Docker

Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels lancée en 2013.

Présentation

Qu’est-ce que Docker ?

Docker est un outil qui permet de **faire tourner des applications dans des conteneurs**. Un **conteneur** est un environnement léger, isolé, qui embarque tout ce qu’il faut pour faire fonctionner un programme : ses fichiers, ses dépendances, sa configuration, etc.

L’idée de Docker, c’est de **standardiser** et **simplifier** le déploiement d’applications. Peu importe le système d’exploitation de la machine (Debian, Ubuntu, Arch, etc.), tant que Docker est installé, on peut exécuter n’importe quel conteneur, toujours de la même manière.

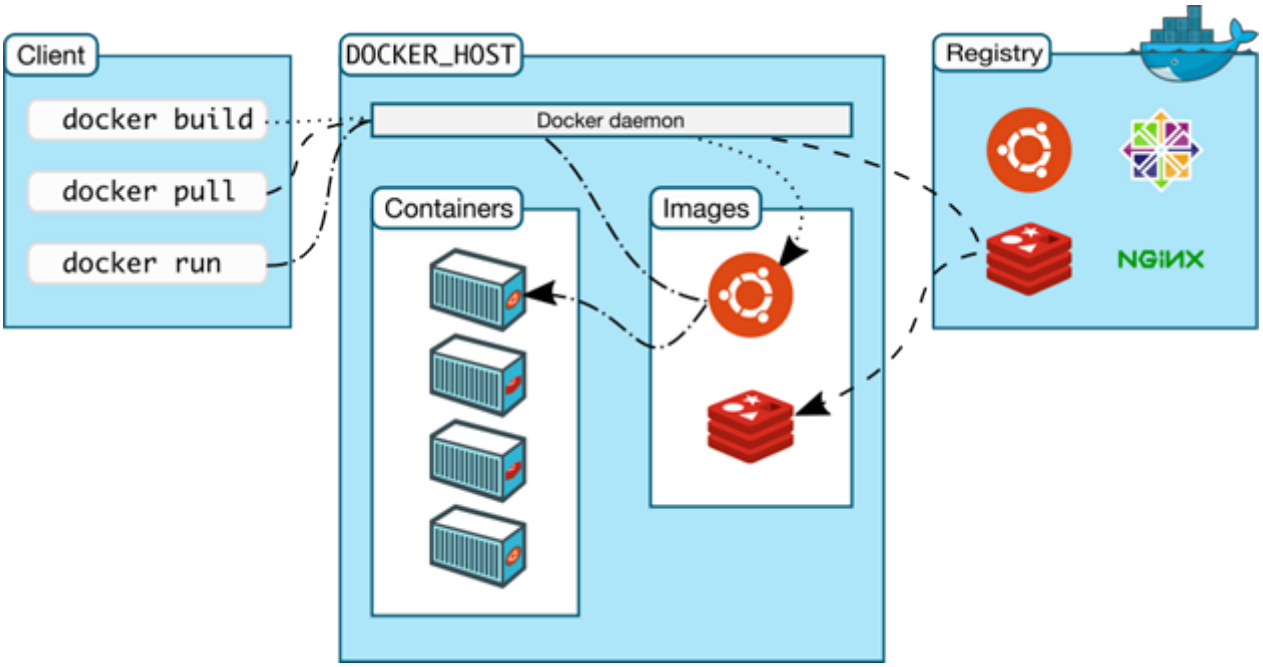
C’est une sorte de **boîte hermétique** pour les applications, qui évite les conflits et facilite les mises à jour.

Comment ça fonctionne ?

Un conteneur Docker est basé sur une **image**. Cette image est comme une recette qui dit : "Voici les fichiers à utiliser, voici comment démarrer l’application, et voici les ports à exposer." Une fois l’image téléchargée ou créée, Docker **exécute** un conteneur à partir de cette image.

Chaque conteneur est **isolé**, mais utilise le **noyau du système hôte**, ce qui le rend **plus léger** et **plus rapide** qu’une machine virtuelle. Docker ne virtualise pas tout un système d’exploitation, seulement ce qui est nécessaire pour faire tourner le service demandé.

Les conteneurs peuvent aussi partager des fichiers, des ports ou des volumes avec le système hôte, ce qui les rend très **flexibles** et **faciles à intégrer** dans un environnement existant.



Pourquoi utiliser Docker ?

Docker est très utilisé dans le monde professionnel et chez les particuliers, car il :

- **Évite les conflits** entre versions de logiciels.
- **Simplifie les déploiements** : une seule commande permet d'installer et de lancer une application complète.
- **Facilite les sauvegardes** et les restaurations : on peut exporter un conteneur et le redéployer ailleurs.
- **Assure une portabilité maximale** : on peut faire tourner la même application sur un PC, un VPS, un NAS, un Raspberry Pi…
- **Offre un excellent support communautaire** : des milliers d'images prêtes à l'emploi sont disponibles sur Docker Hub.

Cas d'usage concrets

En auto-hébergement, Docker est parfait pour faire tourner des services comme :

- Nextcloud (cloud personnel),
- Immich (photos),
- AdGuard Home (filtrage DNS),
- Vaultwarden (gestion de mots de passe),
- Grafana + Prometheus (supervision),
- Et bien d'autres...

Chaque service peut tourner dans son propre conteneur, avec sa configuration indépendante, sans interférer avec les autres.

En résumé

Docker est une **solution simple, rapide et efficace** pour déployer des applications. Grâce aux conteneurs, il permet de **standardiser l'environnement**, **isoler les services**, et **automatiser les déploiements**, sans pour autant complexifier la gestion du serveur.

C'est un outil devenu incontournable pour les administrateurs, les développeurs, et tous ceux qui veulent **héberger leurs services en toute autonomie**, avec une grande souplesse.

Installation

Prérequis :

- Vérifier les règles de firewall
- OS 64bits :
  - Debian 12
  - Debian 11
- Désinstaller anciennes versions

Installation :

Ajout des repo Docker :

```
# AJout des clés GPG:
apt-get update
apt-get install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

# Ajout des repo aux sources apt:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
```

Installation des paquets Docker :

```
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Vérifier l'installation :

```
docker run hello-world
```

Docker est installé !

Commandes utiles

Gestion des Containers

- **`docker run <image>`**
  Crée et exécute un container basé sur une image donnée.
  Exemple : `docker run -d --name my-container nginx`
- **`docker ps`**
  Liste les containers en cours d'exécution.
  Exemple : `docker ps` pour afficher les containers en fonctionnement.
- **`docker ps -a`**
  Liste tous les containers (en cours d'exécution ou arrêtés).
  Exemple : `docker ps -a`
- **`docker start <container>`**
  Démarre un container arrêté.
  Exemple : `docker start my-container`
- **`docker stop <container>`**
  Arrête un container en cours d'exécution.
  Exemple : `docker stop my-container`
- **`docker restart <container>`**
  Redémarre un container.
  Exemple : `docker restart my-container`
- **`docker exec -it <container> <command>`**
  Exécute une commande dans un container en cours d'exécution. Par exemple, pour ouvrir un terminal interactif (bash) :
  Exemple : `docker exec -it my-container bash`
- **`docker logs <container>`**
  Affiche les logs d'un container.
  Exemple : `docker logs my-container`
- **`docker rm <container>`**
  Supprime un container arrêté.
  Exemple : `docker rm my-container`

2. **Gestion des Images**

- **`docker pull <image>`**
  Télécharge une image depuis Docker Hub ou un autre registre.
  Exemple : `docker pull nginx`
- **`docker build -t <image_name> .`**
  Crée une image Docker à partir d'un `Dockerfile` situé dans le répertoire courant.
  Exemple : `docker build -t my-image .`
- **`docker images`**
  Liste toutes les images locales.
  Exemple : `docker images`
- **`docker rmi <image>`**
  Supprime une image locale.
  Exemple : `docker rmi my-image`

3. **Gestion des Volumes**

- **`docker volume create <volume_name>`**
  Crée un volume Docker.
  Exemple : `docker volume create my-volume`
- **`docker volume ls`**
  Liste tous les volumes Docker.
  Exemple : `docker volume ls`
- **`docker volume inspect <volume_name>`**
  Affiche des informations détaillées sur un volume.
  Exemple : `docker volume inspect my-volume`
- **`docker volume rm <volume_name>`**
  Supprime un volume Docker.
  Exemple : `docker volume rm my-volume`

4. **Réseaux Docker**

- **`docker network ls`**
  Liste tous les réseaux Docker.
  Exemple : `docker network ls`
- **`docker network create <network_name>`**
  Crée un réseau Docker.
  Exemple : `docker network create my-network`
- **`docker network inspect <network_name>`**
  Affiche des informations détaillées sur un réseau Docker.
  Exemple : `docker network inspect my-network`
- **`docker network rm <network_name>`**
  Supprime un réseau Docker.

Exemple : `docker network rm my-network`

5. **Gestion des Docker Compose**

- **`docker-compose up`**
  Démarre les services définis dans un fichier `docker-compose.yml`.
  Exemple : `docker-compose up`
- **`docker-compose down`**
  Arrête et supprime les services définis dans un fichier `docker-compose.yml`.
  Exemple : `docker-compose down`
- **`docker-compose ps`**
  Affiche l'état des services Docker Compose.
  Exemple : `docker-compose ps`
- **`docker-compose logs`**
  Affiche les logs des services Docker Compose.
  Exemple : `docker-compose logs`

6. **Autres Commandes Utiles**

- **`docker info`**
  Affiche des informations détaillées sur l'état du moteur Docker.
  Exemple : `docker info`
- **`docker stats`**
  Affiche l'utilisation des ressources pour chaque container en cours d'exécution.
  Exemple : `docker stats`
- **`docker version`**
  Affiche la version de Docker installée.
  Exemple : `docker version`

Portainer

Présentation :

Portainer est une interface utilisateur Web open source pour Docker qui facilite la gestion des conteneurs, images, volumes et réseaux.
Il fournit une interface graphique intuitive qui simplifie le déploiement, la gestion et la surveillance des conteneurs et des services Docker.

Déploiement :

Création du volume de portainer server, qui contiendra la BDD :

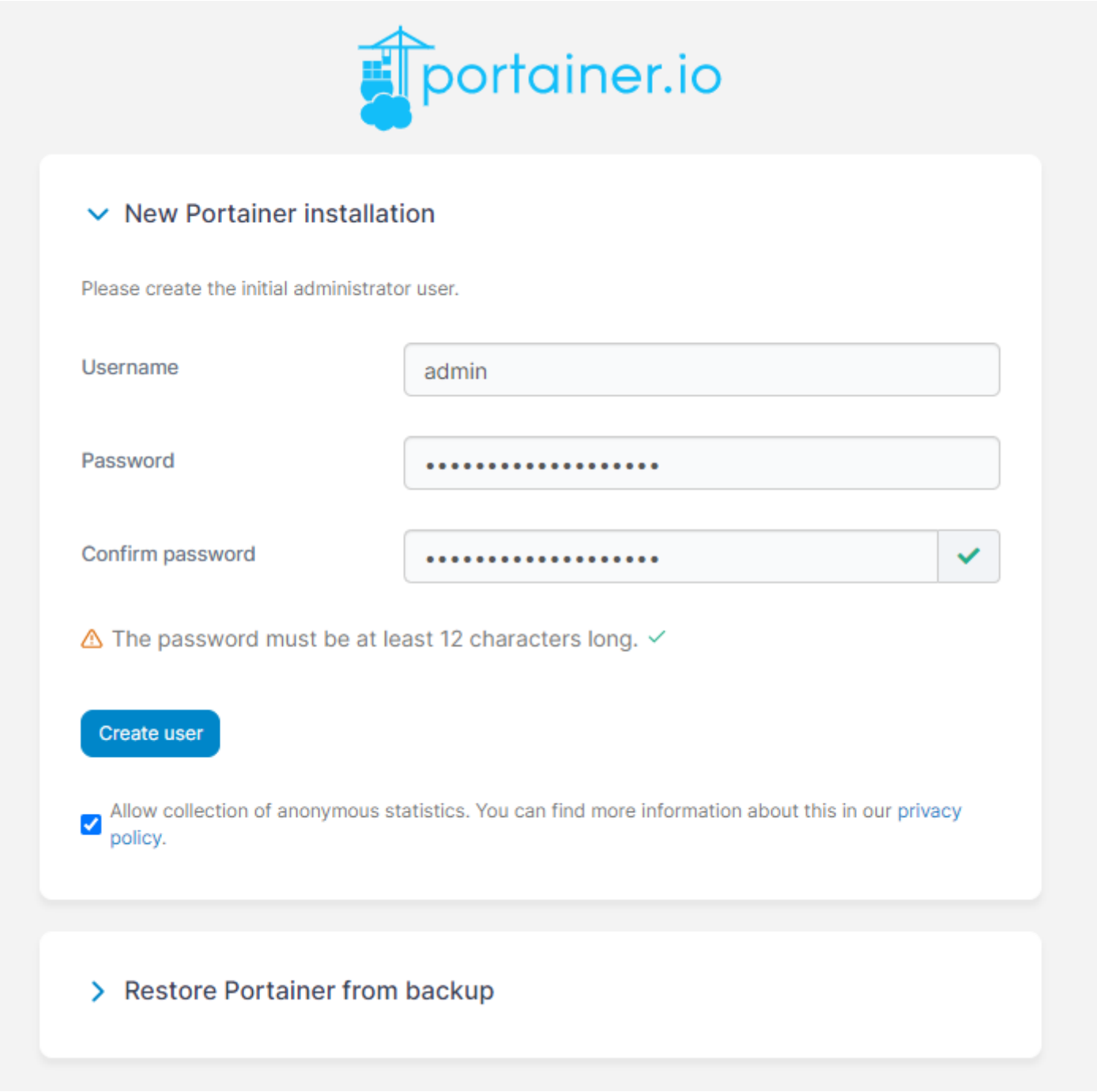
```
docker volume create portainer_data
```

Création du container portainer server :

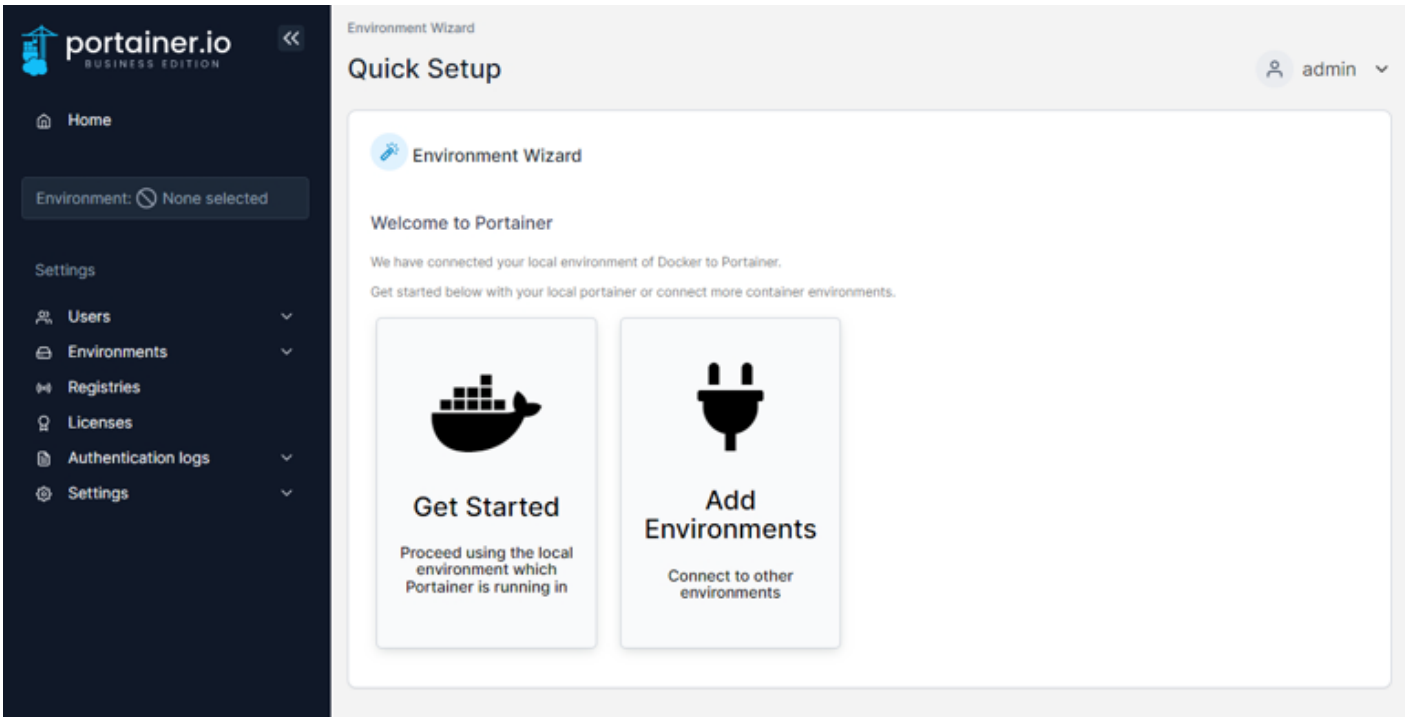
```
docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:lts
```

Interface Web :

Se connecter via https://<IP>:9443 :



Une fois le compte admin et son mot de passe défini, on peut valider. On est alors redirigé vers cette page :


portainer.io
BUSINESS EDITION
Home
Environment: None selected
Settings
Users
Environments
Registries
Licenses
Authentication logs
Settings
Environment Wizard
Quick Setup
admin
Environment Wizard
Welcome to Portainer
We have connected your local environment of Docker to Portainer.
Get started below with your local portainer or connect more container environments.
Get Started
Proceed using the local environment which Portainer is running in
Add Environments
Connect to other environments