Engineer

systemd

det, der starter alt andet på Linux

Hvad det er

Når Linux-kernen er færdig med sin egen opstart, giver den styringen videre til én proces med proces-id 1, og på næsten alle udbredte distributioner er den proces systemd. Det er en system- og servicemanager: den bringer maskinen op, starter og overvåger tjenesterne, mens den kører, og lukker det hele ned i den rigtige rækkefølge. Projektet staver navnet med små bogstaver, aldrig SystemD eller System D, og det afsluttende d er det gamle Unix-tegn for en dæmon. Det hører til familien af init-systemer sammen med SysV-init-scripts og Upstart, som det har afløst på de fleste distributioner, OpenRC og runit, som nogle mindre distributioner stadig vælger, og launchd, der løser samme opgave på macOS.

Alt, hvad systemd styrer, er en unit, beskrevet i en kort tekstfil med sektioner og nøgle-værdi-par. Manualen tæller 11 typer: service, socket, target, device, mount, automount, timer, swap, path, slice og scope. En service kører et program, og en timer starter en anden unit efter en tidsplan. En socket holder en lyttende port og starter sin service, når den første forbindelse kommer, og et target samler units til en tilstand, maskinen kan nå, for eksempel et flerbrugersystem med netværk. Krav og rækkefølge er to forskellige indstillinger, så »det her kræver det dér« og »det her starter efter det dér« står på hver sin linje, og manageren samler dem til én transaktion af start- og stopjob, som kan kontrolleres, før den udføres.

To valg i designet adskiller det fra de scriptbaserede init-systemer, der kom før. Hver tjeneste kører i sin egen kontrolgruppe i kernen, så manageren følger processerne i gruppen og kan stoppe dem samlet, begrænse deres hukommelse og gøre deres CPU-forbrug op. Og projektet er en hel samling og ikke kun en manager: dets eget website nævner en logdæmon, journalen, og komponenter til logins, netværkskonfiguration, tidssynkronisering og navneopslag. Distributionerne vælger selv, hvilke af dem de slår til; manageren på PID 1 er den del, der findes næsten overalt.

Brug begrebet i en beslutning

Brug »Hvad er systemd?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Daglig drift af servere, mail, konti og opdateringer, så det, alle er afhængige af, forbliver kedeligt.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har drevet 10 PostgreSQL- og MS SQL-servere — 10 databaseservere udrullet og holdt i god drift

Har drevet 10 PostgreSQL- og MS SQL-servere Kontakt os →

Hvorfor det betyder noget

På en Linux-server er spørgsmålet sjældent, om man skal køre systemd, for distributionen har allerede valgt det. Den egentlige beslutning er, om man bruger det som supervisor og planlægger, eller om man lægger en anden procesmanager, en containerdæmons genstartspolitik eller cron ovenpå.

Bruger man det direkte, samles opstartsrækkefølge, genstartspolitik, logning, ressourcegrænser og sandboxing i én fil pr. tjeneste, læst af den manager, der allerede kører, uden en ekstra supervisorproces. En unit kan sætte loft over hukommelsen med `MemoryMax=`, gøre styresystemets egne mapper skrivebeskyttede for tjenesten med `ProtectSystem=` og køre under en midlertidig bruger med `DynamicUser=`, og `systemd-analyze security` giver hver tjeneste en eksponeringsscore på en skala fra 0,0 til 10,0. Et job, som en timer starter, logger i modsætning til en crontab-linje til journalen, får samme sandboxing som enhver anden tjeneste og viser sin seneste og næste kørsel i `systemctl list-timers`. Kompetencen kan også flyttes: en ingeniør, der kan læse en unit-fil, kan drive de fleste Linux-servere, uanset distribution.

Omkostningerne er også reelle. Units virker kun på Linux, så et team, der også leverer til macOS, skriver property lists til launchd, og et, der leverer til BSD-systemerne, skriver rc-scripts. I de fleste containere findes systemd slet ikke: applikationen er selv PID 1, og genstartspolitikken hører til det, der kører containeren. Hvor containere og almindelige tjenester deler vært, mindsker Podmans Quadlet afstanden ved at gøre en containerbeskrivelse til en almindelig service-unit, så begge slags arbejde står under den samme supervisor.

Sådan ser det ud i praksis

Virksomhedens egen platform læner sig op ad systemd tre steder.

Produktionsværten i implementeringsbeskrivelsen fra september 2026 om at køre virksomhedens offentligt tilgængelige tjenester på én vært med 512 MB havde én kerne og omkring 464 MB brugbar hukommelse. På den kørte en webserver, der terminerer TLS for syv domæner, git-forgen, onion-tjenesten, én server til henholdsvis Gemini og Gopher samt blokering af indtrængningsforsøg. Automatisering af sikkerhedskopiering er implementeret separat; det dokumenterer ikke aktive krypterede sikkerhedskopier. Med så lidt hukommelse skulle hver fast kørende proces retfærdiggøre sit hukommelsesforbrug, så systemd var supervisoren, uden en ekstra supervisorproces lagt ovenpå, og containerne kørte som Quadlet-units under den samme manager i stedet for under en særskilt containerdæmon. Da spørgsmålet var, om værten også kunne bære en onion-tjeneste, var den laveste målte tilgængelige hukommelse cirka 310 MB af 464 i de registrerede stikprøver over ét døgn.

Implementeringen af sikkerhedskopiering og gendannelsestest bruger systemd-enheder til et databasedump, hvor det er konfigureret, et krypteret, deduplikeret restic-snapshot over SFTP til lager hos en anden udbyder, udtynding efter opbevaringspolitikken og et integritetstjek. Et separat månedligt workflow gendanner en kendt fil og sammenligner dens bytes, og sikkerhedskopiering og gendannelse har hver sit successignal. Timerne afhænger af et konfigureret backuplager; implementeringsbeskrivelsen dateret 2026-10-27 angiver, at de var deaktiveret uden et sådant lager, og dokumenterer ingen gennemført gendannelse i drift. Arbejdet ramte også en af de skarpere kanter i systemd, som er beskrevet nedenfor: adgangssætningen ligger i en beskyttet fil, som enhederne læser, fordi en værdi sendt gennem enhedens miljø får sine escape-tegn fortolket, og en omvendt skråstreg kunne ændre den i al stilhed.

Teknologisiden trækker den generelle linje: almindelige systemd-units som udgangspunkt og Docker og Kubernetes til de arbejdsbyrder, der gør sig fortjent til dem.

Hvor det går galt

En redigeret unit kører videre med de gamle indstillinger. Manageren læser unit-filerne ind i hukommelsen, så en ændring på disken har ingen virkning, før `systemctl daemon-reload` genopbygger dens billede af dem, og `systemctl cat` kan vise en fil, der afviger fra det, den kørende manager bruger. Lav ændringer med `systemctl edit`, der selv genindlæser, eller genindlæs og genstart derefter tjenesten.

En hemmelighed når frem til tjenesten med et tegn for lidt. Værdier i `Environment=` følger de citationsregler, systemd selv bruger, hvor escape-sekvenser i C-stil fortolkes, og en bogstavelig backslash skal skrives dobbelt. Dokumentationen advarer også om, at miljøvariabler er synlige for uprivilegerede klienter over D-Bus. Læg hemmeligheder i en fil, og giv dem videre med `LoadCredential=`, der giver tjenesten en skrivebeskyttet kopi, som kun dens bruger og root kan læse.

En pakkeopdatering fjerner i stilhed en lokal rettelse. Ændringer i leverandørens fil under `/usr/lib/systemd/system/` bliver overskrevet ved næste opdatering af pakken. Læg i stedet ændringerne i en drop-in under `/etc/systemd/system/<unit>.d/`, som lægges oven på originalen og overlever opgraderinger. Kommandolister lægges sammen i stedet for at blive erstattet, så en drop-in, der skifter startkommandoen, først rydder den med en tom tildeling:

```
# /etc/systemd/system/app.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/local/bin/app --config /etc/app/app.toml
Restart=on-failure
```

Et natligt job springer den nat over, hvor serveren var slukket. En kalendertimer, der skulle have kørt, mens maskinen var nede, venter blot på sit næste tidspunkt, fordi `Persistent=` som standard er false. Sæt `Persistent=true` på backups og andre job, der skal køre, så den forsømte kørsel sker, når timeren er aktiv igen, dog med en eventuelt konfigureret tilfældig forsinkelse.

En tjeneste, der er gået ned, bliver nede. `Restart=` er som standard `no`, så en tjeneste, der dør, forbliver død, til nogen opdager det. Sæt `Restart=on-failure` på tjenester, der skal køre hele tiden, og vær klar over, at når en unit rammer sin startgrænse, opgiver manageren den, og en manuel start må vente, til intervallet er gået, eller til nogen kører `systemctl reset-failed`.

Læs mere

- systemd(1) — selve manageren: unit-typerne, opstartsforløbet og de signaler, den reagerer på.
- systemd.unit(5) — unit-filer, afhængigheder, startgrænser og de drop-in-mapper, der tilsidesætter dem.

- systemd.service(5) — servicetyper, genstartspolitik og hvordan kommandolister nulstilles.
- systemd.exec(5) — kørselsmiljøet, sandboxing og credentials til tjenester.
- systemd.timer(5) — kalendertimere og monotone timere, herunder `Persistent=`.
- systemctl(1) — kommandolinjen til at starte, redigere, genindlæse og undersøge units.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er systemd?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/systemd/