Créer un serveur WEB avec Linux

Preambule

Une distribution sur base DEBIAN ou UBUNTU sera utilisée dans ce document.
Nous allons utiliser un package appelé LAMP pour Linux / Apache / MySQL / PHP.

Prerequis :

• Avoir un accès direct au serveur
• Avoir un accès SSH au serveur

1 – Verifier si le serveur est a jour

```
sudo apt update && sudo apt upgrade -y
```

2 – Installation d'Apache

```
sudo apt install apache2
```

- Vérifier le status d'Apache

```
sudo systemctl status apache2
```

Un message disant que le service est actif doit alors apparaitre.

3 – Installation de MySQL – Nous utiliserons plutot MariaDB qui est libre et plus performant.

```
sudo apt install mariadb-server
```

- Sécurisation de l'installation

```
sudo mariadb-secure-installation
```

A ce moment suivez les informations demandées jusqu'à la fin du script.
Attention si vous devez entrer un mot de passe, notez-le bien.

- Démarrage du service

```
sudo systemctl start mariadb
```

• Création d'un utilisateur

```
sudo mariadb
CREATE USER 'user_name'@'%' IDENTIFIED BY 'MotDePasseSecurise!';
```

Remplacer user_name et MotDePasseSecurise ! par vos données.

• Donner les droits de l'utilisateur à la gestion des bases :

```
GRANT ALL PRIVILEGES ON *.* TO 'user_name'@'%' WITH GRANT OPTION;
```

Remplacer user_name pour le nom d'utilisateur précédemment créé.

• Appliquer les changements

```
FLUSH PRIVILEGES ;
```

4 – Installation de PHP

```
sudo apt install php libapache2-mod-php
```

5 – (Facultative) Configuration du pare-feu

Si vous avez un firewall activé, vous devez autoriser le trafic sur les ports 80 (HTTP) et 443 (HTTPS). Pour UFW (un pare-feu simple à utiliser), vous pouvez exécuter :

```
sudo ufw allow 'Apache'
```

6 – Test du serveur

1. **Test d'Apache** : Dans votre navigateur, allez à l'adresse http://votre_adresse_ip ou http://localhost si vous êtes sur le serveur. Vous devriez voir la page d'accueil d'Apache Ubuntu.
2. **Test de PHP** : Créez un fichier info.php dans le répertoire /var/www/html/ :

```
sudo nano /var/www/html/info.php
```

Ajouter :

```
<?php
phpinfo();
?>
```

Enregistrer le fichier

Accéder ensuite à http://localhost/info.php dans un navigateur web. Les informations PHP devraient alors s'afficher à l'écran.

7 – Modification du repertoire /var/www/html/

Il faut maintenant donner les bonnes autorisations au répertoire /var/www/html/ pour que ce dernier soit isolé du reste du système pour des raisons de sécurité.
Pour ce faire nous allons déjà modifier le propriétaire et le groupe.

Lors de l'installation d'Apache, un groupe et un utilisateur ont été créés :

```
www-data
```

Nous allons donc attribuer la propriété du répertoire html à l'utilisateur et au groupe.

```
sudo chown -R www.data:www.data /var/www/html
```

Ensuite nous allons changer les droits sur ce répertoire.

```
sudo chmod -R 755 /var/www/html
```

Maintenant afin que tous les fichiers ou répertoires qui seront créés par la suite conservent les même droits et propriétaires nous allons ajouter une notion d'héritage.

```
sudo chmod g+s /var/www/html
```

Voilà le serveur est maintenant prêt !

LOCALISATION ET PARAMETRAGE DES FICHIER PHP.INI ET APACHE2.CONF

► php.ini

Le fichier php.ini est un fichier de configuration important pour PHP. Il contient les paramètres qui définissent le comportement de PHP.

Ce fichier se trouve à cet emplacement :

```
/etc/php/8.X/apache2/php.ini
```

Le X correspond à la version de PHP qui est installée sur le serveur, au moment de la rédaction de ce document, la version est 8.4.

Après chaque modification de ce fichier il faudra redémarrer le service Apache :

```
sudo systemctl restart apache2
```

► apache2.conf

apache2.conf est le fichier de configuration principal pour le serveur Web Apache sur les systèmes basés sur Debian, tels que Ubuntu.

Ce fichier contient les paramètres de configuration globaux pour Apache, qui s'appliquent à tous les sites Web hébergés sur le serveur.

Il définit les paramètres de base pour le fonctionnement du serveur Web, tels que :

- Le nom du serveur (ServerName)
- Le répertoire racine des documents (DocumentRoot)
- Le fichier index par défaut (DirectoryIndex)
- Les paramètres de sécurité, tels que les restrictions d'accès et les authentifications
- Les paramètres de journalisation, tels que les fichiers de journal des erreurs et des requêtes

apache2.conf est généralement situé dans `/etc/apache2/apache2.conf` sur les systèmes Ubuntu.

Voici quelques exemples de paramètres courants dans apache2.conf :

- ServerName : définit le nom du serveur Web.
- DocumentRoot : définit le répertoire racine des documents pour les sites Web.
- DirectoryIndex : définit le fichier index par défaut pour les répertoires.
- ErrorLog : définit le fichier de journal des erreurs pour le serveur Web.
- CustomLog : définit le fichier de journal des requêtes pour le serveur Web.

Les paramètres de apache2.conf peuvent être modifiés pour ajuster le comportement du serveur Web, mais il est important de prendre soin lors de la modification de ces paramètres pour éviter les problèmes de configuration.

Idem qu'avec php.ini, après chaque modification, il faudra redémarrer le service Apache :

```
sudo systemctl restart apache2
```

AUTOMATISATION DE REDEMARRAGE DES SERVICES

Pour que les service MariaDB et Apache démarrent à chaque redémarrage du serveur il faut le spécifier au système.

```
sudo systemctl enable apache2 && sudo systemctl enable mariadb
```

Pour vérifier le statut d'un service vous pouvez utiliser la commande :

```
sudo systemctl status <nom_du_service>
```

Exemple :

```
sudo systemctl status apache2
```

INSTALLATION DE ADMINER POUR LA GESTION DISTANTE DE LA BASE DE DONNEES

Ouvrir un terminal

• Créer un dossier adminer dans /var/www/html

```
sudo mkdir -p /var/www/html/adminer
```

• Redonner les bonnes autorisations d'utilisateur au répertoire

```
sudo chown www-data :www-data /var/www/html/adminer
```

```
sudo chmod 0755 /var/www/html/adminer
```

• Se rendre dans le répertoire créé

```
cd /var/www/html/adminer
```

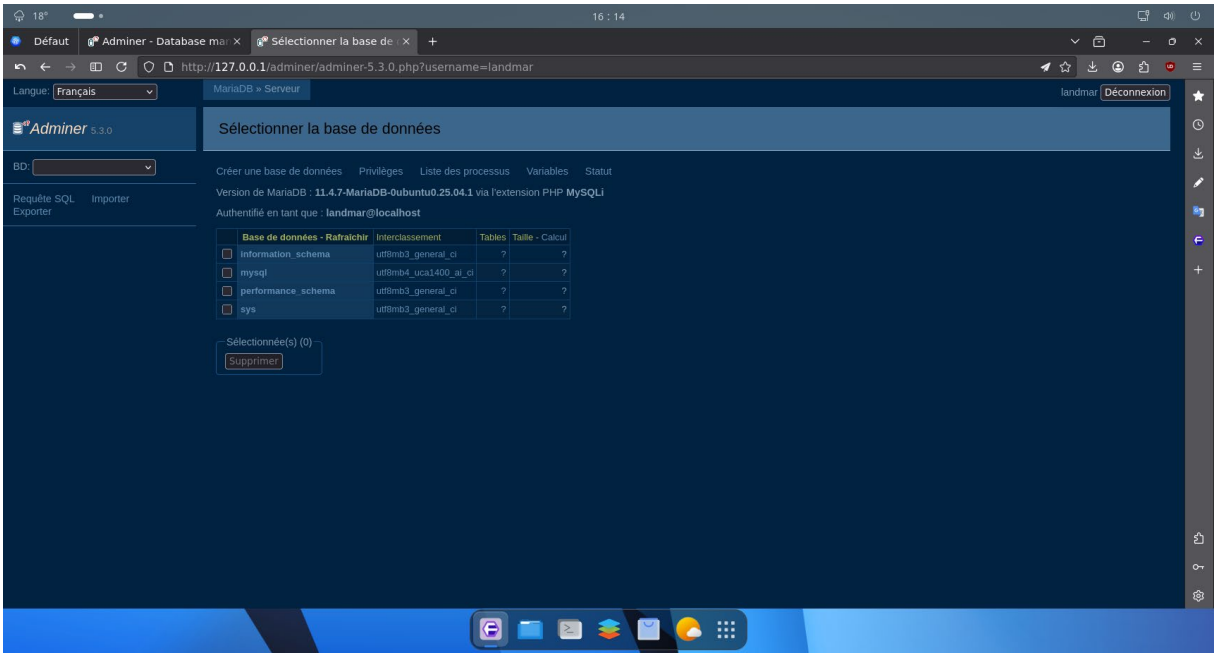
Télécharger le fichier adminer

```
wget https://github.com/vrana/adminer/releases/download/v5.3.0/adminer-5.3.0.php
```

Depuis un navigateur tapez l'adresse : http://IP_DU_Serveur/adminer

Connectez-vous avec les identifiants de la base de données.

Vous arriverez alors sur cette interface :



SUPERVISION DU SERVEUR

Afin de pouvoir administrer le serveur et le superviser et ce à distance, nous allons installer une interface Web de contrôle, nommée Webmin.

Pour ce faire, il va falloir ajouter le dépôts logiciel contenant Webmin, heureusement un script dédié à cet usage est disponible.

Ce denier va mettre à jour les dépôts et installer Webmin.

Voici la ligne de commande à taper :

```
curl -o webmin-setup-repo.sh
https://raw.githubusercontent.com/webmin/webmin/master/webmin-setup-repo.sh
```

Cette commande sert à télécharger le script. En tapant `ls -lh` vous pourrez voir que le fichier webmin-setup-repo.sh est alors présent sur le répertoire en cours.

Maintenant exécutons le script :

```
sudo bash webmin-setup-repo.sh
```

Le dépôt étant maintenant installé, procédons à l'installation de Webmin

```
sudo apt install webmin  --install-recommends
```

Afin de pouvoir accéder à l'interface il va falloir connaitre l'adresse IP du serveur. Pour ce faire il suffit de taper la commande :

```
ip a
```

Webmin écoute les connexions entrantes sur le port 10000.

Maintenant vous pouvez accéder à l'interface dans votre navigateur en tapant cette adresse :

https://[IP_DU_SERVEUR]:10000

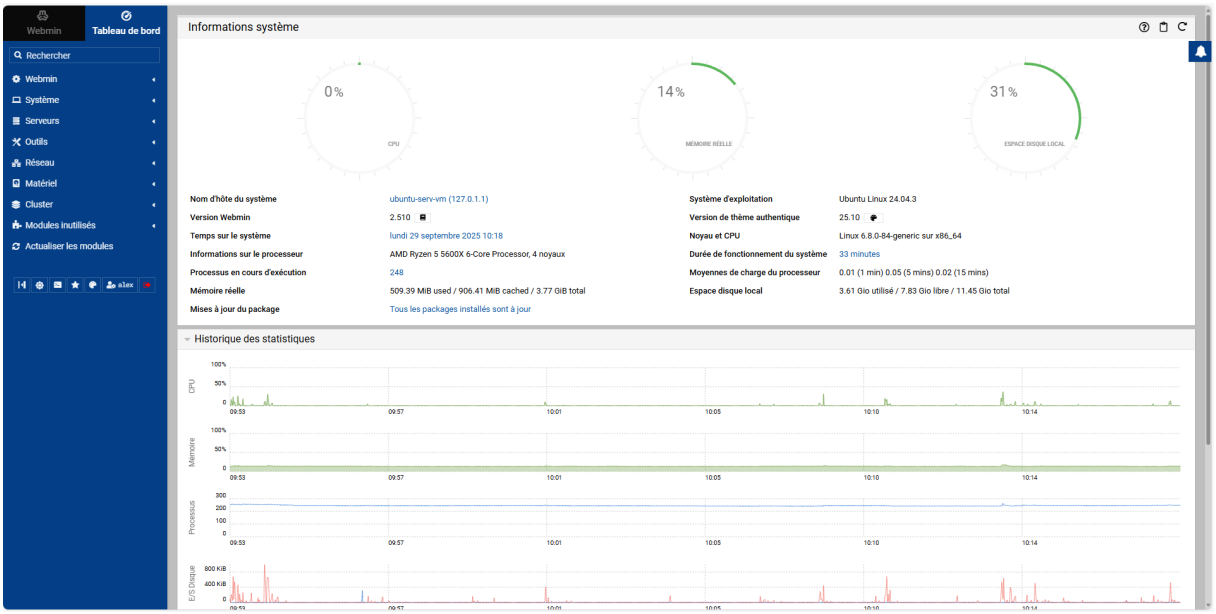
Votre navigateur va sans doute vous prévenir que la connexion n'est pas totalement sécurisée car le certificat utilisé n'est pas considéré comme valide. C'est tout à fait normal car celui-ci est émit uniquement par le serveur et pas par une autorité reconnue. Il suffit de cliquer sur le bouton avancé et d'accepter tout de même la connexion.
Vous serez alors invité à vous identifier avec les mêmes identifiants que ceux pour vous connecter à votre serveur.



Une fois identifié, vous arriverez sur le Dashboard qui est en anglais. Pour passer la langue en Français, rendez-vous dans le menu de gauche puis la zone Webmin et sélectionnez « change language and themes ».

Sélectionnez la langue Française dans language et appliquez les changements. Ensuite rafraichissez l'affichage de votre navigateur avec la touche F5.

Vous pourrez alors afficher le Tableau de bord qui se présente ainsi :

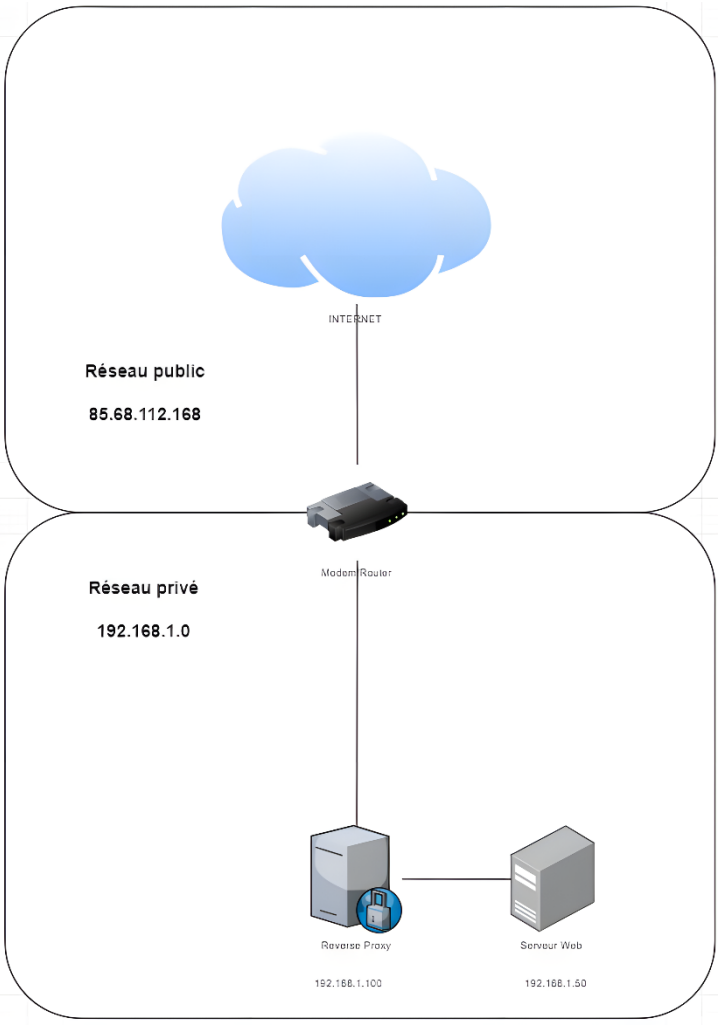


SECURISATION DU SERVEUR

A ce stade, bien que le serveur soit entièrement fonctionnel, il ne faut pas l'exposer directement à Internet. Nous allons voir comment faire pour limiter les expositions et mettre en place un système de reverse proxy qui utilisera un certificat SSL.

Pour mettre en place un reverse proxy nous utiliserons NGNIX en mode Docker afin que celui-ci soit entièrement cloisonné dans un container sécurisé.

Pour ce faire il faut une autre machine dédiée avec un système d'exploitation Linux à jour. Au niveau du matériel un Pc peu puissant fera entièrement l'affaire ou un mini-ordinateur type Raspberry Pi.
Le but d'un reverse proxy est de caché la connexion du serveur en faisant transiter les connexions au travers d'un ordinateur tiers qui lui est fortement sécurisé.



On utilise un prestataire externe qui supporte les connexions SSL et permet la redirection d'un nom de domaine vers une adresse IP.

Par exemple on ouvre un domaine : maconnexion.dns.org et on configure la redirection de maconnexion.dns.org pour qu'elle pointe vers 85.68.112.168, l'adresse public de notre connexion à Internet.

Dans le routeur on va créer une règle NAT qui fait que les connexions qui font un appel au port 80 ou 443 pointent vers l'adresse IP privée du reverse proxy : 192.168.1.100 dans notre exemple.

Le reverse proxy va recevoir les demandes de connexion et il sait que les demandes proviennent de maconnexion.dns.org avec un certificat valide que lui gère.
Il valide la demande et ouvre alors le canal de communication avec le serveur web.

Cela isole le serveur de toute connexion directe à Internet et n'autorise que les connexions au protocole http et https sur port 80 et 443 uniquement.

QU'EST-CE QUE DOCKER

Docker est une plateforme de conteneurisation qui permet d'emballer une application et toutes ses dépendances dans un conteneur léger et portable. Docker Compose est un outil pour définir et lancer des applications multi-conteneurs via un fichier de configuration.

► Docker — résumé

- **Conteneur** : unité légère et isolée qui contient l'application + ses dépendances.
- **Image** : snapshot immuable d'un conteneur (ce que l'on exécute).
- **Docker Engine** : le moteur qui crée et exécute les conteneurs.
- Utilité principale : **garantir que l'application tourne de la même façon sur tous les environnements** (dev, test, prod), simplifier le déploiement et isoler les dépendances.

Exemples d'usage concret :

- Emballer une application Node.js avec toutes ses bibliothèques.
- Isoler une base de données pour les tests sans installer globalement.

► Docker Compose — résumé

- Outil pour orchestrer plusieurs conteneurs liés entre eux.
- On décrit les services (conteneurs), réseaux et volumes dans un fichier YAML typiquement nommé **docker-compose.yml**.
- Commandes courantes : docker compose up (démarrer), docker compose down (arrêter et nettoyer).
- Utile pour : définir et démarrer un ensemble d'éléments (web, db, cache…) en une seule commande.

1 - Installation de Docker :

• Comme toujours, vérifier l'état de mise à jour du système

```
sudo apt update && sudo apt upgrade -y
```

• Installation du système de certificat d'authentification.

```
sudo apt install -y ca-certificates curl gnupg lsb-release
```

• Création de l'environnement de sécurité

```
sudo mkdir -p /etc/apt/keyrings curl -fsSL
https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg echo \ "deb [arch=$(dpkg --print-
architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \ $(lsb_release -cs)
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

• Installation de Docker Engine

```
sudo apt update && sudo apt install -y docker-ce docker-ce-cli
containerd.io docker-buildx-plugin docker-compose-plugin
```

• Vérification de l'installation

`sudo systemctl status docker`	affiche le statut du service
`sudo docker run --rm hello-world`	test de démonstration

• Activation du service au démarrage

```
sudo systemctl enable docker
```

2 - Installation de Portainer, l'interface Web de gestion de Docker.

Création d'un fichier YAML de configuration

Ouvrir un éditeur de texte et copier/coller le code suivant :

```
version: "3.8"
services:
  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: unless-stopped
    ports:
      - "9000:9000"    # UI HTTP
      - "9443:9443"    # UI HTTPS (TLS)
    volumes:
      - portainer_data:/data
      - /var/run/docker.sock:/var/run/docker.sock  # accès Docker
local (server mode)
  agent:
    image: portainer/agent:latest
    container_name: portainer_agent
    restart: unless-stopped
    environment:
      - AGENT_CLUSTER_ADDR=tasks.portainer_agent
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /var/lib/docker/volumes:/var/lib/docker/volumes
    ports:
      - "9001:9001"   # agent (optionnel si communications internes)
volumes:
  portainer_data:
```

Sauvegarder le fichier sous le nom portainer.yaml

Dans un terminal ou se trouve le fichier portainer.yaml, exécuter la commande :
`docker compose up -d`

• Pour mettre à jour l'image Docker :

Taper cette commande, toujours où se trouve le fichier yaml.
`docker pull portainer/portainer-ce:latest && docker compose down && docker compose up -d`

Pour accéder à Portainer, il vous suffit de taper l'adresse : http://localhost:9000 dans votre navigateur web si vous avez une interface graphique ou l'IP du serveur à la place de localhost si vous y accéder depuis un serveur distant.

Installation de NGINX Proxy Manager (NPM)

• Préparation de l'image Docker

Créez un répertoire dédié à NPM et entrez dedans.

```
mkdir -p ~/nginx-proxy-manager
cd ~/nginx-proxy-manager
```

1 – Creation du fichier Docker-Compose

Ouvrez un éditeur de texte (nano par exemple) et copiez/collez le code suivant :

```
version: "3"
services:
  app:
    image: jc21/nginx-proxy-manager:latest
    container_name: nginx-proxy-manager
    restart: unless-stopped
    ports:
      - "80:80"     # HTTP
      - "81:81"     # UI/Admin
      - "443:443"   # HTTPS
    environment:
      DB_MYSQL_HOST: "db"
      DB_MYSQL_PORT: 3306
      DB_MYSQL_USER: "npm"
      DB_MYSQL_PASSWORD: "npm_password_here"
      DB_MYSQL_NAME: "npm"
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt

  db:
    image: mariadb:10.11
    container_name: nginx-proxy-manager-db
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: "change_root_password"
      MYSQL_DATABASE: "npm"
      MYSQL_USER: "npm"
      MYSQL_PASSWORD: "npm_password_here"
    volumes:
      - ./data/mysql:/var/lib/mysql
```

Vérifiez les données et modifiez les parties contenant des mots de passe ou des utilisateurs par vos propres données. Attention à bien sauvegarder toutes ces données !

Lancez le conteneur

```
docker compose up -d
```

Pour accéder à l'interface : http://IP_DU_SERVEUR:81

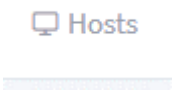
2 - Configuration du reverse proxy

► Pré-requis :

Disposer d'un nom de domaine ou d'un DDNS comme peuvent le proposer :

Voir : https://www.ionos.fr/digitalguide/serveur/outils/apercu-des-fournisseurs-gratuits-de-dyndns/

• Création des Hôtes

Dans l'interface d'administration cliquez sur 

Puis 