Docker par la pratique

Choisir le type d'engine ci-dessous afin de passer aux étapes suivantes:

Installation de Docker

1. Téléchargez et installez Docker sur votre ordinateur.
    1. **Windows**
    2. **Linux**
    3. **MAC**
2. Vérifiez que Docker est installé correctement en exécutant la commande "docker version" dans une fenêtre de terminal.

Installation de Podman

1. Téléchargez et installez Docker sur votre ordinateur.
    1. **Windows**
    2. **Linux**
    3. **MAC**
2. Vérifiez que Podman est installé correctement en exécutant la commande "podman version" dans une fenêtre de terminal.
3. Si besoin faite un alias docker podman

Vérifier l'installation de docker et lancer son premier container

On télécharge l'image hello-world

```
docker pull hello-world
```

On exécute l'image

```
docker run hello-world
```

Lancer en tâche de fond et arrêter un container

1. Lancer le container

```
docker run --name nginx -d nginx
```

2. Voir le container en tache de fond

```
docker ps
```

Exemple :

```
docker ps
CONTAINER ID  IMAGE                             COMMAND
CREATED      STATUS      PORTS       NAMES
f5b4cd727d74  docker.io/library/nginx:latest  nginx -g daemon o...  3
hours ago  Up 3 hours              nginx
```

3. Arrêter le container, pour cela récupérer le container id de la commande ci-dessus

```
docker stop <container_id>
```

Exemple:

```
docker stop f5b4cd727d74
f5b4cd727d74
```

4. Vérifier que le container ne soit plus là:

```
docker ps
```

Exemple:

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

Gestion des images en local

1. Voir les images

```
docker images
```

Exemple:

```
docker images
REPOSITORY                    TAG         IMAGE ID      CREATED       SIZE
localhost/mon-image-docker    latest      0943488d4430  3 hours ago   253
MB
quay.io/podman/hello          latest      094d1dcf105c  19 hours ago  143
kB
docker.io/library/nginx       latest      9e7e7b26c784  3 days ago    139
MB
```

2. Supprimer une image avec la commande `docker rmi -f <image_id>` par exemple l'image **hello**

```
docker rmi -f 0943488d4430
```

3. Puller une image sur une registry dockerhub grâce à la commande `docker pull registry/image:version`

Rendez-vous sur le dockerhub en cliquant sur le lien ci-dessous afin de puller une image alpine en version **3.17.3**

Docker

```
docker pull alpine:3.17.3
```

Création d'une image Docker personnalisée et visualisation des couches

Création d'une image docker

1. Créez un fichier nommé "Dockerfile" dans un répertoire vide.
2. Écrivez les instructions suivantes dans le fichier Dockerfile :

```
FROM debian:latest
RUN apt-get update
RUN apt-get install -y apache2
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

3. Dans le même répertoire, exécutez la commande suivante pour créer une image Docker à partir du fichier Dockerfile :

```
docker build -t mon-image-docker .
```

Visualisation des couches

1. Installer **dive

2. Lancer la commande dive pour inspecter les couches *dive localhost/mon-image-docker* De combien de couche l'image est constituée ?

Lancement d'un conteneur à partir d'une image Docker

1. Exécutez la commande suivante pour lancer un conteneur à partir de l'image Docker que vous venez de créer :

```
docker run -d -p 8080:80 mon-image-docker
```

2. Ouvrez un navigateur web et allez sur l'adresse "**http://localhost:8080**" pour vérifier que le serveur Apache fonctionne correctement.
3. Voir les logs du container en ayant récupérer l'id du container

```
docker logs -f <container_id>
```

Exemple:

```
docker logs -f 4bde2525976a
AH00558: apache2: Could not reliably determine the server's fully
qualified domain name, using 10.88.0.14. Set the 'ServerName' directive
globally to suppress this message
```

Configuration d'un réseau pour les conteneurs Docker

1. Créez un réseau Docker avec la commande suivante :

```
docker network create mon-reseau-docker
```

2. Lancer un conteneur avec la commande suivante pour qu'il soit connecté au réseau créé :

```
docker run --name mon-conteneur-docker --network mon-reseau-docker -d mon-
image-docker
```

3. Exécutez la commande suivante pour vérifier que le conteneur est bien connecté au réseau créé :

```
docker network inspect mon-reseau-docker
```

Exemple:

```
❯ docker network inspect mon-reseau-docker
[
     {
          "name": "mon-reseau-docker",
          "id":
"dacbcfd91cebd10662f1375291f1c23b9471fb0304f724532eea3d14e8227a81",
          "driver": "bridge",
          "network_interface": "podman2",
          "created": "2023-04-15T12:45:33.739809257+02:00",
          "subnets": [
               {
                    "subnet": "10.89.0.0/24",
                    "gateway": "10.89.0.1"
               }
          ],
          "ipv6_enabled": false,
          "internal": false,
          "dns_enabled": true,
          "ipam_options": {
               "driver": "host-local"
          }
     }
]
```

4. On arrête le container

```
docker stop mon-conteneur-docker
```

5. On supprimer le réseau

```
docker network rm  -f mon-reseau-docke
```

Copie d'un fichier local dans un container

1. Puller l'image ci-dessous depuis la registry docker

```
docker pull rockylinux:9.1
```

2. Démarrer le container

```
docker run --name test  -id rockylinux:9.1
```

3. Créer un fichier comme ci-dessous

```
echo "hello world" > file.txt
```

4. Copie du fichier dans le container

```
docker cp $(pwd)/file.txt test:/tmp/
```

5. Vérifiez que le fichier a bien été copié dans le conteneur en utilisant une commande shell pour accéder au conteneur. La commande suivante vous permet d'ouvrir un shell à l'intérieur du conteneur :

```
docker exec -it  test /bin/bash
```

Vous devriez voir le fichier copier comme ci-dessous

```
docker exec -it  test /bin/bash
[root@8454771f216b /]# ls /tmp
file.txt
```

6. Lister le container en cours d'execution

```
docker ps | grep test
8454771f216b  docker.io/library/rockylinux:9.1     /bin/bash   2 days ago
Up 2 days
test
```

7. Suppression du container en récupérant l'id de la commande précédente

```
docker rm -f 8454771f216b
```

Docker persistance avec un volume

1. Créez un volume Docker nommé **mydata** en utilisant la commande `docker volume create`

```
docker volume create mydata
```

2. Lister les volumes `docker volume ls`

```
docker volume ls
DRIVER      VOLUME NAME

local       mydata
```

3. Inspecter le volume avec la commande `docker volume inspect mydata`

```
docker volume inspect mydata
[
     {
          "Name": "mydata",
          "Driver": "local",
          "Mountpoint":
"/var/lib/containers/storage/volumes/mydata/_data",
          "CreatedAt": "2023-04-15T13:01:06.312483116+02:00",
          "Labels": {},
          "Scope": "local",
          "Options": {},
          "MountCount": 0,
          "NeedsCopyUp": true,
          "NeedsChown": true
     }
]
```

4. Créez un conteneur Docker à partir de l'image **nginx** et montez le volume **mydata** dans le répertoire "/usr/share/nginx/html" du conteneur en utilisant l'option "-v".

```
docker run -d --name my-nginx-container -p 8080:80 -v
mydata:/usr/share/nginx/html nginx
```

5. Accéder à la page de votre site avec la commande `curl http://127.0.0.1:8080`

Exemple:

```
curl http://127.0.0.1:8080
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.23.4</center>
</body>
</html>
```

6. Ajoutez un fichier HTML personnalisé dans le répertoire "/usr/share/nginx/html" du conteneur à l'aide de la commande "docker cp".

```
echo '''
<html>
Hello World
</html>
''' > index.html
```

7. Copier le fichier dans le conteneur dans le repertoire `/usr/share/nginx/html/`

8. Accéder à la page de votre site avec la commande. Le serveur nginx va vous répondre Hello World

```
curl http://127.0.0.1:8080
<html>
Hello World
</html>
```

8. Arrêtez et supprimez le conteneur, puis créez un deuxième conteneur à partir de l'image "nginx" et montez le volume "mydata" dans le répertoire "/usr/share/nginx/html" du nouveau conteneur.

9. Vérifiez que le fichier HTML personnalisé que vous avez ajouté dans le premier conteneur est présent dans le répertoire "/usr/share/nginx/html" du deuxième conteneur.

10. Supprimez le volume "mydata" en utilisant la commande `docker volume rm` . Tous les conteneurs doivent être supprimés avant sinon il faut forcer la suppression

Docker persistance avec un bind mount

1. Créer le fichier comme ci-dessous

```
echo '''
<html>
Hello World 2
</html>
''' > index2.html
```

1. Lancer le conteneur présent mais cette fois-ci en utilisant le le bind mount afin de monter le repertoire courant dans le conteneur avec la commande **`docker run -d --name my-nginx-container -p 8080:80 -v ./:/usr/share/nginx/html nginx`**

2. Allez dans le container **/usr/share/nginx/html nginx** et verifier que le fichier index2.html est bien présent

3. Créer le fichier comme ci-dessous

```
echo '''
<html>
Hello World 3
</html>
''' > index3.html
```

4. Allez dans le container **/usr/share/nginx/html nginx** et verifier que le fichier index3.html est bien présent

5. Quel est pour vous la différence entre le bind mount et le volume ?

Docker-compose et scanning des images

Docker-compose

1. Installer docker-compose ou rendez-vous ici

```
sudo curl -L
"https://github.com/docker/compose/releases/download/v2.17.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/bin/docker-compose && sudo chmod
+x /usr/bin/docker-compose
```

2. Verifier que docker-compose via la commande : `docker-compose version`

```
docker-compose version
Docker Compose version 2.17.2
```

3. Installation de nextcloud à l'aide de docker compose.

Nextcloud est un logiciel libre de site d'hébergement de fichiers et une plateforme de collaboration.

4. Ecrire un `docker-compose.yml` permetant de démarrer nextcloud. Netxcloud se compose d'une base de donnée mysql et de l'application nextcloud.

   > Vous pouvez demander à ChatGPT de vous générer le docker-compose

5. Pour lancer Nextcloud avec Docker Compose, il suffit de lancer la commande suivante dans le même répertoire que le fichier `docker-compose.yml` :

```
docker-compose up -d
```

Cette commande construira les images de MySQL et de Nextcloud (si nécessaire) et démarrera les deux conteneurs. L'option `-d` permet de lancer les conteneurs en arrière-plan.

Après avoir lancé les conteneurs, vous pouvez accéder à Nextcloud en visitant `http://localhost:8080` dans votre navigateur. Vous serez invité à créer un compte administrateur et à configurer votre stockage de données. Lors de l'initialisation sélectionner **base de donnée mysql/mariadb** puis remplissez le formulaire avec les informations de connexion à la base de donnée présente dans le `docker-compose.yml`

6. Verifier que les conteneurs fonctionnent `docker-compose ps`

```
❯ docker-compose ps
NAME                IMAGE                                COMMAND
SERVICE             CREATED             STATUS              PORTS
files-app-1         docker.io/library/nextcloud:latest   "apache2-
foreground"     app                 17 hours ago        Up About a minute
80/tcp
files-db-1          docker.io/arm64v8/mysql:oracle       "--default-
authentic…"   db                  17 hours ago        Up About a minute
```

7. Verifier la communication entre le conteneur de la base de donnée et nextcloud

a. Se placer dans le conteneur de netxcloud `docker-compose exec <conteneur> bash` b. Utiliser la commande openssl pour tester la communication avec la base de donnée `openssl s_client -connect db:3306`

8. Arrêter tout et supprimer l'application et la base de donnée à l'aide des commandes `docker-compose down` `docker-compose rm`

Scanning des images netxcloud

1. Installer **trivy**

2. Lancer un scan d'image sur l'image netxcloud avec la commande `trivy image <image> --scanners vuln`

Combien y-a-t-il de vulnérabilités de type CRITICAL,HIGH,MEDIUM découvertes par trivy ? Qu'en pensez-vous ?