

AllWatch

Un micro-superviseur de processus pour Linux : cibles au premier plan, santé par le noyau et le TCP, politiques de relance par cible

Manuel

Version	0.1.0
Commit	`ae9cb8f`
Date du commit	24 novembre 2026
Licence	MIT

Manuel hors ligne, identique aux guides du dépôt à la version et au commit indiqués. Le texte du dépôt fait foi.

Table des matières

Présentation **1**
- Installer sur Debian et Ubuntu . . . 1
- Installer sur Fedora, openSUSE Leap et Arch Linux . . . 1
- Exécution depuis le dépôt de code . . . 2
- Comment ça marche . . . 2
- Configuration . . . 3
- Le contrat qu'une cible doit respecter . . . 3
- Limites . . . 3
- Licence . . . 4
- Crédits . . . 4

Installer et exploiter le service Debian **5**
- Ce que le paquet installe . . . 5
- Installer . . . 5
- Chemins et fichiers . . . 5
- Déclarer une cible . . . 5
- Cible gelée : restart ou alert . . . 6
- Commandes du service . . . 6
- Contrôler la configuration sans lancer de cible . . . 6
- Arrêter une cible sans arrêter le service . . . 7
- Faire tourner le service sous un autre utilisateur . . . 7
- Mettre à jour . . . 7
- Retirer et purger . . . 7
- Autres familles : Fedora, openSUSE Leap et Arch Linux . . . 7
- Limites connues du paquet . . . 8

Dépannage **9**
- Lire d'abord l'état du service . . . 9
- Le service ne démarre pas . . . 9
- La cible n'est pas relancée . . . 9
- La cible est déclarée bloquée alors qu'elle tourne . . . 9
- `--diagnose` renvoie 0 mais la cible n'est pas saine . . . 10
- Le double de test ne répond pas . . . 10
- Le journal grossit sans limite . . . 10
- Des cibles en double après un redémarrage du superviseur . . . 10
- Rien dans le journal . . . 10

Présentation

Le programme affiche le résumé de --help dans la langue du système : anglais par défaut, avec fr, de, es et zh traduits.

AllWatch est un micro-superviseur de processus pour Linux, écrit dans un seul script Bash (environ cent soixante-dix lignes utiles de code). Il lance chaque cible au premier plan, mémorise son PID et juge sa santé uniquement à partir de ce que disent le noyau et le réseau : le processus est-il toujours dans /proc, son port TCP accepte-t-il encore une connexion, et quel code de sortie a-t-il laissé. Il redémarre la cible selon une politique propre à chaque cible et refuse de boucler indéfiniment sur une cible qui meurt aussitôt.

La cible est une boîte noire. Elle ne s'enregistre pas auprès d'AllWatch, ne lui envoie rien et n'a pas besoin de savoir qu'il existe.

Installer sur Debian et Ubuntu

Le paquet Debian installe le superviseur sous /usr/bin/allwatch, l'unité systemd allwatch.service, la configuration d'exemple et la page de manuel en cinq langues. Téléchargez le paquet et SHA256SUMS depuis la page de téléchargement, puis lancez :

```
sudo apt install ./allwatch_0.1.0-7_all.deb
```

Niveau annoncé : **installation et diagnostic validés en conteneur**, sur Debian 12 et 13 et sur Ubuntu 22.04 et 24.04. Le paquet Debian a aussi été éprouvé en vrai : installé comme service sur un Raspberry Pi 4 sous Debian 13, avec une application réelle, pendant deux heures puis six heures (95 événements, 95 réussis, aucun échec), puis retiré et purgé. Le service est installé et activé, avec une configuration qui ne déclare aucune cible ; ajoutez vos cibles dans /etc/allwatch/allwatch.conf, puis lancez sudo systemctl restart allwatch.

Vérifiez la version installée et la configuration sans lancer de cible :

```
allwatch --version
allwatch --help
allwatch --diagnose /etc/allwatch/allwatch.conf
```

--diagnose lit la configuration, le répertoire d'état et le chemin du journal. Il ne lance ni ne sonde aucune cible : il ne prouve pas qu'une cible est saine, il prouve que le superviseur peut lire sa configuration et utiliser ses chemins.

Pour retirer le paquet :

```
sudo apt remove allwatch
sudo apt purge allwatch
```

apt remove arrête le service mais le laisse activé ; apt purge retire aussi la configuration et l'état du service.

Installer sur Fedora, openSUSE Leap et Arch Linux

Le même tag construit aussi des paquets natifs : un .rpm pour Fedora, un .rpm pour openSUSE Leap et un .pkg.tar.zst pour Arch Linux. Téléchargez le fichier de votre famille et SHA256SUMS depuis la page de téléchargement, puis lancez la ligne de votre famille (Fedora, openSUSE Leap, Arch Linux, dans cet ordre) :

```
sudo dnf install ./allwatch-0.1.0-1.fc44.noarch.rpm
sudo zypper install --allow-unsigned-rpm ./allwatch-0.1.0-1.leap160.noarch.rpm
sudo pacman -U ./allwatch-0.1.0-1-any.pkg.tar.zst
```

Niveau annoncé : **installation et diagnostic validés en conteneur**, sur Fedora 43, 44 et 45, sur openSUSE Leap 16.0 et sur Arch Linux. Les paquets ne sont pas présentés comme testés sur un serveur physique. Le service est installé mais pas démarré : ajoutez vos cibles à /etc/allwatch/allwatch.conf, puis lancez sudo systemctl enable --now allwatch. Les commandes allwatch --version, --help et --diagnose sont celles montrées plus haut.

Pour retirer le paquet :

```
sudo dnf remove allwatch
sudo zypper remove allwatch
sudo pacman -R allwatch
```

Le retrait garde l'utilisateur système allwatch et /var/lib/allwatch, ainsi que votre configuration si vous l'avez modifiée (sauvegardée en allwatch.conf.rpmsave ou allwatch.conf.pacsave). Pour les effacer aussi :

```
sudo userdel allwatch
sudo rm -rf /etc/allwatch /var/lib/allwatch
```

Exécution depuis le dépôt de code

Prérequis : Linux (le superviseur lit /proc), Bash 5 ou plus récent, et timeout et awk parmi les utilitaires de base habituels. Pour le double de test dummy-target.sh seulement : setpriv (util-linux) et OpenBSD nc.

```
git clone https://github.com/RonanDavalan/AllWatch.git
cd AllWatch
./allwatch.sh --version
./allwatch.sh --help
./allwatch.sh allwatch.conf
```

La configuration d'exemple lance dummy-target.sh sur le port 9001 sous l'identifiant demo. Dans un second terminal, examinez la cible :

```
cat /tmp/allwatch-runtime/demo.pid
: > /dev/tcp/127.0.0.1/9001 && echo "le port 9001 répond"
```

Éliminez la cible de manière radicale et observez-la revenir avec un nouveau PID (l'intervalle d'interrogation est de 5 secondes) :

```
kill -9 "$(cat /tmp/allwatch-runtime/demo.pid)"
sleep 6
cat /tmp/allwatch-runtime/demo.pid
```

Provoquez le blocage de la cible (elle reste active mais ferme son port). AllWatch le détecte, l'arrête et la redémarre, ce qui prend environ quinze secondes :

```
kill -USR1 "$(cat /tmp/allwatch-runtime/demo.pid)"
sleep 20
grep ' demo ' ~/.local/state/allwatch/allwatch.log | tail -n 6
```

Arrêtez le superviseur avec Ctrl+C (ou SIGTERM) ; il arrête ses cibles et vide son répertoire d'état :

```
ls -A /tmp/allwatch-runtime
```

Comment ça marche

AllWatch est le parent, la cible est l'enfant. Toutes les quelques secondes, il examine chaque cible et la place dans l'un des quatre états suivants :

- **En cours** : le processus existe et son port accepte une connexion.
- **Bloqué** : le processus existe mais son port ne répond pas. Avec ON_FREEZE=restart (le défaut), AllWatch envoie SIGTERM, attend l'arrêt de la cible (5 secondes), envoie SIGKILL si nécessaire, puis considère la cible comme défaillante. Avec ON_FREEZE=alert, il n'envoie aucun signal, journalise ALERTE GEL une fois, laisse la cible surveillée et journalise RETABLI quand le port répond de nouveau.
- **Plantage** : le processus a disparu sans arrêt délibéré. AllWatch lit le code de sortie et redémarre la cible si sa politique le demande.

- **Quitter** : la cible a terminé et sa politique demande de la laisser arrêtée, ou la limite de redémarrage a été atteinte.

Configuration

La configuration est lue une seule fois, au démarrage, ligne par ligne. Les lignes vides et les lignes commençant par # sont ignorées. Paramètres du moteur, une ligne KEY=value chacun :

- POLL_SECONDS=5 : secondes entre deux vérifications.
- PROBE_TIMEOUT=3 : secondes dont dispose un port pour répondre à la sonde.
- MAX_RESTARTS=5 : redémarrages autorisés dans la fenêtre ci-dessous.
- RESTART_WINDOW=60 : durée de cette fenêtre, en secondes.
- ON_FREEZE=restart : une cible bloquée est neutralisée et relancée ; alert la laisse intacte et journalise ALERTE GEL. Toute autre valeur est refusée au démarrage avec le code 1.

Cibles, une par ligne :

```
id port policy command...
```

- id : un nom court, utilisé pour les fichiers d'état et le journal.
- port : le port TCP sur lequel la cible écoute (sondé sur 127.0.0.1).
- policy : l'une des valeurs always, on-failure, transient.
- command... : le reste de la ligne, séparé par des espaces (sans guillemets), exécuté au premier plan depuis le répertoire où allwatch.sh a été lancé.

Politiques :

- always : redémarrer la cible chaque fois qu'elle se termine, même avec le code de sortie 0, sauf si un fichier <id>.stop existe dans le répertoire d'état.
- on-failure : redémarrer après un plantage ou un blocage, mais laisser arrêtée après un code de sortie 0.
- transient : ne jamais la redémarrer.

Si une cible atteint MAX_RESTARTS redémarrages dans les RESTART_WINDOW secondes, AllWatch cesse de la redémarrer, enregistre ALERTE VERROU_CRITIQUE et continue avec les autres cibles.

Le répertoire d'état est /tmp/allwatch-runtime (à remplacer avec ALLWATCH_RUNTIME) ; le journal est un fichier plat, une ligne par événement (date id event), dans ${XDG_STATE_HOME:-$HOME/.local/state}/allwatch/allwatch.log.

Le contrat qu'une cible doit respecter

AllWatch ne s'adapte à rien ; la cible qui rompt le contrat est en faute.

- **Premier plan.** La cible s'exécute au premier plan du processus lancé par AllWatch. Elle ne doit pas se transformer en processus démon ni créer de double processus : AllWatch verrait alors le premier processus se terminer et en lancerait un duplicata.
- **Codes de sortie.** Un arrêt propre et demandé se termine par le code 0. Toute erreur se termine par un code différent de zéro.
- **Un port TCP.** Chaque cible possède un port TCP, ouvert et acceptant les connexions dès son initialisation. Une cible active qui n'accepte pas de connexion dans le délai de la sonde est considérée comme bloquée.
- **SIGTERM.** La cible intercepte SIGTERM, libère ses ressources et se termine avec le code 0 en moins de 5 secondes. Ensuite, AllWatch envoie SIGKILL.

Le titre de la fenêtre, le serveur d'affichage et la sortie propre à la cible ne jouent aucun rôle dans le contrôle d'intégrité.

Limites

- Aucun rechargement à chaud de la configuration : sudo systemctl restart allwatch.

- Aucune vérification fondée sur les titres de fenêtre, X11 ou Wayland.
- Aucune analyse des journaux d'application pour décider d'un redémarrage.
- Aucune surveillance d'une cible qui se transforme en démon ou fait un double fork.
- --diagnose est statique : il lit la configuration et les chemins, il ne sonde aucune cible et ne prouve pas qu'une cible est saine.
- --version, --help et --diagnose ne sont reconnus qu'en première position.
- La commande n'est pas exécutée par un shell : ni guillemets, ni tubes, ni redirections, ni expansion de variables.
- Les valeurs numériques de configuration et les noms de politique ne sont pas validés. Un POLL_SECONDS non numérique arrête le superviseur avec le code 1 ; une politique mal orthographiée agit comme always. ON_FREEZE est validé : une valeur inconnue arrête le superviseur avec le code 1.
- Le journal plat n'est pas rotaté.

Licence

MIT. Voir LICENSE.

Crédits

- **Ronan Davalan** : architecte et arbitre. Il a défini la vision et le périmètre, arbitré les spécifications et les a validées.
- **Claude Sonnet 5.5** (Anthropic) : seul auteur du code (le superviseur et le double de test), écrit en trois sessions de codage le 13 novembre 2026.
- **DeepSeek V4.1 Flash 1017** : documentation de référence, campagne de validation de six heures et rapport de campagne.
- **Gemini 4 Hydrogène** (Google) : échanges de conception et de réflexion sur le superviseur, idée et conception de l'icône.

Voir CONTRIBUTORS.md.

Installer et exploiter le service Debian

Le paquet Debian installe AllWatch comme service systemd. Le service démarre au boot, lit sa configuration une seule fois et attend si aucune cible n'est déclarée. Ce chapitre donne les chemins, les fichiers et les commandes exactes du paquet.

Ce que le paquet installe

```
/usr/bin/allwatch                          le superviseur
/usr/bin/allwatch.sh                       lien de compatibilité
/usr/lib/systemd/system/allwatch.service   l'unité systemd
/etc/allwatch/allwatch.conf                la configuration
/usr/share/man/man1/allwatch.1.gz          page de manuel, anglais
/usr/share/man/fr/man1/allwatch.1.gz       page de manuel, français
/usr/share/man/de/man1/allwatch.1.gz       page de manuel, allemand
/usr/share/man/es/man1/allwatch.1.gz       page de manuel, espagnol
/usr/share/man/zh_CN/man1/allwatch.1.gz    page de manuel, chinois simplifié
/usr/share/doc/allwatch/README.Debian.gz   notes du paquet
/usr/share/doc/allwatch/examples/allwatch.conf.example
/usr/share/doc/allwatch/examples/allwatch.conf  configuration du dépôt de sources
/usr/share/doc/allwatch/examples/dummy-target.sh
```

Le paquet crée l'utilisateur et le groupe système allwatch (domicile /var/lib/allwatch, shell /usr/sbin/nologin). Le service tourne sous cette identité depuis /var/lib/allwatch.

Installer

```
sudo apt install ./allwatch_0.1.0-7_all.deb
```

Le paquet installe le programme et l'unité. Quand systemd tourne, il active et démarre allwatch.service. La configuration livrée ne contient aucune cible active : le service démarre et attend.

Vérifiez l'état du paquet et la version du programme :

```
dpkg -s allwatch | grep -E '^(Status|Version):'
allwatch --version
```

allwatch --version affiche allwatch 0.1.0 ; la version du paquet est 0.1.0-7.

Chemins et fichiers

- /etc/allwatch/allwatch.conf est un fichier de configuration (conffile) appartenant à root, non inscriptible par l'utilisateur du service. C'est la frontière de confiance : les commandes des cibles s'exécutent avec les privilèges de l'utilisateur du service.
- /var/lib/allwatch porte l'état et le journal, mode 0700, propriétaire allwatch:allwatch, créé par systemd. Le journal plat est /var/lib/allwatch/allwatch/allwatch.log.
- /run/allwatch est le répertoire d'exécution, mode 0700, créé à chaque démarrage du service et retiré à l'arrêt. Il porte <id>.pid, <id>.fails et <id>.stop.
- /var/lib/allwatch et /run/allwatch ne survivent pas à une purge ; /run/allwatch ne survit pas à un arrêt.

Déclarer une cible

Éditez /etc/allwatch/allwatch.conf en root et ajoutez une ligne par cible :

```
id port policy command...
```

Exemple réel avec le double de test installé par le paquet :

```
sudo apt install netcat-openbsd
sudo cp /usr/share/doc/allwatch/examples/allwatch.conf.example /etc/allwatch/allwatch.conf
sudo systemctl restart allwatch
```

L'exemple déclare :

```
demo 9001 on-failure /usr/share/doc/allwatch/examples/dummy-target.sh 9001
```

Vérifiez le port :

```
bash -c ': > /dev/tcp/127.0.0.1/9001' && echo "le port 9001 répond"
```

L'utilisateur du service doit pouvoir exécuter la commande. Employez des chemins absolus : la commande est découpée aux espaces, sans guillemets, et n'est pas exécutée par un shell. Pour des arguments contenant des espaces ou de la syntaxe shell, écrivez un script d'enveloppe et appelez-le.

Cible gelée : restart ou alert

ON_FREEZE décide du sort d'une cible encore vivante dont le port TCP ne répond plus après le délai de grâce. ON_FREEZE=restart (le défaut livré) envoie SIGTERM, attend environ cinq secondes, envoie SIGKILL si nécessaire, puis relance la cible selon sa politique. ON_FREEZE=alert n'envoie aucun signal : il journalise ALERTE GEL une fois, laisse la cible vivante et journalise RETABLI quand le port répond de nouveau. Un crash suit la politique de la cible dans les deux cas. Toute autre valeur est refusée au démarrage avec le code 1.

Commandes du service

```
sudo systemctl status allwatch
sudo systemctl start allwatch
sudo systemctl stop allwatch
sudo systemctl restart allwatch
sudo systemctl enable allwatch
sudo systemctl disable allwatch
sudo journalctl -u allwatch -f
sudo tail -f /var/lib/allwatch/allwatch/allwatch.log
```

La configuration est lue une fois au démarrage ; une modification exige sudo systemctl restart allwatch. Il n'y a pas de rechargement à chaud.

Contrôler la configuration sans lancer de cible

Lancez --diagnose avec l'environnement du service et sous l'utilisateur du service :

```
sudo -u allwatch env ALLWATCH_RUNTIME=/run/allwatch \
  XDG_STATE_HOME=/var/lib/allwatch \
  allwatch --diagnose /etc/allwatch/allwatch.conf
```

Il affiche la version, le chemin de la configuration, le chemin d'exécution, le chemin du journal, la conduite sur gel et, pour chaque cible, son identifiant, son port et sa politique. Il renvoie 0 quand la configuration est lisible, qu'un répertoire d'exécution existant appartient à l'utilisateur courant et qu'un répertoire de journal existant est inscriptible. Il renvoie 1 pour une configuration illisible, un ON_FREEZE invalide, un POLL_SECONDS non numérique, un répertoire d'exécution existant qui n'appartient pas à l'utilisateur courant, ou un répertoire de journal existant qui n'est pas inscriptible. Il ne lance aucune cible et ne sonde aucun port.

Arrêter une cible sans arrêter le service

Créez le fichier d'arrêt sous l'utilisateur du service :

```
sudo -u allwatch touch /run/allwatch/demo.stop
```

AllWatch retire le fichier quand la cible se termine et ne la relance pas. Le fichier n'arrête pas à lui seul une cible en cours : envoyez SIGTERM à la cible ou laissez-la se terminer.

Faire tourner le service sous un autre utilisateur

Le service tourne sous allwatch. Si les cibles ont besoin de root ou d'un autre compte, créez un fichier de dépassement (drop-in) :

```
sudo systemctl edit allwatch
```

Ajoutez :

```
[Service]
User=www-data
Group=www-data
```

Puis redémarrez :

```
sudo systemctl restart allwatch
```

Les commandes des cibles héritent de cette identité. Le fichier de configuration reste la propriété de root.

Mettre à jour

```
sudo apt install ./allwatch_0.1.0-7_all.deb
```

/etc/allwatch/allwatch.conf est conservé ; si la version du paquet le modifie, dpkg demande quelle version garder. Le service est redémarré après la mise à jour s'il était actif. /var/lib/allwatch et son journal restent.

Retirer et purger

```
sudo apt remove allwatch
sudo apt purge allwatch
```

apt remove arrête le service, retire le programme, l'unité, les pages de manuel et la documentation du paquet. Il conserve /etc/allwatch/allwatch.conf, /var/lib/allwatch et l'utilisateur et le groupe allwatch. Il ne désactive pas l'unité ; l'état d'activation de deb-systemd-helper reste jusqu'à la purge.

apt purge retire aussi le conffile, /var/lib/allwatch, /etc/allwatch quand il est vide, l'utilisateur et le groupe allwatch quand ils peuvent l'être, et l'état d'activation. Si un processus utilise encore le compte, userdel ou groupdel peut échouer ; vérifiez avec :

```
getent passwd allwatch
getent group allwatch
```

Autres familles : Fedora, openSUSE Leap et Arch Linux

Le .rpm de Fedora et d'openSUSE Leap et le .pkg.tar.zst d'Arch Linux installent les mêmes fichiers aux mêmes chemins que le paquet Debian : /usr/bin/allwatch, /usr/lib/systemd/system/allwatch.service, /etc/allwatch/allwatch.conf, les pages de manuel et /usr/share/doc/allwatch. Installez le fichier de votre famille (Fedora, openSUSE Leap, Arch Linux, dans cet ordre) ; la même ligne met à jour une version installée :

```
sudo dnf install ./allwatch-0.1.0-1.fc44.noarch.rpm
sudo zypper install --allow-unsigned-rpm ./allwatch-0.1.0-1.leap160.noarch.rpm
sudo pacman -U ./allwatch-0.1.0-1-any.pkg.tar.zst
```

Différences avec le paquet Debian : le service est installé mais pas démarré, lancez donc sudo systemctl enable --now allwatch une fois vos cibles déclarées ; /etc/allwatch/allwatch.conf est conservé si vous l'avez modifié, sauvegardé en allwatch.conf.rpmsave (Fedora, openSUSE) ou allwatch.conf.pacsave (Arch) au retrait du paquet ; il n'y a pas de purge, donc le retrait garde l'utilisateur allwatch et /var/lib/allwatch, et le paquet affiche les deux commandes qui les effacent :

```
sudo userdel allwatch
sudo rm -rf /etc/allwatch /var/lib/allwatch
```

Limites connues du paquet

- La validation en conteneur installe le paquet, contrôle --version, --help et --diagnose, puis le retire. Elle ne démarre pas systemd : le service est installé et documenté, mais non validé sous systemd dans le conteneur.
- --diagnose est statique : il ne sonde aucune cible.
- /run/allwatch ne survit pas à un arrêt du service. Après un plantage du superviseur, des cibles peuvent survivre sans fichier PID ; le superviseur redémarré peut en lancer des doublons. Un drop-in avec RuntimeDirectoryPreserve=restart conserve le répertoire à travers un redémarrage du service.

Dépannage

AllWatch n'a pas de commande d'état. L'état d'une cible se lit dans le processus, le port TCP et le journal plat. Ce chapitre donne les commandes exactes des défaillances courantes.

Lire d'abord l'état du service

```
sudo systemctl status allwatch
sudo journalctl -u allwatch -n 50 --no-pager
sudo tail -n 50 /var/lib/allwatch/allwatch/allwatch.log
```

Les lignes du journal sont date id événement. Les événements sont DEMARRAGE, DEMARRE, ADOPTE, PLANTE, ALERTE GEL, RETABLI, NEUTRALISE SIGTERM, NEUTRALISE SIGKILL, CRASHE code=N, RELANCE, ALERTE VERROU_CRITIQUE, QUITTE et ARRET.

Le service ne démarre pas

Le programme écrit ces messages dans la langue du système (LC_ALL, LC_MESSAGES ou LANG) ; l'anglais est la langue par défaut.

- allwatch: configuration illisible : /etc/allwatch/allwatch.conf — le fichier manque ou n'est pas lisible par l'utilisateur du service. Vérifiez sudo -u allwatch test -r /etc/allwatch/allwatch.conf et sudo systemctl cat allwatch.
- allwatch: /run/allwatch n'appartient pas à l'utilisateur — le répertoire d'exécution appartient à un autre utilisateur. Vérifiez ls -ld /run/allwatch /var/lib/allwatch ; l'unité les crée pour allwatch. Si vous avez ajouté un drop-in avec un autre User=, corrigez la propriété ou retirez le drop-in.
- status=203/EXEC — le programme manque ou n'est pas exécutable. Réinstallez le paquet.
- status=1/FAILURE avec une erreur de configuration — lancez --diagnose sous l'utilisateur du service (voir plus bas).

La cible n'est pas relancée

- Politique on-failure : une cible qui se termine avec le code 0 reste arrêtée. C'est le contrat, pas une panne. Utilisez always si elle doit être relancée après un arrêt propre.
- Politique transient : jamais relancée.
- Fichier d'arrêt présent : ls -l /run/allwatch/demo.stop ; retirez-le si l'arrêt n'était pas voulu. Le fichier n'arrête pas à lui seul une cible en cours.
- ALERTE VERROU_CRITIQUE : la cible a atteint MAX_RESTARTS dans RESTART_WINDOW. Corrigez la cause, puis redémarrez le service ; le verrou dure jusqu'au redémarrage du superviseur.
- CRASHE code=N : la cible s'est terminée avec le code N. Lisez son propre journal ; AllWatch ne rapporte que le code.

La cible est déclarée bloquée alors qu'elle tourne

- La sonde est timeout 3 bash -c ': > /dev/tcp/127.0.0.1/9001'. La cible doit écouter sur 127.0.0.1 (ou 0.0.0.0) sur le port déclaré. Une cible qui n'écoute que sur une autre adresse est considérée comme bloquée.
- Mauvais port dans /etc/allwatch/allwatch.conf ; vérifiez avec ss -ltnp | grep 9001.
- La cible est plus lente que PROBE_TIMEOUT=3 ; augmentez-le dans la configuration, puis redémarrez le service.

- La cible est vivante mais son écoute est fermée ; c'est exactement l'état bloqué. Avec ON_FREEZE=restart (le défaut), AllWatch envoie SIGTERM, attend 5 secondes, puis SIGKILL, et la relance. Avec ON_FREEZE=alert, il journalise ALERTE GEL une fois et laisse la cible vivante ; RETABLI est journalisé quand le port répond de nouveau.

`--diagnose` renvoie 0 mais la cible n'est pas saine

--diagnose est statique. Il lit la configuration et les chemins ; il ne sonde aucune cible. Un code de retour 0 prouve seulement que le superviseur peut lire la configuration et utiliser ses répertoires.

Le double de test ne répond pas

- netcat-openbsd manque : sudo apt install netcat-openbsd.
- La commande emploie un chemin relatif. Le service tourne depuis /var/lib/allwatch ; employez le chemin absolu /usr/share/doc/allwatch/examples/dummy-target.sh.
- Le port est déjà utilisé : ss -ltnp | grep 9001.
- L'utilisateur du service ne peut pas exécuter le fichier : sudo -u allwatch test -x /usr/share/doc/allwatch/examples/dummy-target.sh.

Le journal grossit sans limite

AllWatch ne fait pas tourner le journal. Employez logrotate ou un minuteur systemd ; le fichier est /var/lib/allwatch/allwatch/allwatch.log.

Des cibles en double après un redémarrage du superviseur

/run/allwatch est retiré à l'arrêt du service. Une cible peut survivre à un plantage du superviseur sans son fichier PID ; le nouveau superviseur en lance un doublon. Ajoutez un drop-in :

```
sudo systemctl edit allwatch
```

```
[Service]
RuntimeDirectoryPreserve=restart
```

Puis redémarrez. Le répertoire d'exécution est conservé à travers un redémarrage du service, et le nouveau superviseur peut adopter le PID survivant.

Rien dans le journal

Le service tourne sous allwatch et écrit dans /var/lib/allwatch/allwatch/allwatch.log ; vérifiez l'environnement du service avec sudo systemctl show allwatch -p Environment. Si le répertoire n'est pas inscriptible, le journal reste vide ; --diagnose le signale.