CONFIGURATION DE PORTAINER SOUS RASPBERRY PI

Raspberry Pi - Debian Bullseye
Configuration avancée

Tutoriel **DOCKER** - RASPBERRY PI

David GOÏTRÉ

Table des matières

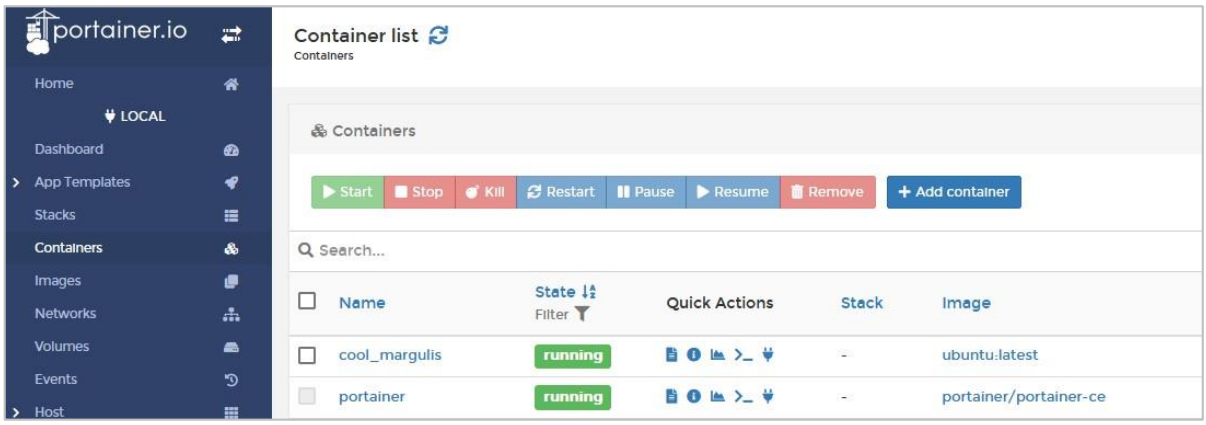
Introduction

PORTAINER est une interface utilisateur de gestion légère qui nous permet de gérer facilement vos différents **environnements Docker** (hôtes Docker ou clusters Swarm). Portainer se veut aussi simple à déployer qu'à utiliser. Il se compose d'un seul conteneur qui peut s'exécuter sur n'importe quel moteur Docker (peut être déployé en tant que conteneur Linux ou conteneur natif Windows, prend également en charge d'autres plates-formes). Portainer nous permet de gérer toutes nos ressources Docker (conteneurs, images, volumes, réseaux et plus!). Il est compatible avec le moteur Docker autonome et avec le mode Docker Swarm. Dans ce tutorial, on travaille sur **PORTAINER 2.xx**.

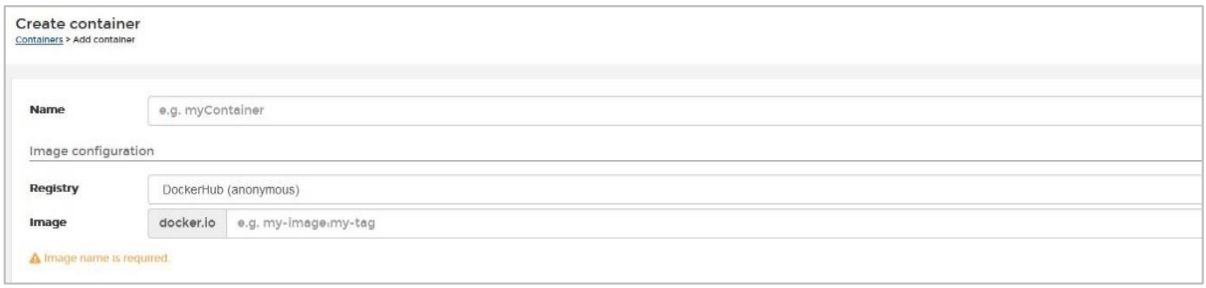
1. Création d'un conteneur

Un conteneur enveloppe l'application d'un logiciel dans une boîte invisible avec tout ce dont il a besoin pour s'exécuter. Cela comprend le système d'exploitation, le code de l'application, le runtime, les outils système et les librairies. Les conteneurs Docker sont construits à partir des images Docker. Ils sont légers, portables et permettent aux développeurs de créer, déployer et exécuter efficacement des applications distribuées. En outre, ils permettent à une application d'être empaquetée et déplacée facilement, augmentant ainsi la simplicité d'une infrastructure.

a) Cliquer sur le bouton **Container,** puis sur le bouton **add container**



b) Saisir les différents paramétrage du conteneur

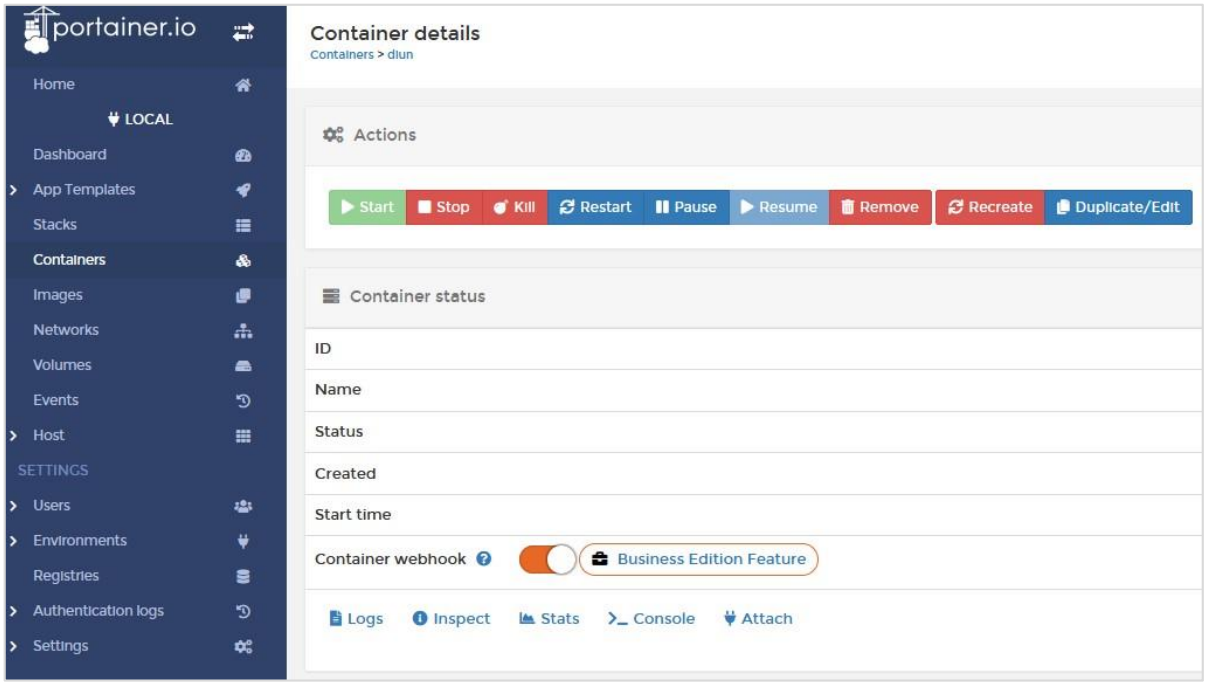


- Choisir un nom pour notre conteneur
- Choisir le nom de l'image présente sur le Docker hub (on peut créer son propre hub)
- Télécharger automatiquement l'image si non dispo en local
- Mapper automatique un port random de l'hôte vers un port utilisé par le service du conteneur
- Activer des ACL permettant de restreindre le management du conteneur à un groupe particulier
- Supprimer automatiquement le conteneur s'il s'agit d'un simple test
- Cliquer sur le bouton **Deploy**

2. Mise à jour d'un conteneur

Portainer permet via son **interface Containers** de gérer nos conteneurs. On peut facilement mettre à jour un conteneur. **Attention faire une sauvegarde du conteneur, avant toute opération**.

a) Cliquer sur **Containers**, pour afficher tous les conteneurs
b) Cliquer sur un conteneur pour l'éditer
c) Cliquer sur le bouton **Recreate**, pour lancer la mise à jour



d) Une fois la mise à jour effectué, cliquer sur le bouton **Restart**

3. Désinstallation d'un conteneur

Comme pour la mise à jour, on peut supprimer le conteneur en cliquant sur le bouton **Remove**, mais on peut aussi le faire via **Docker**.

a) Désinstallation d'un container

```
$ sudo docker ps -a  # liste les contenus
$ sudo docker rm -f container_id  #supprime l'agent correspondant au container_id
```

4. Sauvegarde et restauration d'un conteneur en local

Docker propose également un moyen simple et efficace pour sauvegarder et restaurer un conteneur. Mais évidemment de manière manuelle.

a) Sauvegarde d'un container

```
$ sudo docker commit -p [container-id] backup01  # créer la sauvegarde
$ sudo docker save -o backup01.tar backup01  # compresse la sauvegarde
```

b) Restaurer une sauvegarde d'un container avec la **même configuration de port** que l'original

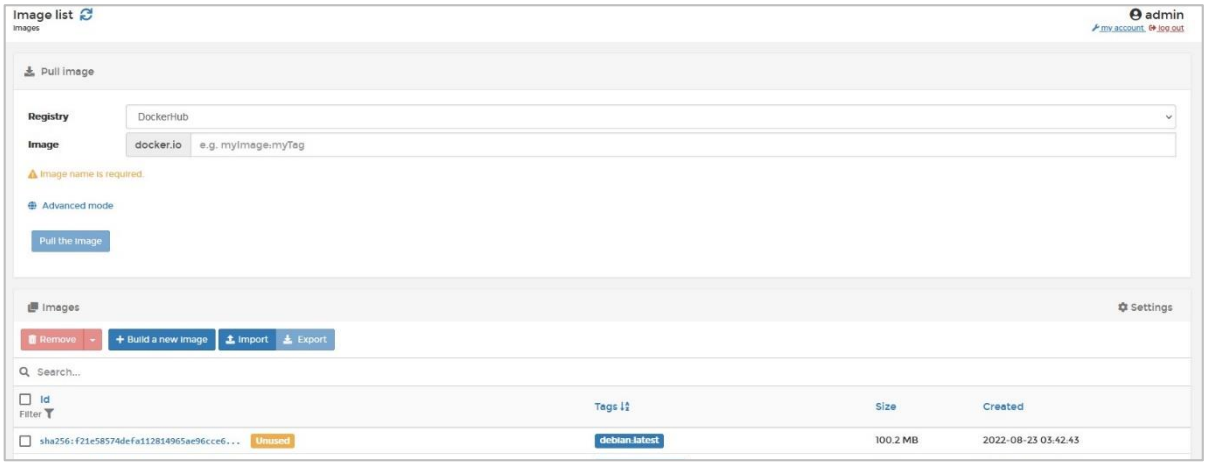
```
$ sudo docker load -i chemin/backup01.tar  # restaure le container
$ sudo docker run -it -p port:port backup01  # instancie l'image
```

5. Création d'une image

Une **image Docker** constitue un modèle immuable qui peut être employé de manière répétée pour créer des conteneurs Docker. L'image contient toutes les informations et dépendances requises pour exécuter un conteneur, y compris toutes les bibliothèques de programme basiques et interfaces utilisateur. Un environnement de ligne de commande **shell** et une implémentation de la **bibliothèque C** standard sont en général présents**.**

Au lieu de virtualiser un ordinateur virtuel (machine) avec son propre système d'exploitation, une image Docker est en général constituée d'une seule application. Il peut s'agir d'un fichier binaire individuel ou d'une combinaison de plusieurs composants logiciel.

a) Cliquer sur le bouton **Images**, puis sur le bouton **Add images**



b) Dans le **Registry** docker.io est sélectionné par défaut, cliquer dessus pour choisir un Hub local

c) Saisir le nom d'une image (ex : debian), ou cliquer sur le bouton **Search**

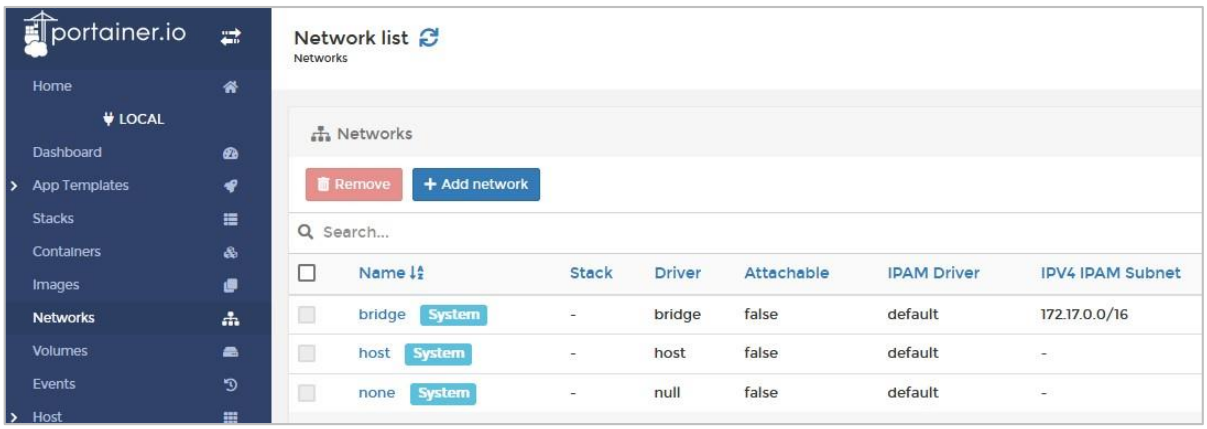
d) Saisir un nom pour l'image (ex : srv-debian)

e) Cliquer sur le bouton **Pull the image**

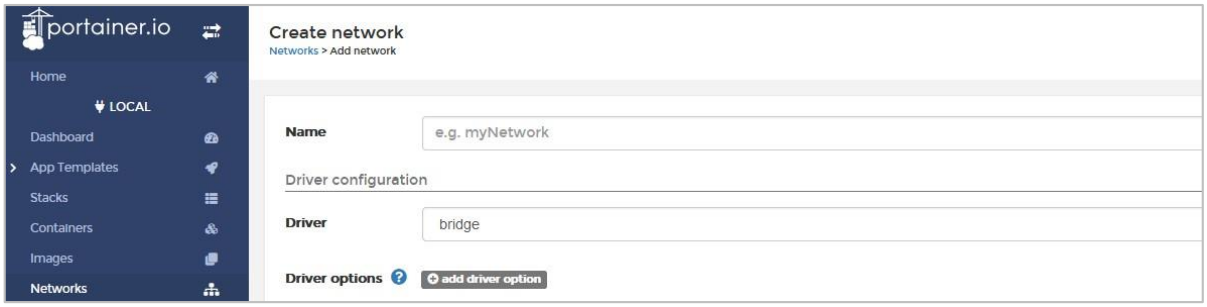
6. Création d'un réseau

Un conteneur enveloppe l'application d'un logiciel dans une boîte invisible avec tout ce dont il a besoin pour s'exécuter. Cela comprend le système d'exploitation, le code de l'application

a) Cliquer sur le bouton **Networks**, puis sur le bouton **Add network**



b) Saisir **un nom** pour le réseau et choisir le **type** (bridge, ipvlan…)



c) Les autres options consiste à remplacer le réseau par défaut (bridge), par un autre, du même type, et mettre tous nos conteneurs dans celui-ci.

7. Mappage d'un port sur un container existant

Il est parfois utile de mapper un port différent sur un container. A partir du Serveur Docker, on arrête le container et on mappe avec le nouveau port (**portlocal:portcontainer**). **Attention cette action créer un nouveau conteneur.**

a) Mapper un port sur le container Ubuntu

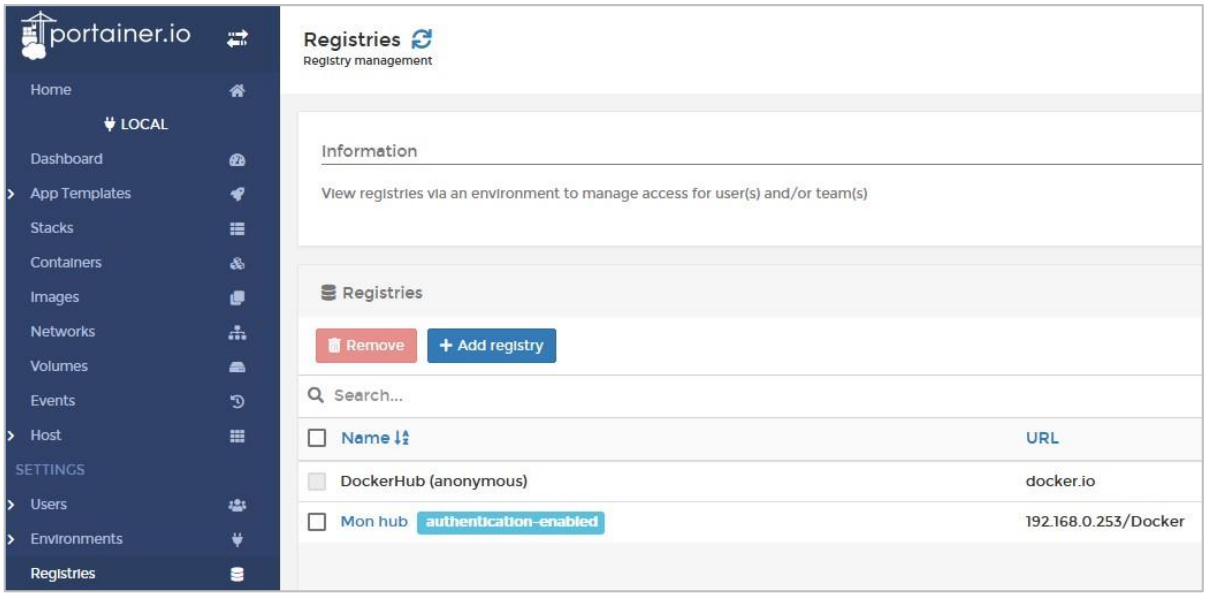
```
$ sudo docker stop <nomdel'image>
$ docker run -d -p 9002:9002 <nomdel'image>
```

8. Création d'un hub

Un Hub est un endroit de stockage des images pour docker. Deux possibilités s'offrent à nous. On peut choisir soit :

- en ligne Docker.io
- en local sur un serveur secondaire par exemple (disponible uniquement sur la version Business)

a) Cliquer sur le bouton **Registries**, puis sur le bouton **Add registry**



9. Création d'un stack

Les Stacks sont un ensemble de conteneurs déployés via un fichier docker-compose. Le fichier peut-être soit copier/coller et éditer directement dans l'éditeur de Portainer, ou Uploader. Cela permet de déployer rapidement un conteneur (ou un ensemble de conteneurs). L'édition est également facilitée.

a) Cliquer sur le bouton **Stacks**, puis sur le bouton **Add stack**

b) Saisir un **nom en minuscule uniquement** pour le stack

c) Construire le stack via l'une des méthodes suivantes :

- **Web editor**, pour créer le stack manuellement
- **Upload** pour télécharger un stack en local
- **Repository** pour
- **Custom template** pour le personnalisé

d) Remplir les différentes options selon la méthode, puis clique sur le bouton **Deploy the stack**

10. Déploiement via la méthode Stack

Nous allons ici déployer, pour l'exemple un stack **Diun**. Ce logiciel permet de gérer les mises à jour des différents containers existants.

a) Ouvrir Portainer est créer un **Stack** :

```
version: "3.5"
  services:
    diun:
      image: crazymax/diun:latest
      container_name: diun
      command: serve
      volumes:
        - "./data:/data"
        - "/var/run/docker.sock:/var/run/docker.sock"
      environment:
        - "TZ=Europe/Paris"
        - "LOG_LEVEL=info"
        - "LOG_JSON=false"
        - "DIUN_WATCH_WORKERS=20"
        - "DIUN_WATCH_SCHEDULE=0 */6 * * *"
        - "DIUN_PROVIDERS_DOCKER=true"
      labels:
        - "diun.enable=true"
      restart: always
```

b) Bien respecter l'indentation du script , sinon cela génèrera une erreur lors du déploiement
c) Cliquer sur le bouton **Deploy the stack** (le déploiement du stack créera automatiquement un conteneur associé).

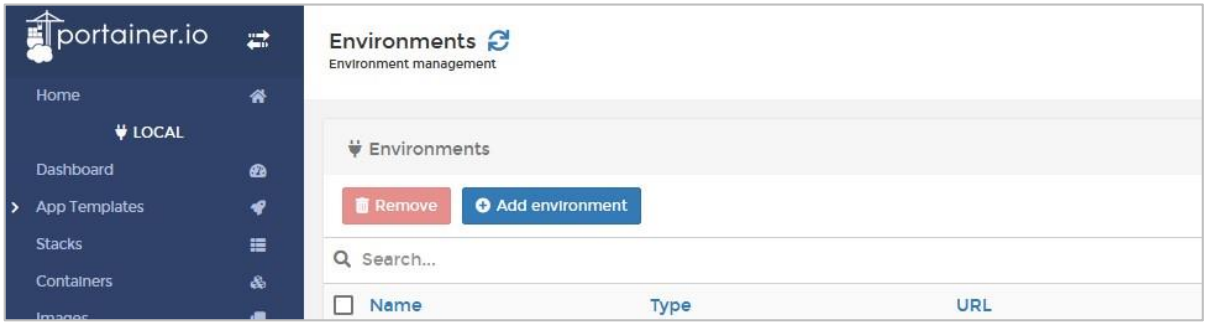
11. Installation de Portainer Agent

Portainer Agent permet de gérer nos conteneurs Docker de plusieurs machines depuis une seule instance Portainer via un poste client.

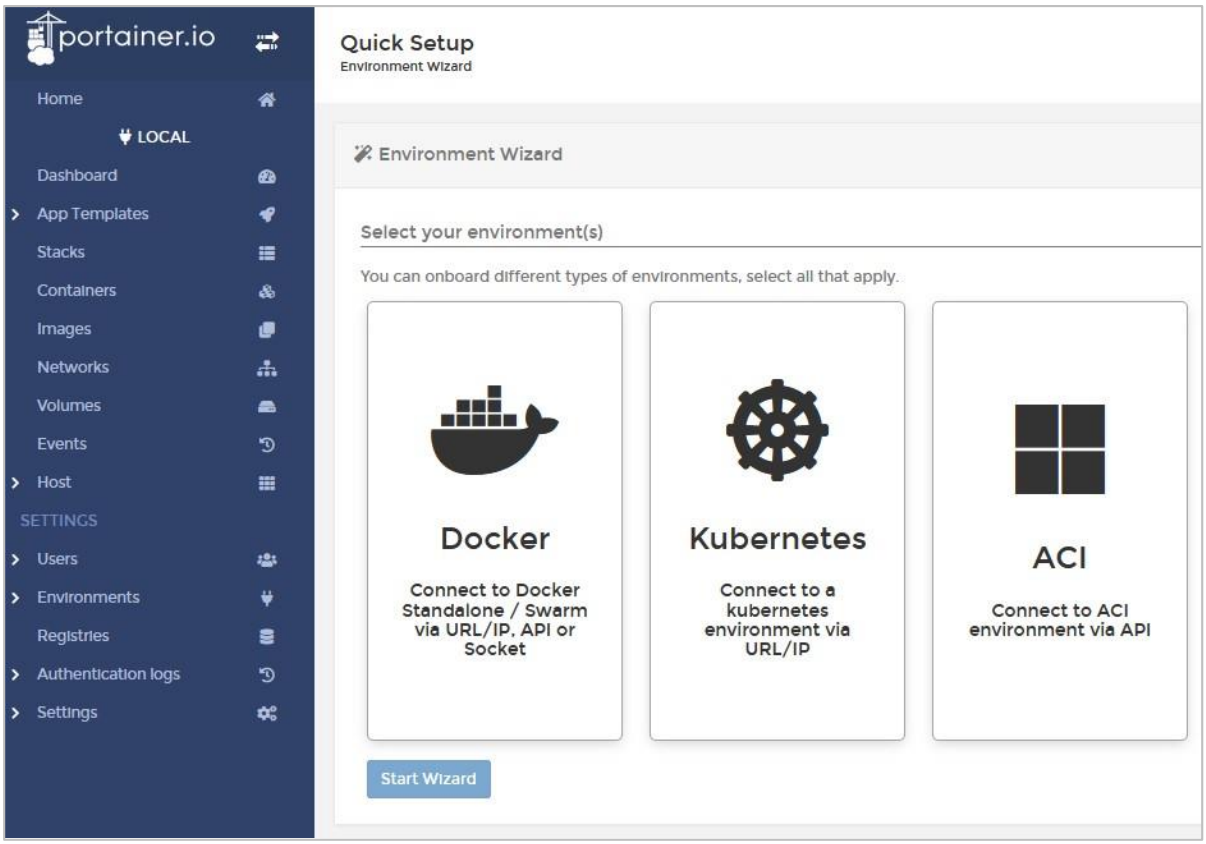
a) Installation de **Portainer Agent**

```
$ sudo docker run -d -p 9001:9001 --name portainer_agent --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v
/var/lib/docker/volumes:/var/lib/docker/volumes portainer/agent:2.11.1
```

b) Cliquer sur le bouton **Environments (Endpoints)**, puis sur le bouton **Add environment**

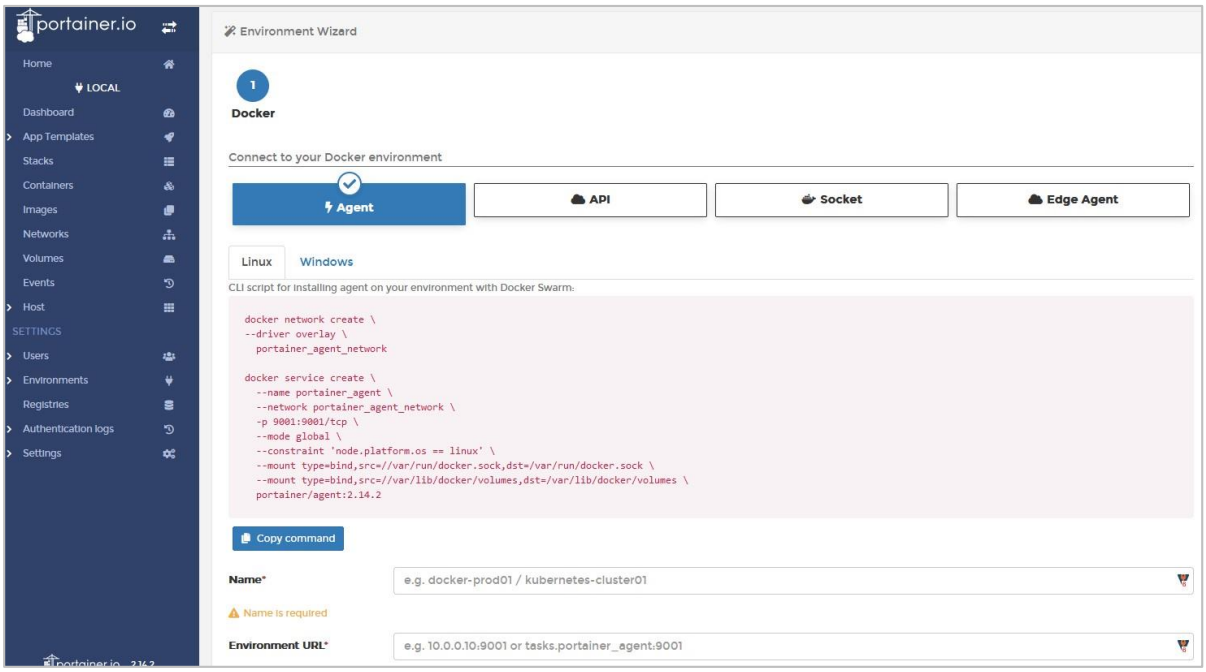


c) Cliquer sur le bouton **Docker**, puis sur le bouton **Start Wizard**



d) Saisir un nom d'Environnement en minuscule : **portainer-agent**

e) Saisir une URL d'environnement : **@IPserveur:9001** puis cliquer sur le bouton **Connect**



f) Cliquer sur le bouton **Finish**

12. Commandes RaspberryPi

Liste des commandes basiques à la gestion du serveur RaspberryPi Docker

```
# sudo chown <user> <nomfile>  # changer le propriétaire d'un fichier
# sudo chown -R user:group </nomdossier>  # changer le propriétaire d'un dossier
# sudo usermod -aG docker <nomuser>  # ajouter un utilisateur au groupe docker
# sudo useradd -u 1500 <nomuser>  # créer un utilisateur avec un UID
# sudo groupadd –g 1010 <nomdugroupe>  # créer un groupe avec un GID
```

13. Liens annexes

Liste de contenu à télécharger pour Docker et Portainer

- Docker et les containers
- Docker et portainer
- Documentation portainer
- Outil de surveillance des mises à jour Diun
- Outil de surveillance des mises à jour Watchtower
- Logiciel de sauvegarde Serveurs
- Installation Linux swag via docker compose
- Linux swag

14. Conclusion

PORTAINER est installé et configuré avec succès sur le serveur **Docker**. On peut désormais créer des de multiples environnements Docker.

Destiné au RaspberryPi (Raspbian) avec Docker, **PORTAINER** peut parfaitement être installer sur différentes plate-formes (NUC, NAS, PC...)