

# DÉPLOIEMENT D'UN SERVICE AVEC DOCKER

## ACTIVITÉ 2 – APPROFONDISSEMENT DANS L'EXPLOITATION DE DOCKER

### Liens :

- Lien vers une formation sur Docker complète mais très simple : <https://blog.microlinux.fr/formation-docker/>

Le document « récapitulatif des commandes Docker » vient en complément.

### L'objectif est ici de :

- comprendre la problématique de persistance des données lorsque l'on utilise la technologie Docker ;
- déployer un premier service qui va nous être utile dans le cadre de Docker : Portainer.

## SOMMAIRE

|                                                                |   |
|----------------------------------------------------------------|---|
| A. La problématique de la persistance des données.....         | 2 |
| B. Gestion de la persistance des données avec les volumes..... | 2 |
| 1. Créer un volume.....                                        | 2 |
| 2. Utiliser un volume dans la commande docker.....             | 3 |
| 3. Autres commandes principales sur les volumes.....           | 4 |
| 3.1 Lister les volumes.....                                    | 4 |
| 3.2 Inspecter un volume.....                                   | 4 |
| 3.3 Supprimer un volume.....                                   | 4 |
| C. Déploiement de Portainer, interface graphique à Docker..... | 5 |

Vous devez faire toutes les manipulations, notamment celles précédées par :



## A. LA PROBLÉMATIQUE DE LA PERSISTANCE DES DONNÉES

Les images servant de base au démarrage d'une instance de conteneur sont en lecture seule et toutes les modifications sur le conteneur sont réalisées dans une couche supplémentaire. Ainsi, les données ajoutées dans un conteneur disparaissent avec lui lors de sa destruction. Pour sauvegarder les modifications dans une autre image, il est nécessaire d'utiliser la commande *commit*.

Cette méthode est adaptée lorsque l'utilisateur est en phase de construction d'une image ou pour déployer un micro-service ne gérant pas de données, mais ne l'est pas pour l'exploitation d'une image existante quand il s'agit, par exemple, d'utiliser le conteneur pour un service en ligne où les données sont modifiées sans cesse ou bien s'il est important de conserver certaines données comme les traces (*logs*).

**Docker a prévu plusieurs mécanismes pour gérer les données persistantes**, nous mettrons en pratique celui le plus couramment utilisé et surtout celui adapté à la situation qui consiste à **utiliser un stockage externe** au conteneur (sur l'hôte ou sur n'importe quel autre support, typiquement des baies SAN ou NAS.) appelé **volume de données**.

**Un volume de conteneur** permet de conserver les données, même en cas de suppression d'un conteneur Docker. Il s'agit également d'une solution pratique pour l'échange de données entre l'hôte et le conteneur.

## B. GESTION DE LA PERSISTANCE DES DONNÉES AVEC LES VOLUMES

Un volume est un répertoire défini sur l'hôte pour y stocker les données utilisables par un conteneur sachant que ce stockage persistera même lorsque le conteneur sera supprimé.

Nous allons lancer un serveur Web avec Apache en y incorporant nos propres fichiers sources. Nous utilisons pour ce-là l'image officielle de Docker : [https://hub.docker.com/\\_/httpd](https://hub.docker.com/_/httpd)

Dans cette image, le répertoire de base par défaut d'Apache est `/usr/local/apache2/htdocs/` : ce sont les fichiers de ce dossier (qui contient les pages HTML du site) qu'il faut pouvoir modifier à volonté et qui ne doivent pas disparaître si le conteneur est supprimé ou remplacé, par exemple par un conteneur utilisant une image avec une version plus récente d'Apache.

**L'idée consiste ici à monter une arborescence de fichiers de l'hôte à l'intérieur du container.** Il est ainsi possible de publier un site Web dont les fichiers sources sont déposés **dans un dossier du serveur hôte** à travers un conteneur.



**Pour faire les choses proprement**, il est nécessaire de suivre les deux étapes suivantes.

### 1. CRÉER UN VOLUME

```
docker volume create <nom_du_volume>
```

 `user@servDockerAR:~$ docker volume create public-html`

Un dossier `public-html` contenant un sous-dossier vide « `_data` » a automatiquement été créé dans `/var/lib/docker/volumes/`.

```
user@servDockerAR:~$ ls -lR /var/lib/docker/volumes/
```

```
...
drwx-----x 3 root root      3 11 août  20:07 public-html

/var/lib/docker/volumes/public-html:
total 1
drwxr-xr-x 2 root root 2 11 août  20:07 _data

/var/lib/docker/volumes/public-html/_data:
total 0
```

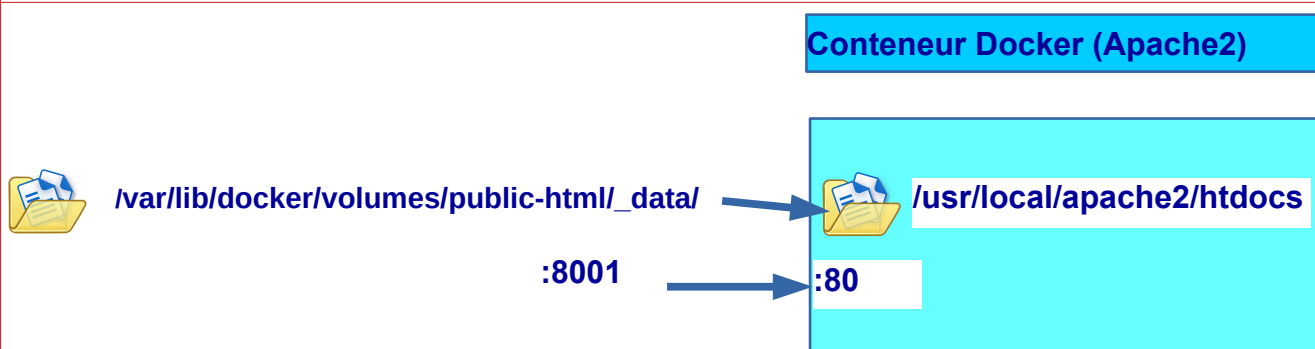
## 2. UTILISER UN VOLUME DANS LA COMMANDE DOCKER

Docker utilise pour cela le **paramètre -v** :

```
docker run --name <nom conteneur> -d -v <nom volume>:<données du conteneur> -p
<adresse IP hôte:port publié sur l'hôte:80 <nom image>
```

```
user@servDockerAR:~$ docker run --name servweb -d -v
public-html:/usr/local/apache2/htdocs -p 192.168.60.111:8001:80 httpd
```

### HÔTE DEBIAN



**i** Au moment de l'initialisation, si des fichiers sont présents dans le dossier du conteneur, ceux-ci sont copiés dans le dossier de l'hôte.

```
user@servDockerAR:~$ ls -l /var/lib/docker/volumes/public-html/_data/
```

```
total 1
```

```
-rw-r--r-- 1 501 staff 45 11 juin 2007 index.html
```

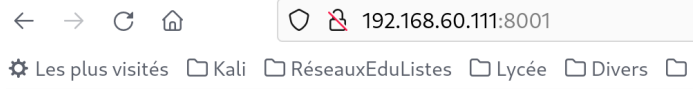
Il suffit donc maintenant de modifier dans `/var/lib/docker/volumes/public-html/_data/` de l'hôte le fichier `index.html` afin de le distinguer de la page par défaut d'Apache et de vérifier que c'est bien cette page qui est publiée.

**Avant modification du fichier index.html**  
de /var/lib/docker/volumes/public-html/\_data/



**It works!**


**Après modification du fichier index.html**  
de /var/lib/docker/volumes/public-html/\_data/



**It works! - Serveur Web AR**

Le conteneur peut maintenant être supprimé et recréé même avec une autre image : si dans la création du conteneur, on fait référence à ce volume via l'option -v, c'est cette page Web qui s'affichera.

 Il est possible de multiplier le paramètre -v autant de fois que nécessaire

 Il est possible de « mapper » n'importe quel dossier de l'hôte sans utiliser la commande docker volume. L'avantage principal d'utiliser cette commande est que les volumes sont connus de Docker et manipulables à travers des commandes.

### 3. AUTRES COMMANDES PRINCIPALES SUR LES VOLUMES

#### 3.1 Lister les volumes

 user@servDockerAR:~\$ docker volume ls

| DRIVER | VOLUME NAME |
|--------|-------------|
| local  | public-html |

#### 3.2 Inspecter un volume

 user@servDockerAR:~\$ docker volume inspect public-html

```
[
  {
    "CreatedAt": "2023-08-11T20:07:52+02:00",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/public-html/_data",
    "Name": "public-html",
    "Options": null,
    "Scope": "local"
  }
]
```

#### 3.3 Supprimer un volume

 user@servDockerAR:~\$ docker volume rm public-html

 Un volume associé à un conteneur en cours d'exécution ne peut pas être supprimé.

## C. DÉPLOIEMENT DE PORTAINER, INTERFACE GRAPHIQUE À DOCKER

Portainer permet de gérer via une interface web les conteneurs locaux ou distants. On trouve l'application sous forme d'image docker.

### Installation de Portainer

Lien officiel : <https://docs.portainer.io/start/install-ce/server/docker/linux>

**1. Création d'un volume** qui va permettre de conserver toutes les données de portainer sur l'espace disque de l'hôte.

```
user@servDockerAR:~$ docker volume create portainer_data
```

**Q1.** Donner le chemin complet du dossier de l'hôte qui va accueillir les données de Portainer

**Il sera ainsi possible de supprimer le conteneur pour le mettre à jour avec une version ultérieure de Portainer, car les données sont persistées sur l'hôte.**

### 2. Lancement du container

```
user@servDockerAR:~$ docker run -d -p 9443:9443 --name portainer --restart=always  
-v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-  
ce:latest
```

Cette commande exécute un conteneur Portainer CE (Portainer Community Edition) en arrière-plan (option -d), exposant ses interfaces via le port 9443 de l'hôte, avec la persistance des données de configuration via le volume créé précédemment et la possibilité de redémarrage automatique (--restart=always) en cas de besoin.

À noter que Portainer génère et utilise un certificat SSL auto-signé pour sécuriser le port 9443. Il est bien sûr possible d'utiliser un autre certificat.

**Nous voyons ici une autre utilisation de l'option -v :**

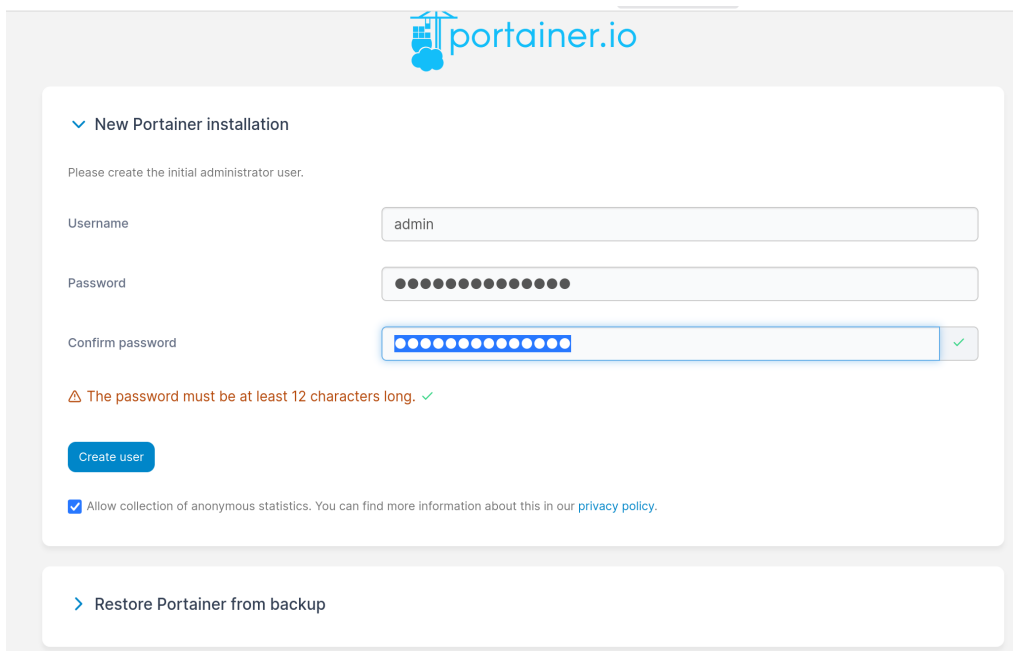
- **-v /var/run/docker.sock:/var/run/docker.sock** : monte le socket Docker de l'hôte dans le conteneur. Cela permet à Portainer d'interagir avec le moteur Docker de l'hôte et de gérer les conteneurs.

**Q2.** Expliquer précisément l'option -p 9443:9443

## Utilisation de Portainer

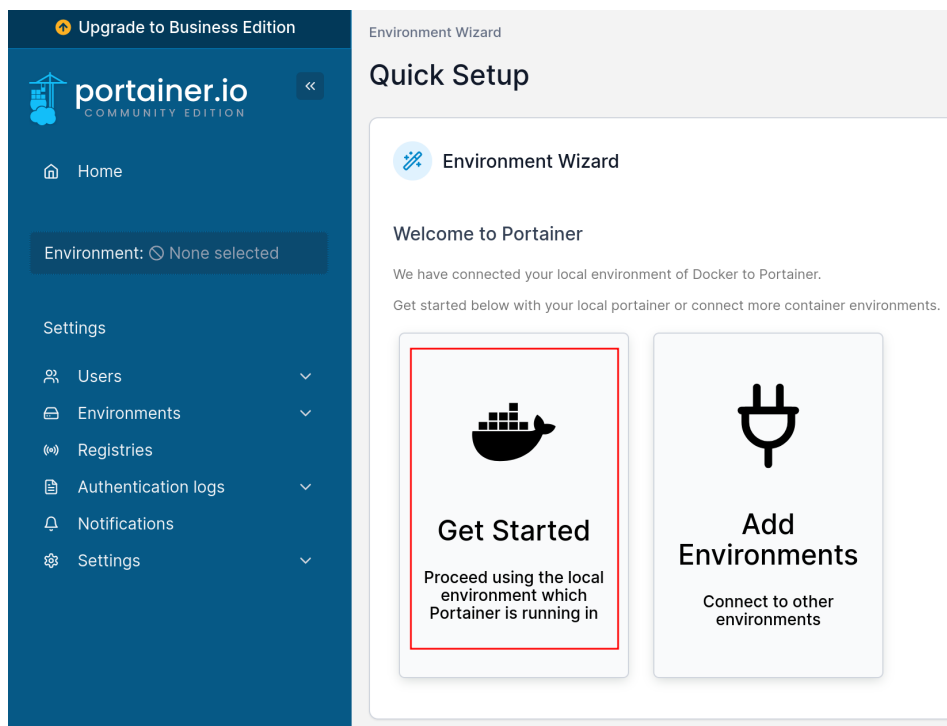
### Q3. Donner l'URL pour accéder à l'application Web

- Choisir un mot de passe (conseillé pour cette activité : portainerAdmin) pour le compte admin.



The screenshot shows the 'New Portainer installation' screen. At the top is the Portainer.io logo. Below it, a section titled 'New Portainer installation' contains the instruction 'Please create the initial administrator user.' There are three input fields: 'Username' with the value 'admin', 'Password' with masked characters, and 'Confirm password' with masked characters and a green checkmark. A message below the fields states 'The password must be at least 12 characters long.' with a green checkmark. A blue 'Create user' button is at the bottom left. At the bottom of the form, there is a checkbox for 'Allow collection of anonymous statistics' which is checked, with a link to the 'privacy policy'. Below the form is a button labeled '> Restore Portainer from backup'.

- Cliquer sur « Create user ».



The screenshot shows the 'Quick Setup' screen of Portainer. On the left is a dark blue sidebar with the Portainer.io logo and a list of menu items: Home, Environment: None selected, Settings, Users, Environments, Registries, Authentication logs, Notifications, and Settings. The main content area is titled 'Environment Wizard' and 'Quick Setup'. It says 'Welcome to Portainer' and 'We have connected your local environment of Docker to Portainer.' Below this, there are two large buttons: 'Get Started' with a ship icon and 'Add Environments' with a plug icon. The 'Get Started' button is highlighted with a red border. Below the 'Get Started' button, it says 'Proceed using the local environment which Portainer is running in'. Below the 'Add Environments' button, it says 'Connect to other environments'.

La vue suivante propose de gérer une instance locale ou distante de Docker.

➤ Sélectionner « Get Started » pour gérer une instance locale.

L'interface, très intuitive, permet d'avoir un aperçu global, de gérer les éléments existants (conteneurs, stacks, images, volumes, réseaux), de créer d'autres conteneurs via notamment des templates, etc. La partie administration permet de gérer les utilisateurs et les registres.

The screenshot shows the Portainer.io Dashboard for a local environment. The left sidebar contains navigation links: Home, local (selected), Dashboard, App Templates, Stacks, Containers, Images, Networks, Volumes, Events, Host, Settings, and Users. The main content area is titled 'Dashboard' and displays 'Environment info' for a local environment (2.1 GB, Standalone 24.0.5). Below this, there are summary cards for 0 Stacks, 6 Images (823.7 MB), 3 Containers (3 running, 0 healthy, 0 stopped, 0 unhealthy), 2 Volumes, and 3 Networks.

The screenshot shows the Portainer.io Home page. The left sidebar has links: Home (selected), Environment: None selected, Settings, Users, Environments, Registries, Authentication logs, Notifications, and Settings. The main content area is titled 'Home' and features a 'Latest News From Portainer' section with a notice about Portainer CE 2.18.4. Below the news, there is an 'Environments' section with a search bar and filters. A table lists the 'local' environment, which is 'Up' and shows details like 'Group: Unassigned', 'No tags', 'Local', '0 stacks', '3 containers', '2 volumes', '6 images', '1 CPU', and '2.1 GB RAM'. There are buttons for 'Live connect' and 'Disconnected'.

Par exemple, un clic sur « containers » affiche une liste détaillée des conteneurs avec leur état ainsi que des informations utiles (nom, image, IP, ports, etc.). Il est possible d'effectuer certaines actions sur ces conteneurs (le démarrer, le stopper, consulter les statistiques, les logs, accéder à une console, etc.) et d'ajouter un conteneur.

The screenshot shows the Portainer.io Container list page. The left sidebar has links: Home, local, Dashboard, App Templates, Stacks, and Containers (selected). The main content area is titled 'Container list' and displays a table of containers. The table has columns for Name, State, Filter, Quick Actions, Stack, Image, Created, IP Address, Published Ports, and Ownership. The containers listed are 'servssh', 'servweb', and 'portainer', all in a 'running' state.

| Name      | State   | Filter | Quick Actions | Stack | Image                         | Created             | IP Address | Published Ports | Ownership      |
|-----------|---------|--------|---------------|-------|-------------------------------|---------------------|------------|-----------------|----------------|
| servssh   | running |        | [Icons]       | -     | aporaf/ubuntu:ssh             | 2023-08-10 15:13:39 | 172.17.0.2 | 22222-22        | administrators |
| servweb   | running |        | [Icons]       | -     | httpd                         | 2023-08-11 20:45:36 | 172.17.0.3 | 8001-80         | administrators |
| portainer | running |        | [Icons]       | -     | portainer/portainer-ce:latest | 2023-08-12 01:03:02 | 172.17.0.4 | 9443-9443       | administrators |

Pour chaque conteneur, on retrouve donc des informations utiles ainsi qu'un accès aux statistiques détaillées, aux logs ainsi qu'à la console (via « Quick actions » ou en cliquant sur le conteneur voulu) :

Containers > servssh

### Container details

⚙️ Actions

▶ Start
 ◻ Stop
 🗑 Kill
 ↺ Restart
 ⏸ Pause
 ▶ Resume
 🗑 Remove
 ↺ Recreate
 📄 Duplicate/Edit

---

📦 Container status

|            |                                                                  |
|------------|------------------------------------------------------------------|
| ID         | 6e5414e11795030fe0d2be13577adb12be9908a5922e9a1e75a9ac6b549e5803 |
| Name       | servssh <a href="#">🔗</a>                                        |
| IP address | 172.17.0.2                                                       |
| Status     | 🟢 Running for 15 hours                                           |
| Created    | 2023-08-10 15:13:39                                              |
| Start time | 2023-08-11 10:41:08                                              |

Container webhook 🔕 ☐ 🏢 Business Edition Feature

[📖 Logs](#)
[🔍 Inspect](#)
[📊 Stats](#)
[📄 Console](#)
[🔗 Attach](#)

Les statistiques sont en temps réel et permettent de suivre le CPU, la mémoire, l'utilisation du réseau ainsi que les processus en cours d'exécution :

Containers > servssh > Stats

### Container statistics

🔔
🔄
👤 admin
▼

🔍 About statistics

This view displays real-time statistics about the container **servssh** as well as a list of the running processes inside this container.

Refresh rate:

📊 Memory usage

📊 CPU usage

📊 Network usage (aggregate)

📊 I/O usage (aggregate)

☰ Processes

| UID  | PID | PPID | C | STIME  | TTY | TIME     | CMD                                                     |
|------|-----|------|---|--------|-----|----------|---------------------------------------------------------|
| root | 521 | 501  | 0 | ao0t11 | ?   | 00:00:00 | sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups |

Toutes les autres possibilités et fonctionnalités sont intéressantes, je vous les laisse découvrir.