

Docker : Guide Complet pour l'Installation, la Gestion des Conteneurs et le Déploiement



Présenté par : Fode Mangane

Table des matières

1. Introduction	1
2. Installation de Docker	1
3. Gestion des conteneurs.....	3
3.1. Lancement et gestion de base	3
3.2. Déploiement de Nginx et exposition des ports.....	4
4. Gestion des images.....	5
4.1. Téléchargement et inspection d'images	5
4.2. Création et sauvegarde d'images	6
4.3. Publication sur Docker Hub	7
5. Réseaux Docker	8
5.1. Types de réseaux et inspection	8
5.2. Communication entre conteneurs	10
5.3. Création de réseaux personnalisés	13
5.4. Utilisation du réseau hôte	15
6. Volumes et persistance des données.....	16
6.1. Montage de chemins d'hôte	16
6.2. Partage de données entre conteneurs.....	17
6.3. Création et utilisation de volumes Docker	18
7. Création d'images avec Dockerfile.....	20
7.1. Structure de base d'un Dockerfile.....	20
7.2. Utilisation des arguments et variables d'environnement	22
7.3. Configuration des utilisateurs et groupes.....	23
8. Docker Compose.....	24
8.1. Création d'applications multi-conteneurs	24
8.2. Gestion du cycle de vie des applications	26
9. Configuration avancée	28
9.1. Configuration du DNS	28
9.2. Configuration de la journalisation.....	29
9.3. Drivers de stockage.....	30
10. Multi-stage builds	30
10.1. Comparaison avec single-stage builds	30
10.2. Exemples pratiques	31
11. Conclusion	32

1. Introduction

Dans un monde numérique où la rapidité de déploiement et la flexibilité des infrastructures sont devenues des exigences incontournables, les technologies de conteneurisation se sont imposées comme des solutions de référence. Parmi elles, **Docker** est aujourd'hui l'outil le plus populaire et le plus utilisé pour simplifier la création, le déploiement et la gestion d'applications dans des environnements légers et isolés appelés **conteneurs**.

Docker permet aux développeurs et administrateurs systèmes de concevoir des applications portables, reproductibles et facilement déployables, quel que soit l'environnement sous-jacent. Grâce à cette technologie, il est désormais possible de standardiser le cycle de vie des applications, de la phase de développement jusqu'à la mise en production, en passant par les environnements de test et d'intégration continue.

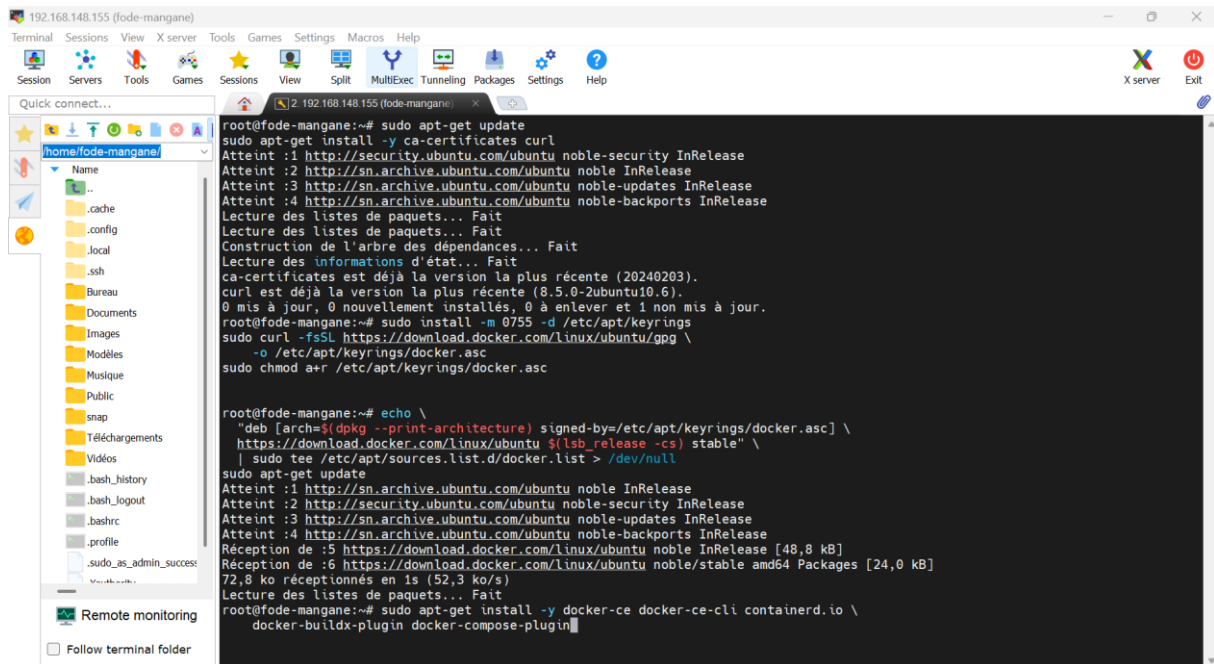
Ce guide complet a pour objectif de vous accompagner, étape par étape, dans la prise en main et la maîtrise de Docker. Il s'adresse aussi bien aux débutants qu'aux utilisateurs intermédiaires désireux de renforcer leurs compétences. À travers des explications détaillées et des manipulations concrètes, vous découvrirez comment :

- Installer et configurer Docker sur un système Ubuntu,
- Créer et gérer des conteneurs et des images Docker,
- Déployer des services comme **Nginx** et les rendre accessibles via des ports spécifiques,
- Organiser et gérer les réseaux Docker pour permettre la communication entre conteneurs,
- Mettre en place des volumes pour assurer la persistance des données,
- Utiliser **Dockerfile** pour automatiser la création d'images personnalisées,
- Et enfin orchestrer des applications multi-conteneurs à l'aide de **Docker Compose**.

À l'issue de ce guide, vous serez capable de concevoir, déployer et gérer efficacement des environnements conteneurisés, en exploitant à la fois les bases et des fonctionnalités avancées comme les multi-stage builds, la gestion des logs, la personnalisation réseau et la publication d'images Docker.

2. Installation de Docker

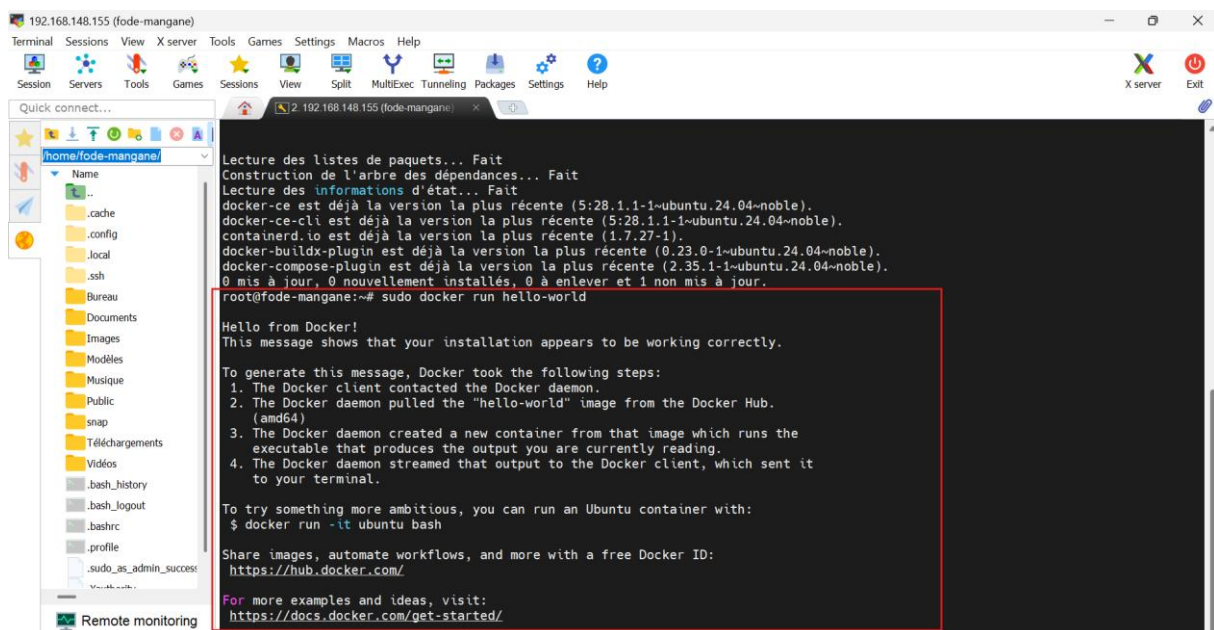
Pour installer Docker sur Ubuntu, nous nous référons à la documentation officielle disponible à l'adresse suivante : docs.docker.com/engine/install/ubuntu. Le processus d'installation se déroule en plusieurs étapes : mise à jour des packages APT, installation des prérequis, ajout de la clé GPG officielle de Docker, ajout du dépôt Docker, puis installation du package Docker Engine.



```
root@fode-mangane:~# sudo apt-get update
sudo apt-get install -y ca-certificates curl
Atteint :1 http://security.ubuntu.com/ubuntu noble-security InRelease
Atteint :2 http://sn.archive.ubuntu.com/ubuntu noble InRelease
Atteint :3 http://sn.archive.ubuntu.com/ubuntu noble-updates InRelease
Atteint :4 http://sn.archive.ubuntu.com/ubuntu noble-backports InRelease
Lecture des listes de paquets... Fait
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
ca-certificates est déjà la version la plus récente (20240203).
curl est déjà la version la plus récente (8.5.0-2ubuntu10.6).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 1 non mis à jour.
root@fode-mangane:~# sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
-o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

root@fode-mangane:~# echo \
'deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable' \
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
Atteint :1 http://sn.archive.ubuntu.com/ubuntu noble InRelease
Atteint :2 http://security.ubuntu.com/ubuntu noble-security InRelease
Atteint :3 http://sn.archive.ubuntu.com/ubuntu noble-updates InRelease
Atteint :4 http://sn.archive.ubuntu.com/ubuntu noble-backports InRelease
Réception de :5 https://download.docker.com/linux/ubuntu noble InRelease [48,8 kB]
Réception de :6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [24,0 kB]
72,8 ko réceptionnés en 1s (52,3 ko/s)
Lecture des listes de paquets... Fait
root@fode-mangane:~# sudo apt-get install -y docker-ce docker-ce-cli containerd.io \
docker-buildx-plugin docker-compose-plugin
```

Une fois l'installation terminée, nous vérifions le bon fonctionnement de Docker en exécutant la commande `docker run hello-world`. Cette commande permet de s'assurer que Docker est correctement installé et opérationnel.



```
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
docker-ce est déjà la version la plus récente (5:28.1.1-1ubuntu.24.04-noble).
docker-ce-cli est déjà la version la plus récente (5:28.1.1-1ubuntu.24.04-noble).
containerd.io est déjà la version la plus récente (1.7.27-1).
docker-buildx-plugin est déjà la version la plus récente (0.23.0-1ubuntu.24.04-noble).
docker-compose-plugin est déjà la version la plus récente (2.35.1-1ubuntu.24.04-noble).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 1 non mis à jour.
root@fode-mangane:~# sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Afin de pouvoir utiliser Docker sans avoir à saisir `sudo` à chaque commande, nous ajoutons l'utilisateur courant au groupe Docker en exécutant la commande suivante : `sudo usermod -aG docker $USER`.

```
root@fode-mangane:~# sudo usermod -aG docker $USER
root@fode-mangane:~#
```

Nous effectuons une déconnexion, puis nous nous reconnectons avant de vérifier la version de Docker.

```

root@fode-mangane:~# logout
fode-mangane@fode-mangane:~$ sudo -i
[sudo] Mot de passe de fode-mangane :
root@fode-mangane:~# docker --version
Docker version 28.1.1, build 4eba377
root@fode-mangane:~#

```

3. Gestion des conteneurs

3.1. Lancement et gestion de base

Nous allons maintenant lancer la création de notre premier conteneur, puis procéder à son démarrage.

```

root@fode-mangane:~# docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
0622fac788ed: Pull complete
Digest: sha256:6015f66923d7afbc53558d7ccffd325d43b4e249f41a6e93eef074c9505d2233
Status: Downloaded newer image for ubuntu:latest
root@4e53ebe6ed0a:/#

```

Nous allons maintenant afficher les conteneurs en cours d'exécution, puis lister tous les conteneurs, qu'ils soient actifs ou non.

```

root@4e53ebe6ed0a:/# exit
exit
root@fode-mangane:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
root@fode-mangane:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
4e53ebe6ed0a   ubuntu    "bash"    3 minutes ago   Exited (0) 44 seconds ago   heuristic_euclid
1c069eea1f5e   hello-world  "/hello"  3 minutes ago   Exited (0) 3 minutes ago   brave_napier
62cefe184b1a   hello-world  "/hello"  20 minutes ago   Exited (0) 20 minutes ago   upbeat_poincare
b222cfad2b80   hello-world  "/hello"  38 minutes ago   Exited (0) 38 minutes ago   blissful_brahmagupta
cdf7ccc31589   hello-world  "/hello"  40 minutes ago   Exited (0) 40 minutes ago   hopeful_jemison
root@fode-mangane:~#

```

Nous allons démarrer l'un des conteneurs inactifs (le premier de la liste), puis le stopper et le supprimer. En listant à nouveau les conteneurs, nous constatons qu'il n'apparaît plus.

```

root@fode-mangane:~# docker start 4e53ebe6ed0a
4e53ebe6ed0a
root@fode-mangane:~# docker stop 4e53ebe6ed0a
4e53ebe6ed0a
root@fode-mangane:~# docker rm 4e53ebe6ed0a
4e53ebe6ed0a
root@fode-mangane:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
1c069eea1f5e   hello-world  "/hello"  8 minutes ago   Exited (0) 8 minutes ago   brave_napier
62cefe184b1a   hello-world  "/hello"  25 minutes ago   Exited (0) 25 minutes ago   upbeat_poincare
b222cfad2b80   hello-world  "/hello"  44 minutes ago   Exited (0) 44 minutes ago   blissful_brahmagupta
cdf7ccc31589   hello-world  "/hello"  45 minutes ago   Exited (0) 45 minutes ago   hopeful_jemison
root@fode-mangane:~#

```

Nous allons maintenant supprimer les autres conteneurs hello-world.

```

root@fode-mangane:~# docker rm 62cefe184b1a
62cefe184b1a
root@fode-mangane:~# docker rm b222cfad2b80
b222cfad2b80
root@fode-mangane:~# docker rm cdf7ccc31589
cdf7ccc31589
root@fode-mangane:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
1c069eea1f5e   hello-world  "/hello"  14 minutes ago   Exited (0) 14 minutes ago   brave_napier
root@fode-mangane:~#

```

Nous allons maintenant créer un nouveau conteneur Ubuntu afin d'obtenir plus d'informations à son sujet. Pour ce faire, nous utiliserons la commande `docker inspect`, suivie de l'ID du conteneur.

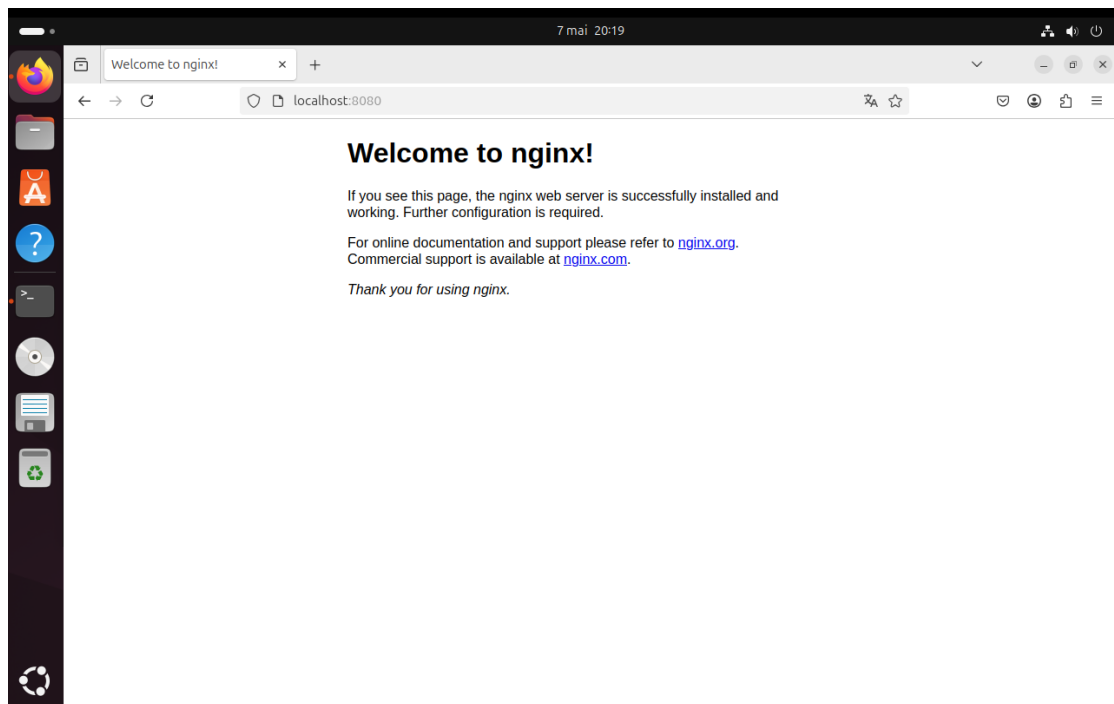
```
root@32891fbf9243:/# exit
exit
root@fode-mangane:~# docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS      PORTS          NAMES
32891fbf9243   ubuntu    "bash"                  57 seconds ago Exited (0) 4 seconds ago   nice_ganguly
1c069eeaf5e    hello-world "/hello"                16 minutes ago Exited (0) 16 minutes ago   brave_napier
root@fode-mangane:~# docker inspect 32891fbf9243
[
  {
    "Id": "32891fbf92431baebbae0e323cfeac018dd7ea5f56b013ba1fb4f916974013da",
    "Created": "2025-05-07T20:00:13.528592752Z",
    "Path": "bash",
    "Args": [],
    "State": {
      "Status": "exited",
      "Running": false,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 0,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2025-05-07T20:00:13.563927642Z",
      "FinishedAt": "2025-05-07T20:01:05.482137348Z"
    },
    "Image": "sha256:a0e45e2ce6e22e73185397d162a64fcf2f80a41c597015cab05d9a7b5913ce",
    "ResolvConfPath": "/var/lib/docker/containers/32891fbf92431baebbae0e323cfeac018dd7ea5f56b013ba1fb4f916974013da/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/32891fbf92431baebbae0e323cfeac018dd7ea5f56b013ba1fb4f916974013da/hostname",
    "HostsPath": "/var/lib/docker/containers/32891fbf92431baebbae0e323cfeac018dd7ea5f56b013ba1fb4f916974013da/hosts",
    "LogPath": "/var/lib/docker/containers/32891fbf92431baebbae0e323cfeac018dd7ea5f56b013ba1fb4f916974013da/32891fbf92431baebbae0e323cfeac018dd7ea5f56b013ba1fb4f916974013da-json.log",
    "Name": "/nice_ganguly",
    "RestartCount": 0,
    "Driver": "overlay2"
  }
]
```

3.2. Déploiement de Nginx et exposition des ports

Nous allons maintenant lancer un conteneur Nginx et configurer l'exposition de son port sur le 8080.

```
root@fode-mangane:~# docker run -d --name webserver -p 8080:80 nginx
2f8a10c3acebef20f5371084ab9368a0a1e1a6ce264330e6f3f9aa6d5ce90072
root@fode-mangane:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS      PORTS          NAMES
2f8a10c3aceb   nginx     "/docker-entrypoint..." 7 seconds ago Up 6 seconds  0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   webserver
root@fode-mangane:~#
```

Nous pouvons maintenant accéder au serveur via le navigateur en utilisant l'adresse <http://localhost:8080>.



4. Gestion des images

4.1. Téléchargement et inspection d'images

Il est possible de travailler avec des images Docker disponibles sur le répertoire public de Docker Hub. Nous allons procéder au téléchargement (pull) de l'image Alpine. Une fois l'image récupérée, nous pouvons vérifier sa présence en listant les images Docker et en précisant le nom ainsi que le tag.

```
root@fode-mangane:~# docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
f18232174bc9: Pull complete
Digest: sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef88c
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
root@fode-mangane:~# docker images alpine
REPOSITORY TAG IMAGE ID CREATED SIZE
alpine latest aded1e1a5b37 2 months ago 7.83MB
root@fode-mangane:~#
```

Nous constatons que les identifiants (ID) des images affichées sont tronqués. Pour les afficher en entier, il suffit d'ajouter l'argument `--no-trunc`. Il est également possible de filtrer les images par nom à l'aide de l'option `--filter`. Enfin, pour obtenir des informations détaillées sur l'image Alpine, nous pouvons utiliser la commande `docker inspect`.

```

root@fode-mangane:~# docker images --no-trunc
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest   sha256:a0e45e2ce6e6e22e73185397d162a64fcf2f80a41c597015cab05d9a7b5913ce  9 days ago    78.1MB
nginx         latest   sha256:a830707172e8069c09cf6c67a04e23e5a1a332d70a90a54999b76273a928b9ce  3 weeks ago    192MB
alpine        latest   sha256:aded1e1a5b3705116fa0a92ba074a5e0b0031647d9c315983ccba2ee5428ec8b  2 months ago   7.83MB
hello-world   latest   sha256:74cc54e27dc41bb10dc4b2226072d469509f2f22f1a3ce74f4a59661a1d44602  3 months ago   10.1kB
root@fode-mangane:~# docker images --filter "reference=alpine"
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
alpine        latest   aded1e1a5b37  2 months ago   7.83MB
root@fode-mangane:~# docker inspect alpine
[
  {
    "Id": "sha256:aded1e1a5b3705116fa0a92ba074a5e0b0031647d9c315983ccba2ee5428ec8b",
    "RepoTags": [
      "alpine:latest"
    ],
    "RepoDigests": [
      "alpine@sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef88c"
    ],
    "Parent": "",
    "Comment": "buildkit.dockerfile.v0",
    "Created": "2025-02-14T03:28:36Z",
    "DockerVersion": "",
    "Author": "",
    "Config": {
      "Hostname": "",
      "Domainname": "",
      "User": ""
    }
  }
]

```

4.2. Création et sauvegarde d'images

Il est possible de créer une image à partir d'un conteneur en réalisant un commit. Lors de cette opération, nous pouvons également attribuer un tag à l'image générée.

```

root@fode-mangane:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
2f8a10c3aceb   nginx     "/docker-entrypoint..." 19 minutes ago Up 19 minutes 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
32891fbf9243   ubuntu    "bash"                   37 minutes ago Exited (0) 36 minutes ago
1c069eeaf1f5e   hello-world "/hello"                 52 minutes ago Exited (0) 52 minutes ago
brave_napier
root@fode-mangane:~# docker commit 32891fbf9243 ubuntu:new
sha256:7d4b6231c190223c9edd73d7e2bc9e23e7bec689213d82aaa559f32301008446
root@fode-mangane:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        new       7d4b6231c190   About a minute ago 78.1MB
ubuntu        latest   a0e45e2ce6e6   9 days ago     78.1MB
nginx         latest   a830707172e8   3 weeks ago    192MB
alpine        latest   aded1e1a5b37   2 months ago   7.83MB
hello-world   latest   74cc54e27dc4   3 months ago   10.1kB
root@fode-mangane:~#

```

Il est possible d'exporter un conteneur sous forme de fichier tar afin de le conserver de manière persistante. Ces fichiers peuvent ensuite être listés à l'aide de la commande `ls -lrt`.

```

root@fode-mangane:~# docker export 32891fbf9243 > ubuntu.tar
root@fode-mangane:~# ls -lrt
total 78732
drwx----- 6 root root    4096 avril 16 23:59 snap
-rw-r--r-- 1 root root 80617472 mai 7 20:46 ubuntu.tar
root@fode-mangane:~#

```

Ce fichier tar peut ensuite être utilisé pour créer une nouvelle image Docker.

```

root@fode-mangane:~# docker import - myubuntu < ubuntu.tar
sha256:a34000e2c15cc5c38badf67bc87e24896a87928dc8f9bde4eed0320f13fdb65a
root@fode-mangane:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
myubuntu      latest   a34000e2c15c   9 seconds ago  78.1MB
ubuntu        new       7d4b6231c190   10 minutes ago 78.1MB
ubuntu        latest   a0e45e2ce6e6   9 days ago     78.1MB
nginx         latest   a830707172e8   3 weeks ago    192MB
alpine        latest   aded1e1a5b37   2 months ago   7.83MB
hello-world   latest   74cc54e27dc4   3 months ago   10.1kB
root@fode-mangane:~#

```

Nous allons à présent lister les images Nginx disponibles. Ensuite, nous sauvegarderons l'image présente dans un fichier au format tar, avant de vérifier sa création en la listant.


```

root@fode-mangane:~# docker images nginx
REPOSITORY TAG IMAGE ID CREATED SIZE
nginx latest a830707172e8 3 weeks ago 192MB
root@fode-mangane:~# docker save -o my_nginx.tar nginx
docker: 'docker save' requires at least 1 argument

Usage: docker save [OPTIONS] IMAGE [IMAGE...]

See 'docker save --help' for more information
root@fode-mangane:~# docker save -o my_nginx.tar nginx
root@fode-mangane:~# ls -lrt my_nginx.tar
-rw-r----- 1 root root 196647424 mai 7 20:53 my_nginx.tar
root@fode-mangane:~#

```

Nous allons maintenant supprimer l'image Nginx présente localement.

```

root@fode-mangane:~# docker rmi -f nginx
Untagged: nginx:latest
Untagged: nginx@sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66

```

```

root@fode-mangane:~# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
myubuntu latest a34000e2c15c 8 minutes ago 78.1MB
ubuntu new 7d4b6231c190 18 minutes ago 78.1MB
ubuntu latest a0e45e2ce6e6 9 days ago 78.1MB
<none> <none> a830707172e8 3 weeks ago 192MB
alpine latest aded1e1a5b37 2 months ago 7.83MB
hello-world latest 74cc54e27dc4 3 months ago 10.1kB
root@fode-mangane:~#

```

Ensuite, nous allons créer une nouvelle image à partir du fichier tar que nous avons précédemment généré.

```

root@fode-mangane:~# docker load < my_nginx.tar
Loaded image: nginx:latest
root@fode-mangane:~# docker images nginx
REPOSITORY TAG IMAGE ID CREATED SIZE
nginx latest a830707172e8 3 weeks ago 192MB
root@fode-mangane:~#

```

4.3. Publication sur Docker Hub

Après avoir créé un compte sur Docker Hub via <https://hub.docker.com/>, nous procédons à la connexion depuis le terminal.

```

root@fode-mangane:~# docker login
USING WEB-BASED LOGIN

Info -> To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: MWNW-TZEV
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser...

WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
root@fode-mangane:~#

```

Nous allons attribuer un tag à notre image en ajoutant notre nom d'utilisateur Docker Hub en préfixe. Une fois le tag appliqué, nous procédons à l'envoi de l'image sur Docker Hub à l'aide de la commande docker push.

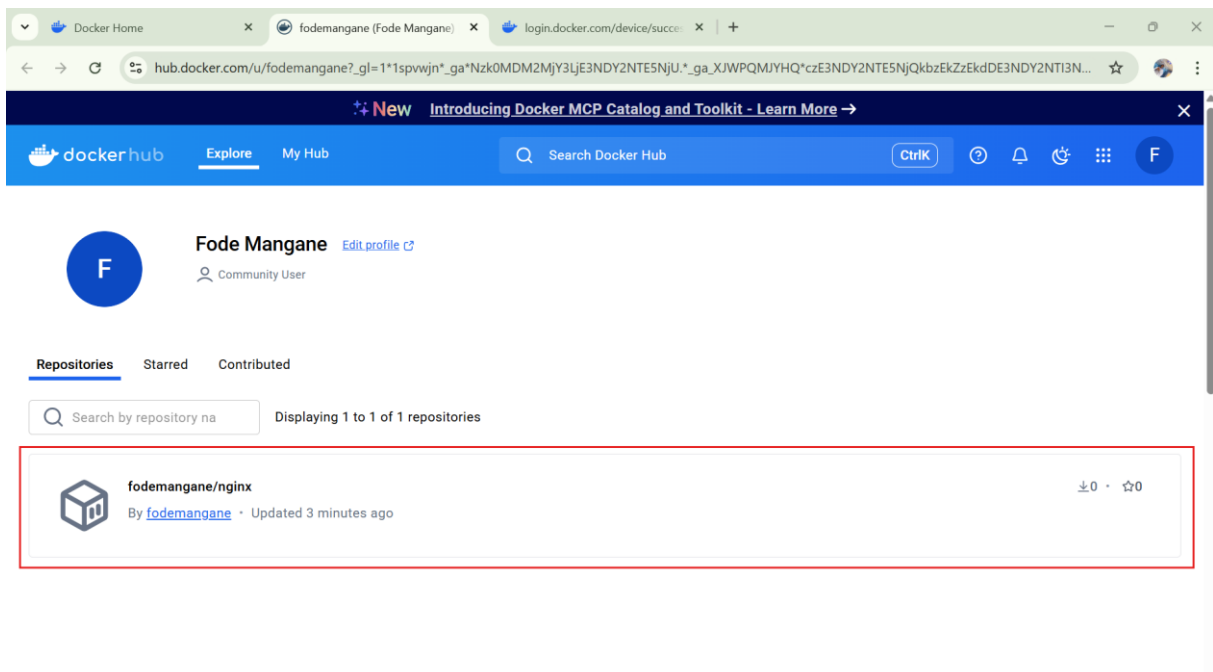
```

root@fode-mangane:~# docker tag nginx:latest fodemangane/nginx:latest
root@fode-mangane:~# docker imageq
docker: unknown command: docker imageq

Run 'docker --help' for more information
root@fode-mangane:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myubuntu            latest             a34000e2c15c       25 minutes ago    78.1MB
ubuntu              new                7d4b6231c190       35 minutes ago    78.1MB
ubuntu              latest             a0e45e2ce6e6       9 days ago        78.1MB
fodemangane/nginx   latest             a830707172e8       3 weeks ago       192MB
nginx               latest             a830707172e8       3 weeks ago       192MB
alpine              latest             aded1e1a5b37       2 months ago      7.83MB
hello-world         latest             74cc54e27dc4       3 months ago      10.1kB
root@fode-mangane:~# docker push fodemangane/nginx:latest
The push refers to repository [docker.io/fodemangane/nginx]
8030dd26ec5d: Mounted from library/nginx
d84233433437: Mounted from library/nginx
f8455d4eb3ff: Mounted from library/nginx
286733b13b0f: Mounted from library/nginx
46a24b5c31d8: Mounted from library/nginx
84accda66bf0: Mounted from library/nginx
6c4c763d22d0: Mounted from library/nginx
latest: digest: sha256:153605abbf752aaefa7f35306698bfaf53f9ec7e2149ef427538caff89e8ab45 size: 1778
root@fode-mangane:~#

```

Nous pouvons maintenant nous rendre sur le site de Docker Hub pour vérifier que notre image a bien été publiée.



5. Réseaux Docker

5.1. Types de réseaux et inspection

Nous allons maintenant nous intéresser à l'aspect réseau de Docker. Pour lister les différents types de réseaux, nous exécutons la commande `docker network ls`. Par défaut, le réseau bridge est appliqué.

```

root@fode-mangane:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ad07dfcd64a         bridge             bridge              local
f07ee5f1f92d        host               host                local
23dc677471fa        none              null                local
root@fode-mangane:~#

```

Nous allons maintenant inspecter le réseau de type bridge afin d'obtenir davantage d'informations à son sujet.

```

root@fode-mangane:~# docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "ad07dfcdd64a0b4d396acc3a410108202ebc7f70fcb3f28e4d7ddb56c239773c",
    "Created": "2025-05-07T20:08:55.743432451Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv4": true,
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "2f8a10c3acebef20f5371084ab9368a0a1e1a6ce264330e6f3f9aa6d5ce90072": {
        "Name": "webserver",
        "EndpointID": "b940f3712e58c262bc58f3ed9314490ea4c831ef233f9e692ebc7ac39e0f0892",
        "MacAddress": "0e:e2:52:79:19:fb",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      }
    }
  }
]

```

Il est possible de lister les différents plugins des pilotes réseau en exécutant la commande `docker info`.

```

root@fode-mangane:~# docker info
Client: Docker Engine - Community
Version: 28.1.1
Context: default
Debug Mode: false
Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version: v0.23.0
    Path: /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version: v2.35.1
    Path: /usr/libexec/docker/cli-plugins/docker-compose
Server:
  Containers: 3
    Running: 1
    Paused: 0
    Stopped: 2
  Images: 6
  Server Version: 28.1.1
  Storage Driver: overlay2
    Backing Filesystem: extfs
    Supports d_type: true
    Using metacopy: false
    Native Overlay Diff: true
    userxattr: false
  Logging Driver: json-file
  Cgroup Driver: systemd
  Cgroup Version: 2
  Plugins:
    Volume: local
    Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
  Swarm: inactive
  Runtimes: io.containerd.runc.v2 runc
  Default Runtime: runc

```

Nous installons le package `bridge-utils`, puis nous affichons les détails du réseau bridge ainsi que son adresse IP.

```

root@fode-mangane:~# apt install bridge-utils
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Paquets suggérés :
  ifupdown
Les NOUVEAUX paquets suivants seront installés :
  bridge-utils
0 mis à jour, 1 nouvellement installés, 0 à enlever et 1 non mis à jour.
Il est nécessaire de prendre 33,9 ko dans les archives.
Après cette opération, 118 ko d'espace disque supplémentaires seront utilisés.
Réception de :1 http://sn.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33,9 kB]
33,9 ko réceptionnés en 2s (19,4 ko/s)
Sélection du paquet bridge-utils précédemment désélectionné.
(Lecture de la base de données... 194622 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../bridge-utils_1.7.1-1ubuntu2_amd64.deb ...
Dépaquetage de bridge-utils (1.7.1-1ubuntu2) ...
Paramétrage de bridge-utils (1.7.1-1ubuntu2) ...
Traitement des actions différées (« triggers ») pour man-db (2.12.0-4build2) ...
root@fode-mangane:~#

```

Pour obtenir davantage d'informations sur les interfaces réseau, nous exécutons la commande `ip a`.

```

root@fode-mangane:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:f6:24:31 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.148.155/24 brd 192.168.148.255 scope global dynamic noprefixroute ens33
        valid_lft 1074sec preferred_lft 1074sec
    inet6 fe80::20c:29ff:fef6:2431/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 42:77:3f:6a:b2:09 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::4077:3fff:fe6a:b209/64 scope link
        valid_lft forever preferred_lft forever
root@fode-mangane:~#

```

5.2. Communication entre conteneurs

Nous démarrons un conteneur, puis nous effectuons à nouveau une inspection de notre réseau en mode pont.

```

root@fode-mangane:~# brctl show
bridge name      bridge id        STP enabled  interfaces
docker0          8000.4273f6ab209  no
root@fode-mangane:~#

```

Nous pouvons l'inspecter pour obtenir davantage d'informations à son sujet.

```

root@fode-mangane:~# docker inspect bridge
[
  {
    "Name": "bridge",
    "Id": "ad07dfcdd64a0b4d396acc3a410108202ebc7f70fcb3f28e4d7ddb56c239773c",
    "Created": "2025-05-07T20:08:55.743432451Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv4": true,
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
    }
  }
]

```

Nous effectuons un ping depuis la machine hôte pour tester l'accessibilité de notre réseau en mode bridge.

```

    "Config": [
      {
        "Subnet": "172.17.0.0/16",
        "Gateway": "172.17.0.1"
      }
    ],
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
  }
]
root@fode-mangane:~# ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1) 56(84) bytes of data.
64 bytes from 172.17.0.1: icmp_seq=1 ttl=64 time=0.047 ms
64 bytes from 172.17.0.1: icmp_seq=2 ttl=64 time=0.091 ms
64 bytes from 172.17.0.1: icmp_seq=3 ttl=64 time=0.094 ms

```

Nous ajoutons un second conteneur que nous connectons à notre réseau en mode bridge. Pour vérifier l'accès à Internet depuis ce conteneur, nous mettons à jour ses paquets, et la commande s'exécute avec succès.

```

root@fode-mangane:~# apt-get update
Atteint :1 http://sn.archive.ubuntu.com/ubuntu noble InRelease
Réception de :2 http://sn.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Réception de :3 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Réception de :4 http://sn.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Atteint :5 https://download.docker.com/linux/ubuntu noble InRelease
Réception de :6 http://sn.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1 066 kB]
Réception de :7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21,5 kB]
Réception de :8 http://sn.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [229 kB]
Réception de :9 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [208 B]
Réception de :10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52,2 kB]
Réception de :11 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Réception de :12 http://sn.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161 kB]
Réception de :13 http://sn.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]

```

Nous allons d'abord effectuer un ping pour vérifier la communication avec l'autre conteneur présent sur le même réseau. (Note : s'assurer que les deux conteneurs sont en cours d'exécution).

```
root@fode-mangane:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2f8a10c3aceb	nginx	"/docker-entrypoint..."	3 hours ago	Up About a minute	0.0.0.0:8080->80/tcp, [::]:8080->80/tcp	webserver
32891fbf9243	ubuntu	"bash"	3 hours ago	Up 20 seconds		nice_ganguly

```
root@fode-mangane:~#
```

Voici les adresses des deux conteneurs : l'adresse du conteneur Nginx est 172.17.0.2, et celle du conteneur Ubuntu est 172.17.0.3.

```
root@fode-mangane:~# docker inspect 2f8a10c3aceb | grep "IPAddress"
"SecondaryIPAddresses": null,
"IPAddress": "172.17.0.2",
"IPAddress": "172.17.0.2",
root@fode-mangane:~# docker inspect 32891fbf9243 | grep "IPAddress"
"SecondaryIPAddresses": null,
"IPAddress": "172.17.0.3",
"IPAddress": "172.17.0.3",
root@fode-mangane:~#
```

Dans notre conteneur Ubuntu, nous procédons à la mise à jour des paquets.

```
root@fode-mangane:~# docker exec -it nice_ganguly bash
root@32891fbf9243:/# apt update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1033 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1080 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [1318 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [22.1 kB]
37% [6 Packages 4166 kB/19.3 MB 22%]
```

Nous allons maintenant installer les paquets nécessaires pour effectuer des pings.

```
root@32891fbf9243:/# apt install -y iputils-ping
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2-bin libpam-cap
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 90.9 kB of archives.
After this operation, 322 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libcap2-bin amd64 1:2.66-5ubuntu2.2 [34.2 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/main amd64 iputils-ping amd64 3:20240117-1build1 [44.3 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libpam-cap amd64 1:2.66-5ubuntu2.2 [12.5 kB]
Fetched 90.9 kB in 1s (96.7 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package libcap2-bin.
(Reading database ... 4381 files and directories currently installed.)
Preparing to unpack .../libcap2-bin_1%3a2.66-5ubuntu2.2_amd64.deb ...
Unpacking libcap2-bin (1:2.66-5ubuntu2.2) ...
Selecting previously unselected package iputils-ping.
Preparing to unpack .../iputils-ping_3%3a20240117-1build1_amd64.deb ...
```

Et voilà, le conteneur Ubuntu parvient à pinger le conteneur Nginx avec succès.

```
root@32891fbf9243:/# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.059 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.094 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.153 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.120 ms
```

Nous effectuons la même opération sur le conteneur Nginx pour vérifier la communication.

```

root@32891fbf9243:~# exit
exit
root@fode-mangane:~# docker exec -it webserver bash
root@2f8a10c3aceb:~# apt update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8792 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [512 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [258 kB]
Fetched 9305 kB in 9s (987 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@2f8a10c3aceb:~# apt install -y iputils-ping
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2-bin libpam-cap
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 96.3 kB of archives.
After this operation, 312 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 libcap2-bin amd64 1:2.66-4 [34.7 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 iputils-ping amd64 3:20221126-1+deb12u1 [47.2 kB]
Get:3 http://deb.debian.org/debian bookworm/main amd64 libpam-cap amd64 1:2.66-4 [14.5 kB]
Fetched 96.3 kB in 1s (126 kB/s)

```

Et voilà, le conteneur Nginx parvient à pinger le conteneur Ubuntu avec succès.

```

root@2f8a10c3aceb:~# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.104 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.102 ms
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.074 ms
64 bytes from 172.17.0.3: icmp_seq=4 ttl=64 time=0.106 ms
64 bytes from 172.17.0.3: icmp_seq=5 ttl=64 time=0.076 ms
64 bytes from 172.17.0.3: icmp_seq=6 ttl=64 time=0.096 ms
64 bytes from 172.17.0.3: icmp_seq=7 ttl=64 time=0.096 ms

```

Ensuite nous faisons un ping vers Google pour vérifier l'accès à Internet.

```

root@2f8a10c3aceb:~# ping google.com
PING google.com (142.250.184.174) 56(84) bytes of data.
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=1 ttl=127 time=149 ms
64 bytes from 174.184.250.142.in-addr.arpa (142.250.184.174): icmp_seq=2 ttl=127 time=143 ms
64 bytes from 174.184.250.142.in-addr.arpa (142.250.184.174): icmp_seq=3 ttl=127 time=52.7 ms
64 bytes from 174.184.250.142.in-addr.arpa (142.250.184.174): icmp_seq=4 ttl=127 time=165 ms

```

5.3. Création de réseaux personnalisés

Nous allons maintenant créer notre propre réseau en mode pont.

```

root@fode-mangane:~# docker network create --driver bridge fode-bridge
276a2ef6eedf16c44c9ad1fb25d1e66cb21d879650db1ebbdd4479915f009d12
root@fode-mangane:~# docker network ls
NETWORK ID        NAME          DRIVER  SCOPE
ad07dfcdd64a      bridge       bridge  local
276a2ef6eedf      fode-bridge  bridge  local
f07ee5f1f92d      host         host    local
23dc677471fa      none         null    local
root@fode-mangane:~#

```

Nous allons créer deux conteneurs et les connecter à notre réseau personnalisé.

```

root@fode-mangane:~# docker run -dt --network fode-bridge --name=Cont1 ubuntu bash
d019b0dcbad554874293b0d6d1e28acb5ce277b060d713e473d8900d406130df
root@fode-mangane:~# docker run -dt --network fode-bridge --name=Cont2 ubuntu bash
ac4a2f5182316d53c6a851a1b2c089f830e60b6e4194147a1c47f2fe486c5a56
root@fode-mangane:~#

```

Nous inspectons notre réseau.


```

root@fode-mangane:~# docker network inspect fode-bridge
[
  {
    "Name": "fode-bridge",
    "Id": "276a2ef6eedf16c44c9ad1fb25d1e66cb21d879650db1ebbdd4479915f009d12",
    "Created": "2025-05-07T23:44:32.701445414Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv4": true,
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "ac4a2f5182316d53c6a851a1b2c089f830e60b6e4194147a1c47f2fe486c5a56": {
        "Name": "Cont2",
        "EndpointID": "2392587caa6c5b9cc3f58f3275d8c8533b0258bf5f7b463494616bfb745852d1",
        "MacAddress": "7a:40:e7:61:a5:93",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
      },
      "d019b0dcbad554874293b0d6d1e28acb5ce277b060d713e473d8900d406130df": {
        "Name": "Cont1",
        "EndpointID": "8f46eabc9412560fd28401b06665217746970b37d1e24c84d3c47c709274d26b",
        "MacAddress": "06:b8:b6:f0:3b:ed",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
root@fode-mangane:~#

```

Nous exécutons le conteneur Cont1 et nous faisons les mises à jour.

```

root@fode-mangane:~# docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
ac4a2f518231   ubuntu    "bash"                   7 minutes ago Up 7 minutes
d019b0dcbad5   ubuntu    "bash"                   7 minutes ago Up 7 minutes
2f8a10c3aceb   nginx     "/docker-entrypoint...." 4 hours ago   Up 53 minutes    0.0.0.0:8080->80/tcp, [::]:8080
32891fbf9243   ubuntu    "bash"                   4 hours ago   Up 52 minutes
1c069eea1f5e   hello-world "/hello"                 4 hours ago   Exited (0) 4 hours ago
brave napier
root@fode-mangane:~# docker exec -it d019b0dcbad5 bash
root@d019b0dcbad5:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1080 kB]

```

Installons les utilitaires ping.

```

root@d019b0dcbad5:/# apt-get install iputils-ping
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2-bin libpam-cap
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 90.9 kB of archives.
After this operation, 322 kB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

Nous allons pinger le conteneur Cont2 qui se trouve sur le même réseau.


```

root@d019b0dcbad5:~# ping cont2
PING cont2 (172.18.0.3) 56(84) bytes of data:
64 bytes from Cont2.fode-bridge (172.18.0.3): icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from Cont2.fode-bridge (172.18.0.3): icmp_seq=2 ttl=64 time=0.097 ms
64 bytes from Cont2.fode-bridge (172.18.0.3): icmp_seq=3 ttl=64 time=0.082 ms
64 bytes from Cont2.fode-bridge (172.18.0.3): icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from Cont2.fode-bridge (172.18.0.3): icmp_seq=5 ttl=64 time=0.125 ms
64 bytes from Cont2.fode-bridge (172.18.0.3): icmp_seq=6 ttl=64 time=0.127 ms
64 bytes from Cont2.fode-bridge (172.18.0.3): icmp_seq=7 ttl=64 time=0.091 ms

```

Nous allons ensuite faire un ping vers Google pour vérifier la connectivité.

```

root@d019b0dcbad5:~# ping google.com
PING google.com (142.250.184.174) 56(84) bytes of data:
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=1 ttl=127 time=323 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=2 ttl=127 time=110 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=3 ttl=127 time=292 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=4 ttl=127 time=330 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=5 ttl=127 time=205 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=6 ttl=127 time=367 ms

```

Nous vérifions maintenant si les conteneurs sur notre réseau personnalisé ont accès au réseau bridge.

```

root@d019b0dcbad5:~# ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1) 56(84) bytes of data:
64 bytes from 172.17.0.1: icmp_seq=1 ttl=64 time=0.057 ms
64 bytes from 172.17.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 172.17.0.1: icmp_seq=3 ttl=64 time=0.122 ms
64 bytes from 172.17.0.1: icmp_seq=4 ttl=64 time=0.109 ms
64 bytes from 172.17.0.1: icmp_seq=5 ttl=64 time=0.103 ms
64 bytes from 172.17.0.1: icmp_seq=6 ttl=64 time=0.108 ms

```

5.4. Utilisation du réseau hôte

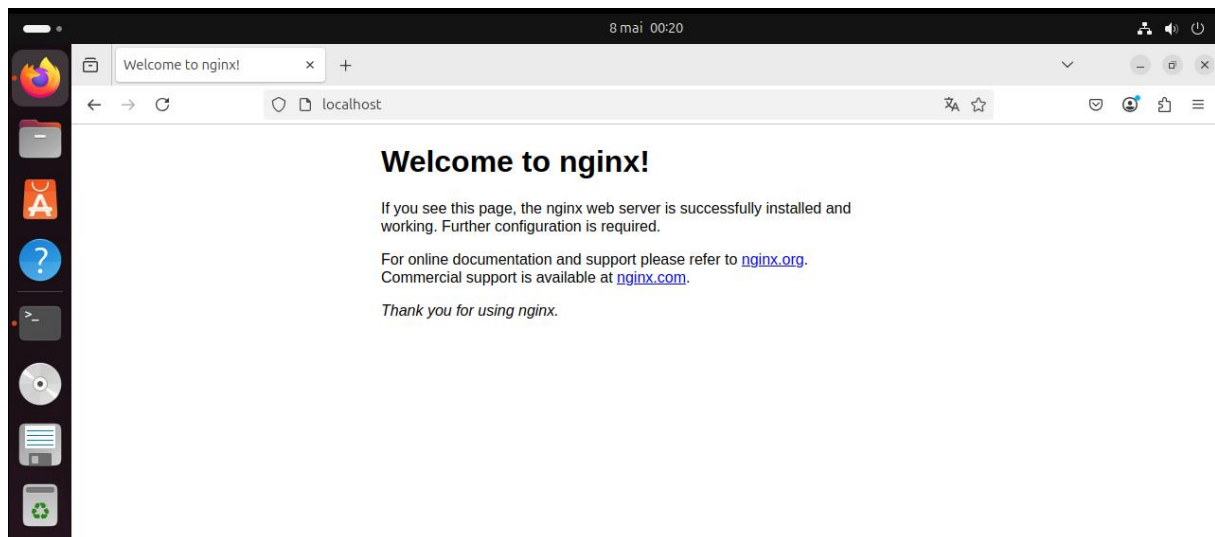
Nous allons démarrer notre conteneur Nginx en mode détaché sur le réseau hôte, puis vérifier s'il a bien été créé.

```

root@fode-mangane:~# docker run --rm -d --network host --name my_nginx nginx
4eb5f2ebdf1bc4e231d006ac84f90b0d9a8e3412a226122e8b753528f30d7e2
root@fode-mangane:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMES
4eb5f2ebdf1    nginx     "/docker-entrypoint..." 4 seconds ago  Up 4 seconds
my_nginx
ac4a2f518231   ubuntu   "bash"                   30 minutes ago Up 30 minutes
Cont2
d019b0dcbad5   ubuntu   "bash"                   30 minutes ago Up 30 minutes
Cont1
2f8a10c3aceb   nginx     "/docker-entrypoint..." 4 hours ago   Up About an hour   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
webserver
32891fbf9243   ubuntu   "bash"                   4 hours ago   Up About an hour
nice_ganguly
root@fode-mangane:~#

```

Notre serveur est désormais accessible depuis le navigateur.



Vérifions quels processus utilisent le port 80. Note : nous installons d'abord les outils nécessaires avec les commandes `sudo apt update` et `sudo apt install net-tools -y`.

```
root@fode-mangane:~# sudo netstat -tulpn | grep :80
tcp        0      0 0.0.0.0:8080        0.0.0.0:*           LISTEN     9464/docker-proxy
tcp        0      0 0.0.0.0:80         0.0.0.0:*           LISTEN     12923/nginx: master
tcp6       0      0 :::8080            :::*                 LISTEN     9470/docker-proxy
tcp6       0      0 :::80              :::*                 LISTEN     12923/nginx: master
```

6. Volumes et persistance des données

6.1. Montage de chemins d'hôte

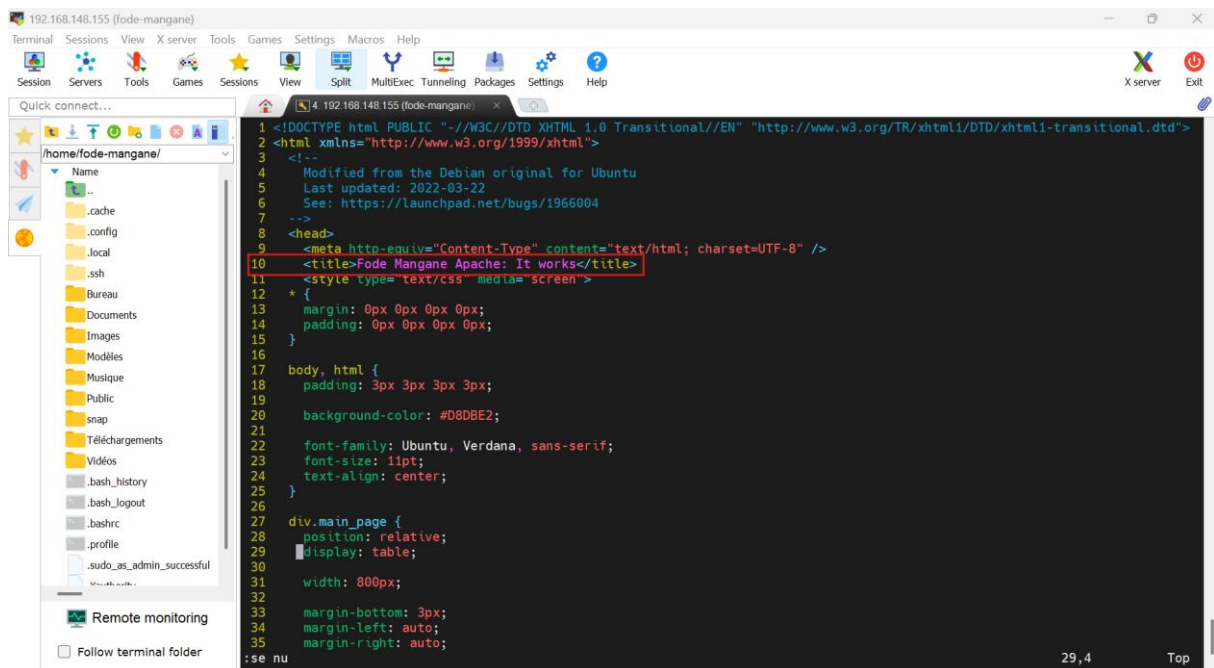
Nous allons maintenant aborder le montage de chemin d'hôte. Nous commençons par créer un répertoire qui servira de volume partagé. Ensuite, nous exécutons un conteneur nommé `volsharing1`, qui sera monté sur le répertoire créé précédemment.

```
root@fode-mangane:/home# mkdir /home/vol-share
root@fode-mangane:/home# docker run -it -p 80:80 -v /home/vol-share:/var/www/html --name volsharing1 ubuntu /bin/bash
root@3b58d6182d1e:/#
```

Nous allons installer et démarrer Apache dans notre conteneur.

```
root@3b58d6182d1e:/# apt update
apt install -y apache2
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [22.1 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1081 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [1318 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1033 kB]
31% [8 Packages 2150 kB/19.3 MB 11%]
root@3b58d6182d1e:/# service apache2 start
* Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
*
root@3b58d6182d1e:/# service apache2 status
* apache2 is running
root@3b58d6182d1e:/#
```

Nous customisons la page d'accueil.

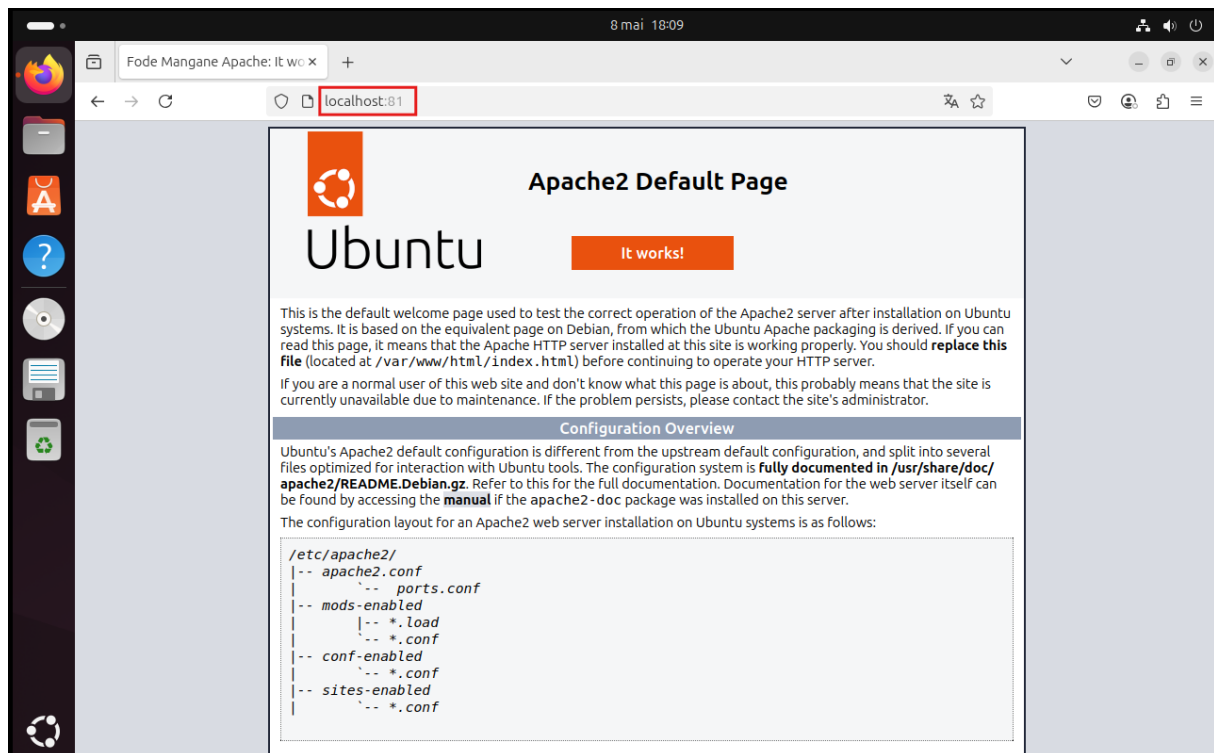


6.2. Partage de données entre conteneurs

Nous allons maintenant créer un deuxième conteneur, volsharing2, et y installer et configurer Apache également.

```
root@fode-mangane:~# docker run -it -p 81:80 -v /home/vol-share:/var/www/html --name volsharing2 ubuntu /bin/bash
root@0bead2c59bcd:/# apt update && apt install -y vim apache2
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1081 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
35% [8 Packages 5087 kB/19.3 MB 26%] [7 Packages 537 kB/1081 kB 50%]
root@0bead2c59bcd:/# service apache2 start
* Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
*
root@0bead2c59bcd:/# service apache2 status
* apache2 is running
root@0bead2c59bcd:/#
```

Nous lançons ensuite un deuxième conteneur et constatons que les modifications effectuées sur le premier conteneur sont également appliquées au second.



6.3. Création et utilisation de volumes Docker

Nous allons maintenant procéder à la création d'un volume Docker, puis lister tous les volumes existants, avant d'inspecter le volume que nous venons de créer.

```
root@fode-mangane:~# docker volume create volume1
volume1
root@fode-mangane:~# docker volume ls
DRIVER      VOLUME NAME
local      volume1
root@fode-mangane:~# docker inspect volume1
[
  {
    "CreatedAt": "2025-05-08T22:51:40Z",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/volume1/_data",
    "Name": "volume1",
    "Options": null,
    "Scope": "local"
  }
]
```

Nous allons créer un conteneur nommé `busybox-cont` et y monter le volume que nous avons créé précédemment.

```
root@fode-mangane:~# docker run -it -v volume1:/data --name busybox-cont busybox:latest
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
97e70d161e81: Pull complete
Digest: sha256:37f7b378a29ceb4c551b1b5582e27747b855bbfaa73fa11914fe0df028dc581f
Status: Downloaded newer image for busybox:latest
/#
```

Nous allons créer un fichier texte dans le volume monté.

```
/ # cd /data
/data # echo "Hi from docker volume" > vol.txt
/data #
```

Nous allons créer un deuxième conteneur qui va être monté sur le même volume. Nous pouvons y retrouver le fichier texte qui a été créé dans l'autre conteneur.

```
root@fode-mangane:~# docker run -it -v volume1:/data --name other-container busybox:latest
/# cd /data
/data # cat vol.txt
Hi from docker volume
/data #
```

Nous allons créer un répertoire local pour stocker les images docker.

```
root@fode-mangane:~# mkdir -p /registry/images
root@fode-mangane:~#
```

Créons un conteneur en local en utilisant une image du Docker Hub puis listons tous les conteneurs en cours d'exécution.

```
root@fode-mangane:~# docker run -d -p 5000:5000 --name registry -v /registry/images:/var/lib/registry --restart always registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
44cf07d57ee4: Pull complete
bbbdd6c6894b: Pull complete
8e82f80af0de: Pull complete
3493bf46cdec: Pull complete
6d464ea18732: Pull complete
Digest: sha256:a3d8aaa63ed8681a604f1dea0aa03f100d5895b6a58ace528858a7b332415373
Status: Downloaded newer image for registry:2
4ebe721ee92d044cfff324f45fcb820116c0e80909f13ce4e25ecee1bc8c1ba9
root@fode-mangane:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
NAMES
4ebe721ee92d   registry:2 "/entrypoint.sh /etc..." 24 seconds ago Up 23 seconds 0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
registry
root@fode-mangane:~#
```

Nous allons procéder à un pull de l'image Jenkins depuis le dépôt public Docker, puis effectuer le tag du conteneur.

```
root@fode-mangane:~# docker pull jenkins/jenkins
Using default tag: latest
latest: Pulling from jenkins/jenkins
cf05a52c0235: Pull complete
d17397ebb26a: Pull complete
79563e7408d6: Pull complete
b013c86cab81: Pull complete
843c26fff869: Pull complete
b461cf81a7b0: Pull complete
9338772faf36: Pull complete
4b03fab713e0: Pull complete
93e0c2618f57: Pull complete
1f21682d0017: Pull complete
e6ceb8dac67e: Pull complete
7d3d841989f8: Pull complete
Digest: sha256:e156a43a586a267930c7fdbc35efbae753dae6417f0f02f5a83b1c6924ee4d73
Status: Downloaded newer image for jenkins/jenkins:latest
docker.io/jenkins/jenkins:latest
root@fode-mangane:~# docker tag jenkins/jenkins localhost:5000/jenkins/jenkins
root@fode-mangane:~#
```

Nous effectuons maintenant le push de l'image vers notre dépôt local.

```
root@fode-mangane:~# docker push localhost:5000/jenkins/jenkins
Using default tag: latest
The push refers to repository [localhost:5000/jenkins/jenkins]
786042a5f98b: Pushed
cf193b809825: Pushed
dd84cd222638: Pushed
a99f75ab4920: Pushed
b4d1e37c11c6: Pushed
e4c0f69e8f38: Pushed
e2b383f49a5e: Pushed
514ff1dda58a: Pushed
d79f989620ee: Pushed
c3542035a14a: Pushed
5491aa6fee97: Pushed
247fffb7158d: Pushed
latest: digest: sha256:49fb9a877c19a9d431d1b532a2aa29491f242fcd17ff4cd9161969aaf0f7e0d0 size: 2833
root@fode-mangane:~#
```

Nous allons maintenant vérifier si l'image a été correctement enregistrée dans notre répertoire.

```
root@fode-mangane:~# cd /registry/images/docker/registry/v2/repositories/  
root@fode-mangane:/registry/images/docker/registry/v2/repositories# ls  
jenkins  
root@fode-mangane:/registry/images/docker/registry/v2/repositories#
```

7. Création d'images avec Dockerfile

7.1. Structure de base d'un Dockerfile

Il est possible de créer une image Docker à partir d'un ensemble d'instructions définies dans un fichier Dockerfile. Nous allons maintenant créer un répertoire appelé Dockerfile et y accéder.

```
root@fode-mangane:/registry/images/docker/registry/v2/repositories# exit  
déconnexion  
fode-mangane@fode-mangane:~$ mkdir Dockerfile  
fode-mangane@fode-mangane:~$ cd Dockerfile/  
fode-mangane@fode-mangane:~/Dockerfile$ vim Dockerfile  
fode-mangane@fode-mangane:~/Dockerfile$
```

Dans notre fichier Dockerfile, nous ajoutons le code suivant :

```
FROM ubuntu  
RUN apt-get update  
RUN apt-get install -y nginx  
COPY index.nginx-debian.html /var/www/html  
EXPOSE 80  
COPY ["/start.sh", "/root/start.sh"]  
ENTRYPOINT /root/start.sh  
~  
~  
~  
~
```

Nous allons créer un message de bienvenue personnalisé dans le fichier index.nginx-debian.html en utilisant l'éditeur vim, depuis le même répertoire.

```
<h1>Bienvenue sur le serveur Nginx personnalisé de Fode Mangane</h1>  
~  
~  
~  
~  
~  
~  
~
```

Nous allons créer un script Shell qui lancera le démon Nginx en éditant le fichier start.sh avec Vim.

```
#!/bin/bash  
  
echo "Lancement du serveur Nginx..."  
nginx -g 'daemon off;'  
~  
~  
~  
~  
~
```

```
fode-mangane@fode-mangane:~/Dockerfile$ chmod +x start-nginx.sh  
fode-mangane@fode-mangane:~/Dockerfile$
```

Nous pouvons maintenant faire le build qui va exécuter notre Dockerfile.

```
fode-mangane@fode-mangane:~/Dockerfile$ sudo docker build .
[+] Building 22.3s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 216B                                0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for ENTRYPOINT to prevent unintended behavior related to OS s 0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest  0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> CACHED [1/5] FROM docker.io/library/ubuntu:latest             0.0s
=> [internal] load build context                                  0.0s
=> => transferring context: 154B                                   0.0s
=> [2/5] RUN apt-get update                                       17.0s
=> [3/5] RUN apt-get install -y nginx                            5.0s
=> [4/5] COPY index.nginx-debian.html /var/www/html              0.0s
=> [5/5] COPY [./start.sh, /root/start.sh]                      0.0s
=> exporting to image                                             0.1s
=> => exporting layers                                           0.1s
=> => writing image sha256:b53052ddb1632b9b1ac6336482154bff2ef464e3b632d67f75a573ba57f48ba 0.0s

1 warning found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for ENTRYPOINT to prevent unintended behavior related to OS signals (line 7)
fode-mangane@fode-mangane:~/Dockerfile$
```

L'image a été construite avec succès. Ce message d'avertissement est normal, Docker recommande simplement d'utiliser le format JSON pour la directive ENTRYPOINT. En vérifiant les images sur notre machine, nous constatons qu'une nouvelle image a été ajoutée, mais elle n'a ni nom ni tag.

```
fode-mangane@fode-mangane:~/Dockerfile$ sudo docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
<none>              <none>       b53052ddb16   4 minutes ago  135MB
myubuntu            latest       a34000e2c15c  27 hours ago  78.1MB
ubuntu              new         7d4b6231c190  28 hours ago  78.1MB
jenkins/jenkins     latest       271510687cf4  2 days ago    471MB
localhost:5000/jenkins/jenkins latest       271510687cf4  2 days ago    471MB
ubuntu              latest       a0e45e2ce6e6  10 days ago   78.1MB
fodemangane/nginx   latest       a830707172e8  3 weeks ago   192MB
nginx               latest       a830707172e8  3 weeks ago   192MB
alpine              latest       aded1e1a5b37  2 months ago  7.83MB
hello-world         latest       74cc54e27dc4  3 months ago  10.1kB
busybox             latest       ff7a7936e930  7 months ago  4.28MB
registry            2           26b2eb03618e  19 months ago 25.4MB
fode-mangane@fode-mangane:~/Dockerfile$
```

Nous allons donc lui attribuer un nom en utilisant son ID.

```
fode-mangane@fode-mangane:~/Dockerfile$ sudo -i
root@fode-mangane:~# docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
<none>              <none>       b53052ddb16   7 minutes ago  135MB
myubuntu            latest       a34000e2c15c  27 hours ago  78.1MB
ubuntu              new         7d4b6231c190  28 hours ago  78.1MB
jenkins/jenkins     latest       271510687cf4  2 days ago    471MB
localhost:5000/jenkins/jenkins latest       271510687cf4  2 days ago    471MB
ubuntu              latest       a0e45e2ce6e6  10 days ago   78.1MB
fodemangane/nginx   latest       a830707172e8  3 weeks ago   192MB
nginx               latest       a830707172e8  3 weeks ago   192MB
alpine              latest       aded1e1a5b37  2 months ago  7.83MB
hello-world         latest       74cc54e27dc4  3 months ago  10.1kB
busybox             latest       ff7a7936e930  7 months ago  4.28MB
registry            2           26b2eb03618e  19 months ago 25.4MB
root@fode-mangane:~# docker tag b53052ddb16 nginxbuilt
root@fode-mangane:~# docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
nginxbuilt          latest       b53052ddb16   7 minutes ago  135MB
myubuntu            latest       a34000e2c15c  27 hours ago  78.1MB
ubuntu              new         7d4b6231c190  28 hours ago  78.1MB
jenkins/jenkins     latest       271510687cf4  2 days ago    471MB
localhost:5000/jenkins/jenkins latest       271510687cf4  2 days ago    471MB
ubuntu              latest       a0e45e2ce6e6  10 days ago   78.1MB
fodemangane/nginx   latest       a830707172e8  3 weeks ago   192MB
nginx               latest       a830707172e8  3 weeks ago   192MB
alpine              latest       aded1e1a5b37  2 months ago  7.83MB
hello-world         latest       74cc54e27dc4  3 months ago  10.1kB
busybox             latest       ff7a7936e930  7 months ago  4.28MB
registry            2           26b2eb03618e  19 months ago 25.4MB
root@fode-mangane:~#
```

Nous allons maintenant lancer cette image pour créer un conteneur Nginx.

```
root@fode-mangane:~# docker run -d -p 82:80 nginxbuilt
ef0c50b579b80559b85cbb77f8d765083f98b304db762a67432abe96f80da850
```



```

root@fode-mangane:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
ef0c50b579b8   nginxbuilt "/bin/sh -c /root/st..." About a minute ago Up About a minute 0.0.0.0:82->80/tcp, [::]:82->80/
tcp           agitated_meninsky
4ebe721ee92d   registry:2 "/entrypoint.sh /etc..." About an hour ago Up About an hour 0.0.0.0:5000->5000/tcp, [::]:500
0->5000/tcp    registry
root@fode-mangane:~# docker exec -it ef0c50b579b8 bash
root@ef0c50b579b8:/# service nginx status
* nginx is running
root@ef0c50b579b8:/#

```

Nous pouvons maintenant accéder à notre serveur Nginx depuis le navigateur et y voir la page que nous avons créée précédemment.



7.2. Utilisation des arguments et variables d'environnement

Il est également possible de créer un Dockerfile qui accepte des arguments lors du processus de build. Nous allons créer un répertoire et y ajouter le Dockerfile.

```

root@fode-mangane:~# mkdir ArgDockerfile
root@fode-mangane:~# cd ArgDockerfile/
root@fode-mangane:~/ArgDockerfile# vim Dockerfile

```

Dans notre cas, nous pouvons transmettre en argument le nom d'un utilisateur. Si aucun nom n'est spécifié, la valeur par défaut sera some_user.

```

FROM busybox
ARG user=some_user

# Créer le répertoire home et ajouter l'utilisateur dans /etc/passwd
RUN mkdir -p /home/$user && \
echo "$user:x:1000:1000:$user:/home/$user:/bin/sh" >> /etc/passwd

USER $user
RUN echo "User is: $user"
~
~
~
~

```

Nous construisons notre image en passant what_user comme valeur pour l'utilisateur.


```

root@fode-mangane:~/ArgDockerfile#
root@fode-mangane:~/ArgDockerfile# docker build --build-arg user=what_user -t test-user .
[+] Building 0.7s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 277B                             0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                   0.0s
=> [1/3] FROM docker.io/library/busybox:latest                 0.0s
=> [2/3] RUN mkdir -p /home/what_user && echo "what_user:x:1000:1000:what_user:/home/what_user:/bin/sh" >> /etc/passwd 0.2s
=> [3/3] RUN echo "User is: what_user"                         0.4s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:0bc851d1f2504777ab6e2f7e2a7c711e8c3a9f53b1b548d31565aafc13b4796d 0.0s
=> => naming to docker.io/library/test-user                     0.0s
root@fode-mangane:~/ArgDockerfile#

```

Pour définir des variables accessibles à l'instruction RUN, il est possible d'utiliser les directives ARG ou ENV. Toutefois, en cas de conflit, la variable définie avec ENV prendra toujours le dessus sur celle définie avec ARG portant le même nom.

```

FROM ubuntu
ARG CONT_IMG_VER
ENV CONT_IMG_VER=v1.0.0
RUN echo $CONT_IMG_VER
~
~
~
~

```

Si nous effectuons le build avec cette commande, la version prise en compte sera celle définie par la variable ENV.

```

root@fode-mangane:~# mkdir VarDockerfile
root@fode-mangane:~# cd VarDockerfile/
root@fode-mangane:~/VarDockerfile# vim Dockerfile
root@fode-mangane:~/VarDockerfile# docker build --build-arg CONT_IMG_VER .
[+] Building 0.1s (6/6) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 113B                             0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                   0.0s
=> [1/2] FROM docker.io/library/ubuntu:latest                   0.0s
=> CACHED [2/2] RUN echo v1.0.0                                 0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:cb7822bf5a0207bcc4afa21a6ee79a78442deed33bc4e6595c830df11a51b175 0.0s
root@fode-mangane:~/VarDockerfile#

```

Il est également possible de monter le conteneur à un volume depuis le Dockerfile.

```

FROM ubuntu
RUN apt-get update && apt-get install -y iputils-ping
VOLUME /data
ENTRYPOINT ["echo"]
~
~
~
~

```

```

root@fode-mangane:~# mkdir DockerVolume
root@fode-mangane:~# cd DockerVolume/
root@fode-mangane:~/DockerVolume# vim Dockerfile
root@fode-mangane:~/DockerVolume# docker build .
[+] Building 17.7s (6/6) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 136B                             0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                   0.0s
=> CACHED [1/2] FROM docker.io/library/ubuntu:latest            0.0s
=> [2/2] RUN apt-get update && apt-get install -y iputils-ping 17.6s
=> exporting to image                                           0.1s
=> => exporting layers                                           0.1s
=> => writing image sha256:5353af35afa4f01b8b75300b7825d1bda8eac8129433b4c85675982db78612d6 0.0s
root@fode-mangane:~/DockerVolume#

```

7.3. Configuration des utilisateurs et groupes

On peut préciser le username ainsi que le groupe auquel il appartient depuis le Dockerfile.

```
FROM ubuntu:latest
RUN apt-get update -y
RUN groupadd -r reseau && useradd -r -g reseau fode
USER fode
CMD ["bash"]
```

```
root@fode-mangane:~# mkdir DockerUser
root@fode-mangane:~# cd DockerUser/
root@fode-mangane:~/DockerUser# vim Dockerfile
root@fode-mangane:~/DockerUser# docker build .
[+] Building 16.2s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 153B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/3] FROM docker.io/library/ubuntu:latest
=> [2/3] RUN apt-get update -y
=> [3/3] RUN groupadd -r reseau && useradd -r -g reseau fode
=> exporting to image
=> => exporting layers
=> => writing image sha256:dcbbc12643ccfb9ac25c77169814979baa8f0994df45896ef337b948683d7363
```

Vérifions que nos images ont bien été créées.

```
root@fode-mangane:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	dcbbc12643cc	2 minutes ago	126MB
<none>	<none>	5353af35afa4	7 minutes ago	128MB
<none>	<none>	cb7822bf5a02	24 minutes ago	78.1MB
test-user	latest	0bc851d1f250	30 minutes ago	4.28MB
nginxbuilt	latest	b53052dabb16	2 hours ago	135MB
myubuntu	latest	a34000e2c15c	29 hours ago	78.1MB
ubuntu	new	7d4b6231c190	29 hours ago	78.1MB
jenkins/jenkins	latest	271510687cf4	2 days ago	471MB
localhost:5000/jenkins/jenkins	latest	271510687cf4	2 days ago	471MB
ubuntu	latest	a0e45e2ce6e6	10 days ago	78.1MB
fodemangane/nginx	latest	a830707172e8	3 weeks ago	192MB
nginx	latest	a830707172e8	3 weeks ago	192MB
alpine	latest	aded1e1a5b37	2 months ago	7.83MB
hello-world	latest	74cc54e27dc4	3 months ago	10.1kB
busybox	latest	ff7a7936e930	7 months ago	4.28MB
registry	2	26b2eb03618e	19 months ago	25.4MB

Lançons le dernier conteneur et vérifions le nom de l'utilisateur ainsi que le groupe auquel il est associé.

```
root@fode-mangane:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	dcbbc12643cc	2 minutes ago	126MB
<none>	<none>	5353af35afa4	7 minutes ago	128MB
<none>	<none>	cb7822bf5a02	24 minutes ago	78.1MB
test-user	latest	0bc851d1f250	30 minutes ago	4.28MB
nginxbuilt	latest	b53052dabb16	2 hours ago	135MB
myubuntu	latest	a34000e2c15c	29 hours ago	78.1MB
ubuntu	new	7d4b6231c190	29 hours ago	78.1MB
jenkins/jenkins	latest	271510687cf4	2 days ago	471MB
localhost:5000/jenkins/jenkins	latest	271510687cf4	2 days ago	471MB
ubuntu	latest	a0e45e2ce6e6	10 days ago	78.1MB
fodemangane/nginx	latest	a830707172e8	3 weeks ago	192MB
nginx	latest	a830707172e8	3 weeks ago	192MB
alpine	latest	aded1e1a5b37	2 months ago	7.83MB
hello-world	latest	74cc54e27dc4	3 months ago	10.1kB
busybox	latest	ff7a7936e930	7 months ago	4.28MB
registry	2	26b2eb03618e	19 months ago	25.4MB

```
root@fode-mangane:~# docker run -it dcbbc12643cc
fode@f277ae0d1918:/# id
uid=999(fode) gid=999(reseau) groups=999(reseau)
```

8. Docker Compose

8.1. Création d'applications multi-conteneurs

Créons le répertoire stackdemo, puis clonons le dépôt disponible à l'adresse <https://github.com/k21academyuk/docker>. Ensuite, accédons au répertoire docker.

```
root@fode-mangane:~# mkdir stackdemo
root@fode-mangane:~# cd stackdemo/
root@fode-mangane:~/stackdemo# git clone https://github.com/k21academyuk/docker
Clonage dans 'docker'...
remote: Enumerating objects: 97, done.
remote: Counting objects: 100% (97/97), done.
remote: Compressing objects: 100% (82/82), done.
remote: Total 97 (delta 22), reused 3 (delta 1), pack-reused 0 (from 0)
Réception d'objets: 100% (97/97), 24.68 Kio | 2.24 Mio/s, fait.
Résolution des deltas: 100% (22/22), fait.
root@fode-mangane:~/stackdemo# ls
docker
root@fode-mangane:~/stackdemo# cd docker/
root@fode-mangane:~/stackdemo/docker# ls
app.py  docker-compose.yml  Ghost_Application  Multi-stage-Guide  Multi-Stage-Guide-2  Nodemaintenance.yaml
root@fode-mangane:~/stackdemo/docker#
```

Assurez-vous que le fichier app.py contient le code suivant. Si ce n'est pas le cas, ajoutez-le manuellement.

```
import time
import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
~
~
~
```

Créons le fichier requirements.txt et y répertorions les bibliothèques nécessaires au projet.

```
flask
redis
~
~
~
```

Créons un Dockerfile.

```
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD ["flask", "run"]
~
~
~
```

Vérifiez que votre fichier docker-compose.yml contient le code suivant. S'il n'est pas présent par défaut, ajoutez-le manuellement.

```
Version: '3'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./code
    environment:
      FLASK_ENV: development
  redis:
    image: "redis:alpine"
```

8.2. Gestion du cycle de vie des applications

Construisons l'application et lançons-la en utilisant la commande docker compose up -d --build.

```
root@fode-mangane:~/stackdemo/docker# docker compose up -d --build
WARN[0000] /root/stackdemo/docker/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 3.3s (13/13) FINISHED
=> [web internal] load build definition from Dockerfile
=> [web internal] load metadata for docker.io/library/python:3.7-alpine
=> [web auth] library/python:pull token for registry-1.docker.io
=> [web internal] load .dockerignore
=> [web internal] transferring context: 2B
=> [web 1/6] FROM docker.io/library/python:3.7-alpine@sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7dff002
=> [web internal] load build context
=> [web internal] transferring context: 3.58kB
=> CACHED [web 2/6] WORKDIR /code
=> CACHED [web 3/6] RUN apk add --no-cache gcc musl-dev linux-headers
=> CACHED [web 4/6] COPY requirements.txt requirements.txt
=> CACHED [web 5/6] RUN pip install -r requirements.txt
=> [web 6/6] COPY . .
=> [web] exporting to image
=> [web] exporting layers
=> [web] writing image sha256:65bb5f9f65c1070f6a6cc0ea5b5753d20c9dc46ecd03d65848b4871130f054e2
=> [web] naming to docker.io/library/docker-web
=> [web] resolving provenance for metadata file
[+] Running 3/3
✔ web Built 0.0s
✔ Container docker-web-1 Started 0.3s
✔ Container docker-redis-1 Started 0.3s
root@fode-mangane:~/stackdemo/docker#
```

Vérifions que l'application fonctionne correctement en accédant à son interface via notre navigateur.



Listons les applications qui s'exécutent depuis le stack.

```
root@fode-mangane:~/stackdemo/docker# docker compose ps
WARN[0000] /root/stackdemo/docker/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
NAME                IMAGE                COMMAND                SERVICE    CREATED        STATUS        PORTS
docker-redis-1      redis:alpine         "docker-entrypoint.s..." redis       13 minutes ago Up 2 minutes  6379/tcp
docker-web-1        docker-web           "flask run"           web        2 minutes ago  Up 2 minutes  0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
root@fode-mangane:~/stackdemo/docker#
```

Nous constatons qu'elles ne sont pas les seules.

```
root@fode-mangane:~/stackdemo/docker# docker ps
CONTAINER ID   IMAGE                COMMAND                CREATED        STATUS        PORTS
23ce3c33eb86   docker-web          "flask run"           3 minutes ago Up 3 minutes  0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
46ff9f52bcd1   docker-web-1        "flask run"           3 minutes ago Up 3 minutes  0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
46ff9f52bcd1   redis:alpine         "docker-entrypoint.s..." 14 minutes ago Up 3 minutes  6379/tcp
root@fode-mangane:~/stackdemo/docker#
```

Modifions notre fichier app.py.

```
import time
import redis
from flask import Flask

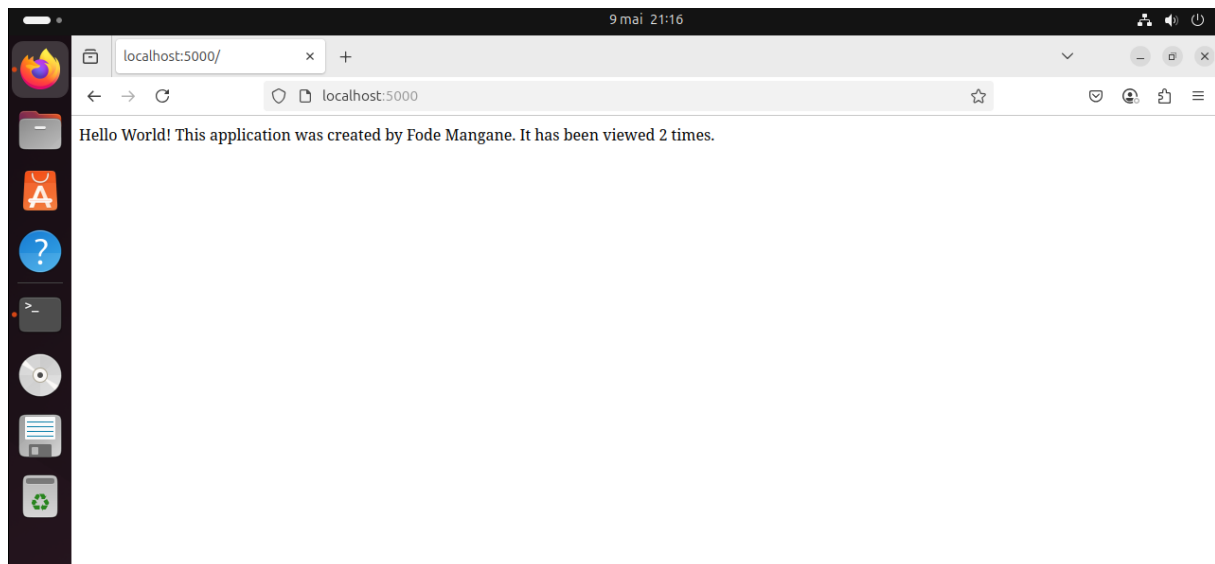
app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits') # Incrémente et retourne le nombre de hits
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! This application was created by Fode Mangane. It has been viewed {} times.\n'.format(count)

if __name__ == "__main__":
    app.run(debug=True)
```

Nous constatons que les modifications apportées ont été prises en compte sans avoir besoin de redémarrer le conteneur.



Nous arrêtons le conteneur.

```
root@fode-mangane:~/stackdemo/docker# docker compose down
WARN[0000] /root/stackdemo/docker/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 3/3
✓ Container docker-redis-1   Removed      0.1s
✓ Container docker-web-1     Removed      0.2s
✓ Network docker_default     Removed      0.1s
root@fode-mangane:~/stackdemo/docker#
```

9. Configuration avancée

9.1. Configuration du DNS

Exécutons un conteneur Nginx présent sur le système et vérifions le contenu du fichier /etc/resolv.conf du conteneur.

```
2f8a10c3aceb  nginx          "/docker-entrypoint..."  2 days ago    Up 24 seconds    0.0.0.0:8080->80/tcp,
[::]:8080->80/tcp  webserver
32891fbf9243  ubuntu         "bash"          2 days ago    Exited (255) 28 hours ago
nice_ganguly
1c069eea1f5e  hello-world    "/hello"        2 days ago    Exited (0) 2 days ago
brave_napier
root@fode-mangane:~/stackdemo/docker# docker exec -it 2f8a10c3aceb bash
root@2f8a10c3aceb:/# cat /etc/resolv.conf
# Generated by Docker Engine.
# This file can be edited; Docker Engine will not make further changes once it
# has been modified.

nameserver 192.168.148.2
search localdomain

# Based on host file: '/run/systemd/resolve/resolv.conf' (legacy)
# Overrides: []
root@2f8a10c3aceb:/#
```

Nous allons créer un fichier daemon.json qui servira de DNS externe pour tous nos conteneurs. Nous y ajouterons l'adresse IP de notre serveur DNS.


```
root@fode-mangane:~# cd stackdemo/docker/
root@fode-mangane:~/stackdemo/docker# docker inspect -f "{{.HostConfig.LogConfig.Type}}" nginxlog
journald
root@fode-mangane:~/stackdemo/docker# cd
```

Vérifions maintenant le driver de stockage.

```
root@fode-mangane:~/stackdemo/docker# cd
root@fode-mangane:~# docker info | grep -i "Storage Driver"
Storage Driver: overlay2
```

```
"dns": ["8.8.8.8"],
"log-driver": "syslog",
"storage-driver": "vfs"
```

```
root@fode-mangane:~# docker info | grep -i "Storage Driver"
Storage Driver: vfs
```

10.1. Comparaison avec single-stage builds

```
root@fode-mangane:~# git clone https://github.com/k21academyuk/Multi-stage-Guide
Clonage dans 'Multi-stage-Guide'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 23 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
Réception d'objets: 100% (23/23), 5.71 Kio | 1.43 Mio/s, fait.
Résolution des deltas: 100% (7/7), fait.
root@fode-mangane:~#
```

```

root@fode-mangane:~/# cd Multi-stage-Guide/
root@fode-mangane:~/Multi-stage-Guide# sudo docker build -f Dockerfile.singlestage -t single_stage
[+] Building 247.5s (11/11) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile.singlestage    0.0s
=> => transferring dockerfile: 158B                                0.0s
=> [internal] load metadata for docker.io/library/node:12         3.9s
=> [auth] library/node:pull token for registry-1.docker.io        0.0s
=> [internal] load .dockerignore                                    0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load build context                                    0.0s
=> => transferring context: 39.22kB                                   0.0s
=> [1/5] FROM docker.io/library/node:12@sha256:01627afeb110b3054ba4a1405541ca095c8bfca1cb6f2be9479c767a2711879e 202.6s
=> resolve docker.io/library/node:12@sha256:01627afeb110b3054ba4a1405541ca095c8bfca1cb6f2be9479c767a2711879e 0.0s
=> sha256:01627afeb110b3054ba4a1405541ca095c8bfca1cb6f2be9479c767a2711879e 776B / 776B 0.0s
=> sha256:3a69ea1270dbaf4e2f0477361be4b7a43400e559c6abdafa69d73f7c755f43475 2.21kB / 2.21kB 0.0s
=> sha256:6c8de432cf7f7d8c58899f61982d1662ec6b73fb3ef92f862ba170dcc5b64fa9 7.68kB / 7.68kB 0.0s
=> sha256:9bed1e86f01ee95c76d2c8b4385a47ae336e6d293afade9368469d99daa9369f 11.30MB / 11.30MB 78.6s
=> sha256:5f196cde425181bc7e4411865a2e002932b7b6b0ffce787c04c1bdeaf1f204f20 45.43MB / 45.43MB 26.8s

```


Vérifions la taille de notre image.

```
root@fode-mangane:~/Multi-stage-Guide# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
single_stage   latest    ab9e3318059d   3 minutes ago  922MB
```

Nous allons ensuite exécuter le Dockerfile pour le Multistage.

```
root@fode-mangane:~/Multi-stage-Guide# cd Multi-stage-Guide/
root@fode-mangane:~/Multi-stage-Guide# ls
app.go Dockerfile.multistage Dockerfile-security Dockerfile.singlestage package.json server.js
root@fode-mangane:~/Multi-stage-Guide# docker build -f Dockerfile.multistage -t multi_stage .
[+] Building 235.5s (14/14) FINISHED
=> [internal] load build definition from Dockerfile.multistage
=> => transferring dockerfile: 231B
=> [internal] load metadata for gcr.io/distroless/nodejs:latest
=> [internal] load metadata for docker.io/library/node:12
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 39.22kB
=> [stage-1 1/3] FROM gcr.io/distroless/nodejs:latest@sha256:b534f9b5528e69baa7e8caf7bcc1d93ecf59faa15d289221decf588 196.2s
=> => resolve gcr.io/distroless/nodejs:latest@sha256:b534f9b5528e69baa7e8caf7bcc1d93ecf59faa15d289221decf5889a2ed3877 0.0s
=> => sha256:b534f9b5528e69baa7e8caf7bcc1d93ecf59faa15d289221decf5889a2ed3877 741B / 741B
=> => sha256:9d75631b3941ccfd290d43720fb3e7e89353b4c6402d9b586ac0f71411dfbaee 918B / 918B
=> => sha256:cf46b336f6597ef45686d6f517a12c43466b85fbdff5cfff01ff7d8109db47046 967B / 967B
=> => sha256:7dcffaf987694bb0a0863ae2c3b582125b1c20d3148f0412f901b918b9a8e22d 817.61kB / 817.61kB
=> => sha256:383e1c5dd0c1830143b1230e90292ebd4219911e0512b70d250c8907c4899110 821.00kB / 821.00kB
=> => sha256:c59673e9fae3f9d588110a25acdf7240f3a5d97c40fb86ccc71c23bf7abbea53 8.02MB / 8.02MB
=> => extracting sha256:383e1c5dd0c1830143b1230e90292ebd4219911e0512b70d250c8907c4899110 0.1s
```

En comparant les tailles des deux images, nous constatons que l'image multi-stage occupe beaucoup moins d'espace que l'image single_stage, qui fait 922 Mo.

Note : J'ai rencontré un problème d'espace disque sur mon serveur Docker, ce qui m'a poussé à effectuer un nettoyage complet des images, conteneurs arrêtés, volumes anonymes et du cache de build en utilisant la commande `docker system prune -a --volumes`. C'est pour cette raison que l'image single_stage n'est plus visible actuellement.

```
root@fode-mangane:~/Multi-stage-Guide# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
multi_stage    latest    9bc1927829f6   39 seconds ago  166MB
```

10.2. Exemples pratiques

On peut utiliser un autre exemple avec le dépôt https://github.com/k21academyuk/Multi-stage-Guide_1

```
root@fode-mangane:~# git clone https://github.com/k21academyuk/Multi-stage-Guide_1
Clonage dans 'Multi-stage-Guide_1'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 18 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
Réception d'objets: 100% (18/18), 4.65 Kio | 4.65 Mio/s, fait.
Résolution des deltas: 100% (4/4), fait.
root@fode-mangane:~#
```

Faisons le build de ce conteneur avec le Dockerfile single stage et multistage.

```

root@fode-mangane:~# cd Multi-stage-Guide_1/
root@fode-mangane:~/Multi-stage-Guide_1# ls
app.go Dockerfile Dockerfile.multi Dockerfile-release Makefile
root@fode-mangane:~/Multi-stage-Guide_1# docker build -f Dockerfile -t single1-saml .
[+] Building 1.8s (11/11) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 204B                                0.0s
=> [internal] load metadata for docker.io/library/golang:1.6-alpine 1.7s
=> [auth] library/golang:pull token for registry-1.docker.io       0.0s
=> [internal] load .dockerignore                                   0.0s
=> => transferring context: 2B                                       0.0s
=> [1/5] FROM docker.io/library/golang:1.6-alpine@sha256:269d188232cd9a6194f71650780cb2e903a76958182def1008cc7255f6f45 0.0s
=> [internal] load build context                                   0.0s
=> => transferring context: 2.18kB                                   0.0s
=> CACHED [2/5] RUN mkdir /app                                     0.0s
=> CACHED [3/5] ADD . /app/                                        0.0s
=> CACHED [4/5] WORKDIR /app                                       0.0s
=> CACHED [5/5] RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main . 0.0s
=> exporting to image                                              0.0s
=> => exporting layers                                              0.0s
=> => writing image sha256:b51a3fbbaa24170985a947571ce3918a37ccf824886fd2f6d63fb89625ca60a76 0.0s
=> => naming to docker.io/library/single1-saml                    0.0s
root@fode-mangane:~/Multi-stage-Guide_1#

```

Nous comparons les deux images et nous notons la même chose que précédemment : le multistage occupe beaucoup moins d'espace mémoire.

```

root@fode-mangane:~/Multi-stage-Guide_1# docker images
REPOSITORY      TAG       IMAGE ID       CREATED        SIZE
single1-saml    latest    b51a3fbbaa241 About an hour ago 291MB
single1-small   latest    b51a3fbbaa241 About an hour ago 291MB
root@fode-mangane:~/Multi-stage-Guide_1#

```

11. Conclusion

Ce guide complet sur Docker a couvert tous les aspects essentiels de cette technologie de conteneurisation, depuis l'installation jusqu'aux fonctionnalités avancées. Voici les points clés à retenir :

Installation et configuration : Docker s'installe facilement sur Ubuntu en suivant la documentation officielle, avec la possibilité d'utiliser Docker sans sudo en ajoutant l'utilisateur au groupe Docker.

Gestion des conteneurs : Nous avons appris à créer, démarrer, arrêter et supprimer des conteneurs, ainsi qu'à exposer des ports pour accéder aux services comme Nginx.

Gestion des images : Le guide a expliqué comment télécharger, inspecter, créer et sauvegarder des images Docker, et comment les publier sur Docker Hub.

Réseaux Docker : Nous avons exploré les différents types de réseaux, la communication entre conteneurs, la création de réseaux personnalisés et l'utilisation du réseau hôte.

Volumes et persistance des données : Le montage de chemins d'hôte, la création de volumes Docker et le partage de données entre conteneurs ont été couverts en détail.

Création d'images avec Dockerfile : La structure de base d'un Dockerfile, l'utilisation des arguments et variables d'environnement, et la configuration des utilisateurs et groupes ont été expliqués.

Docker Compose : La création et la gestion d'applications multi-conteneurs ont été démontrées à travers un exemple pratique.

Configuration avancée : Le guide a abordé la configuration du DNS, de la journalisation et des drivers de stockage.

Multi-stage builds : Les avantages des builds multi-étapes par rapport aux builds à étape unique ont été illustrés, montrant comment les images multi-stage occupent beaucoup moins d'espace disque.

Docker simplifie considérablement le déploiement d'applications en garantissant leur fonctionnement identique dans tous les environnements. Cette technologie est devenue incontournable dans le développement moderne, le déploiement continu et les architectures de microservices. La maîtrise de Docker constitue donc un atout majeur pour tout professionnel de l'informatique.