`, `home`, `lib`, `lib64`, `media`, `mnt`, `opt`, `proc`, `root`, `run`, `sbin`, `srv`, `sys`, `tmp`, `usr`, `var`

c)

Oui, il est très différent car ma machine contient beaucoup plus d'éléments.

2)

a)

Dans la commande `docker run -ti --rm rancher/cowsay cowsay "Hello Docker"` :

- `docker run` : Créer un conteneur Docker.
- `-ti` : Indique à Docker d'ouvrir un terminal interactif (TTY) dans le conteneur.
- `--rm` : Spécifie que le conteneur doit être supprimé automatiquement après son arrêt.
- `rancher/cowsay` : Nom de l'image Docker à utiliser pour créer le conteneur.
- `cowsay "Hello Docker"` : Commande à exécuter à l'intérieur du conteneur.

b)

D'après **2); a)**, `rancher/cowsay` est le nom de l'image à utiliser et lancer pour créer le conteneur.

c)

D'après **2); a)**, `cowsay "Salut Linux"` est la commande à exécuter à l'intérieur du Docker. Cette commande permet de faire dire à la vache qui s'affiche dans le terminal Docker `Salut Linux`.

d)

Le binaire `cowsay` de trouve dans le conteneur Docker qu'on a exécuté.

e)

En considérant que le conteneur n'est pas présent, Docker vérifie d'abord si l'image a déjà été téléchargé précédemment. Si ce n'est pas le cas, Docker télécharge l'image puis la lance dans un conteneur.

f)

La commande est plus rapide lors du second lancement car l'image à déjà été téléchargé pour la première fois. Cette image est stocké sur le PC pour éviter de la télécharger tout le temps.

Exercice 5

1)

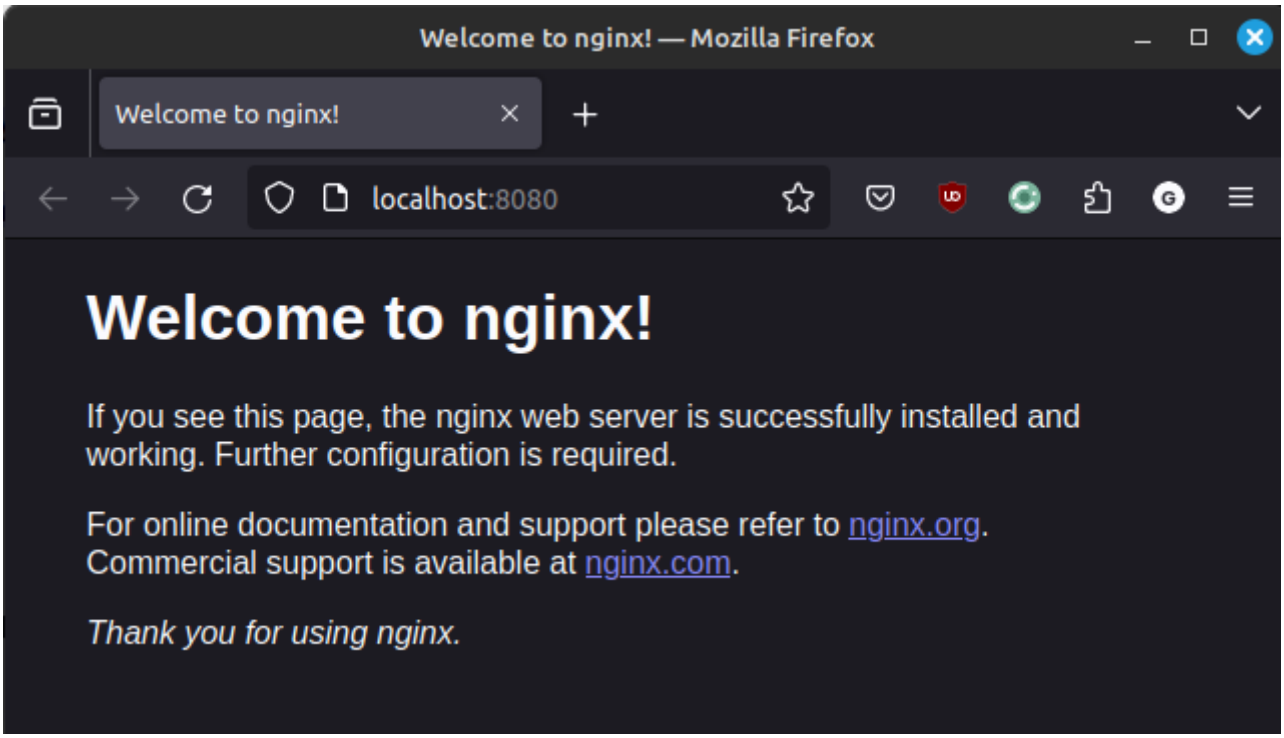
a)

Dans la commande `docker run -ti -p8080:80 nginx` :

- `docker run` : Créer un conteneur Docker.
- `-ti` : Indique à Docker d'ouvrir un terminal interactif (TTY) dans le conteneur.
- `-p8080:80` : Le port 80 du conteneur sera mappé vers le port 8080 de l'hôte Docker. En d'autres termes, les requêtes HTTP envoyées au port 8080 de l'hôte seront redirigées vers le port 80 du conteneur Nginx.
- `nginx` : Nom de l'image Docker à utiliser pour créer le conteneur.

b)

On voit ceci :



C'est la page d'accueil de nginx.

c)

Grâce à la commande `docker ps`, on peut voir que le nom du conteneur est `dreamy_greider`.
Voici le résultat que la console m'a retournée :

```
CONTAINER ID   IMAGE     COMMAND                  CREATED
STATUS              PORTS                                     NAMES
```

```
579fdabad382   nginx     "/docker-entrypoint.…"   About a minute ago   Up
About a minute   0.0.0.0:8080->80/tcp, :::8080->80/tcp   dreamy_greider
```

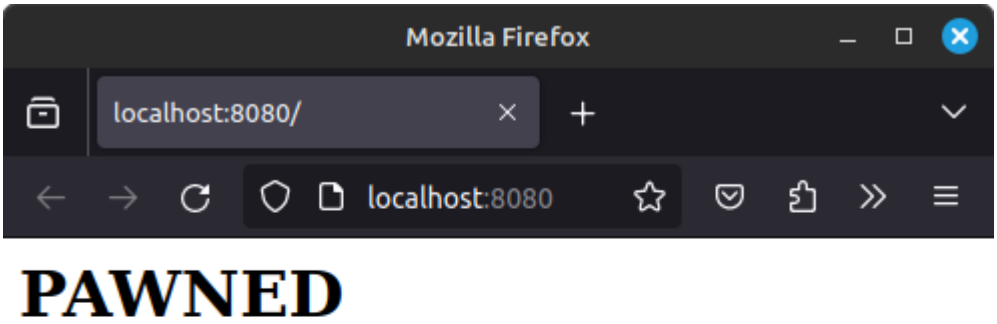
e)

D'après le fichier `default.conf`, on peut voir que le chemin de la racine du serveur WEB est `/usr/share/nginx/html`.

f)

i)

Le contenu de la page a bien changé. Voici ce qui s'affiche à présent sur `http://localhost:8080` :



PAWNED

2)

a)

La commande qui était à taper est la suivante :
`docker run -ti -p8080:80 -v ./ex5/html:/usr/share/nginx/html nginx`

b)

Le mot de passe de la page secrète est `geek123`.

Exercice 6

1)

a)

Pour lancer le serveur Redis, il faut taper la commande suivante :
`docker run -ti -p6379:6379 redis`

b)

Redis est un magasin de données clé-valeur (aussi appelé "dictionnaire" ou "table de hachage") à code source ouvert, en réseau, avec une durabilité optionnelle. Il est écrit en ANSI-C (C89).

d)

Après avoir effectué les commandes demandés, on obtient :

```
SET lol 123
+OK
GET lol
$3
123
```

2)

Le script Python se trouve dans `./ex6/main.py`.
L'output du programme Python sorti sur la console a été stocké dans le fichier `./ex6/occurrences.txt` grâce à la commande `python3 main.py > occurrences.txt`