D O C U M E N T A T I O N T E C H N I Q U E

Docker

Installation, Configuration et Mise en Production

BTS SIO – Services Informatiques aux Organisations
Environnement : Debian / Linux

1. Présentation de Docker

Docker est un logiciel libre destiné aux développeurs et administrateurs systèmes, dont l'objectif est de faciliter le développement, la diffusion et le déploiement d'applications web autonomes.

Son principal intérêt est d'assembler les briques d'une application en conteneurs pouvant être partagés sous forme d'images, et de les exécuter indépendamment de la plateforme et de l'environnement cible.

Origines

Docker a été initialement développé par Solomon Hykes pour un projet interne à la société dotCloud, une plateforme PaaS (Platform as a Service) fondée en 2008. Le projet a été distribué en open source en 2013 et a rapidement fédéré une communauté très active.

Principe de fonctionnement

Docker utilise des fonctionnalités natives du noyau Linux et fournit des outils simplifiés permettant :

- La duplication et la suppression de conteneurs
- L'accessibilité des conteneurs via les API et CLI
- La migration de conteneurs (à froid ou à chaud)

2. Installation de Docker

Avant d'installer Docker, il convient de vérifier qu'aucune ancienne version n'est déjà présente sur le système.

2.1 Commandes d'installation

```
apt update
wget https://get.docker.com/
```

Figure 1 – Commandes de mise à jour et téléchargement du script Docker

> **apt update** : met à jour la liste des paquets disponibles depuis les dépôts configurés. Cette commande ne met pas à jour les paquets eux-mêmes, mais s'assure que le système connaît les dernières versions disponibles.
>
> **wget https://get.docker.com/** : télécharge le script officiel d'installation de Docker. Ce script détecte automatiquement la distribution Linux et configure les dépôts Docker, les clés GPG et installe le moteur Docker.

2.2 Exécution du script d'installation

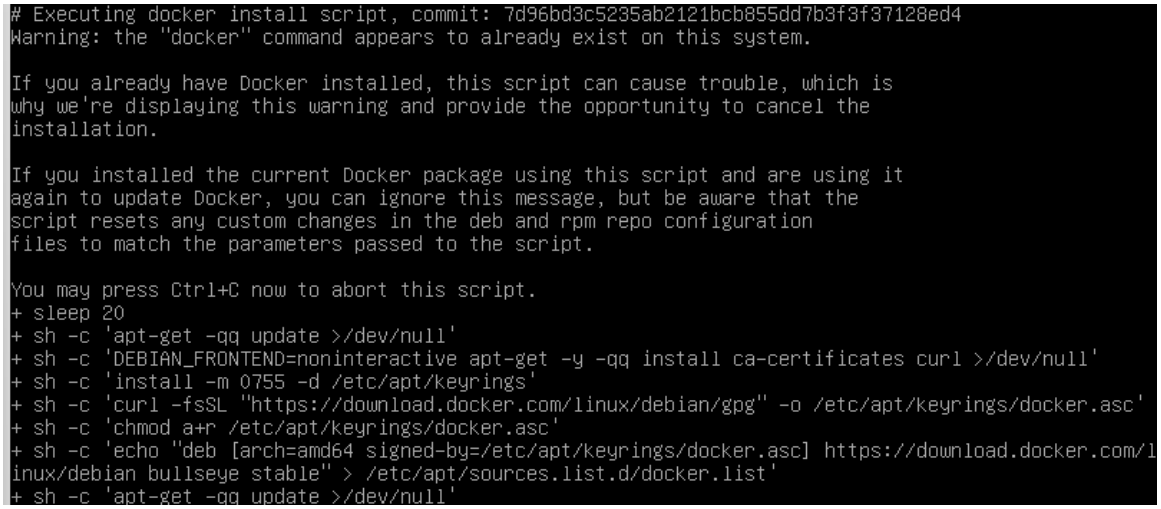

Figure 2 – Sortie du script d'installation Docker

> **bash index.html** : exécute le script d'installation téléchargé par wget. Le script effectue automatiquement les opérations suivantes :

- Mise à jour des dépôts (apt-get update)
- Installation des certificats et dépendances (ca-certificates, curl)
- Téléchargement et installation de la clé GPG officielle Docker
- Ajout du dépôt Docker dans les sources APT
- Installation du moteur Docker Engine

3. Vérification de l'installation

3.1 Test avec hello-world

La première vérification consiste à lancer un conteneur de test :

Figure 3 – Résultat de docker images après le test hello-world

> **docker run hello-world** : télécharge l'image hello-world depuis Docker Hub et l'exécute dans un conteneur. Si le message de bienvenue s'affiche, Docker est correctement installé.
>
> **docker images** : liste toutes les images Docker présentes localement. On retrouve ici l'image hello-world:latest avec son identifiant (IMAGE ID), sa taille disque (DISK USAGE) et la taille du contenu (CONTENT SIZE).

3.2 Liste des commandes conteneur

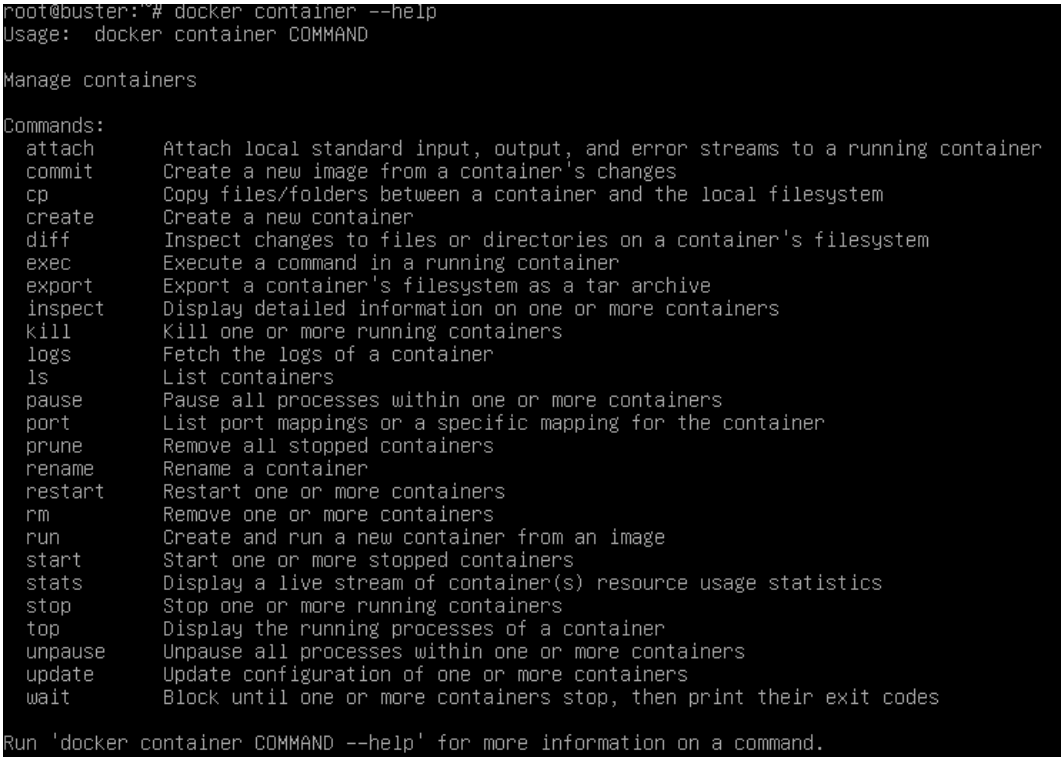

Figure 4 – Sortie de docker container --help

> **docker container --help** : affiche la liste complète des sous-commandes disponibles pour gérer les conteneurs. Parmi les plus utilisées : run (créer et lancer), ls (lister), stop (arrêter), rm (supprimer), exec (exécuter une commande dans un conteneur actif), logs (consulter les journaux).

3.3 Vérification du service Docker

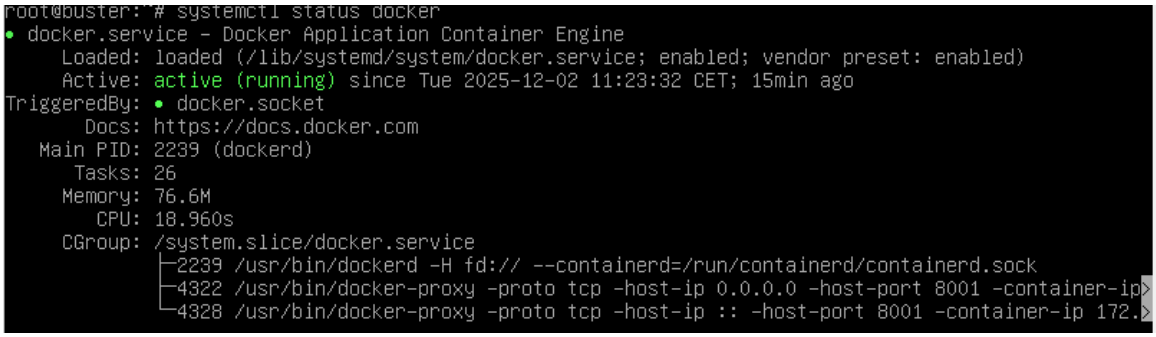

Figure 5 – Statut du service Docker

> **systemctl status docker** : vérifie l'état du service Docker via systemd. Les informations clés à observer :

- **Active: active (running)** – confirme que le daemon Docker tourne correctement
- **Loaded: loaded (...; enabled)** – le service est activé au démarrage du système
- **Main PID** – identifiant du processus principal dockerd
- **Memory / CPU** – ressources consommées par le daemon Docker

4. Gestion des images et conteneurs

4.1 Déploiement de DokuWiki

Téléchargement de l'image

```
root@buster:~# docker pull mprasil/dokuwiki
Using default tag: latest
latest: Pulling from mprasil/dokuwiki
Digest: sha256:507db13c7bda01572e86fc829eb32b447328b28a0dc4314b908a6987eb3d9efe
Status: Image is up to date for mprasil/dokuwiki:latest
docker.io/mprasil/dokuwiki:latest
```

Figure 6 – Téléchargement de l'image mprasil/dokuwiki

> **docker pull mprasil/dokuwiki** : télécharge l'image mprasil/dokuwiki depuis Docker Hub. Le tag « latest » est utilisé par défaut. Le Digest (SHA256) permet de vérifier l'intégrité de l'image téléchargée.

Vérification des images

```
root@buster:~# docker images
                                                                    i Info →  U In Use
IMAGE                    ID             DISK USAGE   CONTENT SIZE   EXTRA
hello-world:latest       f7931603f70e       25.9kB         9.52kB    U
mprasil/dokuwiki:latest  507db13c7bda        511MB          125MB    U
```

Figure 7 – Liste des images Docker après le pull

> docker images affiche désormais deux images : hello-world (25.9 kB) et mprasil/dokuwiki (511 MB). La colonne CONTENT SIZE indique la taille réelle du contenu de l'image.

Création et lancement du conteneur

```
root@buster:~# docker run -d -p 8003:80 --name wiki03 mprasil/dokuwiki
45d19aac3bd6d0a9d92dfb6f01ebd0f0cc2276faeb1a30e758e773dc44f4aaba
```

Figure 8 – Lancement du conteneur DokuWiki

> **docker run -d -p 8003:80 --name wiki03 mprasil/dokuwiki** :

- **-d** (detach) : lance le conteneur en arrière-plan. Le terminal reste disponible pour d'autres commandes.
- **-p 8003:80** (publish) : redirige le port 8003 de la machine hôte vers le port 80 du conteneur. Toute requête sur le port 8003 de l'hôte sera transmise au serveur web du conteneur.
- **--name wiki03** : attribue le nom « wiki03 » au conteneur pour le référencer facilement (au lieu de l'ID hexadécimal).
- **mprasil/dokuwiki** : nom de l'image source sur Docker Hub à utiliser pour créer le conteneur.

Le retour de la commande est l'identifiant complet (SHA256) du conteneur créé.

Liste des conteneurs

```
root@buster:~# docker container ls -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                      P
ORTS                                       NAMES
45d19aac3bd6   mprasil/dokuwiki  "/startup.sh run"        3 minutes ago    Up 3 minutes                0
.0.0.0:8003->80/tcp, [::]:8003->80/tcp     wiki03
830f05ea80b8   mprasil/dokuwiki  "/startup.sh run"        4 minutes ago    Created
                                           wiki02
b6309dadf099   mprasil/dokuwiki  "/startup.sh run"        20 minutes ago   Up 20 minutes               0
.0.0.0:8001->80/tcp, [::]:8001->80/tcp     wiki01
f7902ea796bd   hello-world       "/hello"                 24 minutes ago   Exited (0) 24 minutes ago
                                           magical_meninsky
```

Figure 9 – Liste de tous les conteneurs (actifs et arrêtés)

> **docker container ls -a** : liste tous les conteneurs, y compris ceux qui sont arrêtés. Les colonnes importantes :

- **CONTAINER ID** – identifiant court unique du conteneur
- **IMAGE** – image source utilisée pour créer le conteneur
- **STATUS** – état actuel : « Up » (en cours), « Created » (créé mais pas démarré), « Exited » (arrêté)
- **PORTS** – redirections de ports configurées (ex: 0.0.0.0:8003->80/tcp)
- **NAMES** – nom attribué au conteneur

Accès à l'interface web

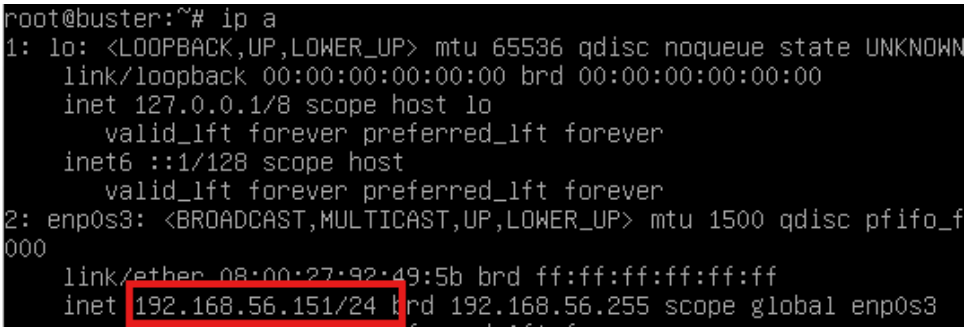

Figure 10 – Récupération de l'adresse IP de la machine hôte

> **ip a** : affiche les interfaces réseau et leurs adresses IP. Ici l'interface enp0s3 porte l'adresse 192.168.56.151/24. C'est cette adresse, combinée au port du conteneur, qui permet d'accéder au service.

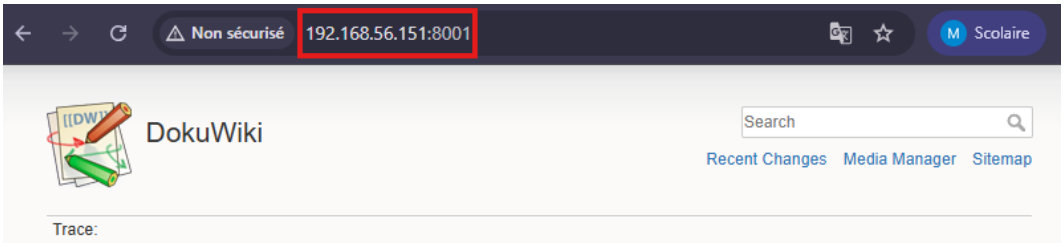

Figure 11 – Page d'accueil DokuWiki accessible via le navigateur

> En saisissant http://192.168.56.151:8001 dans le navigateur, on accède directement à l'interface DokuWiki servie par le conteneur. Le port 8001 de l'hôte est redirigé vers le port 80 du conteneur wiki01.

4.2 Déploiement d'Ubuntu

Téléchargement de l'image Ubuntu

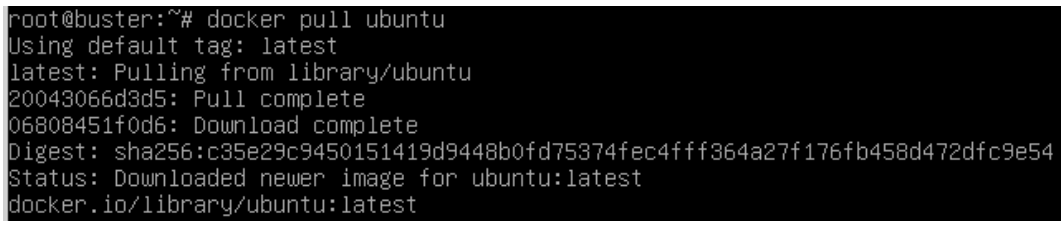

Figure 12 – Téléchargement de l'image Ubuntu

> **docker pull ubuntu** : télécharge l'image officielle Ubuntu depuis Docker Hub. Les layers (couches) sont téléchargées individuellement puis assemblées.

Lancement en arrière-plan

```
root@buster:~# docker run -d -p 8004:80 --name Ubuntu ubuntu
0fe20906b014ea0f8951b233663a51b6b81877fbec49a93d7152169e1ceafedb
```

Figure 13 – Création du conteneur Ubuntu en mode détaché

> **docker run -d -p 8004:80 --name Ubuntu ubuntu** : crée et lance un conteneur nommé « Ubuntu » en arrière-plan avec redirection du port 8004 vers le port 80.

Accès interactif au conteneur

```
root@debian0:~# docker run -ti ubuntu /bin/bash
root@82a22b1b4d08:/# _
```

Figure 14 – Ouverture d'un shell bash dans le conteneur Ubuntu

> **docker run -ti ubuntu /bin/bash** :

- **-t** (tty) : alloue un pseudo-terminal au conteneur, ce qui permet d'avoir un prompt interactif.
- **-i** (interactive) : maintient l'entrée standard (stdin) ouverte, permettant de taper des commandes.
- **/bin/bash** : commande à exécuter dans le conteneur. Ici on démarre un shell Bash.

> Le prompt change (root@82a22b1b4d08:/#) indiquant qu'on est maintenant à l'intérieur du conteneur, connecté en tant que root.

Exploration du conteneur

```
root@82a22b1b4d08:/# ls -l
total 48
lrwxrwxrwx   1 root root    7 Apr 22  2024 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Apr 22  2024 boot
drwxr-xr-x   5 root root  360 Dec  4 12:50 dev
drwxr-xr-x   1 root root 4096 Dec  4 12:50 etc
drwxr-xr-x   3 root root 4096 Oct 13 14:09 home
lrwxrwxrwx   1 root root    7 Apr 22  2024 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Apr 22  2024 lib64 -> usr/lib64
drwxr-xr-x   2 root root 4096 Oct 13 14:02 media
drwxr-xr-x   2 root root 4096 Oct 13 14:02 mnt
drwxr-xr-x   2 root root 4096 Oct 13 14:02 opt
dr-xr-xr-x 138 root root    0 Dec  4 12:50 proc
drwx------   2 root root 4096 Oct 13 14:09 root
drwxr-xr-x   4 root root 4096 Oct 13 14:09 run
lrwxrwxrwx   1 root root    8 Apr 22  2024 sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Oct 13 14:02 srv
dr-xr-xr-x  13 root root    0 Dec  4 12:50 sys
drwxrwxrwt   2 root root 4096 Oct 13 14:09 tmp
drwxr-xr-x  12 root root 4096 Oct 13 14:02 usr
drwxr-xr-x  11 root root 4096 Oct 13 14:09 var
root@82a22b1b4d08:/# _
```

Figure 15 – Contenu de la racine du conteneur Ubuntu

> **ls -l** : liste le contenu de la racine (/) du conteneur. On retrouve l'arborescence Linux standard (bin, boot, dev, etc, home, usr, var...). Le conteneur Ubuntu est un système Linux minimal et isolé du système hôte.

5. Docker Compose

Docker Compose permet de définir et gérer des applications multi-conteneurs à l'aide d'un fichier YAML.

5.1 Installation et vérification

```
root@debian0:~# apt-get install docker-compose version
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
E: Impossible de trouver le paquet version
root@debian0:~# apt-get install docker-compose-plugin
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets suivants seront mis à jour :
  docker-compose-plugin
1 mis à jour, 0 nouvellement installés, 0 à enlever et 80 non mis à jour.
Il est nécessaire de prendre 7 708 ko dans les archives.
Après cette opération, 45,3 Mo d'espace disque seront libérés.
Réception de :1 https://download.docker.com/linux/debian bullseye/stable amd64 docker-compose-plugin
 amd64 5.0.0-1~debian.11~bullseye [7 708 kB]
7 708 ko réceptionnés en 2s (4 394 ko/s)
Lecture des fichiers de modifications (« changelog »)... Terminé
(Lecture de la base de données... 37923 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../docker-compose-plugin_5.0.0-1~debian.11~bullseye_amd64.deb ...
Dépaquetage de docker-compose-plugin (5.0.0-1~debian.11~bullseye) sur (2.40.3-1~debian.11~bullseye)
...
Paramétrage de docker-compose-plugin (5.0.0-1~debian.11~bullseye) ...
root@debian0:~# docker compose version
Docker Compose version v5.0.0
```

Figure 16 – Installation de Docker Compose et vérification de la version

> **apt-get install docker-compose-plugin** : installe Docker Compose en tant que plugin du CLI Docker. La première tentative avec « docker-compose version » échoue car l'ancien paquet standalone n'existe pas ; il faut installer « docker-compose-plugin ».
>
> **docker compose version** : vérifie la version installée. Ici Docker Compose v5.0.0. Notez l'espace (docker compose) et non le tiret (docker-compose) pour la nouvelle syntaxe plugin.

6. Portainer – Interface de gestion

Portainer est une interface web open source qui permet de créer, modifier, surveiller et gérer des conteneurs Docker graphiquement.

6.1 Création du volume persistant

```
root@debian0:~# docker volume create portainer-data
portainer-data
```

Figure 17 – Création du volume portainer-data

> **docker volume create portainer-data** : crée un volume Docker nommé portainer-data. Les volumes permettent de persister les données même si le conteneur est supprimé et recréé. Les données de configuration de Portainer seront stockées dans ce volume.

6.2 Configuration initiale

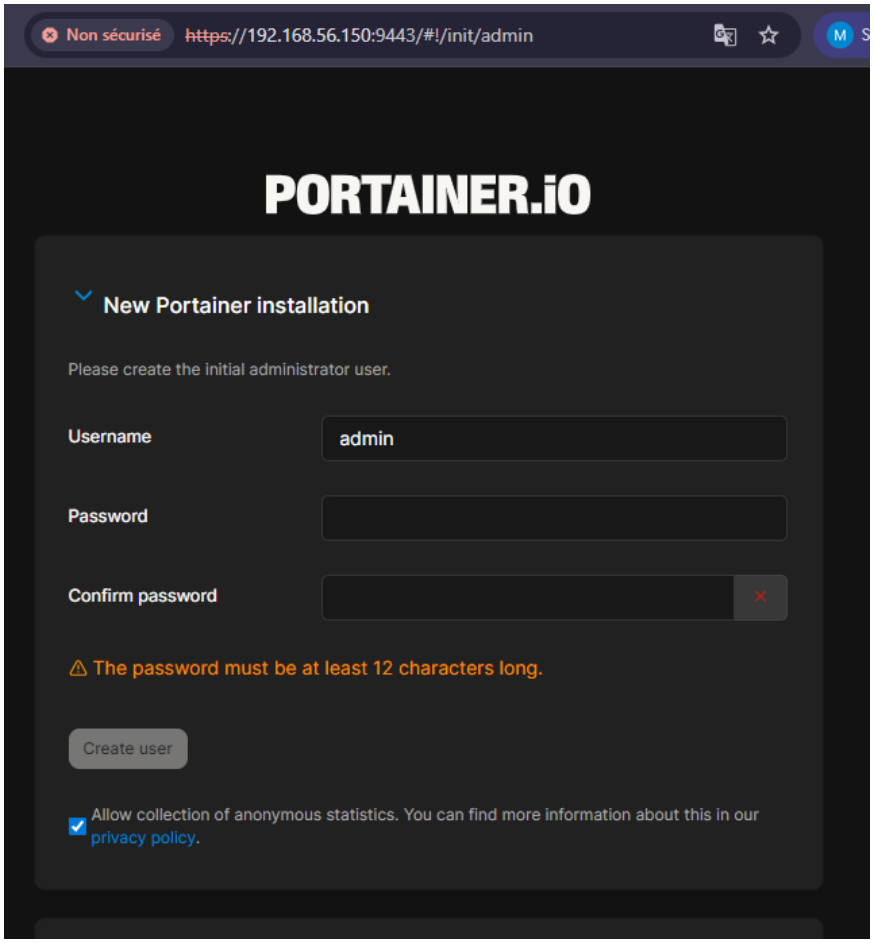

Figure 18 – Page de création du compte administrateur Portainer

> Lors du premier accès à Portainer via https://<IP>:9443, l'interface demande de créer un compte administrateur. Le mot de passe doit comporter au minimum 12 caractères.

6.3 Connexion

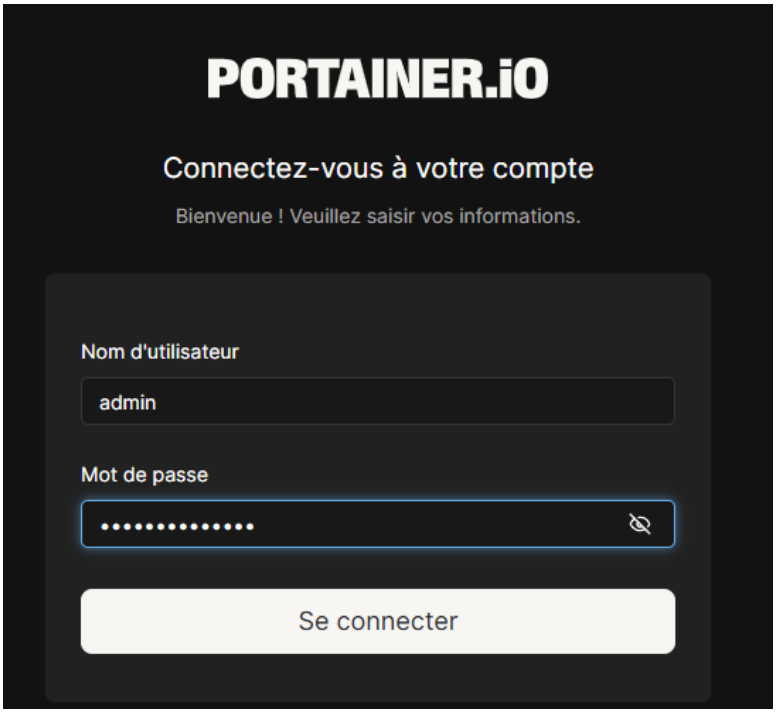

Figure 19 – Page de connexion à Portainer

> Après la création du compte, la page de connexion permet de s'authentifier avec les identifiants définis précédemment.

6.4 Choix de l'environnement

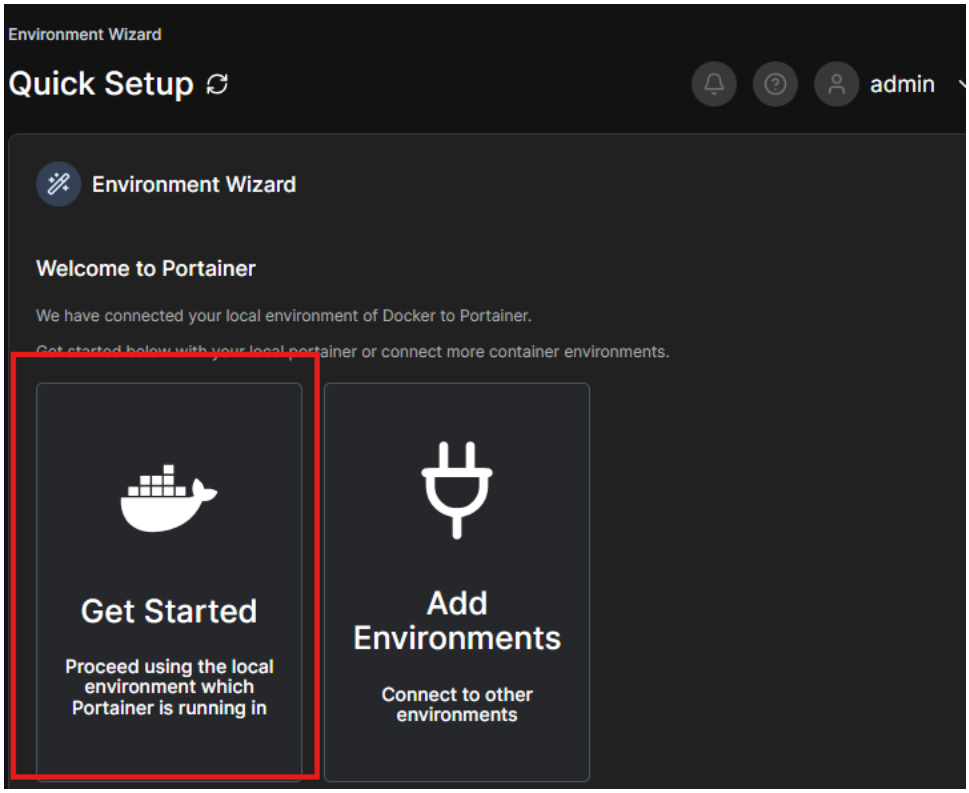

Figure 20 – Écran de sélection de l'environnement (Get Started)

> L'écran « Quick Setup » propose deux options : « Get Started » pour utiliser l'environnement Docker local (là où Portainer est installé), ou « Add Environments » pour se connecter à des environnements Docker distants. Pour un usage local, cliquer sur « Get Started ».

6.5 Dashboard

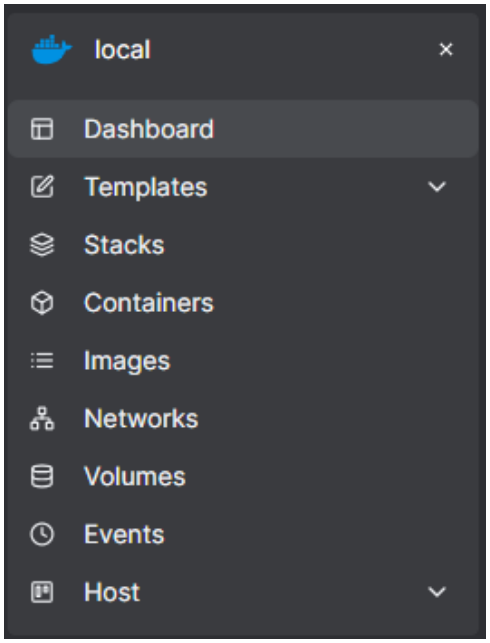

Figure 21 – Menu latéral du Dashboard Portainer

> Le menu latéral donne accès à toutes les fonctionnalités : Dashboard (vue d'ensemble), Stacks (applications multi-conteneurs), Containers (gestion individuelle), Images (gestion des images), Networks (réseaux Docker), Volumes (stockage persistant), Events (journal d'événements) et Host (informations système).

7. Déploiement de stacks via Portainer

7.1 Stack Nextcloud

Nextcloud est une solution de cloud privé auto-hébergée.

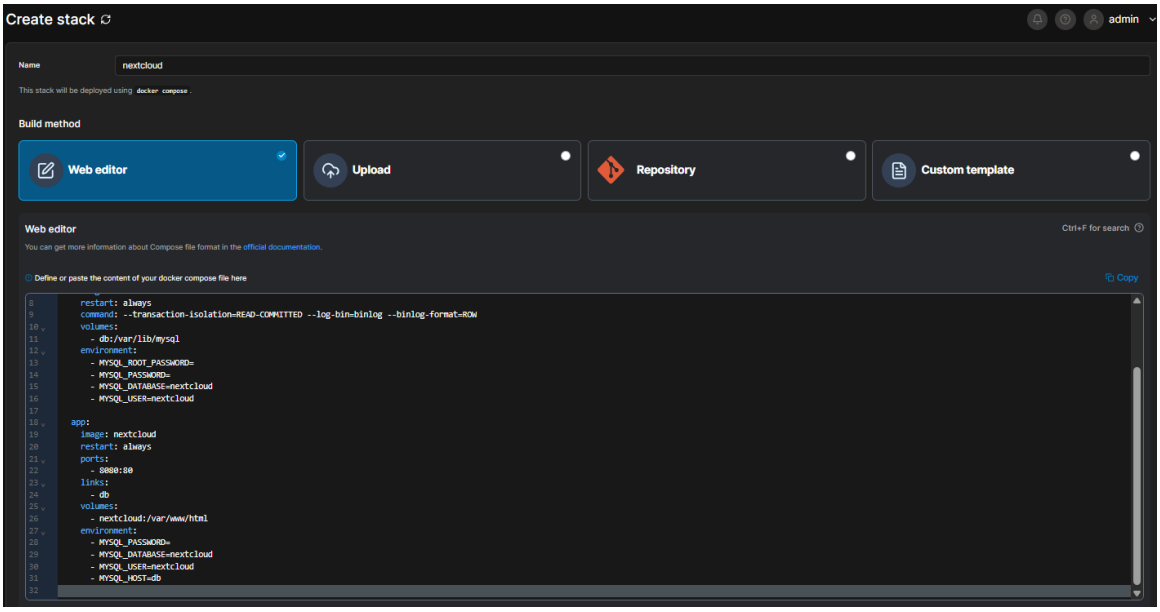

Figure 22 – Éditeur de stack Portainer avec la configuration Nextcloud

> Dans Portainer, la section Stacks > Add stack permet de définir un fichier docker-compose directement dans un éditeur web. Le stack Nextcloud comprend deux services : une base de données MySQL (db) et l'application Nextcloud (app) connectée sur le port 8080.

Les variables d'environnement (MYSQL_ROOT_PASSWORD, MYSQL_DATABASE, MYSQL_USER) configurent automatiquement la base de données lors du premier démarrage.

Configuration de Nextcloud

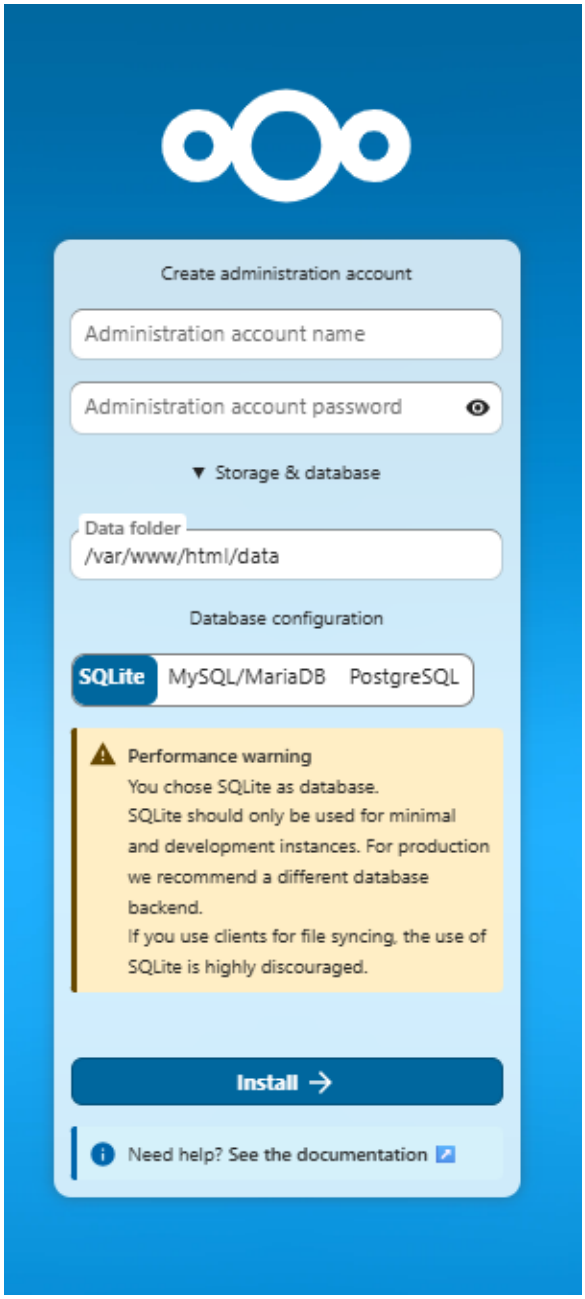

Figure 23 – Assistant de configuration Nextcloud

> Lors du premier accès (http://192.168.56.150:8080), Nextcloud affiche un assistant de configuration. Il faut créer un compte administrateur, choisir le dossier de données (/var/www/html/data) et sélectionner le type de base de données (SQLite pour le test, MySQL/MariaDB pour la production).

Applications recommandées

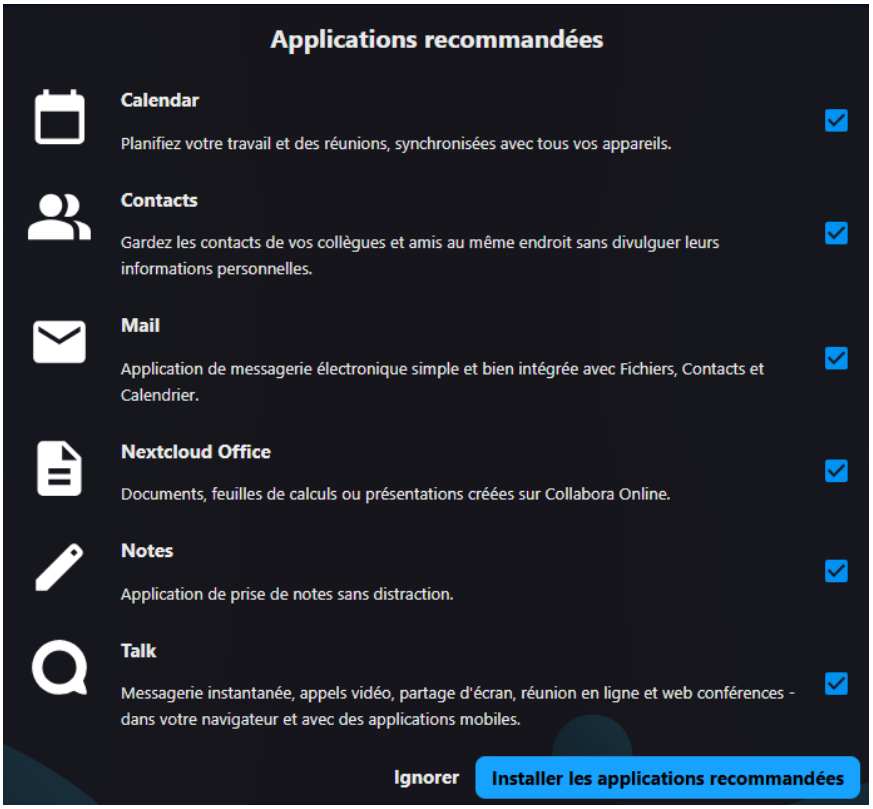

Figure 24 – Sélection des applications recommandées Nextcloud

> Nextcloud propose d'installer des applications intégrées : Calendar (agenda), Contacts (carnet d'adresses), Mail (messagerie), Nextcloud Office (suite bureautique), Notes (prise de notes) et Talk (visioconférence). Ces applications étendent les fonctionnalités de la plateforme.

Interface Nextcloud opérationnelle

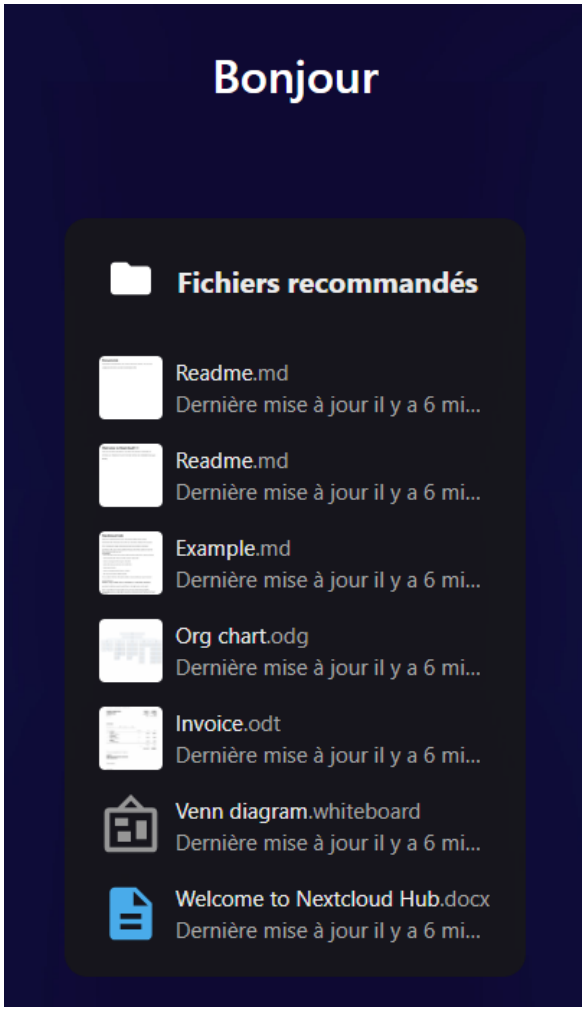

Figure 25 – Page d'accueil Nextcloud après installation

> L'interface Nextcloud est désormais fonctionnelle avec les fichiers recommandés (Readme.md, Example.md, etc.). L'utilisateur peut désormais stocker, partager et gérer ses fichiers via cette interface cloud privée.

7.2 Stack Nginx

Configuration initiale du stack

```yaml
version: '3.8'

services:
    web:
      image: nginx:latest
      container_name: nginx-web-simple
      ports:
        - "8085:80"
      restart: always

```

Figure 26 – Fichier docker-compose Nginx (version initiale)

> Ce fichier docker-compose définit un service web unique utilisant l'image nginx:latest. Le conteneur est nommé nginx-web-simple, le port 8085 de l'hôte est redirigé vers le port 80 du conteneur, et restart: always garantit le redémarrage automatique en cas de panne.

Page par défaut de Nginx

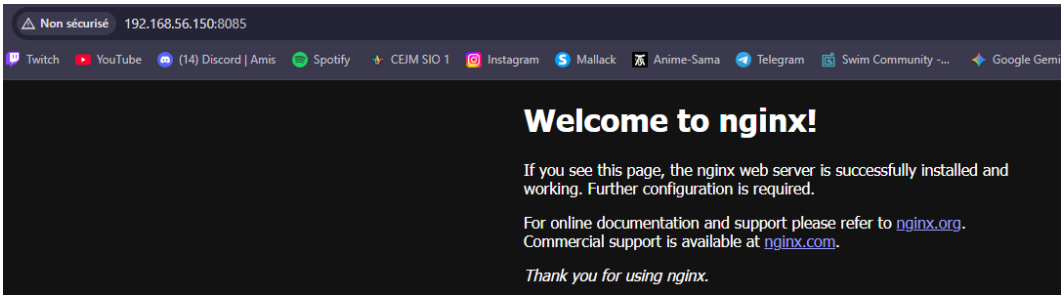

Figure 27 – Page d'accueil par défaut de Nginx

> En accédant à http://192.168.56.150:8085, la page « Welcome to nginx! » confirme que le serveur web fonctionne correctement dans le conteneur.

Personnalisation avec un volume

```yaml
version: '3.8'

services:
    web:
      image: nginx:latest
      container_name: nginx-web-simple
      ports:
        - "8085:80"
      volumes:
        - /opt/nginx-content/html:usr/share/nginx/html:ro
      restart: always

```

Figure 28 – Fichier docker-compose Nginx modifié avec montage de volume

> **volumes:** la directive volumes monte le répertoire local /opt/nginx-content/html dans le conteneur à l'emplacement /usr/share/nginx/html (racine web de Nginx). Le suffixe **:ro** (read-only) empêche le conteneur de modifier les fichiers source.

Résultat de la personnalisation

Mon Serveur Web Personnalisé !

Ce site est servi par NGINX via docker et Portainer.

Figure 29 – Page web personnalisée servie par Nginx

> Après la mise à jour du stack et le redéploiement, la page personnalisée « Mon Serveur Web Personnalisé ! » remplace la page par défaut de Nginx. Le contenu HTML créé dans /opt/nginx-content/html/index.html est servi directement par le conteneur.

7.3 Stack MariaDB + Adminer

Configuration de la base de données

```yaml
# Votre Base de Données
db:
  image: mariadb:10.11
  container_name: mariadb-db
  restart: always
  environment:
    MARIADB_ROOT_PASSWORD: mot_de_passe_robuste
    MARIADB_DATABASE: mon_site
    MARIADB_USER: utilisateur
    MARIADB_PASSWORD: mot_de_passe_utilisateur
  volumes:
    - db_data:/var/lib/mysql
  networks:
    - app-network

networks:
  app-network:
    driver: bridge

volumes:
  db_data:
```

Figure 30 – Configuration du service MariaDB dans le stack

> Le service db utilise l'image mariadb:10.11. Les variables d'environnement configurent automatiquement :

- **MARIADB_ROOT_PASSWORD** : mot de passe du compte root de la base de données
- **MARIADB_DATABASE** : nom de la base de données créée au démarrage (mon_site)
- **MARIADB_USER / MARIADB_PASSWORD** : compte utilisateur avec droits sur la base mon_site

> Le volume db_data:/var/lib/mysql assure la persistance des données. Le réseau app-network (driver bridge) permet la communication entre les services du stack.

Interface Adminer

Figure 31 – Page de connexion Adminer

> Adminer est un outil d'administration de base de données léger (un seul fichier PHP). Accessible sur le port 8086, il permet de se connecter à MariaDB en indiquant « db » comme serveur (nom du service Docker, résolu automatiquement grâce au réseau Docker interne).

8. Apache Guacamole

Apache Guacamole est une passerelle de bureau à distance accessible via un navigateur web, supportant les protocoles VNC, RDP et SSH.

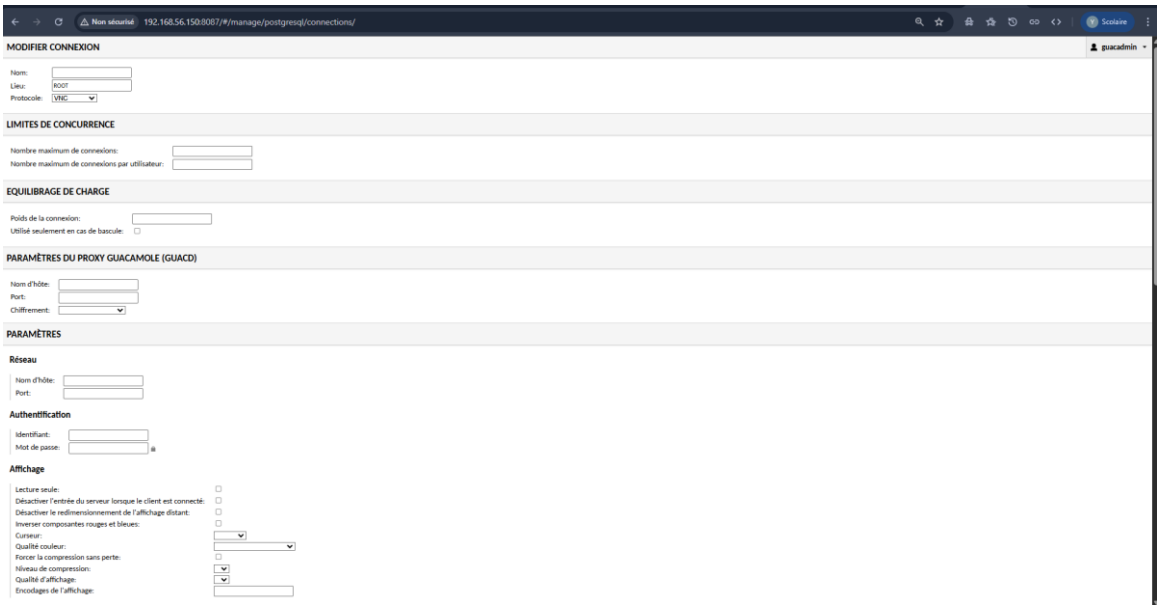

Figure 32 – Interface de configuration d'une connexion dans Guacamole

> L'interface Guacamole permet de configurer des connexions distantes. Les paramètres incluent le nom de la connexion, le protocole (VNC, RDP, SSH), les limites de concurrence, les paramètres du proxy guacd (nom d'hôte, port, chiffrement), les paramètres réseau (hôte et port de la machine cible), l'authentification (identifiant/mot de passe) et les options d'affichage (qualité, compression, encodage).

9. Exercice – Création de 2 stacks personnalisés

Objectif : Créer 2 stacks différents de votre choix, chacun composé d'un ensemble de conteneurs qui communiquent entre eux via un réseau Docker dédié.

9.1 Stack 1 : WordPress + MySQL

WordPress est le CMS le plus utilisé au monde. Ce stack déploie WordPress avec une base de données MySQL dédiée.

Architecture

- **wordpress** : serveur web Apache + PHP avec WordPress (port 8090)
- **mysql** : SGBDR MySQL 8.0 pour le stockage des données
- Communication via le réseau interne wp-network

Étapes de déploiement

1. Ouvrir Portainer > Stacks > Add stack

2. Nommer le stack : wordpress
3. Saisir le fichier docker-compose suivant :

```
version: '3.8'

services:
  db:
    image: mysql:8.0
    container_name: wp-mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword123
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wpuser
      MYSQL_PASSWORD: wppassword123
    volumes:
      - wp_db_data:/var/lib/mysql
    networks:
      - wp-network

  wordpress:
    image: wordpress:latest
    container_name: wp-app
    restart: always
    depends_on:
      - db
    ports:
      - "8090:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: wpuser
      WORDPRESS_DB_PASSWORD: wppassword123
    volumes:
      - wp_app_data:/var/www/html
    networks:
      - wp-network

networks:
  wp-network:
    driver: bridge

volumes:
  wp_db_data:
  wp_app_data:
```

4. Cliquer sur « Deploy the stack »
5. Accéder à WordPress : http://<adresse_IP>:8090
6. Suivre l’assistant d’installation WordPress

Vérification

```
docker exec -it wp-app ping db
```

> Le conteneur WordPress doit pouvoir résoudre le nom « db » et atteindre MySQL via le réseau interne wp-network.

9.2 Stack 2 : Gitea + PostgreSQL

Gitea est une forge logicielle légère et auto-hébergée, similaire à GitHub.

Architecture

- **gitea** : serveur Gitea avec interface web (port 3000) et SSH (port 2222)
- **postgres** : SGBDR PostgreSQL 16
- Communication via le réseau interne gitea-network

Étapes de déploiement

1. Ouvrir Portainer > Stacks > Add stack
2. Nommer le stack : gitea
3. Saisir le fichier docker-compose suivant :

```
version: '3.8'

services:
  db:
    image: postgres:16
    container_name: gitea-postgres
    restart: always
    environment:
      POSTGRES_USER: gitea
      POSTGRES_PASSWORD: giteapassword123
      POSTGRES_DB: giteadb
    volumes:
      - gitea_db_data:/var/lib/postgresql/data
    networks:
      - gitea-network

  gitea:
    image: gitea/gitea:latest
    container_name: gitea-app
    restart: always
    depends_on:
      - db
    ports:
      - "3000:3000"
      - "2222:22"
    environment:
      USER_UID: 1000
      USER_GID: 1000
```

```
      GITEA__database__DB_TYPE: postgres
      GITEA__database__HOST: db:5432
      GITEA__database__NAME: giteadb
      GITEA__database__USER: gitea
      GITEA__database__PASSWD: giteapassword123
    volumes:
      - gitea_app_data:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    networks:
      - gitea-network

networks:
  gitea-network:
    driver: bridge

volumes:
  gitea_db_data:
  gitea_app_data:
```

4. Cliquer sur « Deploy the stack »
5. Accéder à Gitea : http://<adresse_IP>:3000
6. Configurer la base de données PostgreSQL (hôte db:5432) et créer un compte administrateur

Vérification

```
docker exec -it gitea-app ping db
docker exec -it gitea-postgres psql -U gitea -d giteadb -c '\dt'
```

> La première commande vérifie la connectivité réseau. La seconde vérifie que la base de données est accessible en listant ses tables.

Récapitulatif

Caractéristique	Stack WordPress	Stack Gitea
Application	WordPress (CMS)	Gitea (Forge Git)
Base de données	MySQL 8.0	PostgreSQL 16
Port web	8090	3000
Réseau Docker	wp-network	gitea-network
Volumes	wp_db_data, wp_app_data	gitea_db_data, gitea_app_data
Communication	WordPress → MySQL (3306)	Gitea → PostgreSQL (5432)

Ces deux stacks illustrent le principe fondamental de Docker Compose : la création d'ensembles de conteneurs interconnectés, isolés dans leur propre réseau, avec des volumes persistants pour la sauvegarde des données.