

cgroups

Control Groups (cgroups) spielen eine zentrale Schlüsselrolle in `systemd`. Sie bilden das fundamentale Kernel-Feature, mit dem `systemd` Prozesse **gruppiert, verfolgt, isoliert und bezüglich ihrer Systemressourcen beschränkt**.

Vor `systemd` nutzten klassische Init-Systeme (wie SysVinit) Prozess-IDs (PIDs) und PID-Dateien (`.pid`), um zu verfolgen, welche Prozesse zu welchem Dienst gehörten. Wenn ein Dienst Tochterprozesse erzeugte (z. B. ein Apache-Webserver mit vielen Worker-Prozessen) und abstürzte oder unsauber beendet wurde, blieben verwaiste Prozesse („Zombie-Prozesse“) im System zurück.

Mit cgroups gelingt `systemd` die vollständige und lückenlose Prozesskontrolle.

1. Die Kernaufgaben von cgroups in systemd

A. Lückenloses Prozess-Tracking (Keine "Verwaisten Prozesse")

Jeder Dienst, den `systemd` startet, wird in seine eigene cgroup platziert.

- Wenn ein Prozess neue Tochterprozesse startet (`fork()`), erben diese **automatisch** dieselbe cgroup.
- Selbst wenn sich ein Dämon versucht zu tarnen oder von seinem Elternprozess absplattet, bleibt er in der cgroup gefangen.
- **Ergebnis:** Wenn Sie `systemctl stop mein-dienst` ausführen, sendet `systemd` das Stopp-Signal an die *gesamte cgroup*. Dadurch werden verlässlich **alle** dazugehörigen Prozesse beendet.

B. Ressourcenbeschränkung und -kontrolle (Resource Limiting)

Über cgroups kann `systemd` genau festlegen, wie viel Hardware-Ressourcen eine Unit oder eine Gruppe von Units verbrauchen darf. Sie können das direkt in der Unit-Datei konfigurieren:

- **Arbeitsspeicher (RAM):**

- `MemoryMax=500M` – Der Dienst darf maximal 500 MB RAM nutzen. Wächst er darüber hinaus, greift der Out-Of-Memory (OOM) Killer gezielt für diesen Dienst.
- `MemoryHigh=400M` – Drosselt den Prozess sanft ab 400 MB, bevor das harte Limit erreicht wird.

- **CPU-Leistung:**

- `CPUQuota=50%` – Beschränkt den Dienst auf maximal 50% eines CPU-Kerns.
- `CPUWeight=100` – Priorisierung der CPU-Zeit im Vergleich zu anderen cgroups (Standard ist 100).

- **Festplatten-I/O:**

- `IOReadBandwidthMax=/dev/sda 10M` – Beschränkt das Lesen von der Festplatte auf 10 MB/s.

- **Tasks/Prozesse (Fork-Bomb-Schutz):**

- `TasksMax=100` – Der Dienst darf maximal 100 Unterprozesse/Threads gleichzeitig öffnen.

2. Die cgroup-Hierarchie in systemd

`systemd` baut beim Systemstart eine baumartige Struktur unter dem virtuellen Dateisystem `/sys/fs/cgroup/` auf. Diese Struktur unterteilt die Last des Systems in drei Hauptzweige (Slices):

Plaintext

```
/sys/fs/cgroup/
├─ system.slice    --> Alle Systemdienste (z. B. nginx, sshd, docker)
├─ user.slice      --> Angemeldete Benutzer & deren Sitzungen/Prozesse
└─ machine.slice   --> Virtuelle Maschinen (KVM/QEMU) & LXC/Systemd-Container
```

- **Slices (`.slice`):** Logische Gruppen, um Ressourcen hierarchisch zu verteilen (z. B. "Systemdienste bekommen insgesamt maximal 70% CPU, angemeldete User 30%").
- **Scopes (`.scope`):** Automatisch erstellte Gruppen für Prozesse, die nicht von `systemd` selbst gestartet wurden (z. B. eine SSH-Benutzersitzung).

- **Services** (`.service`): Die einzelnen Dienste innerhalb der Slices.

3. Befehle zum Anzeigen und Steuern von cgroups

Der Befehl `systemd-cgls` (cgroup list)

Zeigt den gesamten cgroup-Baum des Systems inklusive aller laufenden PIDs an:

Bash

```
systemd-cgls
```

Der Befehl `systemd-cgtop` (cgroup top)

Ähnlich wie der normale `top`-Befehl, zeigt aber den Live-Ressourcenverbrauch (CPU, RAM, I/O) aufgeteilt nach cgroups/Services an:

Bash

```
systemd-cgtop
```

Ressourcen zur Laufzeit anpassen

Sie können Limits eines laufenden Dienstes dynamisch anpassen, ohne die Unit-Datei manuell zu bearbeiten:

Bash

```
# Setzt das Arbeitsspeicher-Limit für Nginx sofort auf 1 GB  
systemctl set-property nginx.service MemoryMax=1G
```

4. Die Bedeutung von cgroups für Container (Docker, Podman)

Cgroups sind zusammen mit Linux **Namespaces** die fundamentale Grundlage moderner Container-Technologien:

- **Namespaces** sorgen für die *Isolierung* (Der Container sieht nur seine eigenen Dateien, Netzwerke und Prozesse).
- **CGroups** sorgen für die *Ressourcenbegrenzung* (Der Container kann das Wirtssystem nicht lahmlegen).

Da `systemd` cgroups nativ verwaltet, integrieren sich Container-Runtimes wie Docker, Podman oder `containerd` meist direkt mit dem `systemd`-cgroup-Treiber.

Created 2026-09-24 13:27:28 UTC by Admin

Updated 2026-09-25 09:16:12 UTC by Admin