Engineer

Konfiguration

de inställningar som avgör hur ett system beter sig

Vad det är

Konfiguration är allt som ett program läser in utanför sin egen källkod för att avgöra hur det ska köras på just ett visst ställe: porten det lyssnar på, databasen det ansluter till, kontona det släpper in, funktionerna det slår på. I vardagligt tal blir det nästan alltid config eller konfig, och filerna som bär den kallas konfigurationsfiler. NIST definierar begreppet i SP 800-128, sin vägledning för säkerhetsinriktad konfigurationshantering, som ”the possible conditions, parameters, and specifications with which an information system or system component can be described or arranged”, alltså de möjliga villkor, parametrar och specifikationer som ett informationssystem eller en systemkomponent kan beskrivas eller ordnas med. Ordet att hålla fast vid är beskrivas: en konfiguration är en beskrivning, och systemet kan hålla med om den eller låta bli.

Ett program hämtar sällan sin konfiguration från ett enda ställe. Standardvärden är inkompilerade i binären, en huvudfil ligger på disken med en katalog av drop-in-fragment bredvid sig, och miljövariabler och kommandoradsflaggor kommer in vid start. Programmet slår ihop allt detta till en enda uppsättning värden, med en regel för vad som händer när två källor sätter samma nyckel, och den regeln är varje programs egen. OpenSSH-servern behåller det första värde den läser för de flesta nyckelord, medan andra verktyg låter en senare fil skriva över en tidigare. Samma filer inlästa i en annan ordning ger ett annat system.

Formaten varierar: rader i INI-stil, TOML, YAML, JSON och format som bara ett enda program använder. TOML 1.0.0 vill vara ”a minimal configuration file format”, ett minimalt format för konfigurationsfiler, medan YAML 1.2.2 är ett allmänt dataspråk som konfigurationsfiler lånar, och vart och ett har sina egna regler för hur text blir till värden.

Använd begreppet i ett beslut

Använd ”Vad är konfiguration?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Hela uppsättningen nedskriven som filer, så att den kan återskapas exakt och aldrig bara finns i en enda persons huvud.

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. Skrev 1 000 RBAC-regler som säkrar Linux-servrar — 1,000 åtkomstregler, skrivna för att täppa till en säkerhetslucka

Skrev 1 000 RBAC-regler som säkrar Linux-servrar Hör av er →

Varför det spelar roll

Konfigurationen är det som låter ett och samma bygge tjäna många ställen. Samma program körs på en bärbar dator, en staging-server och i produktion, och det som skiljer dem åt är konfigurationen. The Twelve-Factor App, en metod för tjänster som ska driftsättas, drar gränsen just där: config är ”everything that is likely to vary between deploys”, allt som sannolikt skiljer sig mellan driftsättningar, som referenser till underliggande tjänster, inloggningsuppgifter till externa tjänster och värdnamnet för varje driftsättning. Metoden kräver en strikt åtskillnad mellan konfiguration och kod, och den erbjuder ett test som är värt att använda överallt: om ”the codebase could be made open source at any moment, without compromising any credentials”, alltså om kodbasen när som helst skulle kunna bli öppen källkod utan att en enda inloggningsuppgift röjs.

Samma text markerar också den motsatta kanten. Intern koppling som är likadan i varje driftsättning, som ett webbramverks tabell över routes, faller utanför definitionen och är enligt texten ”best done in the code”. Ett värde som är detsamma i varje driftsättning är kod utklädd till inställning, och en fil ger det bara ytterligare ett ställe att vara fel på.

Konfigurationen är också där ett system uttrycker sitt skydd: vem som får logga in, vilka portar som svarar, vad en brandvägg släpper igenom. En server är exakt så härdad som de inställningar den faktiskt använder, och mellan en fil och de inställningarna står tolken, prioritetsreglerna, de inkompilerade standardvärdena, programversionen och miljön. Vilken som helst av dem kan ändra utfallet medan filen är oförändrad, så den pålitliga vanan är att fråga programmet vad det använder och behandla det svaret som konfigurationen.

Så ser det ut i praktiken

Den här webbplatsen konfigureras i en enda TOML-fil, hugo.toml, som bland mycket annat bestämmer webbplatsens språk, taxonomier och monteringar. En tabell i hugo.toml visar hur mycket några få rader kan avgöra:

```
[frontmatter]
  date = ["date", ":git", ":fileModTime"]
  lastmod = ["lastmod", ":git", ":fileModTime"]
```

Varje lista är en uppslagsordning: ett datum som står i själva sidan, sedan datumet för den senaste commit som rörde filen och sist filens ändringstid. En flagga nära början av filen, `enableGitInfo = true`, är det som över huvud taget låter Hugo läsa git-historiken. Kommentaren ovanför tabellen förklarar den första posten: det bokstavliga fältet i front matter måste namnges, annars ignoreras ett `lastmod` som står i en sida i tysthet, medan webbplatsen byggs som vanligt.

Den tabellen är åtgärden som beskrivs i lagningen av en webbplatskarta där de flesta sidor delade en tidsstämpel. Före åtgärden hämtades datumet från filens ändringstid, en källa som beskrev filen och inte innehållet: innehållsexporten skapar alla filer på nytt varje gång den körs, så webbplatskartan uppgav samma ändringsögonblick för 172 av webbplatsens 176 sidor. Uppslagsordningen flyttade datakällan till versionshistoriken. Vid det dokumenterade implementeringsskedet berörde en synkronisering efter en månads innehållsändringar åtta filer av 186.

På en server går samma lärdom igen i härdningen av SSH med kontroller i tre steg. En automatiseringsroll hade skrivit konfigurationen för SSH-daemonen i månader när en åtkomstrapport i september 2026 läste av de inställningar som daemonen faktiskt tillämpade och fann två som inte stämde med rollens fil. Operativsystemets egna standardvärden låg ovanför drop-in-filen, och för de två inställningarna vinner den första förekomsten, så lösningen blev ett filnamnsprefix som sorteras före leverantörens. Runt den kontrollerar rollen nu i tre steg: kandidatfilen syntaxkontrolleras före installationen, den sammansatta konfigurationen kontrolleras efteråt, och de gällande värdena läses tillbaka och 24 direktiv kontrolleras mot dem. Två direktiv lämnades medvetet utanför, eftersom daemonen inte längre implementerar dem.

Tre skydd mot brute force via SSH som aldrig fungerade visar samma sak från andra hållet. Vid en granskning i augusti 2026 stod alla tre i konfigurationen och i varje rapport, och alla tre hade varit utan verkan ända sedan servern byggdes. I konfigurationen var jailen för spärrning inställd på port 22 fast daemonen hade flyttats till 1986, hastighetsbegränsningen fanns i brandväggen men under en bredare regel som matchade trafiken innan den nådde begränsningen, och spärråtgärden anropade en binär för paketfiltrering som paketsystemet rekommenderar utan att kräva den, och på den här värden kunde programmet inte hittas. Granskningen kontrollerade det körande systemet i stället för filerna, och två regler kom ut av den: ett säkerhetsskydd får en automatisk kontroll i stället för en kommentar, och en brandvägg verifieras genom var en regel ligger snarare än genom att den finns.

Var det går fel

Var och en av situationerna nedan är en fil som stämmer som text men är fel som beteende, och för var och en finns en kontroll som läser beteendet.

- En annan källa vinner än den man räknade med. OpenSSH-manualen för serverns konfiguration säger: ”Unless noted otherwise, for each keyword, the first obtained value will be used.” Ett standardvärde från leverantören som läses före din drop-in-fil slår den alltså, och i ett verktyg där det sista värdet vinner blir samma upplägg omvänt. Läs prioritetsregeln i programmets egen manual och fråga sedan programmet: `sshd -t` kontrollerar att en konfiguration är giltig, och `sshd -T` skriver ut den som faktiskt gäller.
- En TOML-nyckel hamnar i fel tabell. I TOML 1.0.0 hör paren under en `[table]`-rubrik till den tabellen ”until the next header or EOF”, och indrag ignoreras. En nyckel som läggs till längst ned i en fil hamnar i den sista tabellen även när den står helt till vänster. Håll tabellerna sist och de fristående nycklarna ovanför dem. Att definiera en nyckel eller en tabell två gånger är ogiltigt, så en dubblett är ett fel att rätta och aldrig ett sätt att skriva över.
- Ett YAML-värde utan citattecken byter typ. Med kärnschemat, som YAML 1.2.2 rekommenderar som standard, blir ett värde utan citattecken som matchar ett av dess mönster ett tal, ett booleskt värde eller null: `1.10` blir flyttalet 1,1, `0o17` blir heltalet 15, `0123` kommer tillbaka som 123, `FALSE` är ett booleskt värde och `~` är null. Sätt citattecken runt varje värde som måste förbli text, med versionssträngar och identifierare med inledande nollor först. Specifikationen räknar också upprepade nycklar i en mappning till de sätt som en inläsning kan bli ofullständig på, så en nyckel som skrivs två gånger är en defekt.

- En inställning ser aktiv ut men gör ingenting. En regel som skuggas av en tidigare, ett direktiv som en nyare version har tagit bort, en port som tjänsten sedan länge har lämnat: alla ligger kvar i filen och ser avsiktliga ut. Jailen på port 22 och hastighetsbegränsningen under en bredare regel visades båda som friska. En kontroll som läser varje fil för sig släpper igenom dem, så kontrollera ordning och förekomst mot det körande systemet.
- En hemlighet följer med filen. Ett lösenord i en konfigurationsfil går dit filen går: till repot, till säkerhetskopiorna, till felrapporten. Twelve-Factor lägger config i miljövariabler delvis för att det, till skillnad från konfigurationsfiler, finns ”little chance of them being checked into the code repo accidentally”.

Läs mer

- NIST CSRC Glossary: configuration — varje definition som publikationerna från NIST ger begreppet, med sitt källdokument.
- TOML v1.0.0 — hela specifikationen, kort nog att läsa i ett svep.
- YAML Ain’t Markup Language, version 1.2.2 — specifikationen; kapitel 10 innehåller kärnschemat som avgör vad ett värde utan citattecken blir.
- The Twelve-Factor App: Config — argumenten för att hålla värden som hör till en viss driftsättning utanför koden, med testet om öppen källkod.
- sshd_config(5) — manualen för konfigurationen av OpenSSH-servern, med regeln om första värdet, Include och Match.
- sshd(8) — manualen för daemonen, där `-t` kontrollerar en konfiguration och `-T` skriver ut den som gäller.

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 konfiguration?” 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/configuration/