Lab3:Publication d’image

1. Créer une page d’accueil personnalisée

index.html :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>My First Container</title>
</head>
<body>
    <h1>Mon Conteneur Nginx est Actif !</h1>
    <p>
        Félicitations ! Ce contenu a été servi par votre image Docker personnalisée
        qui utilise le serveur web Nginx.
    </p>
    <p>
        <small>Ce fichier remplace la page d'accueil par défaut (index.html).</small>
    </p>
</body>
</html>
```

2. Lancer un conteneur Nginx

```
docker run -d --name nginx-custom -p 8080:80 nginx
```

3. Copier la page HTML dans le conteneur

```
docker cp index.html nginx-custom:/usr/share/nginx/html/index.html
```

Cela remplace la page d’accueil par défaut à l’intérieur du conteneur.

Vérifier ici:

http://localhost:8080/

4. Créer une image à partir du conteneur modifié

```
docker commit nginx-custom monuserdocker/nginx-custom:v1
```

⚠ Remplace monuserdocker par ton nom Docker Hub.

- docker commit: Crée une nouvelle image à partir des modifications apportées au système de fichiers du conteneur.
- nginx-custom: C'est le **nom** (ou l'**ID)** du conteneur Docker en cours d'exécution (ou récemment arrêté) dont tu veux sauvegarder l'état.
- monuserdocker: Ton nom d'utilisateur Docker Hub (ou le dépôt cible).

- **nginx-custom:v1** → Le nom du dépôt de l'image et le tag de l'image

5. Se connecter à Docker Hub

docker login

6. Pousser l'image vers Docker Hub

docker push monuserdocker/nginx-custom:v1

Pourquoi ne pas utiliser docker commit ?

docker commit :

❌ ne garde *aucune trace* de ce que on a modifié
❌ impossible à versionner
❌ crée souvent des images plus lourdes
❌ complique le travail en équipe
❌ rend les builds automatiques impossibles (CI/CD)

Autre solution?

1.Créer un Dockerfile

Dans le même dossier où se trouve index.html

```
FROM nginx:latest

COPY index.html /usr/share/nginx/html/index.html
```

2.Construire l'image

Dans ce même dossier (très important !) :

```
docker build -t nginx-custom .
```

Le . indique à Docker d'utiliser ce dossier comme contexte et donc de trouver le Dockerfile ici.

Le drapeau (-t pour *tag*) permet de donner un nom à l'image que vous créez. Ici, l'image sera nommée nginx-custom

3.Taguer l'image pour Docker Hub

docker tag nginx-custom monuserdocker/nginx-custom:latest

⚠ Remplace monuserdocker par ton nom Docker Hub.

4.Pousser l'image sur Docker Hub

docker push monuserdocker/nginx-custom:latest

Pourquoi un Dockerfile est beaucoup mieux ?

1. Reproductibilité

Un Dockerfile décrit exactement comment créer l'image :

```
COPY index.html /usr/share/nginx/html/index.html
```

N'importe qui peut reconstruire la même image avec :

```
docker build .
```

✔ Même résultat
✔ Sans manipulation manuelle
✔ Sans dépendre d'un conteneur existant

2. Historique clair (Git)

Tu peux versionner ton Dockerfile dans Git.
Tu sais quand, par qui, quoi a changé.

3. Automatisable (CI/CD)

GitHub Actions, GitLab CI, Jenkins… savent construire une image à partir d'un Dockerfile.

4. Images plus propres et plus légères

Le Dockerfile part d'une base claire (ex : `nginx:alpine`).
Pas d'artefacts inutiles dans l'image.

5. Maintenabilité

Dans 6 mois, tu peux revoir ton Dockerfile pour comprendre ce qu'il fait.
Avec `docker commit`, impossible de savoir ce qui existe dans l'image sans fouiller.

6. Collaboration facilitée

Toute l'équipe peut partager le même Dockerfile.

Conclusion

👉 **Le Dockerfile est la seule méthode fiable, propre et professionnelle pour créer une image.**
👉 `docker commit` dépanne, mais ne doit pas être utilisé en production.