RAIDWatch - RAID- und Pool-Überwachung auf UGREEN NAS (UGOS Pro)

Handbuch (DE/EN) - Version v1.01 - Stand 08.02.2026

Community-Lösung, kein offizielles UGREEN-Produkt. Verwendung auf eigene Verantwortung.

Note: The English manual starts on page 8.

0. Schnellstart (DE)

- Ordner anlegen (z.B. /volume1/docker/raidwatch) und raidwatch.sh + raidwatch.env hineinkopieren.
- raidwatch.sh ausführbar machen: chmod +x /volume1/docker/raidwatch/raidwatch.sh
- raidwatch.env anpassen (SMTP, MAIL_TO, REPORT_LANG).
- Testmail senden: /volume1/docker/raidwatch/raidwatch.sh --test-mail
- Automatisierung einrichten (Cron), empfohlen: --once alle 5 Minuten.

0.1 Changelog (DE)

v1.01 (08.02.2026)

Neu:

- Cron: PATH verbessert, damit mdadm auch unter Cron gefunden wird.
- HTML-Report: komplette Tabellenzeile bei ALARM/DEGRADED/REBUILD eingefärbt.
- Mail-Footer: Skript-Version v1.01.

Behoben:

- MD RAID: Defekt/Reserve korrekt ermittelt, inklusive spare rebuilding.
- Rebuild: Fortschritt sowie Speed/ETA robuster aus /proc/mdstat gelesen.
- Doku: RAIDWATCH_ENV Beispiel auf raidwatch.env geändert, Link zu crontab-generator.org korrigiert.

1. Einleitung

RAIDWatch ist ein Bash-Skript zur Überwachung von RAID- und Storage-Status auf UGREEN NAS. Es prüft je nach Konfiguration MD RAID (mdadm), BTRFS sowie optional ZFS, erstellt einen Outlook-freundlichen Report und versendet ihn per SMTP.

2. Funktionsumfang

- MD RAID Monitoring (mdadm) inklusive Rebuild-/Degraded-Erkennung.
- BTRFS Monitoring (Pools/Volumes), sofern vorhanden.
- Optional ZFS-Check (standardmäßig deaktiviert).
- HTML- oder Text-Report per SMTP (Outlook freundlich).
- Stabile Schacht-Zuordnung bei MD RAID über mdadm RaidDevice-Slot (robust bei Hot-Swap).
- Baymap-Datei (Serial -> Bay) als Zusatzinfo, um bei fehlenden Disks Modell und Seriennummer weiter anzeigen zu können.
- Statuslogik: OK, INFO, ALARM, RECOVERY (inkl. Cooldown und optionalem Versand bei Statusänderungen).
- Log-Rotation und Log-Retention.

3. Paket-Inhalt und Ordnerstruktur

Im Paket enthalten:

- raidwatch.sh - Hauptskript (Checks, Report, SMTP-Versand).
- raidwatch.env - Konfiguration (Sprache, SMTP, Checks, Logging).
- raidwatch-logs/ - wird automatisch erstellt (Logs, State, Baymap).

Empfohlener Speicherort ist ein Ordner auf einem Daten-Volume, z.B.:

```
/volume1/docker/raidwatch/
```

4. Voraussetzungen

- UGREEN NAS mit UGOS Pro.
- SSH-Zugriff (für Installation und Scheduling).
- Bash, mdadm, coreutils (auf UGOS in der Regel vorhanden).
- Python 3 (SMTP-Versand via smtplib) (auf UGOS in der Regel vorhanden).
- Für BTRFS-Checks: btrfs-Tools (meist vorhanden, wenn BTRFS genutzt wird).

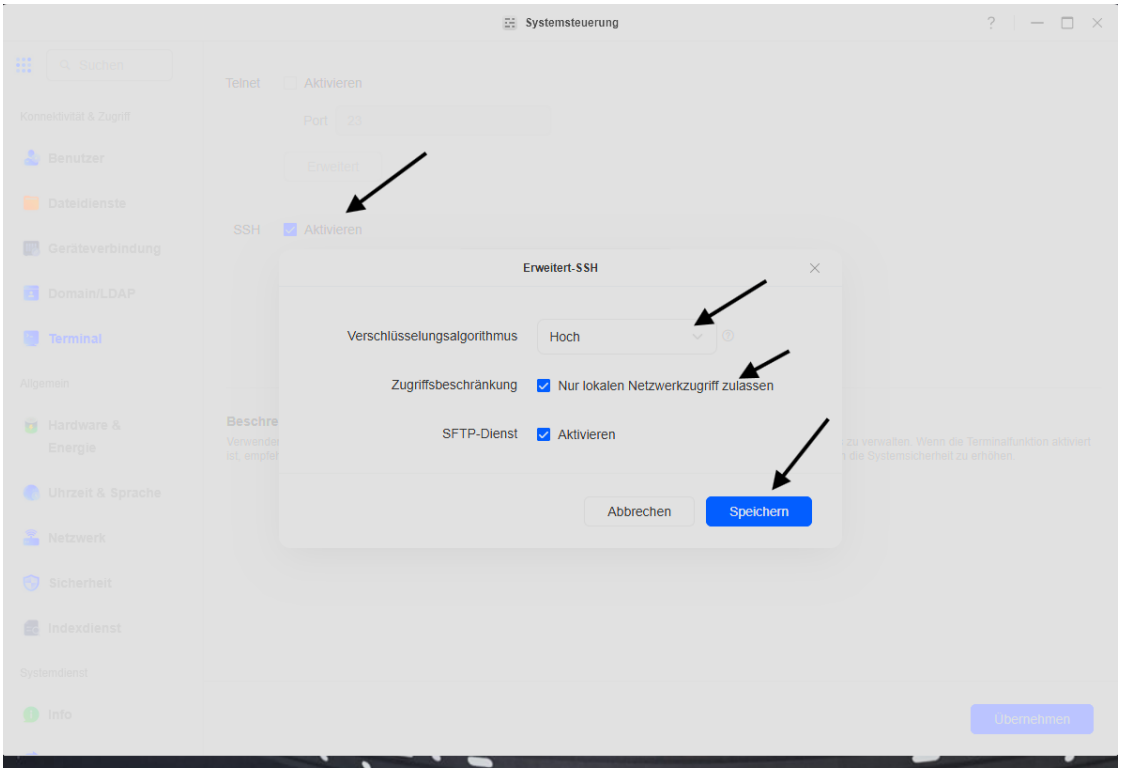
Optional: Zugriff auf einen SMTP-Server (Port 587 STARTTLS oder 465 SMTPS).

5. Installation auf UGREEN NAS

Schnellstart (7 Schritte)

1. Dateien auf das NAS kopieren (z.B. nach /volume1/docker/raidwatch/).
2. raidwatch.env anpassen (MAIL_TO, SMTP_* , REPORT_LANG usw.).
3. SSH auf dem NAS aktivieren: Systemsteuerung -> Terminal -> SSH aktivieren. Empfehlung: "Nur lokalen Netzwerkzugriff zulassen".

Beispiel (UGOS Pro): SSH nur im lokalen Netzwerk aktivieren.



4. PuTTY Verbindung zum NAS herstellen (Windows). Beispiel: Host/IP 192.168.120.4, Port 22.
5. Root-Rechte erlangen: sudo -i
6. Skript testen: ./raidwatch.sh --test-mail und danach ./raidwatch.sh --once --force
7. Cronjob anlegen: crontab -e und einen Eintrag hinzufügen (z.B. alle 5 Minuten). Weitere Infos: https://crontab-generator.org/

Beispiel Cron-Eintrag:

```
*/5 * * * * /volume1/docker/raidwatch/raidwatch.sh --once >/dev/null 2>&1
```

5.1 Dateien kopieren

1) Projektordner anlegen (Beispiel):

```
mkdir -p /volume1/docker/raidwatch
```

2) raidwatch.sh und raidwatch.env in diesen Ordner kopieren (per SMB oder Datei-App).

3) Skript ausführbar machen:

```
chmod +x /volume1/docker/raidwatch/raidwatch.sh
```

5.2 Konfiguration raidwatch.env

Öffne raidwatch.env und passe die Variablen an. Kurzüberblick (vollständige Liste siehe unten):

Variable	Beschreibung	Beispiel/Default
NAS_NAME	Anzeigename im Betreff/Report.	DH4300PLUS
REPORT_LANG	de oder en.	de
SMTP_SERVER	SMTP Hostname/IP.	smtp.example.com
SMTP_PORT	Port 587 (STARTTLS) oder 465 (SMTPS).	587
SMTP_USER	SMTP Benutzer.	nas@example.com
SMTP_PASS	SMTP Passwort (App-Passwort empfohlen).	********
MAIL_TO	Empfänger (kommagetrennt möglich).	you@example.com

5.3 Vollständige ENV-Referenz

Alle Optionen aus raidwatch.env (Defaultwerte siehe Datei):

Bereich	Variablen (Auszug)	Kurzbeschreibung
Allgemein	NAS_NAME, REPORT_LANG, SUBJECT_PREFIX	Name, Sprache, Betreff-Präfix (Betreff max. 64 Zeichen).
SMTP	SMTP_SERVER, SMTP_PORT, SMTP_USER, SMTP_PASS	SMTP Verbindung und Login.
Mail	MAIL_FROM, MAIL_TO, MAIL_FORMAT, HTML_SHOW_DETAILS	Absender/Empfänger, HTML/Text, Detailsblock.
Checks	CHECK_MDRAID, CHECK_BTRFS, CHECK_ZFS	Aktiviert/Deaktiviert einzelne Checks.
Alarm	ALERT_ON_CHANGE, ALERT_COOLDOWN_SECONDS	Mail bei Änderungen, Cooldown gegen Spam.
Logging	LOG_DIR, LOG_RETENTION_DAYS, LOG_MAX_MB, LOG_ROTATE_COUNT	Log-Verzeichnis und Rotation.
Loop	SLEEP_SECONDS	Intervall bei --loop.

6. Verwendung

6.1 Hilfe und Test

Hilfe anzeigen:

```
/volume1/docker/raidwatch/raidwatch.sh --help
```

Testmail (ohne Checks):

```
/volume1/docker/raidwatch/raidwatch.sh --test-mail
```

Einmaliger Check (empfohlen für Cron/UGOS Task Scheduler):

```
/volume1/docker/raidwatch/raidwatch.sh --once
```

Mail immer senden (auch ohne Änderung), z.B. zum Testen:

```
/volume1/docker/raidwatch/raidwatch.sh --once --force
```

6.2 Dauerschleife

Optional kann das Skript in einer Schleife laufen. Intervall über SLEEP_SECONDS:

```
/volume1/docker/raidwatch/raidwatch.sh --loop
```

Hinweis: Für Dauerbetrieb ist Cron meist robuster als ein Hintergrundprozess.

6.3 Sprache per ENV/CLI

REPORT_LANG kann in raidwatch.env gesetzt werden oder temporär per CLI überschrieben werden:

```
REPORT_LANG=en /volume1/docker/raidwatch/raidwatch.sh --help
```

Die CLI-Variante hat Priorität vor der env-Datei. Das gilt auch für --help.

6.4 Alternative ENV-Datei

Wenn du mehrere Konfigurationen nutzen möchtest, kannst du den Pfad per RAIDWATCH_ENV setzen:

```
RAIDWATCH_ENV=/volume1/docker/raidwatch/raidwatch.env /volume1/docker/raidwatch/raidwatch.sh --once
```

7. Checks und Statuslogik

7.1 Status (OK, INFO, ALARM, RECOVERY)

- OK: kein Problem erkannt, normalerweise keine Mail.
- ALARM: Problem erkannt (z.B. RAID degradiert). Mail sofort, danach nach Cooldown und optional bei Änderungen.
- RECOVERY: vorher ALARM, jetzt wieder OK. Mail einmalig.
- INFO: Änderung ohne Problem (nur wenn ALERT_ON_CHANGE=1).

7.2 MD RAID (mdadm)

RAIDWatch liest /proc/mdstat und mdadm --detail. Status wie OK, DEGRADED oder Rebuild/Resync wird erkannt.

Die Schacht-Zuordnung erfolgt bei MD RAID über die mdadm-Spalte RaidDevice (Slotindex + 1). Dadurch bleibt die Bay-Nummer stabil, selbst wenn /dev/sdX nach Hot-Swap wechselt.

7.3 Baymap und fehlende Platten

Solange das Array vollständig ist, schreibt RAIDWatch eine Baymap-Datei (raidwatch_baymap.json). Darin wird die Seriennummer einer Platte dem Slot/Bay zugeordnet.

Wenn später eine Platte fehlt, kann RAIDWatch damit weiterhin Modell und Seriennummer für den fehlenden Slot anzeigen. Hinweis: Wenn beim allerersten Start bereits Platten fehlen, kann die Baymap diese Slots nicht historisch füllen.

7.4 BTRFS

Wenn CHECK_BTRFS=1, werden BTRFS-Volumes geprüft. Warnungen/Fehler werden im Report markiert, sofern erkennbar.

7.5 ZFS (optional)

Wenn CHECK_ZFS=1, wird zpool status ausgewertet. Unter UGOS ist ZFS meist nicht aktiv, daher Standard aus.

8. Automatisierung

8.1 Cron (empfohlen)

Beispiel: alle 5 Minuten einen Check (ohne Konsolen-Ausgabe):

```
*/5 * * * * /volume1/docker/raidwatch/raidwatch.sh --once >/dev/null 2>&1
```

Cron bearbeiten (als root):

```
crontab -e
```

8.2 Cooldown und Statusänderungen

ALERT_COOLDOWN_SECONDS verhindert wiederholte Mails bei dauerhaftem Fehler.

ALERT_ON_CHANGE=1 sendet zusätzlich bei Statuswechseln (z.B. Rebuild-Fortschritt).

9. Logs, State und Dateien

- Logdatei: raidwatch_YYYY-MM.log (Rotation nach LOG_MAX_MB).
- State: raidwatch_state.env (letzter Status/Hash für Change-Detection).
- Baymap: raidwatch_baymap.json (Serial -> Slot/Bay, Zusatzinfo).

Empfehlung: LOG_DIR auf ein Daten-Volume legen, um die Systempartition zu schonen.

10. Troubleshooting

- Keine Mail: SMTP-Daten prüfen, Ports 587/465, Firewall, App-Passwort verwenden.
- Testmail OK, aber keine Alarmmails: Prüfe, ob überhaupt ein Problem erkannt wird. Bei OK wird nicht immer gesendet.
- Bay falsch: mdadm --detail prüfen, ob RaidDevice sauber gelistet wird.

Zum Debuggen: Logdatei und Detailsblock im Report prüfen (HTML_SHOW_DETAILS=1).

11. Sicherheitshinweise

- Skript läuft typischerweise als root, daher nur aus vertrauenswürdiger Quelle verwenden.
- RAIDWatch ersetzt kein Backup. Ein degradiertes RAID ist ein Alarm, kein Backup.

12. Update und Deinstallation

Update: raidwatch.sh durch neue Version ersetzen, raidwatch.env behalten. Danach einmal --once ausführen.

Deinstallation: Ordner löschen (inkl. Logs). Cron-Aufgabe entfernen, falls gesetzt.

13. Hinweis

Dieses Paket ist eine Community-Lösung. Keine Gewährleistung. Nutzung auf eigene Verantwortung.

RAIDWatch - RAID and Pool Monitoring for UGREEN NAS (UGOS Pro)

Handbook (DE/EN) - Version v1.01 - Updated 08.02.2026

Community solution, not an official UGREEN product. Use at your own risk.

0. Quick start (EN)

- Create a folder (for example /volume1/docker/raidwatch) and copy raidwatch.sh + raidwatch.env into it.
- Make the script executable: chmod +x /volume1/docker/raidwatch/raidwatch.sh
- Edit raidwatch.env (SMTP, MAIL_TO, REPORT_LANG).
- Send a test email: /volume1/docker/raidwatch/raidwatch.sh --test-mail
- Schedule it cron. Recommended: run --once every 5 minutes.

0.1 Changelog (EN)

v1.01 (2026-02-08)

Added:

- Cron: improved PATH handling so mdadm is found when running via cron.
- HTML report: full-row highlighting for ALARM/DEGRADED/REBUILD rows.
- Mail footer: script version v1.01.

Fixed:

- MD RAID: correct Failed/Spare counters, including spare rebuilding.
- Rebuild: more reliable parsing of progress and speed/ETA from /proc/mdstat.
- Docs: RAIDWATCH_ENV example uses raidwatch.env, fixed crontab-generator.org link.

1. Introduction

RAIDWatch is a Bash script that monitors storage health on UGREEN NAS. Depending on your settings it checks MD RAID (mdadm), BTRFS, and optionally ZFS, builds an Outlook-friendly report and sends it via SMTP.

2. Features

- MD RAID monitoring (mdadm), including rebuild and degraded detection.
- BTRFS monitoring (pools/volumes) when present.
- Optional ZFS checks (disabled by default).
- SMTP email delivery with HTML or plain text reports (Outlook friendly).
- Stable bay/slot mapping for MD RAID using mdadm RaidDevice slots (robust after hot-swap).
- Baymap file (serial -> bay) as additional info to keep showing model/serial even if a disk is missing.
- Status logic: OK, INFO, ALERT, RECOVERY (with cooldown and optional change notifications).
- Log rotation and retention.

3. Package contents and folder layout

Included files:

- raidwatch.sh - main script (checks, report, SMTP).
- raidwatch.env - configuration (language, SMTP, checks, logging).
- raidwatch-logs/ - auto-created (logs, state, baymap).

Recommended location on a data volume, for example:

```
/volume1/docker/raidwatch/
```

4. Requirements

- UGREEN NAS running UGOS Pro.
- SSH access (for installation and scheduling).
- Bash, mdadm, coreutils (usually present on UGOS).
- Python 3 (SMTP sending via smtplib). (usually present on UGOS).
- For BTRFS checks: btrfs tools (typically available if BTRFS is used).

Optional: access to an SMTP server (587 STARTTLS or 465 SMTPS).

5. Installation on UGREEN NAS

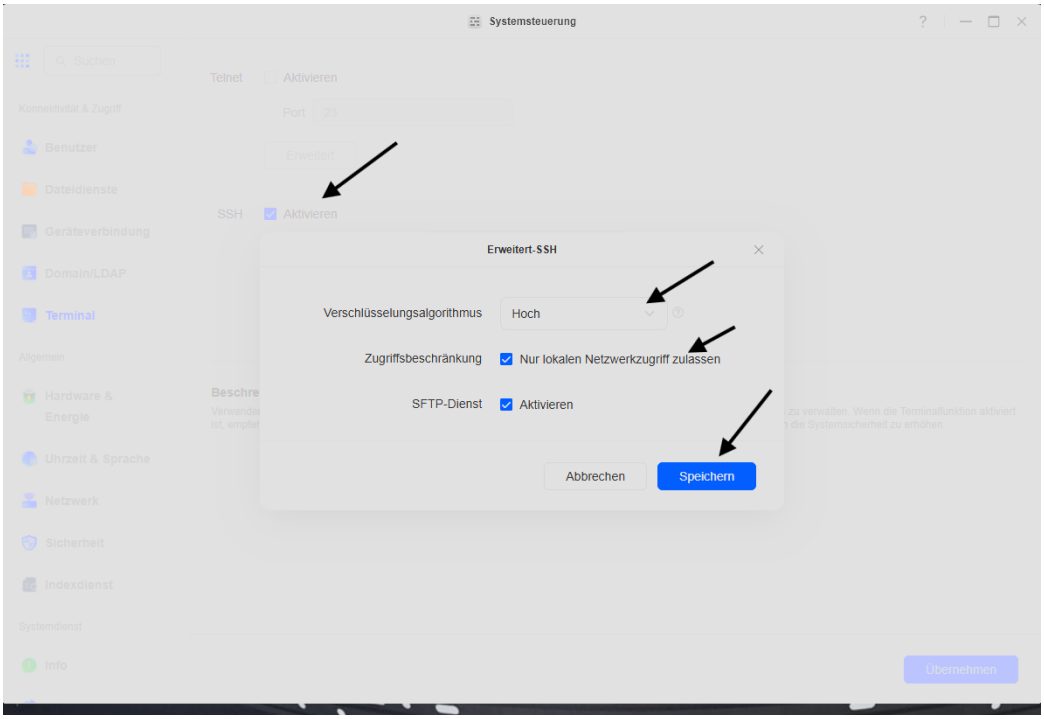
Quick start (7 steps)

1. Copy the files to the NAS (for example to /volume1/docker/raidwatch/).

2. Edit raidwatch.env (MAIL_TO, SMTP_* , REPORT_LANG, etc.).

3. Enable SSH on the NAS: Control Panel -> Terminal -> Enable SSH. Recommendation: allow local network access only.

Example (UGOS Pro): enable SSH for local network access only.



4. Connect via PuTTY (Windows). Example: Host/IP 192.168.120.4, Port 22.

5. Become root: sudo -i

6. Test the script: ./raidwatch.sh --test-mail and then ./raidwatch.sh --once --force

7. Create the cron job: crontab -e and add an entry (for example every 5 minutes). More info: https://crontab-generator.org/

Example cron entry:

```
*/5 * * * * /volume1/docker/raidwatch/raidwatch.sh --once >/dev/null 2>&1
```

5.1 Copy files

1) Create a folder (example):

```
mkdir -p /volume1/docker/raidwatch
```

2) Copy raidwatch.sh and raidwatch.env into that folder (via SMB or the file app).

3) Make the script executable:

```
chmod +x /volume1/docker/raidwatch/raidwatch.sh
```

5.2 Configure raidwatch.env

Edit raidwatch.env and adjust the variables. Quick overview (full list below):

Variable	Description	Example/Default
NAS_NAME	Display name in subject/report.	DH4300PLUS
REPORT_LANG	de or en.	de
SMTP_SERVER	SMTP hostname/IP.	smtp.example.com
SMTP_PORT	587 (STARTTLS) or 465 (SMTPS).	587
SMTP_USER	SMTP user.	nas@example.com
SMTP_PASS	SMTP password (app password recommended).	********
MAIL_TO	Recipient (comma-separated supported).	you@example.com

5.3 Full env reference

All options from raidwatch.env (see file for defaults):

Area	Variables (excerpt)	Short description
General	NAS_NAME, REPORT_LANG, SUBJECT_PREFIX	Name, language, subject prefix (subject max 64 chars).
SMTP	SMTP_SERVER, SMTP_PORT, SMTP_USER, SMTP_PASS	SMTP connection and authentication.
Mail	MAIL_FROM, MAIL_TO, MAIL_FORMAT, HTML_SHOW_DETAILS	From/To, HTML/text, details block.
Checks	CHECK_MDRAID, CHECK_BTRFS, CHECK_ZFS	Enable/disable individual checks.
Alerts	ALERT_ON_CHANGE, ALERT_COOLDOWN_SECONDS	Notify on changes, cooldown against spam.
Logging	LOG_DIR, LOG_RETENTION_DAYS, LOG_MAX_MB, LOG_ROTATE_COUNT	Log directory and rotation.
Loop	SLEEP_SECONDS	Interval for --loop.

6. Usage

6.1 Help and testing

Show help:

```
/volume1/docker/raidwatch/raidwatch.sh --help
```

Send a test email (no checks):

```
/volume1/docker/raidwatch/raidwatch.sh --test-mail
```

Run once recommended for cron:

```
/volume1/docker/raidwatch/raidwatch.sh --once
```

Always send (even without changes), useful for testing:

```
/volume1/docker/raidwatch/raidwatch.sh --once --force
```

6.2 Loop mode

Optionally keep running in a loop. Interval is controlled by SLEEP_SECONDS:

```
/volume1/docker/raidwatch/raidwatch.sh --loop
```

Note: For long-term operation, cron or the UGOS scheduler is usually more robust than a background process.

6.3 Language via ENV/CLI

Set REPORT_LANG in raidwatch.env, or override temporarily on the command line:

```
REPORT_LANG=en /volume1/docker/raidwatch/raidwatch.sh --help
```

The CLI value has priority over the env file, including for --help output.

6.4 Alternate env file

Use RAIDWATCH_ENV to point to a different config file:

```
RAIDWATCH_ENV=/volume1/docker/raidwatch/raidwatch.env /volume1/docker/raidwatch/raidwatch.sh --once
```

7. Checks and status logic

7.1 Status (OK, INFO, ALERT, RECOVERY)

- OK: no problem detected, usually no email.
- ALERT: problem detected (for example degraded RAID). Sends immediately, then after cooldown and optionally on changes.
- RECOVERY: previously ALERT, now back to OK. Sends once.
- INFO: change without an error (only if ALERT_ON_CHANGE=1).

7.2 MD RAID (mdadm)

RAIDWatch reads /proc/mdstat and mdadm --detail and detects states like DEGRADED and rebuild/resync progress.

Bay/slot numbers come from the mdadm RaidDevice slot index (plus 1). This remains stable even if /dev/sdX changes.

7.3 Baymap and missing disks

While the array is healthy, RAIDWatch writes a baymap file (raidwatch_baymap.json) mapping disk serial numbers to bays.

If a disk is later missing, this allows RAIDWatch to keep showing the missing slot's model and serial. If disks are already missing during the first run, those slots cannot be filled historically.

7.4 BTRFS

With CHECK_BTRFS=1, BTRFS volumes are checked and warnings are highlighted in the report when detectable.

7.5 ZFS (optional)

With CHECK_ZFS=1, zpool status is parsed. On UGOS, ZFS is typically not used, so default is off.

8. Scheduling

8.1 Cron (recommended)

Example: run every 5 minutes (no console output):

```
*/5 * * * * /volume1/docker/raidwatch/raidwatch.sh --once >/dev/null 2>&1
```

Edit cron (as root):

```
crontab -e
```

8.2 Cooldown and status changes

ALERT_COOLDOWN_SECONDS prevents repeated emails during persistent failures.

ALERT_ON_CHANGE=1 also sends notifications on status changes (for example rebuild progress).

9. Logs, state and files

- Log file: raidwatch_YYYY-MM.log (rotation via LOG_MAX_MB).
- State: raidwatch_state.env (last status/hash for change detection).
- Baymap: raidwatch_baymap.json (serial -> bay, additional info).

Tip: set LOG_DIR to a data volume to reduce writes to the system partition.

10. Troubleshooting

- No email: verify SMTP settings, ports 587/465, firewall, use an app password.
- Test email works but no alerts: OK states usually do not send. Create a real failure scenario to test alerts.
- Wrong bay: check mdadm --detail output and whether RaidDevice is listed correctly.

For debugging, check the log file and enable the HTML details block (HTML_SHOW_DETAILS=1).

11. Security notes

- The script typically runs as root. Only use it from trusted sources.
- RAIDWatch is not a backup. A degraded RAID is an alert, not a backup strategy.

12. Update and uninstall

Update: replace raidwatch.sh with the new version, keep raidwatch.env. Run --once once afterwards.

Uninstall: delete the folder (including logs). Remove any cron if configured.

13. Disclaimer

This is a community project. No warranty. Use at your own risk.