Auto-hébergement du portfolio sur Raspberry Pi 5

Stack web, exposition Internet, sécurisation et exploitation

Objectif : héberger un portfolio personnel sur un Raspberry Pi 5, accessible depuis Internet, avec une configuration reproductible (runbook), un minimum de durcissement et des pratiques d’exploitation (sauvegardes, mises à jour, logs).

Équipement	Raspberry Pi 5 (4/8 Go) + microSD ou SSD USB (recommandé)
OS	Raspberry Pi OS (64-bit) / Debian (stable)
Stack	Apache2 + PHP + (optionnel) MariaDB
Réseau	IP fixe (ou réservation DHCP), redirection de ports, DNS/DDNS
Sécurité	SSH par clé, pare-feu (UFW), Fail2ban, HTTPS (Let's Encrypt)

Pré-requis (à vérifier avant de commencer)

Accès administrateur au routeur/box Internet (NAT/port forwarding).Un nom de domaine (recommandé) ou un service DDNS.Accès SSH local au Raspberry Pi (clavier/écran ou SSH via LAN).

Convention de nommage

Dans cette procédure, le serveur s’appelle **pi-portfolio** et le site est servi pour **portfolio.example.tld**. Adapte ces valeurs à ton contexte.

1. Installation et configuration de base

1) Flasher l'OS (Raspberry Pi Imager) en 64-bit. Active **SSH** et définis un utilisateur non-root.

2) Premier boot, puis mise à jour complète :

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

3) Définir hostname + timezone :

```
sudo hostnamectl set-hostname pi-portfolio
sudo timedatectl set-timezone Europe/Paris
```

4) Vérifier l'adresse IP et préparer une IP fixe (ou une réservation DHCP sur le routeur).

2. Durcissement minimal (SSH + pare-feu + mises à jour)

Objectif : réduire la surface d'attaque avant d'exposer le serveur.

SSH par clé (recommandé) : génère une clé côté client puis copie la clé publique :

```
ssh-keygen -t ed25519
ssh-copy-id user@IP_DU_PI
```

Ensuite, côté serveur, ajuste /etc/ssh/sshd_config :

```
sudo nano /etc/ssh/sshd_config

# Recommandations
PasswordAuthentication no
PermitRootLogin no
PubkeyAuthentication yes
```

Recharge SSH :

```
sudo systemctl reload ssh
```

Pare-feu UFW (ouvrir uniquement SSH/HTTP/HTTPS) :

```
sudo apt install -y ufw
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow OpenSSH
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
sudo ufw status verbose
```

Fail2ban (protection brute-force) :

```
sudo apt install -y fail2ban
sudo systemctl enable --now fail2ban
sudo fail2ban-client status
```

Mises à jour automatiques (optionnel mais utile) :

```
sudo apt install -y unattended-upgrades
```

```
sudo dpkg-reconfigure --priority=low unattended-upgrades
```

3. Installation de la stack Web (Apache + PHP + MariaDB optionnel)

Installer Apache et démarrer le service :

```
sudo apt install -y apache2
sudo systemctl enable --now apache2
```

Installer PHP (version selon Debian/RPi OS) :

```
sudo apt install -y php libapache2-mod-php php-cli php-curl php-xml php-mbstring
```

MariaDB (optionnel - utile si ton portfolio consomme une base) :

```
sudo apt install -y mariadb-server
sudo systemctl enable --now mariadb
sudo mysql_secure_installation
```

Page de test PHP (à supprimer ensuite) :

```
echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php
```

4. Déploiement du portfolio (vhost propre + permissions)

Créer un répertoire dédié :

```
sudo mkdir -p /var/www/portfolio
sudo chown -R $USER:www-data /var/www/portfolio
sudo chmod -R 2750 /var/www/portfolio
```

Déployer les fichiers (exemples) :

```
# Option A: depuis ton PC
scp -r ./site/* user@IP_DU_PI:/var/www/portfolio/

# Option B: git (si repo)
cd /var/www/portfolio && git clone <repo> .
```

Créer un VirtualHost Apache :

```
sudo nano /etc/apache2/sites-available/portfolio.conf

<VirtualHost *:80>
  ServerName portfolio.example.tld
  DocumentRoot /var/www/portfolio
  <Directory /var/www/portfolio>
    AllowOverride All
    Require all granted
  </Directory>
  ErrorLog ${APACHE_LOG_DIR}/portfolio_error.log
  CustomLog ${APACHE_LOG_DIR}/portfolio_access.log combined
</VirtualHost>
```

Activer le site + modules utiles :

```
sudo a2enmod rewrite headers
sudo a2ensite portfolio.conf
sudo a2dissite 000-default.conf
sudo apache2ctl configtest
sudo systemctl reload apache2
```

5. HTTPS avec Let's Encrypt (Certbot)

Installer Certbot et le plugin Apache :

```
sudo apt install -y certbot python3-certbot-apache
```

Générer le certificat :

```
sudo certbot --apache -d portfolio.example.tld
```

Vérifier le renouvellement automatique :

```
sudo certbot renew --dry-run
systemctl list-timers | grep certbot || true
```

6. Exposition Internet (routeur, DNS, DDNS) - points clés

NAT / Port forwarding : rediriger TCP 80 et 443 vers l'IP du Raspberry Pi.

DNS : pointer portfolio.example.tld vers ton IP publique (ou utiliser un service DDNS).

Conseil : si possible, utilise un proxy/DNS type Cloudflare pour simplifier certains réglages et ajouter une couche anti-DDoS (optionnel).

7. Exploitation : sauvegardes, logs, supervision

Sauvegarde (exemple rsync vers un disque externe monté sur /mnt/backup) :

```
sudo mkdir -p /mnt/backup/portfolio
sudo rsync -a --delete /var/www/portfolio/ /mnt/backup/portfolio/
```

Automatiser via cron (tous les jours à 02h00) :

```
sudo crontab -e

0 2 * * * rsync -a --delete /var/www/portfolio/ /mnt/backup/portfolio/ >/var/log/
backup_portfolio.log 2>&1
```

Logs : suivre les accès Apache et le journal système :

```
sudo tail -f /var/log/apache2/portfolio_access.log
sudo journalctl -u apache2 -f
```

Surveillance : au minimum, vérifier l'état des services :

```
systemctl status apache2
systemctl status fail2ban
ufw status verbose
```

8. Validation (checklist de fin)

Le site répond en HTTP puis redirige vers HTTPS.Ports exposés : uniquement 80/443 (+ 22 si nécessaire) ; le reste est filtré.SSH : accès par clé, root désactivé.Certificat OK et renouvellement configuré.Une stratégie de sauvegarde existe (même simple).

Annexe - commandes utiles

```
Voir l’IP / réseau : ip a
Tester HTTP/HTTPS : curl -I http://portfolio.example.tld ; curl -I https://portfolio.example
Vérifier ports ouverts : sudo ss -tulpn
Permissions web : namei -l /var/www/portfolio
Apache modules : apache2ctl -M
```