Services maison : le guide du local

Faire tourner ses propres services à la maison — messagerie, tableau de bord, sauvegarde, domotique — sans abonnement, sans nuage, sans envoyer ses données chez quelqu'un d'autre.

Version 1.0 — 2026-11-18

---

1. Le principe

Un service local a trois propriétés qu'un service cloud n'a pas :

- rien ne sort de votre réseau, sauf ce que vous décidez ;
- personne ne peut fermer le service en changeant ses conditions ;
- la panne est visible : c'est votre machine, vous pouvez regarder.

En contrepartie, vous devenez l'exploitant. Ce guide traite exactement de cela : comment installer, surveiller et sauvegarder un service pour qu'il tienne des années sans attention.

La règle d'or : un service que vous ne surveillez pas est un service qui tombera sans prévenir.

---

2. Choisir ce qu'on auto-héberge

Tout n'est pas worth la peine. Critères :

Auto-héberger	Laisser au cloud
données personnelles (santé, notes, photos)	service à fort débit (vidéo)
service que vous utilisez tous les jours	outil que vous oubliez
protocole ouvert (MQTT, ActivityPub, CalDAV)	application fermée sans API
besoin d'automatisation	service gratuit suffisant

Le piège classique : auto-héberger pour le principe, puis passer plus de temps à maintenir l'outil qu'à l'utiliser. Si un service gratuit vous convient et ne manipule pas vos données, gardez-le.

---

3. systemd : l'unité comme contrat

Une unité systemd décrit comment le service doit vivre : ce qu'il faut avant de le démarrer, ce qu'il faut faire s'il échoue, ce qu'il ne doit jamais pouvoir faire.

```
[Unit]
Description=Mon service
After=network.target mosquitto.service
Wants=mosquitto.service

[Service]
Type=simple
User=monuser
Group=gpio
ExecStart=/usr/bin/python3 /home/monuser/service/serveur.py
```

```
Restart=on-failure
RestartSec=10
NoNewPrivileges=true
ProtectSystem=full
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Quatre lignes qui changent tout :

- Restart=on-failure — un plantage ne laisse pas le service mort ;
- User= — le service tourne sans droits superutilisateur ;
- NoNewPrivileges, ProtectSystem=full — s'il est compromis, il ne peut pas écrire sur le système ;
- After= — il ne démarre pas avant son dépendance.

Ne jamais ajouter Restart=always sans réfléchir : un service qui plante en boucle redémarre en boucle. Préférer on-failure et comprendre la cause.

Timers plutôt que démons

Pour une tâche périodique, un *timer* est presque toujours préférable à un démon : il n'y a pas de processus permanent, donc rien à maintenir, et un échec ne laisse rien de coincé.

```
[Timer]
OnBootSec=2min
OnUnitActiveSec=60
AccuracySec=5
Unit=mon-service.service
```

---

4. Boucle locale : le défaut, pas l'exception

Un service qui n'a pas besoin d'être joignable depuis l'internet ne doit écouter que sur 127.0.0.1.

```
# Grafana
Environment=GF_SERVER_HTTP_ADDR=127.0.0.1
Environment=GF_USERS_ALLOW_SIGN_UP=false
Environment=GF_AUTH_ANONYMOUS_ENABLED=false
```

Deux raisons : une surface d'attaque nulle, et l'impossibilité d'exposer quelque chose par erreur. Pour accéder au service depuis l'extérieur, un tunnel (chiffrement de bout en bout, authentification) vaut mieux qu'une ouverture de port.

Vérifier : ss -ltnp | grep <port> doit montrer 127.0.0.1, jamais 0.0.0.0.

---

5. Surveiller : relever, pas supposer

Un système fiable ne se contente pas d'agir : il relit l'état qu'il croit avoir produit.

- « J'ai écrit le GPIO, donc le relais est fermé » → non. Relire la broche.
- « J'ai envoyé le message » → non. Lire le code de retour.
- « Le service est actif » → non. systemctl is-active et le journal.

Un bon tableau de surveillance pour une machine seule :