Auto-hébergement fiable sur Raspberry Pi

Superviser, sauvegarder et prouver un Pi qui tourne déjà — sans capteur, sans nuage payant, sans abonnement.

Version 1.0 — 2026-11-20. Tous les extraits de ce guide sont copiés de fichiers qui existent sur le Pi d'origine (Raspberry Pi 5 B Rev 1.1, BCM2712 + RP1, 16 Go, Raspberry Pi OS Lite / Debian 13 trixie, utilisateur athena, couche dans ~/Documents/Athena). Chaque chapitre indique ses fichiers sources et la commande qui permet de revérifier ce qui est affirmé.

Ce que ce guide n'est pas : un tutoriel de capteurs. Aucun matériel n'est requis. Si un extrait vient d'un fichier qui n'existe pas encore, c'est écrit noir sur blanc dans le chapitre, pas maquillé en exemple.

Sommaire

Partie I — Le système de surveillance

1. Pourquoi superviser, et quoi
2. Huit unités systemd, et le rôle de chacune
3. Le piège User=
4. La sonde de santé : ce qu'elle écrit et où
5. La page de statut sur 127.0.0.1

Partie II — Les données

6. Sauvegarder le vault : le bundle git
7. Prouver que la sauvegarde marche : la restauration testée
8. Le doctor
9. Synchroniser sans y penser

Partie III — Aller plus loin

10. Surveiller les paiements entrants
11. La validation quotidienne
12. Le backlog et le cron

Partie IV — Les pièges

13. Les dix pannes qui coûtent du temps
14. Checklist de déploiement

Partie I — Le système de surveillance

1. Pourquoi superviser, et quoi

Un Pi ne prévient pas quand il faillit. Il ne dit rien quand l'alimentation s'affaisse, quand le SoC se bride thermiquement, quand une unité systemd meurt en silence, quand la carte SD se remplit. Le journal système contient la trace — mais personne ne lit journalctl en continu.

La supervision utile n'est pas un tableau de bord décoratif. C'est une mesure périodique écrite quelque part, plus une règle explicite qui décide quand la mesure devient une alerte.

Sur cette machine, six grandeurs décident de presque tout :

Grandeur	Source réelle	Seuil retenu
température SoC	vcgencmd measure_temp	≥ 78 °C
bridage (throttle)	vcgencmd get_throttled	≠ 0x0
disque racine	shutil.disk_usage("/")	≥ 90 %
unités en échec	systemctl --failed	> 0
horloge	timedatectl show -p NTPSynchronized --value	non synchronisée
Wi-Fi	/proc/net/wireless	≤ −85 dBm

Ces seuils ne sont pas des opinions : ils sont écrits dans le code, en tête de sys/health_collect.py :

```
TEMP_WARN_C = 78.0
DISK_WARN_PCT = 90.0
WLAN_WEAK_DBM = -85.0
ALERT_REPEAT_S = 6 * 3600
KEEP_SAMPLES_S = 90 * 24 * 3600
KEEP_ALERTS_S = 180 * 24 * 3600
```

Deux durées de rétention, deux usages : 90 jours d'échantillons pour voir une tendance, 180 jours d'alertes pour retrouver un incident. Au-delà, la sonde élague elle-même (fonction prune) — une base qui grossit sans fin finit par remplir le disque qu'elle était censée surveiller.

Le bridage mérite un mot. get_throttled renvoie un masque de bits, pas un booléen. 0x0 = rien ne s'est produit. Toute autre valeur signale une sous-tension ou un bridage thermique, passé ou présent. Le code distingue -1 (lecture impossible) de 0 (rien) — et ne lève pas d'alerte sur une lecture impossible. C'est le genre de détail qui évite une nuit d'alertes fantômes quand vcgencmd n'est pas dans le PATH du service.

Relecture réelle du Pi, le 2026-11-20 à 00:27 EST :

```
$ vcgencmd measure_temp
temp=42.2'C
$ vcgencmd get_throttled
throttled=0x0
$ vcgencmd measure_volts
volt=0.8325V
$ df -h / | tail -1
```

```
/dev/mmcblk0p2  115G   12G   99G  11% /
```

Fichiers sources : sys/health_collect.py, data/health.db.
Revérifier : vcgencmd measure_temp && vcgencmd get_throttled.

2. Huit unités systemd, et le rôle de chacune

La couche tient sur huit unités : quatre couples service + timer, un service continu, et un couple de maintenance. Rien d'autre. Un timer vaut mieux qu'un démon qui boucle : systemd gère la périodicité, le rattrapage après extinction, la journalisation et la relance.

Voici le tableau complet, avec la périodicité relue dans les fichiers state/units/*.timer :

Unité	Ce qu'elle exécute	Périodicité
athena-health.service + .timer	sys/health_collect.py → data/health.db	OnUnitActiveSec=60, OnBootSec=45, AccuracySec=5
athena-health-web.service	sys/health_web.py (page de statut)	continu, Restart=on-failure, RestartSec=5
athena-vault-backup.service + .timer	sys/vault_backup.sh	OnCalendar=*-*-* 03:17:00, Persistent=true
athena-vault-doctor.service + .timer	sys/vault_doctor.sh	OnCalendar=Sun *-*-* 04:17:00, Persistent=true
athena-git-sync.service + .timer	sys/git_sync.sh	OnBootSec=10min, OnUnitActiveSec=2h
athena-site-maintenance.service + .timer	sys/maintenance_site.sh	OnCalendar=*-*-* 04:47:00, Persistent=true

Le fichier de collecte, en entier — c'est le gabarit à copier :

```
[Unit]
Description=Athéna — collecteur de santé (échantillon Pi 5 vers SQLite)
After=network.target

[Service]
Type=oneshot
User=athena
Group=athena
ExecStart=/usr/bin/python3 /home/athena/Documents/Athena/sys/health_collect.py
Nice=5
IOSchedulingClass=idle
```

Et son timer :

```
[Unit]
Description=Athéna — collecte santé toutes les 60 s

[Timer]
OnBootSec=45
OnUnitActiveSec=60
AccuracySec=5
Unit=athena-health.service

[Install]
WantedBy=timers.target
```