

AllWatch

A tiny process supervisor for Linux: foreground targets, kernel and TCP health, per-target restart policies

Manual

Version 0.1.0
Commit ae9cb8f
Commit date 24 November 2026
License MIT

Offline manual, identical to the guides of the repository at the version and commit shown. The text of the repository is the reference.

Contents

Overview **1**
- Install on Debian and Ubuntu . . . 1
- Install on Fedora, openSUSE Leap and Arch Linux . . . 1
- Run from the source repository . . . 2
- How it works . . . 2
- Configuration . . . 2
- The contract a target must respect . . . 3
- Limits . . . 3
- License . . . 4
- Credits . . . 4

Install and run the Debian service **5**
- What the package installs . . . 5
- Install . . . 5
- Paths and files . . . 5
- Declare a target . . . 5
- Frozen target: restart or alert . . . 6
- Service commands . . . 6
- Check the configuration without starting a target . . . 6
- Stop a target without stopping the service . . . 6
- Run as another user . . . 7
- Upgrade . . . 7
- Remove and purge . . . 7
- Other families: Fedora, openSUSE Leap and Arch Linux . . . 7
- Known limits of the package . . . 8

Troubleshooting **9**
- Read the service state first . . . 9
- The service does not start . . . 9
- The target is not restarted . . . 9
- The target is reported hung but it is running . . . 9
- `--diagnose` returns 0 but the target is unhealthy . . . 10
- The test double does not answer . . . 10
- The log grows without limit . . . 10
- Duplicate targets after a supervisor restart . . . 10
- Nothing in the log . . . 10

Overview

The program prints its --help summary in the language of the system: English by default, with fr, de, es and zh translated.

AllWatch is a tiny process supervisor for Linux, written in a single Bash script (about one hundred and seventy useful lines of code). It starts each target in the foreground, remembers its PID, and judges its health only from what the kernel and the network say: is the process still in /proc, does its TCP port still accept a connection, and what exit code did it leave. It restarts the target according to a per-target policy and refuses to loop forever on a target that dies at once.

The target is a black box. It does not register with AllWatch, does not send it anything and does not need to know it exists.

Install on Debian and Ubuntu

The Debian package installs the supervisor as /usr/bin/allwatch, the allwatch.service systemd unit, the example configuration and the manual page in five languages. Download the package and SHA256SUMS from the download page, then run:

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

Level announced: **installation and diagnosis validated in a container**, on Debian 12 and 13 and on Ubuntu 22.04 and 24.04. The Debian package was also run for real: installed as a service on a Raspberry Pi 4 under Debian 13, with a real application, for a two-hour run and then a six-hour run (95 events, 95 passed, none failed), then removed and purged. The service is installed enabled, with a configuration that declares no target; add your targets to /etc/allwatch/allwatch.conf, then run sudo systemctl restart allwatch.

Check the installed version and the configuration without starting a target:

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

--diagnose reads the configuration, the state directory and the log path. It does not start or probe a target, so it does not prove that a target is healthy; it proves that the supervisor can read its configuration and use its paths.

To remove the package:

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

apt remove stops the service but leaves it enabled; apt purge also removes the configuration and the service state.

Install on Fedora, openSUSE Leap and Arch Linux

The same tag also builds native packages: an .rpm for Fedora, an .rpm for openSUSE Leap and a .pkg.tar.zst for Arch Linux. Download the file for your family and SHA256SUMS from the download page, then run the line of your family (Fedora, openSUSE Leap, Arch Linux, in this order):

```
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
```

Level announced: **installation and diagnosis validated in a container**, on Fedora 43, 44 and 45, on openSUSE Leap 16.0 and on Arch Linux. The packages are not presented as tested on a physical server. The service is installed but not started: add your targets to /etc/allwatch/allwatch.conf, then

run sudo systemctl enable --now allwatch. The commands allwatch --version, --help and --diagnose are the ones shown above.

To remove the package:

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

Removal keeps the allwatch system user and /var/lib/allwatch, and your configuration when you edited it (saved as allwatch.conf.rpmsave or allwatch.conf.pacsave). To erase them as well:

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

Run from the source repository

Requirements: Linux (the supervisor reads /proc), Bash 5 or later, and timeout and awk from the usual core utilities. For the test double dummy-target.sh only: setpriv (util-linux) and OpenBSD nc.

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

The example configuration launches dummy-target.sh on port 9001 under the id demo. In a second terminal, look at the target:

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

Kill the target the hard way and see it come back with a new PID (the poll interval is 5 seconds):

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

Make the target hang (it stays alive but closes its port). AllWatch notices, stops it and restarts it, which takes about fifteen seconds:

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

Stop the supervisor with Ctrl+C (or SIGTERM); it stops its targets and empties its state directory:

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

How it works

AllWatch is the parent, the target is the child. Every few seconds it looks at each target and puts it in one of four states:

- **Up**: the process exists and its port accepts a connection.
- **Hung**: the process exists but its port does not answer. With ON_FREEZE=restart (the default), AllWatch sends SIGTERM, waits for the target to stop (5 seconds), sends SIGKILL if needed, then treats the target as failed. With ON_FREEZE=alert, it sends no signal, logs ALERTE GEL once, keeps the target under watch and logs RETABLI when the port answers again.
- **Crashed**: the process is gone without a deliberate stop. AllWatch reads the exit code and restarts the target if its policy says so.
- **Quit**: the target ended and its policy says to leave it alone, or the restart limit was reached.

Configuration

The configuration is read once, at startup, line by line. Empty lines and lines starting with # are ignored. Engine parameters, one KEY=value per line:

- POLL_SECONDS=5: seconds between two checks.
- PROBE_TIMEOUT=3: seconds a port has to answer the probe.
- MAX_RESTARTS=5: restarts allowed inside the window below.
- RESTART_WINDOW=60: length of that window, in seconds.
- ON_FREEZE=restart: a hung target is neutralized and restarted; alert leaves it untouched and logs ALERTE GEL. Any other value is refused at startup with exit code 1.

Targets, one per line:

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

- id: a short name, used for the state files and the log.
- port: the TCP port the target listens on (probed on 127.0.0.1).
- policy: one of always, on-failure, transient.
- command...: the rest of the line, split on spaces (no quoting), run in the foreground from the directory where allwatch.sh was started.

Policies:

- always: restart the target whenever it ends, even with exit code 0, unless a file <id>.stop exists in the state directory.
- on-failure: restart it after a crash or a hang, but leave it stopped after an exit code 0.
- transient: never restart it.

If a target reaches MAX_RESTARTS restarts within RESTART_WINDOW seconds, AllWatch stops restarting it, logs ALERTE VERROU_CRITIQUE and carries on with the other targets.

The state directory is /tmp/allwatch-runtime (override with ALLWATCH_RUNTIME); the log is a flat file, one line per event (date id event), in ${XDG_STATE_HOME:-$HOME/.local/state}/allwatch/allwatch.log.

The contract a target must respect

AllWatch adapts to nothing; a target that breaks the contract is the one at fault.

- **Foreground.** The target runs in the foreground of the process AllWatch starts. It must not daemonize or double-fork: AllWatch would see the first process end and start a duplicate.
- **Exit codes.** A clean, requested stop exits with code 0. Any failure exits with a non-zero code.
- **A TCP port.** Each target owns one TCP port, open and accepting connections as soon as it is initialized. A target that is alive but does not accept a connection within the probe timeout is considered hung.
- **SIGTERM.** The target catches SIGTERM, releases its resources and exits with code 0 within 5 seconds. After that, AllWatch sends SIGKILL.

The window title, the display server and the target's own output play no part in the health check.

Limits

- No hot reload of the configuration: sudo systemctl restart allwatch.
- No check based on window titles, X11 or Wayland.
- No parsing of application logs to decide a restart.
- No supervision of a target that daemonizes or double-forks.
- --diagnose is static: it reads the configuration and the paths, it does not probe a target and it does not prove that a target is healthy.
- --version, --help and --diagnose are recognized only as the first argument.
- The command is not run through a shell: no quoting, pipes, redirections or variable expansion.
- Numeric configuration values and policy names are not validated. A non-numeric POLL_SECONDS stops the supervisor with exit 1; a mistyped policy acts as always. ON_FREEZE is validated: an unknown value stops the supervisor with exit 1.
- The flat log is not rotated.

License

MIT. See LICENSE.

Credits

- **Ronan Davalan**: architect and arbiter. He set the vision and the scope, arbitrated the specifications and validated them.
- **Claude Sonnet 5.5** (Anthropic): sole author of the code (the supervisor and the test double), written in three coding sessions on 13 November 2026.
- **DeepSeek V4.1 Flash 1017**: reference documentation, six-hour validation campaign and campaign report.
- **Gemini 4 Hydrogène** (Google): design and thinking exchanges on the supervisor, idea and design of the icon.

See CONTRIBUTORS.md.

Install and run the Debian service

The Debian package installs AllWatch as a systemd service. The service starts at boot, reads its configuration once, and waits if no target is declared. This chapter gives the exact paths, files and commands of the package.

What the package installs

```
/usr/bin/allwatch                          the supervisor
/usr/bin/allwatch.sh                       compatibility symlink
/usr/lib/systemd/system/allwatch.service   the systemd unit
/etc/allwatch/allwatch.conf                the configuration
/usr/share/man/man1/allwatch.1.gz          manual page, English
/usr/share/man/fr/man1/allwatch.1.gz       manual page, French
/usr/share/man/de/man1/allwatch.1.gz       manual page, German
/usr/share/man/es/man1/allwatch.1.gz       manual page, Spanish
/usr/share/man/zh_CN/man1/allwatch.1.gz    manual page, Simplified Chinese
/usr/share/doc/allwatch/README.Debian.gz   package notes
/usr/share/doc/allwatch/examples/allwatch.conf.example
/usr/share/doc/allwatch/examples/allwatch.conf  source repository configuration
/usr/share/doc/allwatch/examples/dummy-target.sh
```

The package creates the system user and group allwatch (home /var/lib/allwatch, shell /usr/sbin/nologin). The service runs as that user from /var/lib/allwatch.

Install

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

The package installs the program and the unit. When systemd is running, it enables and starts allwatch.service. The shipped configuration has no active target, so the service starts and waits.

Check the package state and the program version:

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

allwatch --version prints allwatch 0.1.0; the package version is 0.1.0-7.

Paths and files

- /etc/allwatch/allwatch.conf is a conffile owned by root and not writable by the service user. It is the trust boundary: target commands run with the service user's privileges.
- /var/lib/allwatch holds the state and the log, mode 0700, owner allwatch:allwatch, created by systemd. The flat log is /var/lib/allwatch/allwatch/allwatch.log.
- /run/allwatch is the runtime directory, mode 0700, created at each service start and removed when the service stops. It holds <id>.pid, <id>.fails and <id>.stop.
- /var/lib/allwatch and /run/allwatch do not survive a purge; /run/allwatch does not survive a stop.

Declare a target

Edit /etc/allwatch/allwatch.conf as root and add one line per target:

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

A real example with the test double installed by the package:

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

The example declares:

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

Check the port:

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

The service user must be able to execute the command. Use absolute paths: the command is split on spaces without quoting and is not run through a shell. For arguments with spaces or shell syntax, write a wrapper script and call it.

Frozen target: restart or alert

ON_FREEZE decides what happens when a target is still alive but its TCP port no longer answers after the grace period. ON_FREEZE=restart (the shipped default) sends SIGTERM, waits about five seconds, sends SIGKILL if needed, and restarts the target according to its policy. ON_FREEZE=alert sends no signal: it logs ALERTE GEL once, leaves the target running and logs RETABLI when the port answers again. A crash follows the target policy under both values. Any other value is refused at startup with exit code 1.

Service commands

```
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
```

The configuration is read once at startup; a change requires sudo systemctl restart allwatch. There is no hot reload.

Check the configuration without starting a target

Run --diagnose with the service environment and as the service user:

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

It prints the version, the configuration path, the runtime path, the log path, the freeze policy and each target's id, port and policy. It exits 0 when the configuration is readable, an existing runtime directory is owned by the current user and an existing log directory is writable. It exits 1 for an unreadable configuration, an invalid ON_FREEZE, a non-numeric POLL_SECONDS, an existing runtime directory not owned by the current user, or an existing log directory that is not writable. It starts no target and probes no port.

Stop a target without stopping the service

Create the stop file as the service user:

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

AllWatch removes the file when the target ends and does not restart it. The file does not stop a running target by itself; send SIGTERM to the target or let it end.

Run as another user

The service runs as allwatch. If the targets need root or another account, create a drop-in:

```
sudo systemctl edit allwatch
```

Add:

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

Then restart:

```
sudo systemctl restart allwatch
```

The target commands inherit that identity. The configuration file remains root-owned.

Upgrade

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

/etc/allwatch/allwatch.conf is preserved; if the packaged version changes, dpkg asks which version to keep. The service is restarted after the upgrade if it was active. /var/lib/allwatch and its log remain.

Remove and purge

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

apt remove stops the service, removes the program, the unit, the manual pages and the package documentation. It keeps /etc/allwatch/allwatch.conf, /var/lib/allwatch and the allwatch user and group. It does not disable the unit; the deb-systemd-helper enablement state remains until purge.

apt purge also removes the conffile, /var/lib/allwatch, /etc/allwatch when empty, the allwatch user and group when they can be removed, and the enablement state. If a process still uses the account, userdel or groupdel may fail; check with:

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

Other families: Fedora, openSUSE Leap and Arch Linux

The Fedora and openSUSE Leap .rpm and the Arch Linux .pkg.tar.zst install the same files at the same paths as the Debian package: /usr/bin/allwatch, /usr/lib/systemd/system/allwatch.service, /etc/allwatch/allwatch.conf, the manual pages and /usr/share/doc/allwatch. Install the file of your family (Fedora, openSUSE Leap, Arch Linux, in this order); the same line upgrades an installed version:

```
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
```

Differences from the Debian package: the service is installed but not started, so run sudo systemctl enable --now allwatch once your targets are declared; /etc/allwatch/allwatch.conf is kept when you edited it and saved as allwatch.conf.rpmsave (Fedora, openSUSE) or allwatch.conf.pacsave (Arch) when the package is removed; there is no purge, so removal keeps the allwatch user and /var/lib/allwatch, and the package prints the two commands that erase them:

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

Known limits of the package

- The container validation installs the package, checks --version, --help and --diagnose, then removes it. It does not start systemd, so the service is installed and documented but not validated under systemd in the container.
- --diagnose is static: it does not probe targets.
- /run/allwatch does not survive a service stop. After a supervisor crash, targets may survive without PID files; the restarted supervisor can start duplicates. A drop-in with RuntimeDirectoryPreserve=restart keeps the directory across a service restart.

Troubleshooting

AllWatch has no status command. The state of a target is read from the process, the TCP port and the flat log. This chapter gives the exact commands for the common failures.

Read the service state first

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

The log lines are date id event. The events are DEMARRAGE, DEMARRE, ADOPTE, PLANTE, ALERTE GEL, RETABLI, NEUTRALISE SIGTERM, NEUTRALISE SIGKILL, CRASHE code=N, RELANCE, ALERTE VERROU_CRITIQUE, QUITTE and ARRET.

The service does not start

The program writes these messages in the language of the system (LC_ALL, LC_MESSAGES or LANG); English is the default.

- allwatch: cannot read configuration: /etc/allwatch/allwatch.conf — the file is missing or not readable by the service user. Check sudo -u allwatch test -r /etc/allwatch/allwatch.conf and sudo systemctl cat allwatch.
- allwatch: /run/allwatch is not owned by the current user — the runtime directory is owned by another user. Check ls -ld /run/allwatch /var/lib/allwatch; the unit creates them as allwatch. If you added a drop-in with another User=, fix the ownership or remove the drop-in.
- status=203/EXEC — the program is missing or not executable. Reinstall the package.
- status=1/FAILURE with a configuration error — run --diagnose as the service user (see below).

The target is not restarted

- Policy on-failure: a target that exits with code 0 is left stopped. This is the contract, not a failure. Use always if it must restart after a clean exit.
- Policy transient: never restarted.
- Stop file present: ls -l /run/allwatch/demo.stop; remove it if the stop was not intended. The file itself does not stop a running target.
- ALERTE VERROU_CRITIQUE: the target reached MAX_RESTARTS in RESTART_WINDOW. Fix the cause, then restart the service; the lock lasts until the supervisor restarts.
- CRASHE code=N: the target exited with code N. Read its own log; AllWatch only reports the code.

The target is reported hung but it is running

- The probe is timeout 3 bash -c ': > /dev/tcp/127.0.0.1/9001'. The target must listen on 127.0.0.1 (or 0.0.0.0) on the declared port. A target that listens only on another address is considered hung.
- Wrong port in /etc/allwatch/allwatch.conf; check with ss -ltnp | grep 9001.
- The target is slower than PROBE_TIMEOUT=3; raise it in the configuration, then restart the service.

- The target is alive but its listener is closed; this is exactly the hang state. With ON_FREEZE=restart (the default), AllWatch sends SIGTERM, waits 5 seconds, then SIGKILL, and restarts it. With ON_FREEZE=alert, it logs ALERTE GEL once and leaves the target running; RETABLI is logged when the port answers again.

`--diagnose` returns 0 but the target is unhealthy

--diagnose is static. It reads the configuration and the paths; it does not probe a target. A return code 0 only proves that the supervisor can read the configuration and use its directories.

The test double does not answer

- netcat-openbsd is missing: sudo apt install netcat-openbsd.
- The command uses a relative path. The service runs from /var/lib/allwatch; use the absolute path /usr/share/doc/allwatch/examples/dummy-target.sh.
- The port is already used: ss -ltnp | grep 9001.
- The service user cannot execute the file: sudo -u allwatch test -x /usr/share/doc/allwatch/examples/dummy-target.sh.

The log grows without limit

AllWatch does not rotate the log. Use logrotate or a systemd timer; the file is /var/lib/allwatch/allwatch/allwatch.log.

Duplicate targets after a supervisor restart

/run/allwatch is removed when the service stops. A target may survive a supervisor crash without its PID file; the new supervisor starts a duplicate. Add a drop-in:

```
sudo systemctl edit allwatch
```

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

Then restart. The runtime directory is kept across a service restart, so the new supervisor can adopt the surviving PID.

Nothing in the log

The service runs as allwatch and writes to /var/lib/allwatch/allwatch/allwatch.log; check the service environment with sudo systemctl show allwatch -p Environment. If the directory is not writable, the log stays empty; --diagnose reports it.