

# Hintergrund-Prozesse (Supervisor)

## Hintergrund-Prozesse via Supervisor

Wenn dein Projekt einen Prozess braucht, der **dauerhaft im Hintergrund laeuft** (z.B. Symfony Messenger, Laravel Queue-Worker, ein Node.js-Daemon, ein eigener Watcher), kannst du ihn per Supervisor starten lassen. Der Prozess startet dann automatisch mit dem Container und wird bei Absturz neu gestartet — du musst nichts weiter tun.

“ **Cron vs. Supervisor** — Cron startet einen Prozess **zu festen Zeitpunkten** und laesst ihn nach getaner Arbeit sterben. Supervisor startet einen Prozess **einmal** und haelt ihn **dauerhaft am Leben**. Fuer Job-Queues (Messenger, Sidekiq, RQ etc.) willst du Supervisor.

## Wo liegen die Config-Dateien?

Per SSH in deinen Container einloggen und dann:

```
cd ~/.webunit/supervisor.d/
```

Dieses Verzeichnis gehoert dir und ist vor Web-Zugriff geschuetzt (`https://deine-domain/.webunit/...` liefert 403).

Pro Prozess legst du eine `<name>.conf`-Datei an. Der Dateiname ist egal, die Endung `.conf` ist Pflicht.

## Erstes Beispiel: Symfony Messenger

Datei: `~/.webunit/supervisor.d/messenger.conf`

```
[program:messenger-consume]
command=/usr/local/bin/php /var/www/current/bin/console messenger:consume async --time-limit=3600 --memory-limit=256M
```

```
autostart=true
autorestart=true
stdout_logfile=~/.webunit/logs/messenger.log
stderr_logfile=~/.webunit/logs/messenger.err
```

Aktivieren:

```
webunit-supervisor reload
```

Status pruefen:

```
webunit-supervisor status
```

Ausgabe:

|                   |         |                         |
|-------------------|---------|-------------------------|
| apache2           | RUNNING | pid 56, uptime 0:01:38  |
| sshd              | RUNNING | pid 57, uptime 0:01:38  |
| messenger-consume | RUNNING | pid 129, uptime 0:00:05 |

Fertig — der Worker laeuft, wird bei Container-Restart automatisch gestartet und bei Absturz neu gestartet.

## Weitere Beispiele

### Laravel Queue-Worker

```
[program:laravel-queue]
command=/usr/local/bin/php /var/www/artisan queue:work --sleep=3 --tries=3 --max-time=3600
autostart=true
autorestart=true
stdout_logfile=~/.webunit/logs/queue.log
stderr_logfile=~/.webunit/logs/queue.err
```

### Node.js-Daemon

```
[program:my-node-daemon]
command=/usr/bin/node /var/www/current/daemon.js
```

```
directory=/var/www/current
environment=NODE_ENV="production",PORT="4000"
autostart=true
autorestart=true
stdout_logfile=~/.webunit/logs/node.log
stderr_logfile=~/.webunit/logs/node.err
```

## Mehrere Worker parallel

Manche Queue-Systeme profitieren von mehreren parallelen Workern. Supervisor kennt dafür `numprocs`:

```
[program:messenger-consume]
command=/usr/local/bin/php /var/www/current/bin/console messenger:consume async --time-limit=3600
numprocs=3
process_name=%(program_name)s_%(process_num)02d
autostart=true
autorestart=true
stdout_logfile=~/.webunit/logs/messenger_%(process_num)02d.log
stderr_logfile=~/.webunit/logs/messenger_%(process_num)02d.err
```

Das startet 3 Instanzen: `messenger-consume_00`, `_01`, `_02`.

## Einfacher Bash-Loop (zum Testen)

```
[program:testworker]
command=/bin/bash -c "while true; do date; sleep 5; done"
autostart=true
autorestart=true
stdout_logfile=~/.webunit/logs/testworker.log
stderr_logfile=~/.webunit/logs/testworker.err
```

## Kommandos

Alle Kommandos rufst du im Container-SSH ohne `sudo` auf.

| Kommando | Was es macht |
|----------|--------------|
|----------|--------------|

|                                                       |                                                                                                        |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>webunit-supervisor reload</code>                | Liest <code>*.conf</code> neu ein und startet neue/geaenderte Programme. Sanitize-Check laeuft vorher. |
| <code>webunit-supervisor status</code>                | Zeigt Status aller Programme (deine + apache2 + sshd).                                                 |
| <code>webunit-supervisor restart &lt;name&gt;</code>  | Startet ein Programm neu.                                                                              |
| <code>webunit-supervisor tail &lt;name&gt;</code>     | Zeigt letzte Zeilen von stdout.                                                                        |
| <code>webunit-supervisor tail-err &lt;name&gt;</code> | Zeigt letzte Zeilen von stderr.                                                                        |
| <code>webunit-supervisor validate</code>              | Prueft nur die Configs, ohne Reload.                                                                   |
| <code>webunit-supervisor help</code>                  | Alle Kommandos.                                                                                        |

“ `restart` und `tail` auf `apache2` oder `sshd` werden abgelehnt — das sind System-Prozesse.

# Config-Format (Kurz-Referenz)

Wichtige Direktiven fuer `[program:...]`:

| Direktive                                          | Bedeutung                                                                        |
|----------------------------------------------------|----------------------------------------------------------------------------------|
| <code>command=</code>                              | Vollstaendiger Pfad zum Binary + Argumente. <b>Kein</b> <code>sudo</code> .      |
| <code>directory=</code>                            | Arbeitsverzeichnis vor dem Start.                                                |
| <code>environment=KEY="value",KEY2="value2"</code> | Zusaetzliche Umgebungsvariablen.                                                 |
| <code>autostart=true</code>                        | Beim Container-Start automatisch starten.                                        |
| <code>autorestart=true</code>                      | Nach Absturz neu starten.                                                        |
| <code>startsecs=1</code>                           | Sekunden Wartezeit, bis der Start als erfolgreich gilt.                          |
| <code>startretries=3</code>                        | Nach so vielen Fehl-Starts aufgeben und <code>FATAL</code> markieren.            |
| <code>stopsignal=TERM</code>                       | Signal zum Stoppen ( <code>TERM</code> , <code>INT</code> , <code>KILL</code> ). |
| <code>stopwaitsecs=10</code>                       | Wartezeit auf sauberes Beenden, bevor <code>KILL</code> .                        |
| <code>stdout_logfile=&lt;pfad&gt;</code>           | Datei fuer stdout. Standard sind Dateien unter <code>~/.webunit/logs/</code> .   |
| <code>stderr_logfile=&lt;pfad&gt;</code>           | Datei fuer stderr.                                                               |
| <code>stdout_logfile_maxbytes=50MB</code>          | Rotate nach 50 MB (Standard).                                                    |
| <code>stdout_logfile_backups=10</code>             | Anzahl rotierter Backups (Standard 10).                                          |
| <code>numprocs=N</code>                            | N parallele Instanzen des Programms.                                             |

| Direktive                                                    | Bedeutung                    |
|--------------------------------------------------------------|------------------------------|
| <code>process_name=%(program_name)s_%(process_num)02d</code> | Namensschema bei numprocs>1. |

Vollstaendige Referenz: <http://supervisord.org/configuration.html#program-x-section-values>

# Was NICHT erlaubt ist

Der Sanitize-Check beim `reload` blockt:

- `user=root` oder `user=0` — Programme laufen als Container-User.
- `umask=...` — greift zu tief in Prozess-Umgebung ein.
- `stopasuser=root` oder `stopasuser=0` — analog.
- Programm-Namen `apache2`, `sshd`, `supervisord` — reserviert fuer System-Prozesse.
- Andere Sektionen als `[program:...]`, `[group:...]` oder `[eventlistener:...]` — die anderen (z.B. `[supervisord]`) wuerden die globale Container-Config beeinflussen.

Bei Fehlern zeigt `webunit-supervisor validate` (oder `reload`) genau welche Zeile in welcher Datei das Problem ist:

```
FEHLER: ~/.webunit/supervisor.d/messenger.conf:3 - user=root ist nicht erlaubt (nur Container-User)
```

# Logs

- **stdout / stderr** deiner Prozesse landen in den Dateien, die du in der Config angibst.  
Empfehlung: `~/.webunit/logs/<name>.log`.
- **Rotation:** Supervisor rotiert automatisch bei 50 MB, haelt 10 Backups. Aendern via `stdout_logfile_maxbytes=` und `stdout_logfile_backups=`.
- **Live mitlesen:** `webunit-supervisor tail <name>` (stdout) bzw. `tail-err` (stderr) — zeigt die letzten ~1000 Zeilen.
- **Selbst gucken:** `tail -f ~/.webunit/logs/messenger.log`.
- **Web-Zugriff geblockt:** `/.webunit/*` liefert immer 403, deine Logs sind also nicht von aussen einsehbar.

# Was passiert bei Container-Neubau?

Deine `~/webunit/supervisor.d/`-Configs bleiben erhalten (liegen im `/var/www`-Volume). Nach dem Neubau starten sie automatisch mit dem Container.

Die generierte `README.txt` im Verzeichnis kommt nur **einmal** beim allerersten Container-Start rein — wenn du sie loeschst, wird sie nicht neu geschrieben.

# Troubleshooting

## Prozess ist im Status `FATAL`

`webunit-supervisor status` zeigt `FATAL`. Das heisst Supervisor hat den Prozess mehrfach versucht zu starten und aufgegeben. Ursache findest du in `stderr`:

```
webunit-supervisor tail-err <name>
```

Typische Fehler:

- Falscher Pfad im `command=`
- Fehlende Berechtigungen auf einer Datei
- Datenbank nicht erreichbar
- Node-Modules / vendor/ nicht installiert

Fix, dann:

```
webunit-supervisor restart <name>
```

## Prozess ist im Status `EXITED`

Er ist normal beendet (Exit-Code 0), Supervisor startet ihn aber nicht neu. Vermutlich hast du `autorestart=false` oder der Command endet tatsaechlich (z.B. `time-limit` bei Messenger). Bei Messenger ist das gewollt — `autorestart=true` startet ihn dann wieder.

## `webunit-supervisor: command not found`

Das kann eigentlich nicht passieren — das Binary liegt im PATH. Wenn doch, ist entweder dein Container aelter als 2026-09-03 (dann sag uns Bescheid, wir rebuilden) oder der PATH ist in deinem `.bashrc` ueberschrieben.

## Config aendert sich nicht

Nach dem Editieren einer `.conf` musst du **explizit** `reload` fahren. Supervisor sieht Datei-Änderungen nicht von allein.

```
webunit-supervisor reload
```

## "Sanitize-Check fehlgeschlagen — Reload abgebrochen"

Der Reload wird abgebrochen wenn **irgendeine** deiner `.conf` einen Fehler hat. Fix die betroffene Datei (die genaue Zeile steht in der Fehlermeldung), dann nochmal `reload`.

---

Revision #13

Created 3 September 2026 12:27:02 by webunitadmin

Updated 7 September 2026 15:16:15 by webunitadmin