

Pratique : Container 101

Découvrir les conteneurs avec Docker

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020

emmanuel.bruno@univ-tln.fr

2026-09-30

Table des matières

1	Objectif du TP	1
2	Prérequis	1
3	Setup initial	2
4	Exercice 1 — Lancer un premier conteneur	2
4.1	Objectif	2
5	Exercice 2 — Volumes : partager des fichiers	6
5.1	Objectif	6
6	Exercice 3 — Mode interactif et installation	7
7	Exercice 4 — Créer une image simple	11
8	Exercice 5 — Construire une image applicative	12
9	Exercice 6 — (Optionnel) Réseaux Docker	13
10	Synthèse	13

1 Objectif du TP

💡 Astuce

Donner le goût des conteneurs en pratiquant :

- Les commandes essentielles : `run`, `ps`, `stop`, `rm`
- Les ports, volumes et réseaux
- Le mode interactif (exploration)
- La création d'une image simple avec un Dockerfile

Ce TP **n'est pas un TP de production**. Il vise à construire une **intuition correcte** des conteneurs.

2 Prérequis

- Docker installé et fonctionnel :

```
docker --version
```

```
Docker version 29.8.0, build 88096ef
```

- Accès à un terminal (bash/zsh)
- Éditeur de texte minimal

⚠ Avertissement

Si l'installation de Docker localement est difficile, vous pouvez utiliser un environnement en ligne tel que Play with Docker ou Katacoda.

💡 Astuce

Les commandes présentées dans ce TP utilisent les **formes longues** des options pour plus de clarté. **Formes longues ↔ courtes**

- **-detach** : -d
- **-publish** : -p
- **-volume** : -v
- **-interactive** : -i
- **-tty** : -t
- **-env** : -e
- **-name** : (pas de forme courte distincte)
- **-rm** : (forme longue déjà explicite)

3 Setup initial

Créer un dossier de travail et s'y positionner (\$TP_DIR="~/tp0-containers-101"):

Nettoyage optionnel des restes d'exécutions précédentes :

```
docker rm --force tp0-nginx tp0-app
2>/dev/null || true
docker network rm tp0-net 2>/dev/null
|| true
docker image rm tp0-app:0.1 2>/dev/
null || true
```

4 Exercice 1 — Lancer un premier conteneur

4.1 Objectif

Voir un conteneur **en cours d'exécution**.

La commande suivante lance un conteneur `nginx` (serveur web) en arrière-plan, nommé `tp0-nginx`, exposant le port 80 du conteneur sur le port 8080 de l'hôte :

```
docker run --detach \
  --name tp0-nginx \
  --publish 8080:80 \
  nginx:stable
```

```
Unable to find image 'nginx:stable'
locally
stable: Pulling from library/nginx
```

```
6b37362b3da7: Pulling fs layer
```

```
b9614c6dbd86: Pulling fs layer
```

```
3a55ec62cf16: Pulling fs layer
```

```
f3cb4f951034: Pulling fs layer
```

```
2aa35770407f: Pulling fs layer
```

```
af529266d394: Pulling fs layer
```

```
757049f14b94: Pulling fs layer
```

```
f3cb4f951034: Waiting
```

```
2aa35770407f: Waiting
```

```
af529266d394: Waiting
```

```
757049f14b94: Waiting
```

3a55ec62cf16:
Downloading 626B/626B

3a55ec62cf16: Download complete

b9614c6dbd86:
Downloading 341.2kB/33.33MB

6b37362b3da7:
Downloading 310.7kB/29.83MB

6b37362b3da7:
Downloading 5.914MB/29.83MB

b9614c6dbd86:
Downloading 5.483MB/33.33MB

6b37362b3da7:
Downloading 13.07MB/29.83MB

b9614c6dbd86:
Downloading 10.63MB/33.33MB

6b37362b3da7:
Downloading 19.61MB/29.83MB

b9614c6dbd86:
Downloading 15.78MB/33.33MB

6b37362b3da7:
Downloading 25.84MB/29.83MB

b9614c6dbd86:
Downloading 20.54MB/33.33MB

6b37362b3da7: Verifying Checksum

6b37362b3da7: Download complete

b9614c6dbd86:
Downloading 26.68MB/33.33MB

6b37362b3da7:

Extracting 327.7kB/29.83MB

b9614c6dbd86: Verifying Checksum

b9614c6dbd86: Download complete

6b37362b3da7:
Extracting 3.277MB/29.83MB

6b37362b3da7:
Extracting 6.226MB/29.83MB

2aa35770407f:
Downloading 404B/404B

2aa35770407f: Download complete

6b37362b3da7:
Extracting 8.847MB/29.83MB

af529266d394:
Downloading 1.21kB/1.21kB

af529266d394: Verifying Checksum

af529266d394: Download complete

6b37362b3da7:
Extracting 10.81MB/29.83MB

6b37362b3da7:
Extracting 13.11MB/29.83MB

757049f14b94:
Downloading 1.399kB/1.399kB

757049f14b94: Verifying Checksum

757049f14b94: Download complete

6b37362b3da7:
Extracting 16.38MB/29.83MB

6b37362b3da7:
Extracting 19.66MB/29.83MB

6b37362b3da7:
Extracting 22.94MB/29.83MB

f3cb4f951034:
Downloading 955B/955B

f3cb4f951034: Verifying Checksum

f3cb4f951034: Download complete

6b37362b3da7:
Extracting 25.56MB/29.83MB

6b37362b3da7:
Extracting 26.21MB/29.83MB

6b37362b3da7:
Extracting 28.18MB/29.83MB

6b37362b3da7:
Extracting 28.84MB/29.83MB

6b37362b3da7:
Extracting 29.16MB/29.83MB

6b37362b3da7:
Extracting 29.82MB/29.83MB

6b37362b3da7:
Extracting 29.83MB/29.83MB

6b37362b3da7: Pull complete

b9614c6dbd86:
Extracting 360.4kB/33.33MB

b9614c6dbd86:
Extracting 2.884MB/33.33MB

b9614c6dbd86:

Extracting 6.488MB/33.33MB

b9614c6dbd86:
Extracting 9.732MB/33.33MB

b9614c6dbd86:
Extracting 12.98MB/33.33MB

b9614c6dbd86:
Extracting 14.78MB/33.33MB

b9614c6dbd86:
Extracting 18.02MB/33.33MB

b9614c6dbd86:
Extracting 20.91MB/33.33MB

b9614c6dbd86:
Extracting 24.15MB/33.33MB

b9614c6dbd86:
Extracting 27.03MB/33.33MB

b9614c6dbd86:
Extracting 30.28MB/33.33MB

b9614c6dbd86:
Extracting 30.64MB/33.33MB

b9614c6dbd86:
Extracting 31MB/33.33MB

b9614c6dbd86:
Extracting 33.33MB/33.33MB

b9614c6dbd86: Pull complete

3a55ec62cf16: Extracting 626B/626B

3a55ec62cf16: Extracting 626B/626B

3a55ec62cf16: Pull complete

```
f3cb4f951034: Extracting 955B/955B
f3cb4f951034: Extracting 955B/955B
f3cb4f951034: Pull complete
2aa35770407f: Extracting 404B/404B
2aa35770407f: Extracting 404B/404B
2aa35770407f: Pull complete

af529266d394:
Extracting 1.21kB/1.21kB

af529266d394:
Extracting 1.21kB/1.21kB

af529266d394: Pull complete

757049f14b94:
Extracting 1.399kB/1.399kB

757049f14b94:
Extracting 1.399kB/1.399kB

757049f14b94: Pull complete
Digest: sha256:b972f831f200b19ef0767938224f9711e74cd783718738cd7405d5cabf75c442
Status: Downloaded newer image for
nginx:stable
fe1074ad3fedaa76678b00d3f49f98287785dd5592af6703aae8859818225fa1
```

Vérifier qu'il est bien lancé :

```
docker container ls
```

CONTAINER COMMAND STATUS NAMES	ID	IMAGE CREATED PORTS
fe1074ad3fed "/docker-entrypoint...."		nginx:stable 2 seconds

```
ago Up Less than a second
0.0.0.0:8080->80/tcp, [::]:8080->80/
tcp tp0-nginx
d496ee28e239 ghcr.io/ebpro/jupyter-
base:quarto-full-fix-chromium-ci-
headless-render "bash -lc '\n #
[dia..." 50 seconds ago Up 46 seconds
amazing_johnson
```

Il est accessible via le port 8080 de l'hôte avec un navigateur web ou curl :

Tester :

```
curl --max-time 30 --include http://
localhost:8080
```

```
HTTP/1.1 200 OK [1mServer[0m:
nginx/1.30.5 [1mDate[0m: Wed, 30 Sep
2026 06:51:26 GMT [1mContent-Type[0m:
text/html [1mContent-Length[0m: 896
[1mLast-Modified[0m: Tue, 15 Sep
2026 13:27:58 GMT [1mConnection[0m:
keep-alive [1mETag[0m: «6aa947de-380»
[1mAccept-Ranges[0m: bytes
```

```
<!DOCTYPE html> Welcome to nginx! Wel-
come to nginx! If you see this page, nginx
is successfully installed and working. Further
configuration is required for the web server,
reverse proxy, API gateway, load balancer,
content cache, or other features. For online
documentation and support please refer to
nginx.org. To engage with the community
please visit community.nginx.org. For enter-
prise grade support, professional services,
additional security features and capabilities
please refer to f5.com/nginx. Thank you for
using nginx. Les 10 derniers logs du conteneur
peuvent être consultés avec :
```

```
docker logs tp0-nginx --tail 10
```

```

2026/09/30 06:51:25 [notice] 1#1:
start worker process 52
2026/09/30 06:51:25 [notice] 1#1:
start worker process 53
2026/09/30 06:51:25 [notice] 1#1:
start worker process 54
2026/09/30 06:51:25 [notice] 1#1:
start worker process 55
2026/09/30 06:51:25 [notice] 1#1:
start worker process 56
2026/09/30 06:51:25 [notice] 1#1:
start worker process 57
2026/09/30 06:51:25 [notice] 1#1:
start worker process 58
2026/09/30 06:51:25 [notice] 1#1:
start worker process 59
2026/09/30 06:51:25 [notice] 1#1:
start worker process 60
172.17.0.1 - - [30/Sep/2026:06:51:26
+0000] "GET / HTTP/1.1" 200 896 "-"
"curl/8.5.0" "-"

```

Le conteneur peut être arrêté avec :

```
docker stop tp0-nginx
```

```
tp0-nginx
```

Les conteneurs y compris arrêtés peuvent être listés avec :

```
docker container ls --all
```

CONTAINER	ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
fe1074ad3fed		nginx:stable	"/docker-entrypoint...."	4 seconds ago	Exited (0)	Less than a second ago	tp0-nginx
d496ee28e239		ghcr.io/ebpro/jupyter-					

```

base:quarto-full-fix-chromium-ci-
headless-render "bash -lc '\n #
[diagram] 52 seconds ago Up 48 seconds
amazing_johnson

```

et supprimé avec :

```
docker rm --force tp0-nginx
```

```
tp0-nginx
```

! Rappel

Un conteneur = un service/processus isolé

5 Exercice 2 — Volumes : partager des fichiers

5.1 Objectif

Pour relier le système de fichier de l'hôte et celui du conteneur, on utilise des **volumes**.

Créer un dossier `www/` sur l'hôte avec une page HTML simple :

```

mkdir -p www
echo "<h1>Hello from host</h1>" >
www/index.html

```

Lancer un serveur web nginx en montant le dossier `www/` de l'hôte dans le dossier `/usr/share/nginx/html` du conteneur (en lecture seule avec `:ro`) :

```

docker run --detach \
  --name tp0-nginx \
  --publish 8080:80 \
  --volume "$PWD/www":/usr/share/
nginx/html:ro \
  nginx:stable

```

Il est accessible via le port 8080 de l'hôte avec un navigateur web ou `curl` :

`curl` est utilisé ici pour afficher les headers HTTP et le contenu de la page.

```
curl --max-time 30 --include http://localhost:8080
```

```
HTTP/1.1 200 OK
Server: nginx/1.30.5
Date: Wed, 30 Sep 2026 06:51:34 GMT
Content-Type: text/html
Content-Length: 25
Last-Modified: Wed, 30 Sep 2026 06:51:27 GMT
Connection: keep-alive
ETag: "6abcb16f-19"
Accept-Ranges: bytes

<h1>Hello from host</h1>
```

Modifier `www/index.html` sur l'hôte et rafraîchir la page ou relancer `curl`.

```
echo "<h1>Hello from host - updated</h1>" > www/index.html
curl --max-time 30 --include http://localhost:8080
```

Nettoyage :

! Rappel

Les données ne vivent pas dans le conteneur, il doit être éphémère

6 Exercice 3 — Mode interactif et installation

Il est important de **comprendre la différence entre un conteneur et une image**. Il est possible d'exécuter des conteneurs en mode

interactif pour explorer leur contenu et interagir avec eux.

Il est déconseillé d'installer des logiciels directement dans un conteneur en production.

```
docker run --interactive --tty --rm ubuntu:22.04
```

Dans le conteneur, exécuter les commandes suivantes pour explorer le système, créer un fichier et installer `curl`.

```
cat /etc/os-release
uname -a
echo "Hello from inside the container" > /tmp/hello.txt
apt update && apt install -y curl
curl --max-time 30 --include https://example.com
```

quitter le conteneur avec la commande `exit` (ou `Ctrl+D`).

```
Unable to find image 'ubuntu:22.04' locally
22.04: Pulling from library/ubuntu
```

```
20c3783cc497: Pulling fs layer
```

```
20c3783cc497:
Downloading 310.7kB/29.75MB
```

```
20c3783cc497:
Downloading 7.782MB/29.75MB
```

```
20c3783cc497:
Downloading 14.94MB/29.75MB
```

```
20c3783cc497:
Downloading 21.79MB/29.75MB
```

20c3783cc497: Verifying Checksum

20c3783cc497: Download complete

20c3783cc497:

Extracting 327.7kB/29.75MB

20c3783cc497:

Extracting 2.949MB/29.75MB

20c3783cc497:

Extracting 5.898MB/29.75MB

20c3783cc497:

Extracting 7.864MB/29.75MB

20c3783cc497:

Extracting 9.175MB/29.75MB

20c3783cc497:

Extracting 11.14MB/29.75MB

20c3783cc497:

Extracting 14.42MB/29.75MB

20c3783cc497:

Extracting 18.02MB/29.75MB

20c3783cc497:

Extracting 21.3MB/29.75MB

20c3783cc497:

Extracting 24.25MB/29.75MB

20c3783cc497:

Extracting 25.56MB/29.75MB

20c3783cc497:

Extracting 25.89MB/29.75MB

20c3783cc497:

Extracting 27.85MB/29.75MB

20c3783cc497:

Extracting 28.51MB/29.75MB

20c3783cc497:

Extracting 29.49MB/29.75MB

20c3783cc497:

Extracting 29.75MB/29.75MB

20c3783cc497: Pull complete

Digest: sha256:b8b6ee6aa931ecd9d0d952abc34dc0e5f7c6a30c

Status: Downloaded newer image for
ubuntu:22.04

debconf: delaying package
configuration, since apt-utils is not
installed

Selecting previously unselected
package openssl.

(Reading database ... 4393 files and
directories currently installed.)

Preparing to unpack .../00-openssl_3.0.2-0ubuntu1.30_amd64.deb ...

Unpacking

openssl (3.0.2-0ubuntu1.30) ...

Selecting previously unselected
package ca-certificates.

Preparing to unpack .../01-ca-certificates_20260601~22.04.1ubuntu1~22.04.1.deb ...

Unpacking

ca-certificates (20260601~22.04.1) ...

Selecting previously unselected
package libnghttp2-14:amd64.

Preparing to unpack .../02-libnghttp2-14_1.43.0-1ubuntu0.4_amd64.deb ...

Unpacking

libnghttp2-14:amd64

(1.43.0-1ubuntu0.4) ...
Selecting previously unselected
package libpsl5:amd64.

Preparing to unpack .../03-libpsl5_0.21.0-1.2build2_amd64.deb ...

Unpacking

libpsl5:amd64 (0.21.0-1.2build2) ...

Selecting previously unselected
package publicsuffix.

Preparing to unpack .../04-publicsuffix_20211207.1025-1ubuntu0.22.04.1.deb ...

Unpacking

publicsuffix (20211207.1025-1) ...

```

Selecting previously unselected
package libbrotli1:amd64.
Preparing to unpack .../05-libbrotli1_1.0.
Unpacking
libbrotli1:amd64 (1.0.9-2build6) ...
Selecting previously unselected
package libsasl2-modules-db:amd64.
Preparing to unpack .../06-libsasl2-
modules-db_2.1.27+dfsg2-3ubuntu1.2_amd64.
Unpacking libsasl2-modules-db:amd64
(2.1.27+dfsg2-3ubuntu1.2) ...
Selecting previously unselected
package libsasl2-2:amd64.
Preparing to unpack .../07-libsasl2-2_2.1.
Unpacking libsasl2-2:amd64
(2.1.27+dfsg2-3ubuntu1.2) ...
Selecting previously unselected
package libldap-2.5-0:amd64.
Preparing to unpack .../08-libldap-2.5-0_2.
Unpacking libldap-2.5-0:amd64
(2.5.20+dfsg-0ubuntu0.22.04.1) ...
Selecting previously unselected
package librtmp1:amd64.
Preparing to unpack .../09-librtmp1_2.4+20
Unpacking librtmp1:amd64 (2.4+20151223.git
Selecting previously unselected
package libssh-4:amd64.
Preparing to unpack .../10-libssh-4_0.9.6-
Unpacking libssh-4:amd64
(0.9.6-2ubuntu0.22.04.8) ...
Selecting previously unselected
package libcurl4:amd64.
Preparing to unpack .../11-libcurl4_7.81.0
Unpacking libcurl4:amd64
(7.81.0-1ubuntu1.29) ...
Selecting previously unselected
package curl.
Preparing to unpack .../12-curl_7.81.0-1ub
Unpacking
curl (7.81.0-1ubuntu1.29) ...
Selecting previously unselected
package libldap-common.
Preparing to unpack .../13-libldap-common_

```

```

Unpacking libldap-common
(2.5.20+dfsg-0ubuntu0.22.04.1) ...
Selecting previously unselected
package libsasl2-modules:amd64.
Preparing to unpack .../14-libsasl2-modules_2.1.27+dfsg2
Unpacking libsasl2-modules:amd64
(2.1.27+dfsg2-3ubuntu1.2) ...
Setting up libpsl5:amd64
(0.21.0-1.2build2) ...
Setting up libbrotli1:amd64
(1.0.9-2build6) ...
Setting up libsasl2-modules:amd64
(2.1.27+dfsg2-3ubuntu1.2) ...
Setting up libnghttp2-14:amd64
(1.43.0-1ubuntu0.4) ...
Setting up libldap-common
(2.5.20+dfsg-0ubuntu0.22.04.1) ...
Setting up libsasl2-modules-db:amd64
(2.1.27+dfsg2-3ubuntu1.2) ...
Setting up librtmp1:amd64 (2.4+20151223.gitfa8646d.1-2bu
Setting up libsasl2-2:amd64
(2.1.27+dfsg2-3ubuntu1.2) ...
Setting up libssh-4:amd64
(0.9.6-2ubuntu0.22.04.8) ...
Setting up openssl
(3.0.2-0ubuntu1.30) ...
Setting up publicsuffix
(20211207.1025-1) ...
Setting up libldap-2.5-0:amd64
(2.5.20+dfsg-0ubuntu0.22.04.1) ...
Setting up ca-certificates
(20260601~22.04.1) ...
debconf: unable to initialize
frontend: Dialog
debconf: (TERM is not set, so the
dialog frontend is not usable.)
debconf: falling back to frontend:
Readline
debconf: unable to initialize
frontend: Readline
debconf: (Can't locate Term/
ReadLine.pm in @INC (you may
need to install the Term::ReadLine .

```



```
cat /tmp/hello.txt # ✕ disparu
curl --max-time 30 --include https://
example.com # ✕ curl non installé
```

! Rappel

Ce qui n'est pas dans une image est perdu

7 Exercice 4 — Créer une image simple

Pour rendre l'installation reproductible et persistante, il faut créer une image. L'approche standard est d'écrire un fichier **Dockerfile** décrivant l'image.

Créer un Dockerfile qui installe curl (RUN) dans une image basée sur ubuntu:22.04 (FROM) et qui lance bash par défaut (CMD).

```
1 FROM ubuntu:22.04
2 RUN apt update && apt install -
y curl
3 CMD ["bash"]
```

Construire l'image Docker à partir d'un Dockerfile dans le dossier courant (.) avec le tag tp0-ubuntu-curl :

```
docker build --tag tp0-ubuntu-curl .
```

Tester en mode interactif en exécutant la commande curl --max-time 30 --include https://example.com dans le conteneur :

```
docker run --interactive --tty --rm
tp0-ubuntu-curl
```

la commande peut aussi être passée en paramètre du docker run :

```
docker run --rm tp0-ubuntu-curl \
    curl --max-time 30 --include
https://example.com
```

```
% Total    % Received % Xferd  Average
Speed    Time      Time      Time      Current
                                Dload
Upload    Total    Spent    Left    Speed

  0      0    0      0    0      0      0
0  -:--:-- -:--:-- -:--:--      0
100    713    0    713    0      0   5958
0  -:--:-- -:--:-- -:--:--   5991
HTTP/2 200
date: Wed, 30 Sep 2026 06:52:22 GMT
content-type: text/
html; charset=utf-8
server: cloudflare
last-modified: Mon, 28 Sep 2026
16:19:32 GMT
allow: GET, HEAD
accept-ranges: bytes
age: 1926
cf-cache-status: HIT
cf-ray: a4314dee2c8cf82e-CDG
```

```
<!doctype
html><html      lang=en><head><meta
charset=utf-8><link      rel=icon
href=data:,><meta      name=viewport
content="width=device-width,initial-
scale=1"><title>Example      Domain</
title><style>html{color:scheme:light
dark;background:light-
dark(#eee,#222)}
body{font:16px/1.6      system-ui,sans-
serif;max-width:30em;min-height:100vh;margin:auto;padding:
2em      20vh;box-sizing:border-
box;display:grid;place-
content:center;text-align:center}</
style></head><body><p>This      domain
is      for      use      in      documentation
```

```
examples without needing permission.
This is not a service, avoid
relying on it for testing
and monitoring purposes.</p><a
href=https://iana.org/help/example-
domains>Learn more</a><script src=/
s.js></script></body></html>
```

8 Exercice 5 — Construire une image applicative

En suivant la même approche, il est possible de créer une image contenant une application Python simple.

Créer une structure de dossiers `app/` avec une application serveur Web minimale en Python :

```
app
├─ Dockerfile
└─ app.py
```

1 directory, 2 files

L'application Python

```
1 from http.server import
SimpleHTTPRequestHandler, HTTPServer
2
3 server = HTTPServer(('0.0.0.0',
5000), SimpleHTTPRequestHandler)
4 print("Listening on 5000")
5 server.serve_forever()
```

Un fichier `Dockerfile` qui construit l'image de l'application Python en utilisant l'image officielle `python:3.11-slim` comme base, en copiant le fichier `app.py` dans le conteneur et en définissant la commande par défaut pour exécuter l'application.

```
1 FROM python:3.11-slim
2 WORKDIR /app
```

```
3 COPY app.py .
4 CMD ["python", "app.py"]
```

Construire l'image Docker **depuis le dossier `app/`** avec le tag `tp0-app:0.0.1` :

```
docker build --tag tp0-app:0.0.1 .
```

Exécuter le conteneur en mappant le port 5000 du conteneur (sur lequel l'application écoute) sur le port 5000 de l'hôte :

```
docker run --detach \
--name tp0-app \
--publish 5000:5000 \
tp0-app:0.0.1
```

```
ea14f0b80a990ec322e5b3c48ad0a5b880982c209c7d84bab30837b
```

Tester l'application avec `curl` depuis l'hôte :

```
curl --max-time 30 --include http://
localhost:5000
```

```
HTTP/1.0 200 OK
Server:
SimpleHTTP/0.6 Python/3.11.16
Date: Wed, 30 Sep 2026 06:52:34 GMT
Content-type: text/
html; charset=utf-8
Content-Length: 224
```

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Directory listing for /</
title>
</head>
<body>
<h1>Directory listing for /</h1>
```

```
<hr>
<ul>
<li><a href="app.py">app.py</a></li>
</ul>
<hr>
</body>
</html>
```

Nettoyer le conteneur (arrêt et suppression) :

```
docker stop tp0-app
docker rm tp0-app
```

```
tp0-app
tp0-app
```

9 Exercice 6 — (Optionnel) Réseaux Docker

Il est possible de faire fonctionner plusieurs conteneurs ensemble via un réseau Docker. Dans cet exercice, nous allons créer un réseau Docker et y connecter le conteneur de l'application Python et et autre conteneur `curl` pour tester l'application.

Créer un réseau (virtuel) Docker nommé `tp0-net` :

```
docker network create tp0-net
```

```
465a9236df759676c63ad81f5d2c665254bdb4f22
```

Lancer le conteneur de l'application Python dans ce réseau :

```
docker run --detach \
  --name tp0-app \
  --network tp0-net \
  tp0-app:0.0.1
```

```
ed8d7a9916023ce4d18ec16a54113aa87f122bd833ba755468efc07
```

Tester l'application avec un conteneur `curl` connecté au même réseau Docker, en utilisant le nom du conteneur `tp0-app` comme `host-name` pour le service :

```
#| echo: true
#| output: asis
docker run --rm \
  --network tp0-net \
  curlimages/curl --max-time 30 --
  include http://tp0-app:5000
```

Nettoyage :

! Rappel

Les conteneurs communiquent via des réseaux virtuels et des noms DNS internes

10 Synthèse

! Rappel

- Un conteneur n'est pas un mini-serveur, C'est un **processus isolé**, créé à partir d'une image
- Une **include image** est un **modèle immuable** pour créer des conteneurs à l'aide d'un **Dockerfile** qui décrit sa construction et peut être versionné.
- Les données doivent être stockées en dehors du conteneur via des **volumes**
- Les conteneurs communiquent via des **réseaux virtuels**

Les conteneurs permettent de déployer des applications complexes de manière reproductible et en isolant les différentes parties.