Engineer

Dæmon

et program, der kører i baggrunden uden opsyn

Hvad det er

En dæmon er en langlivet proces, som intet menneske starter i hånden: styresystemet eller en servicemanager sætter den i gang, den har ingen terminal, og den bliver kørende for at besvare forespørgsler eller holde øje med en del af maskinen. På engelsk staves ordet daemon, og mange dæmoner bærer det i navnet som et afsluttende d: `sshd` tager imod SSH-forbindelser, `crond` kører planlagte job, og `systemd-journald` samler logge. Windows kalder den samme slags program en tjeneste, og det gør systemd også, som i dag passer de fleste dæmoner på Linux.

Den klassiske Unix-opskrift, som Linux-manualen `daemon(7)` stadig gennemgår i femten nummererede trin, lader programmet rive sig løs selv. Det lukker de fildeskriptorer, det har arvet, nulstiller sine signalhåndteringer, forker, kalder `setsid()` for at forlade terminalens session og forker en gang til, så det ikke kan få en terminal igen. Det første barn afslutter, så PID 1 adopterer den overlevende proces, som sender sine standardstrømme til `/dev/null`, skifter til `/`, skriver en PID-fil og opgiver sine privilegier. C-bibliotekets kald `daemon()` klarer en del af arbejdet, og manualen fraråder det netop, fordi det kun gennemfører en delmængde af trinene.

Den nye stil, som samme manual kalder new-style daemons, og som både systemd og Apples launchd forventer, vender opskriften på hovedet. Programmet bliver i forgrunden, for servicemanageren har allerede startet det løsrevet, med et rent miljø, den rigtige bruger og den rigtige arbejdsmappe. Det skriver sine loglinjer til standard error, hvor servicemanageren samler dem op, melder, når det er klar til at tage imod arbejde, lukker pænt ned på SIGTERM og genindlæser sin konfiguration på SIGHUP. Apples vejledning til launchd-job er kontant om den gamle vane: et job må ikke gøre sig selv til dæmon, og forker det og afslutter, konkluderer launchd, at processen er død.

To naboer bliver tit forvekslet med den. Et planlagt job, som cron eller en systemd-timer kører, starter, gør sit arbejde og afslutter, mens en dæmon bliver. Og macOS deler ordet i to: en launch daemon betjener hele systemet og indlæses ved opstart, mens en launch agent tilhører én bruger og kun kører, så længe brugeren er logget ind. Linux trækker samme skel med de instanser af systemd, der kører for hver enkelt bruger.

Brug begrebet i en beslutning

Brug »Hvad er en dæmon?« 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

Hver dæmon på en server er en løbende forpligtelse. Den optager hukommelse, uanset om nogen bruger den, og nogen skal patche den, hver gang leverandøren udsender en rettelse. Lytter den på en netværksport, hører den til angrebsfladen i hver time, den kører. Det nyttige spørgsmål er derfor sjældent, hvilken dæmon man skal installere, men om arbejdet overhovedet kræver en proces, der altid kører.

Tre former dækker det meste. Noget, der sker efter en tidsplan, som en backup, en rapport eller fornyelsen af et certifikat, behøver ingen fast proces: en timer starter det, det kører til ende, og mellem kørslerne koster det intet. Noget, der besvarer lejlighedsvise forespørgsler, kan socket-aktiveres: servicemanageren ejer den lyttende socket og starter dæmonen ved første forbindelse, og da socketen forbliver bundet under en genstart, venter nye forespørgsler i kø i stedet for at blive afvist. Kun arbejde, der skal reagere hele tiden, som en webserver, en database eller en mailserver, fortjener en proces, der altid er oppe. På en lille maskine afgør det valg, om det hele kan være i hukommelsen.

Den anden beslutning kommer, hver gang en dæmon skrevet af andre skal i produktion, og fire spørgsmål afgør den. Bliver den i forgrunden, når man beder den om det? Logger den til standard error, eller insisterer den på sine egne logfiler? Kan den fortælle servicemanageren, at den er klar, og ikke blot at den er startet? Og kræver den root, eller kan servicemanageren række den den ene

privilegerede ressource, den har brug for, for eksempel en socket på port 443? En dæmon, der svarer godt på alle fire, er billig at drive i årevis. En, der dumper på dem alle, ankommer med et wrapper-script, sin egen logrotation og en undtagelse i den næste sikkerhedsgennemgang.

Sådan ser det ud i praksis

Virksomhedens egen platform viser de valg i tre konkrete stykker arbejde.

At levere sitet over Gemini og Gopher kostede to fast kørende processer, ikke et ekstra build. Da sitet blev spejlet til begge protokoller, skrev det build, der skriver HTML'en, også 777 Gopher-menuer og det samme antal Gemini-dokumenter i den dokumenterede implementeringsfase, og to små dæmoner på den samme vært tog sig af at levere dem. En separat sammenligning af builds registrerede nul ændrede HTML-bytes, da outputformaterne blev tilføjet; resultatet beskriver ikke senere redaktionelle eller visuelle ændringer.

Onion-spejlet på Tor viser netværkssiden af samme valg: dæmonen bygger udgående kredsløb, så ingen ringer ind, og firewallen får ingen ny åben port. Rollen, der sætter onion-spejlet op, installerer dæmonen fra distributionens egen pakke, skriver en konfiguration med den udgående proxyport slået fra og afviser at overskrive en identitet, den ikke selv har genereret. Ét trin gav en lære, der gælder dæmoner i al almindelighed: nøglen landede korrekt, kørslen meldte ingen ændring, og den nye adresse blev aldrig offentliggjort, fordi intet i konfigurationen på det tidspunkt pegede på den nye mappe. Nu stopper rollen med en fejl, når en identitet ligger på disken, uden at nogen tjeneste nævner den.

Rapporteringen viser, hvad en stille dæmon kan skjule. Seks indsamlere, der kun læser og gengiver værten som Markdown, blev bygget om, da det stod klart, at en blind indsamling og en sund vært gav den samme side: et spejl, hvis dæmon havde skiftet logformat, viste det samme nul som et spejl, ingen havde besøgt. Indsamlerne melder nu blandt andet, hvor mange journallinjer de har læst, over for hvor mange de har genkendt, så en dæmon, der ændrer det, den skriver, dukker op som et hul og ikke som ro.

Hvor det går galt

Tjenesten starter og stopper igen et sekund senere. En dæmon i gammel stil forker og lader forælderprocessen afslutte, sådan som opskriften foreskriver. Under standardindstillingen `Type=simple` er den proces, systemd startede, hovedprocessen, så når den afslutter, regnes tjenesten for stoppet, og med standarden `KillMode=control-group` bliver barnet, der skulle fortsætte, dræbt sammen med resten af enhedens kontrolgruppe. Kør programmet med dets forgrundsflag. Til de få, der ikke kan ændres, erklærer man `Type=forking` sammen med `PIDFile=`, en type manualen fraråder til ny kode.

Klienter forbinder for tidligt og bliver afvist. Med `Type=simple` regner systemd tjenesten for startet, så snart processen er forket, så enheder, der er ordnet efter den, starter, mens dæmonen stadig læser sin konfiguration. Ifølge servicemanualen melder `systemctl start` oven i købet succes under den type, selv når programfilen mangler, eller den konfigurerede bruger ikke findes. Vælg `Type=exec`, der venter, til programmet faktisk er eksekveret, og lad dæmonen selv melde sig klar med `Type=notify`, når andre afhænger af den:

```
#include <systemd/sd-daemon.h>

int main(void) {
    load_config();
    open_listeners();
    sd_notify(0, "READY=1"); /* enheder ordnet efter denne starter nu */
    return serve_forever();
}
```

Et stopscript dræber en fremmed proces. En forkende dæmon skriver sit proces-id i en fil, så scripts kan sende den signaler senere. Efter et nedbrud bliver filen liggende, kernen giver det samme nummer til en helt anden proces, og det næste stop sender SIGTERM til det forkerte program. Servicemanualen anbefaler helt at undgå PID-filer i nye projekter. En servicemanager, der selv har startet processen, følger den gennem dens kontrolgruppe og har ingen fil, der kan blive forældet.

En dæmon dør ved opstart og efterlader intet spor. Den gamle opskrift sender standard output og standard error til `/dev/null`, så en fejlmeddelelse, der kommer, før dæmonen har åbnet sin egen logfil, forsvinder sporløst. Under

en servicemanager skal begge strømme blive, hvor de er: alt, hvad dæmonen skriver til standard error, lander i journalen, hvor `journalctl -u` læser det enhed for enhed.

Én udnyttet fejl giver adgang til hele maskinen. En dæmon, der kører som root, fordi den skulle bruge port 80 ved opstart, beholder root for hver eneste forespørgsel bagefter. Lad servicemanageren binde den privilegerede socket og række den videre, og overlad det at opgive privilegier til servicemanageren, som `daemon(7)` anbefaler; indstillinger som `NoNewPrivileges=` og `ProtectSystem=` begrænser, hvad en kompromitteret proces kan røre ved.

Læs mere

- daemon(7) — de femten trin i den klassiske opskrift, og hvad en dæmon i ny stil gør i stedet.
- daemon(3) — C-bibliotekets genvej og den fejl med sessionslederen, som dens egen manual beskriver.
- sd_notify(3) — sådan fortæller en dæmon systemd, at den er klar, genindlæser eller stopper.
- systemd.socket(5) — socket-aktivering, hvor servicemanageren holder porten og starter dæmonen efter behov.
- setsid — POSIX-definitionen af kaldet, der løsriver en proces fra sin terminal.
- Creating Launch Daemons and Agents — Apples regler for launchd-job, blandt andet hvorfor et job aldrig må forke og afslutte.

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 en dæmon?« 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/daemon/