**Engineer**

# YAML

формат налаштувань, який людина ще може читати

## Що це таке

YAML — це текстовий формат даних, у якому структуру задають відступи, двокрапки й дефіси замість дужок і лапок; саме тому його зазвичай обирають для конфігурації, яку люди пишуть і переглядають вручну. Назва — рекурсивний акронім «YAML Ain't Markup Language». Файли мають розширення `.yaml` або `.yml`, а RFC 9512, опублікований у лютому 2024 року, зареєстрував тип медіа `application/yaml` і назвав `.yaml` бажаним розширенням.

Під скупою пунктуацією лежить невелика модель даних. Є відображення (mappings), що пов'язують ключі зі значеннями, послідовності, тобто впорядковані списки, і скаляри: рядки, числа, булеві значення та null. Навколо цієї моделі мова додає те, що потрібно людині з файлом у редакторі: коментарі, які починаються з `#`, блокові скаляри, що зберігають розриви рядків через `|` або згортають їх через `>`, кілька документів в одному файлі, розділених `---`, якорі й псевдоніми (`&name` і `*name`), завдяки яким одне значення можна використати ще раз деінде, і теги, що прямо називають тип.

Перший фреймворк для YAML написали на Perl у 2001 році. Версія 1.1 специфікації вийшла 2005-го, версія 1.2 — 2009-го; головною її метою було зробити YAML строгою надмножиною JSON, і з неї прибрали чимало старих правил неявної типізації. Редакція 1.2.2, оприлюднена в жовтні 2021 року, виправила помилки й уточнила формулювання, не змінивши саму мову. Утім, багато програм досі читає файли за старими правилами: документація Ansible зазначає, що PyYAML і libyaml реалізують YAML 1.1.

Отже, YAML стоїть між двома сусідами. JSON — його строга підмножина, лаконічна й однозначна, створена для обміну даними між програмами. TOML — пласкіший формат конфігурації, де кожен рядок береться в лапки, тож тип значення ніколи не доводиться вгадувати. YAML уміє більше за обидва, і саме ця додаткова свобода породжує його проблеми.

Застосуйте поняття до рішення

Застосуйте «Що таке YAML?» 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

Справжній вибір стосується формату для файлу, який люди підтримуватимуть роками: описів розгортань, визначень конвеєрів, конфігурації серверів. Плейбуки Ansible пишуть на YAML, як і конфігураційні файли багатьох інших інфраструктурних інструментів, тож у такій роботі рішення часто ухвалено заздалегідь. Залишається вирішити, наскільки суворо писати ці файли.

Головний аргумент на користь YAML — читабельність. Рецензент читає diff, не рахуючи дужок, коментар пояснює, чому значення саме таке, а багаторядковий скрипт чи сертифікат уміщується в блоковий скаляр без екранування. З іншого боку, пробіли мають значення: ключ, зсунутий на два пробіли далі, може опинитися в іншому відображенні, а файл і далі розбирається. Табуляцію для відступів заборонено зовсім. Значення без лапок отримують тип за написанням, і правила написання в 1.1 та 1.2 різняться. Шаблони додають згори ще один синтаксис: в Ansible значення, що починається з `{{`, треба брати в лапки, інакше парсер сприйме його як відображення.

Вирішальне питання — як перевірятимуть файл. YAML з лінтером і звичкою брати рядки в лапки — зручний формат для конфігурації, яку команда редагує спільно. YAML, який ніхто не перевіряє, — формат, у якому

коректний файл може казати щось інше, ніж мав на увазі автор. Для даних, що передаються між програмами, надійніший вибір — JSON, а для короткого файлу налаштувань із кількома рівнями вкладеності простіший TOML.

## Як це виглядає на практиці

Інфраструктурний репозиторій компанії охоплює розгорнуті публічні сервіси та конфігурацію інших компонентів, зокрема ролі бази даних, резервного копіювання, поштового транспорту й DNS. Зафіксована реалізація налічувала 19 плейбуків Ansible і 34 ролі, загалом 12 065 рядків — це історичний підрахунок, а не поточний розмір репозиторію. Автоматизація резервного копіювання й пошти має окремий від конфігурації крок введення в дію. Склад компонентів задається даними, а не прапорцями. Кожен хост належить до групи рівня, YAML-змінні якої перелічують ролі для нього, тож запуску розгортання не потрібні аргументи, щоб привести всі хости до стану, оголошеного їхніми групами. Тут YAML слугує мовою оголошеного стану, а успіх перевіряють, а не припускають, бо хост, уже приведений до заданого стану, має звітувати про нуль змін. Використання YAML уповільнює внесення змін: кожна зміна займає більше часу, ніж редагування файла на сервері, а оскільки часткове застосування змін гірше за відмову від виконання, перед приведенням до заданого стану тепер виконується попередній перевірочний сценарій.

Дванадцять тисяч рядків YAML мусить прочитати машина, перш ніж їх побачить людина. На задокументованому етапі реалізації в серпні 2026 року YAML була однією з мов, які перевіряли ворота коміту з 22 лінтерів, разом із HTML, CSS, JavaScript, Python, Markdown і shell. Кожна діагностика там зупиняє коміт, бо рівня попереджень у цих воротах немає, а вбудовані придушення заборонено цілком: коментар ігнорування чи директива вимкнення у файлі перевірку не обходять. Для YAML це важливо, бо більшість пасток нижче — цілком коректний синтаксис, і зловити їх до того, як вони потраплять на хост, може лише правило, яке їх відхиляє.

## Де виникають проблеми

Пастки нижче мають спільну рису: файл розбирається без помилок, але означає щось інше.

**Код країни перетворюється на false.** За правилами YAML 1.1 значення `yes`, `no`, `on`, `off`, `y` і `n` без лапок є булевими, причому в кількох варіантах регістру. Базова схема (core schema) YAML 1.2 залишає тільки `true` і `false` у трьох написаннях, а RFC 9512 попереджає, що парсер за старим синтаксисом може прочитати `{insecure: n}` як false, а не як літеру n. Список північних країн показує проблему:

```
країни:
    - DK
    - SE
    - NO # YAML 1.1 читає це як false
```

Беріть у лапки кожен рядок, який можна прочитати як щось інше, записуйте булеві значення лише як `true` і `false` і доручіть лінтеру стежити за обома правилами.

**Номер версії втрачає цифру.** Числа без лапок перетворюються, тож `version: 1.10` надходить як число з рухомою комою 1.1, а за правилами 1.1 початковий нуль позначає вісімкове число, і код `010` стає вісімкою. Документація Ansible саме тому радить брати версію на кшталт `"1.0"` у лапки. Беріть у лапки ідентифікатори, коди й версії: це рядки, у яких просто трапляються цифри.

**Ключ опиняється під чужим батьком.** Відступ визначає, якому відображенню належить ключ, тож рядок, зсунутий на два пробіли, переносить налаштування в інше місце без жодного повідомлення про помилку. Специфікація також вимагає, щоб кожен ключ у відображенні був унікальним, а повторений ключ глибоко в довгому блоці легко проґавити під час рецензії, тому yamllint має окреме правило для кожної з цих двох помилок. Запускайте його до того, як файл піде в роботу.

**Завантаження файлу запускає код.** Теги можуть називати типи конкретної мови програмування, і завантажувач, який їх виконує, збудує будь-який об’єкт, що його замовить файл. Документація PyYAML порівнює використання `yaml.load` (з небезпечним завантажувачем) із `pickle.load`: такий виклик може створювати об’єкти й викликати функції Python. Вибір завантажувача важливий; у поточних версіях його треба вказувати явно. `yaml.safe_load` розпізнає лише базові теги YAML. RFC 9512

просить, щоб виконання коду під час десеріалізації було типово вимкнене. Використовуйте безпечний завантажувач для звичайних конфігураційних файлів.

**Короткий файл заповнює пам'ять.** Якорі й псевдоніми роблять документ YAML графом, а не деревом. Коли кожен якір двічі повторює попередній, кожен новий рядок подвоює розмір, і тридцять коротких рядків описують понад мільярд текстових значень, щойно програма обходить результат як звичайне дерево. RFC 9512 називає це експоненційним розростанням даних, атакою billion laughs, і відповідає на неї процесорами, що обмежують глибину рекурсії якорів. Приймайте YAML ззовні лише через такий процесор і перевіряйте це обмеження на тому парсері, який справді використовуєте.

## Що почитати далі

- YAML 1.2.2 specification — сама мова, зокрема базова схема, яка визначає типи значень без лапок.
- RFC 9512: YAML Media Type — реєстрація `application/yaml` із нотатками з безпеки про булеві значення, виконання коду й розгортання псевдонімів.
- YAML specification changes — що змінила кожна редакція, від 1.2.1 до 1.2.2.
- Ansible: YAML Syntax — підмножина, якою користується Ansible, і її пастки з лапками.
- yamllint rules — правила для булевих значень, дубльованих ключів, вісімкових чисел і відступів.
- PyYAML documentation — різниця між `load` і `safe_load` словами самої бібліотеки.

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 YAML?» 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/yaml/**