Engineer

Конфігурація

налаштування, які визначають, як поводиться система

Що це таке

Конфігурацією називають усе, що програма зчитує поза власним кодом, щоб знати, як працювати в конкретному місці: на якому порту слухати, до якої бази даних підключатися, кого пускати і які функції вмикати. У розмові інженери здебільшого кажуть «конфіг», англійською config, а файли з нею називають конфігураційними файлами або файлами конфігурації. NIST у SP 800-128, посібнику з управління конфігурацією з фокусом на безпеку, визначає її як можливі стани, параметри та специфікації, якими можна описати або впорядкувати інформаційну систему чи її компонент. Ключове слово тут — «описати»: конфігурація є описом, і система, яку вона описує, може з ним розходитися.

Рідко програма бере конфігурацію з одного місця. Значення за замовчуванням вкомпільовано в бінарний файл, на диску лежить головний файл, а поруч із ним — каталог додаткових фрагментів (drop-in), змінні середовища й прапорці командного рядка додаються під час запуску. Програма зводить усе це в один набір значень за правилом, яке вирішує, що робити, коли два джерела задають той самий ключ, і кожна програма має своє правило. Сервер OpenSSH для більшості ключових слів залишає перше прочитане значення, а інші інструменти дають пізнішому файлу перекрити раніший. Ті самі файли, прочитані в іншому порядку, дають іншу систему.

Формати бувають різні: рядки в стилі INI, TOML, YAML, JSON і формати, якими користується лише одна програма. TOML 1.0.0 прагне бути мінімальним форматом конфігураційних файлів, а YAML 1.2.2 — загальна мова даних, яку конфігураційні файли запозичують. Кожен формат має власні правила, за якими текст стає значеннями.

Застосуйте поняття до рішення

Застосуйте «Що таке конфігурація?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Уся конфігурація записана у файлах, тож її можна відтворити точно, і вона не живе лише в чиїйсь голові.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Саме конфігурація дає змогу одній збірці служити в багатьох місцях. Та сама програма працює на ноутбуці, на тестовому хості (staging) і в продакшні, а різниця між ними — у її конфігурації. The Twelve-Factor App, методологія для сервісів, які розгортають, проводить межу саме тут: конфігурація — це все, що, імовірно, змінюватиметься між розгортаннями, як-от адреси баз даних і допоміжних сервісів, облікові дані до зовнішніх сервісів і канонічне ім’я хоста кожного розгортання. Методологія вимагає суворо відокремлювати конфігурацію від коду й пропонує перевірку, яку варто застосовувати будь-де: чи можна було б будь-якої миті відкрити кодову базу, не скомпрометувавши жодних облікових даних.

Той самий текст позначає й протилежну межу. Внутрішні налаштування, однакові в кожному розгортанні, наприклад таблиця маршрутів вебфреймворку, до конфігурації в цьому сенсі не належать, і їм краще жити в коді. Значення, однакове скрізь, — це код у костюмі налаштування, а файл лише дає йому ще одне місце, де помилитися.

Конфігурація — це також місце, де система заявляє свій рівень захисту: хто може входити, які порти відповідають, що пропускає брандмауер. Хост захищений рівно настільки, наскільки захищені значення, які він фактично використовує, а між файлом і цими значеннями стоять парсер, правила пріоритету, вкомпільовані значення за замовчуванням, версія програми й середовище. Кожен із них може змінити результат, поки файл лишається тим самим, тож надійна звичка — питати програму, що вона використовує, і вважати конфігурацією саме цю відповідь.

Як це виглядає на практиці

Цей сайт налаштовано в одному файлі TOML — hugo.toml, який серед іншого задає мови сайту, таксономії та монтування (mounts). Одна таблиця в ньому показує, скільки можуть вирішити кілька рядків:

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

Кожен список — це порядок пошуку: спершу дата, записана в самій сторінці, потім дата останнього коміту, який торкався файлу, і нарешті час зміни файлу. Прапорець у верхній частині файлу, `enableGitInfo = true`, узагалі дозволяє Hugo читати історію git. Коментар над таблицею пояснює, навіщо перший елемент: якщо саме поле з front matter не назване в списку, Hugo мовчки ігнорує записаний у сторінці `lastmod`, а сайт збирається як завжди.

Ця таблиця і є виправленням із випадку, коли карта сайту показувала однаковий час зміни майже для всіх сторінок. Пошукові системи отримували з неї дату, за якою 172 сторінки зі 176 змінилися в одну й ту саму мить: дату брали з часу зміни файлу, а експорт вмісту під час кожного запуску наново записує всі файли. Ланцюжок резервних джерел перевів дату на історію версій. На тому зафіксованому етапі реалізації повторна синхронізація після місяця правок вмісту зачепила вісім із 186 файлів.

На сервері той самий урок проходить крізь посилення захисту SSH із триетапною перевіркою. Роль автоматизації вже кілька місяців записувала конфігурацію демона SSH, коли у вересні 2026 року звіт про доступ запитав у самого демона, які значення той застосовує, і два з них розійшлися з файлом ролі. Власні значення операційної системи за замовчуванням стояли активними раніше за drop-in-файл, а для цих двох параметрів діє значення, прочитане першим. Розв’язок — додати до назви drop-in-файлу префікс, щоб у лексичному порядку він ішов перед файлом постачальника. Відтоді роль перевіряє себе в три етапи: синтаксис файлу-кандидата перевіряють до встановлення, зібрану конфігурацію — після, а підсумкові значення демона зчитують назад і звіряють із ними 24 директиви. Дві директиви свідомо пропущено, бо демон їх більше не реалізує.

Протилежний бік тієї самої вади видно в перевірці трьох засобів SSH проти перебору паролів, проведеній у серпні 2026 року. Усі три значилися в конфігурації й у кожному звіті, проте від самого створення хоста жоден із них так і не спрацював. Набір правил блокування Fail2ban (jail) був налаштований на порт 22, а демон SSH приймав з'єднання на порту 1986. Обмеження частоти в брандмауері стояло нижче за ширше правило, яке перехоплювало трафік першим. Дія блокування залежала від бінарного файлу пакетного фільтра, який система пакетів лише рекомендує. Перевірка читала систему в роботі, а не файли, і дала два нові правила: засіб безпеки отримує перевірку через assert, а не коментар, а брандмауер перевіряють за тим, де стоїть правило, а не за тим, чи воно є.

Де виникають проблеми

Кожна вада нижче — це файл, правильний як текст і хибний як поведінка, і для кожної є перевірка, що читає саме поведінку.

- Перемагає хибне джерело. Посібник OpenSSH до конфігурації сервера каже, що для кожного ключового слова використовується перше отримане значення, якщо не зазначено інше. Тож значення постачальника, прочитане раніше за власний drop-in, перемагає його, а в інструменті, де перемагає останнє значення, усе навпаки. Правило пріоритету варто прочитати в посібнику самої програми, а потім запитати саму програму: `sshd -t` перевіряє, чи конфігурація коректна, а `sshd -T` виводить чинну.
- Ключ TOML потрапляє до чужої таблиці. У TOML 1.0.0 пари ключ–значення під заголовком `[table]` належать цій таблиці до наступного заголовка або до кінця файлу, а відступи ігноруються. Ключ, дописаний у кінці файлу, опиняється в останній таблиці, навіть якщо стоїть без жодного відступу. Таблиці варто тримати в кінці, а звичайні ключі — над ними. Визначати ключ або таблицю двічі заборонено, тож дублікат — це помилка, яку треба виправити, а не спосіб щось перевизначити.
- Значення YAML без лапок змінює тип. За базовою схемою (core schema), яку YAML 1.2.2 рекомендує за замовчуванням, значення без лапок, що відповідає одному з її шаблонів, стає числом, булевим значенням або null: `1.10` — це дробове число 1,1, `0o17` — ціле 15, `0123` повертається як 123, `FALSE` — булеве значення, а `~` — null. Кожне значення, яке має лишитися текстом, слід брати в лапки, насамперед номери версій та ідентифікатори з нулями на початку. Специфікація також відносить повторені ключі у відображенні (mapping) до випадків, коли завантаження лишається неповним, тож ключ, записаний двічі, — це дефект.

- Налаштування виглядає чинним і нічого не робить. Правило, перекрите іншим, що стоїть вище, директива, яку новіший випуск прибрав, порт, з якого сервіс давно пішов: кожне лишається у файлі й має вигляд свідомого рішення. Jail на порту 22 й обмеження частоти під ширшим правилом обидва виглядали справними в кожному статусі. Перевірка, яка читає кожен файл окремо, їх пропускає, тож порядок і наявність треба звіряти із системою в роботі.
- Секрет мандрує разом із файлом. Пароль у файлі конфігурації потрапляє туди ж, куди й сам файл: у репозиторій, у резервні копії, у звіт про ваду. The Twelve-Factor App тримає конфігурацію у змінних середовища зокрема тому, що їх, на відміну від конфігураційних файлів, малоймовірно випадково закомітити в репозиторій коду.

Що почитати далі

- NIST CSRC Glossary: configuration — усі визначення, які публікації NIST дають цьому терміну, кожне з документом-джерелом.
- TOML v1.0.0 — повна специфікація, достатньо коротка, щоб прочитати її за один раз.
- YAML Ain't Markup Language, version 1.2.2 — специфікація; у розділі 10 описано базовий набір правил (Core Schema), який вирішує, чим стане значення без лапок.
- The Twelve-Factor App: Config — аргументи на користь того, щоб тримати значення конкретного розгортання поза кодом, разом із перевіркою на open source.
- sshd_config(5) — посібник із конфігурації сервера OpenSSH: правило першого значення, Include і Match.
- sshd(8) — посібник демона, де `-t` перевіряє конфігурацію, а `-T` виводить чинну.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке конфігурація?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/configuration/