La gestion des ports

Redirection des ports

La plupart des applications dockerisées écoutent sur des ports spécifiques. Par exemple : serveurs web, bases de données, API, etc.

Pour les rendre accessibles depuis l'extérieur, **il faut rediriger ces ports** avec l'option `--port`, `-p`.
Cette option permet de lier un port du conteneur à un port de la machine hôte.

Dans le détail, lorsqu'une requête est envoyée à un port de la machine hôte, Docker la redirige automatiquement vers le port correspondant à l'intérieur du conteneur.

La syntaxe de `-p` est la suivante :

```
-p [IP:]PORT_HOTE:PORT_CONTENEUR
```

- **`PORT_CONTENEUR`** : le port sur lequel l'application à l'intérieur du conteneur écoute.
- **`PORT_HOTE`** : le port de la machine hôte sur lequel les requêtes externes seront reçues.
- **`IP` (facultative)** : permet de restreindre l'accès à une IP spécifique, par exemple `127.0.0.1`.

> La redirection de ports docker permet de répondre à des **cas d'usages fréquents** : contrôle d'accès aux applications, gestion des conflits de port avec des applications similaires, configuration d'architectures réseaux complexes, etc.

Exemples avec strm/helloworld-http

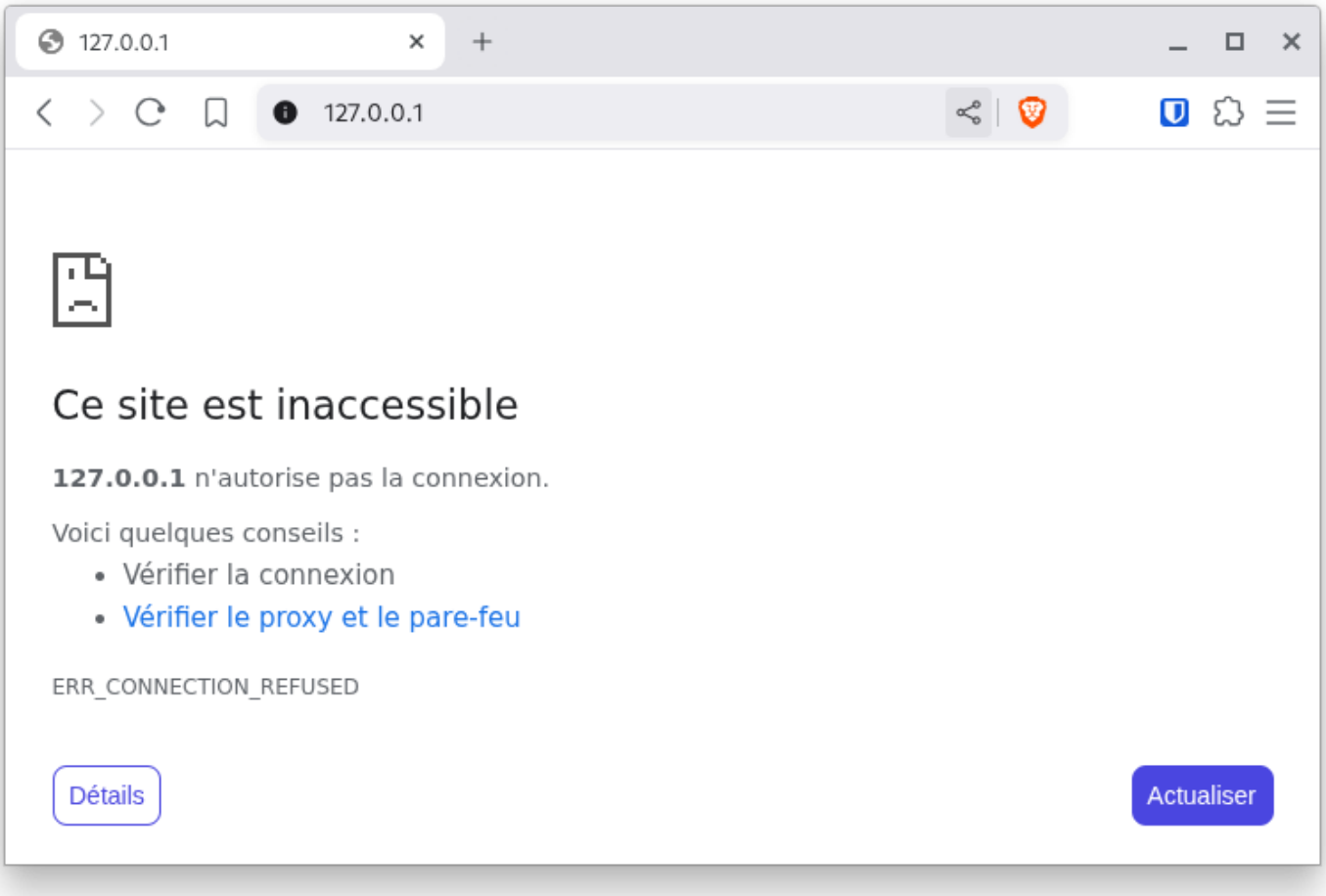
L'image strm/helloworld-http permet d'afficher un message test et écoute sur le port `80`, port `HTTP` par défaut.

Une fois lancé, un `docker ps -a` nous confirme que le port `80` du conteneur est ouvert.

```
debian@debian:~$ docker run -ti --rm --name=test_web_1 strm/helloworld-http
Serving HTTP on 0.0.0.0 port 80 ...
```

```
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND      CREATED          STATUS          PORTS     NAMES
0047600361d4   strm/helloworld-http   "/main.sh"   33 seconds ago   Up 32 seconds   80/tcp    test_web_1
debian@debian:~$
```

Néanmoins, le port `80` de la machine hôte **ne répond pas** :



> Le port `80` de la machine hôte **n'est pas automatiquement redirigé** vers celui du conteneur.

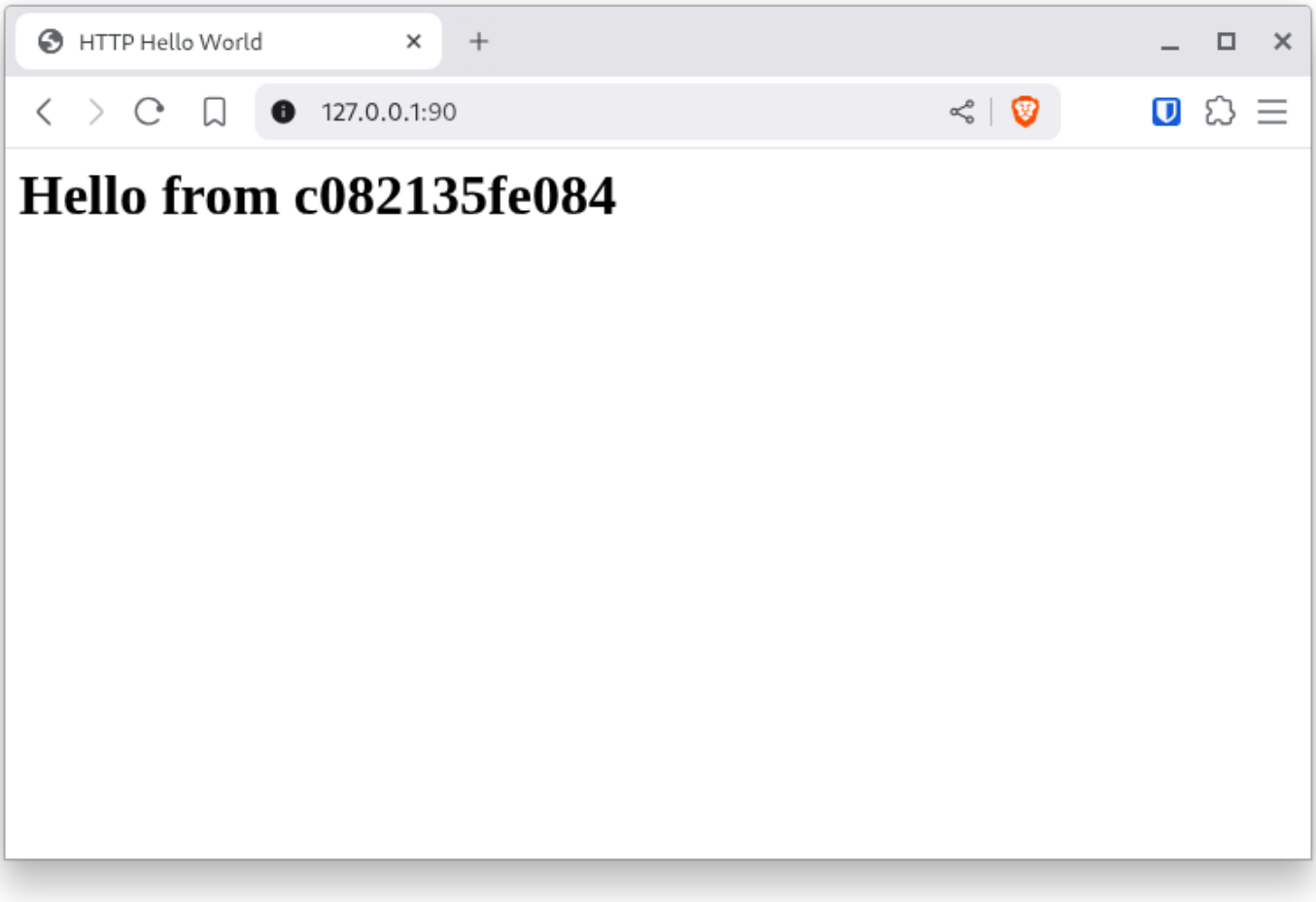
Il faut lancer le container avec l'argument `-p`. Il est tout à fait possible d'indiquer un autre port pour l'hôte, par exemple `90`.

```
debian@debian:~$ docker run -ti --rm -p 90:80 --name=test_web_2 strm/helloworld-http
Serving HTTP on 0.0.0.0 port 80 ...
```

Un `docker ps -a` nous confirme que le port `90` de la machine hôte est redirigé vers le port `80` du conteneur.

```
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE                 COMMAND      CREATED         STATUS         PORTS                                   NAMES
c082135fe084   strm/helloworld-http  "/main.sh"   3 seconds ago   Up 3 seconds   0.0.0.0:90->80/tcp, [::]:90->80/tcp   test_web_2
debian@debian:~$
```

La page web est accessible depuis `127.0.0.1:90`. Elle est également accessible depuis le LAN, toujours sur le port `90`.



Il est possible de n'écouter que depuis l'IP `127.0.0.1` avec `-p 127.0.0.1:90:80`.

```
debian@debian:~$ docker run -d -p 127.0.0.1:90:80 --name=test_web_3 strm/helloworld-http
53ec777e586ab6c00b200a5c64bf7aa69e5facc83402ffdb4ae57117bbb90238
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE                 COMMAND      CREATED         STATUS         PORTS                     NAMES
53ec777e586a   strm/helloworld-http  "/main.sh"   3 seconds ago   Up 3 seconds   127.0.0.1:90->80/tcp   test_web_3
debian@debian:~$
```

> Cela peut-être utile pour faire fonctionner une application derrière un reverse proxy (traefik, apache, nginx, etc.) afin de lui attribuer un nom de domaine.

Revision #14
Created 16 August 2025 18:32:21 by Thibaud FRICHET

Updated 29 August 2025 18:32:02 by Thibaud FRICHET