Engineer

systemd

det som startar allt annat i Linux

Vad det är

Den process som får process-id 1 på nästan varje stor Linux-distribution heter systemd. Det är en system- och tjänstehanterare: den tar upp maskinen vid uppstart, startar och vaktar tjänsterna medan maskinen är igång och stänger ner allt i rätt ordning när den ska av. Projektet skriver namnet med gemener, aldrig SystemD eller System D, och det avslutande d:et är den gamla Unix-markeringen för en daemon. Det hör till familjen init-system. Där finns också SysV-initskripten och Upstart, som det har ersatt på de flesta distributioner, OpenRC och runit, som en del mindre distributioner fortfarande väljer, och launchd, som gör samma jobb på macOS.

Allt som hanteraren tar hand om är en enhet, unit i den engelska dokumentationen, och varje enhet beskrivs i en kort textfil med sektioner och par av nyckel och värde. Manualen räknar upp 11 typer: service, socket, target, device, mount, automount, timer, swap, path, slice och scope. En service kör ett program, och en timer startar en annan enhet enligt en tidsplan. En socket håller en lyssnande port öppen och startar sin service först när den första anslutningen kommer in. Ett target samlar enheter till ett tillstånd som maskinen kan nå, t.ex. ett fleranvändarsystem med nätverk. Beroenden och ordning är skilda inställningar, så ”det här behöver det där” och ”det här startar efter det där” blir två olika rader. Av dem bygger hanteraren en enda transaktion av start- och stoppjobb som den kan kontrollera innan något körs.

Två designval skiljer systemd från de skriptbaserade init-systemen som kom före. Varje tjänst körs i en egen kontrollgrupp (cgroup) i Linux, så hanteraren håller reda på processerna i gruppen och kan stoppa dem tillsammans, begränsa deras minne och redovisa deras processortid. Dessutom är projektet en hel svit och inte bara en hanterare: den egna webbplatsen räknar upp en loggdaemon,

journalen, och komponenter för inloggning, nätverkskonfiguration, tidssynkronisering och namnuppslagning. Distributionerna väljer själva vilka av tilläggen de slår på. Hanteraren på PID 1 är den del som finns nästan överallt.

Använd begreppet i ett beslut

Använd ”Vad är systemd?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Daglig skötsel av servrar, e-post, konton och uppdateringar, så att det som alla förlitar sig på förblir odramatiskt.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu — 10 databasservrar som driftsattes och hölls i gott skick

Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu Hör av er →

Varför det spelar roll

På en Linux-server är frågan sällan om du ska köra systemd, för distributionen har redan valt det. Det verkliga beslutet är om du låter systemd vara både processövervakare och schemaläggare, eller om du staplar en annan processhanterare, en containerdaemons omstartspolicy eller cron ovanpå.

Låter du systemd sköta det hamnar startordning, omstartspolicy, loggar, resursgränser och sandlådeisolering (sandboxing) i en fil per tjänst, läst av hanteraren som redan kör, utan en extra process för tjänstehantering. En enhet kan sätta ett tak för minnet med `MemoryMax=`, göra operativsystemets egna kataloger skrivskyddade för sin tjänst med `ProtectSystem=` och köra under en tillfällig användare med `DynamicUser=`. Kommandot `systemd-analyze security` ger dessutom varje tjänst ett exponeringsvärde på en skala från 0,0 till 10,0. Ett jobb som en timer startar skiljer sig från en rad i crontab på tre sätt: det loggar till journalen, det får samma sandlådeisolering som vilken tjänst som helst, och det visar sin senaste och nästa körning i `systemctl list-timers`. Kunskapen går också att ta med sig: den som kan läsa en enhetsfil kan sköta de flesta Linux-servrar, oavsett distribution.

Kostnaderna finns också. Enheter fungerar bara på Linux, så ett team som även levererar till macOS skriver property lists för launchd, och ett som levererar till BSD-systemen skriver rc-skript. I de flesta containrar finns ingen systemd alls: applikationen är själv PID 1, och omstartspolicyn hör till det som kör containern. Där containrar och vanliga tjänster delar värd minskar Quadlet i Podman avståndet genom att göra en containerbeskrivning till en vanlig tjänsteenhet, så att båda sorternas arbetslast står under samma processövervakare.

Så ser det ut i praktiken

Företagets egen plattform lutar sig mot systemd på tre ställen.

I implementeringsbeskrivningen från september 2026 hade produktionsvärddatorn som beskrivs i företagets publika tjänster på en värddator med 512 MB en kärna och ungefär 464 MB användbart minne. På den kördes en webbserver som terminerar TLS för sju domäner, git-forgen, onion-tjänsten, en server vardera för Gemini och Gopher samt spärrning av intrångsförsök. Automatisering av säkerhetskopiering har implementerats separat; det fastställer inte att krypterade säkerhetskopior är aktiva. Med så lite minne behövde varje ständigt körande process motivera sin minnesförbrukning, så systemd var processhanteraren, utan någon ytterligare tillsynsprocess ovanpå, och containrarna kördes som Quadlet-enheter under samma hanterare i stället för under en egen containerdaemon. När frågan var om värddatorn också klarade en onion-tjänst var det lägsta uppmätta tillgängliga minnet ungefär 310 MB av 464 under ett dokumenterat dygns stickprov.

Implementeringen av säkerhetskopiering och återställningstest använder systemd-enheter för en databasdump där det är konfigurerat, en krypterad, deduplicerad restic-ögonblicksbild via SFTP till lagring hos en annan leverantör, gallring enligt bevarandepolicyn och en integritetskontroll. Ett separat månatligt arbetsflöde återställer en känd fil och jämför dess byte, och säkerhetskopiering och återställning har separata signaler för lyckade körningar. Timerenheterna är beroende av ett konfigurerat repo; implementeringsbeskrivningen daterad 2026-10-27 anger att de var inaktiverade utan ett sådant och redovisar ingen genomförd återställning i drift. Arbetet stötte också på en av de vassare kanterna i systemd, som beskrivs nedan: lösenfrasen ligger i en skyddad fil som enheterna läser, eftersom ett värde som skickas via enhetens miljö får sina escape-tecken tolkade, och ett omvänt snedstreck skulle kunna ändra lösenfrasen i tysthet.

Teknikstacken drar den allmänna gränsen: vanliga systemd-enheter som standard, och Docker och Kubernetes för de arbetslaster som förtjänar dem.

Var det går fel

En redigerad enhet kör vidare med sina gamla inställningar. Hanteraren läser in enhetsfilerna i minnet, så en ändring på disken gör ingenting förrän `systemctl daemon-reload` bygger om hanterarens bild av dem, och `systemctl cat` kan visa en

fil som skiljer sig från den som den körande hanteraren använder. Gör ändringar med `systemctl edit`, som laddar om åt dig, eller ladda om och starta sedan om tjänsten.

En hemlighet når tjänsten med ett tecken för lite. Värden i `Environment=` går igenom de citatregler som systemd själv använder, där escape-sekvenser i C-stil tolkas och ett bokstavligt omvänt snedstreck måste skrivas dubbelt. Dokumentationen varnar också för att miljövariabler syns för oprivilegierade klienter över D-Bus. Lägg hemligheter i en fil och lämna över dem med `LoadCredential=`, som ger tjänsten en skrivskyddad kopia som bara dess användare och root kan läsa.

En paketuppgradering tar tyst bort en lokal rättelse. Ändringar i leverantörens fil under `/usr/lib/systemd/system/` skrivs över vid nästa uppdatering av paketet. Lägg ändringarna i en drop-in-fil under `/etc/systemd/system/<unit>.d/` i stället. Den läggs ovanpå originalet och överlever uppgraderingar. Kommandolistor läggs ihop i stället för att ersättas, så en drop-in som byter startkommando först tömmer listan med en tom tilldelning:

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

Ett nattligt jobb hoppar över natten då servern var avstängd. En kalendertimer som skulle ha körts medan maskinen var nere väntar bara på sin nästa tid, eftersom `Persistent=` som standard är false. Sätt `Persistent=true` på säkerhetskopiering och andra jobb som måste köras, så går den missade körningen när timern är aktiv igen, med en slumpmässig fördröjning om en sådan är konfigurerad.

En tjänst som har kraschat förblir nere. `Restart=` är som standard `no`, så en tjänst som dör ligger kvar död tills någon märker det. Sätt `Restart=on-failure` på tjänster som ska köra hela tiden, och räkna med att hanteraren ger upp en enhet som når sin startgräns. En manuell start får då vänta tills intervallet har gått eller tills någon kör `systemctl reset-failed`.

Läs mer

- systemd(1) — själva hanteraren: enhetstyperna, uppstartsförloppet och de signaler den svarar på.
- systemd.unit(5) — enhetsfiler, beroenden, startgränser och de drop-in-kataloger som åsidosätter dem.
- systemd.service(5) — tjänstetyper, omstartspolicy och hur kommandolistor nollställs.
- systemd.exec(5) — körmiljön, alternativen för sandlådeisolering och credentials till tjänster.
- systemd.timer(5) — kalendertimrar och monotona timrar, bl.a. `Persistent=` .
- systemctl(1) — kommandoraden för att starta, redigera, ladda om och inspektera enheter.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är systemd?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/systemd/