Engineer

YAML

ein für Menschen lesbares Format für Einstellungen

Was es ist

YAML ist ein Textformat für verschachtelte Daten. Es kennzeichnet Struktur durch Einrückungen, Doppelpunkte und Bindestriche statt durch Klammern und Anführungszeichen. Deshalb ist es eine übliche Wahl für Konfigurationen, die Menschen von Hand schreiben und prüfen. Der Name ist ein rekursives Akronym: „YAML Ain't Markup Language", also „YAML ist keine Auszeichnungssprache". Dateien enden auf `.yaml` oder `.yml`. Der im Februar 2024 veröffentlichte RFC 9512 registrierte den Medientyp `application/yaml` und nannte `.yaml` als bevorzugte Dateiendung.

Hinter den wenigen Satzzeichen steckt ein kleines Datenmodell. Es gibt Zuordnungen, die Schlüssel und Werte paaren, Sequenzen als geordnete Listen und Skalare: Zeichenfolgen, Zahlen, boolesche Werte und null. Darum herum ergänzt die Sprache, was Menschen beim Bearbeiten einer Datei brauchen: mit `#` beginnende Kommentare, Blockskalare, die mit `|` Zeilenumbrüche erhalten oder mit `>` zusammenfalten, mehrere durch `---` getrennte Dokumente in einer Datei, Anker und Aliasse (`&name` und `*name`), mit denen ein Wert anderswo wiederverwendet werden kann, und Tags, die einen Typ ausdrücklich angeben.

Das erste YAML-Framework wurde 2001 in Perl geschrieben. Version 1.1 der Spezifikation folgte 2005 und Version 1.2 im Jahr 2009. Deren Hauptziel war, YAML zu einer echten Obermenge von JSON zu machen; viele ältere Regeln zur impliziten Typisierung wurden entfernt. Die im Oktober 2021 veröffentlichte Revision 1.2.2 korrigierte Fehler und präzisierte Formulierungen, ohne die Sprache zu ändern. Viele Programme lesen jedoch noch nach den älteren Regeln. Die Ansible-Dokumentation hält fest, dass PyYAML und libyaml YAML 1.1 implementieren.

YAML steht damit zwischen zwei Nachbarn. JSON ist eine echte Teilmenge: knapp und eindeutig, für den Datenaustausch zwischen Programmen gemacht. TOML ist ein flacheres Konfigurationsformat, in dem jede Zeichenfolge in Anführungszeichen steht und ihr Typ daher nie erraten wird. YAML bietet mehr als beide. Genau diese zusätzliche Freiheit ist die Ursache seiner Schwierigkeiten.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist YAML?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Die gesamte Einrichtung wird in Dateien festgehalten, damit sie sich exakt wiederherstellen lässt und nicht nur im Kopf einer einzelnen Person existiert.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Die eigentliche Entscheidung lautet, welches Format eine über Jahre von Menschen gepflegte Datei bekommen soll: Deployment-Beschreibungen, CI-Abläufe oder Serverkonfigurationen. Ansible-Playbooks sind YAML, ebenso die Konfigurationsdateien vieler anderer Infrastrukturwerkzeuge. Für einen großen Teil dieser Arbeit ist die Entscheidung daher schon vorgelagert gefallen. Offen bleibt, wie streng man das Format schreibt.

Für YAML spricht die Lesbarkeit. Ein Prüfer kann einen Änderungsvergleich lesen, ohne Klammern zu zählen. Ein Kommentar erklärt den Grund für einen Wert, und ein mehrzeiliges Skript oder Zertifikat passt ohne Escapezeichen in einen Blockskalar. Dagegen steht, dass Leerraum Bedeutung trägt: Ein um zwei Leerzeichen zu weit eingerückter Schlüssel kann in einer anderen Zuordnung landen, während die Datei weiterhin gültig eingelesen wird. Tabulatoren sind zum Einrücken ganz verboten. Nicht in Anführungszeichen gesetzte Werte erhalten ihren Typ nach ihrer Schreibweise; die Regeln dafür unterscheiden sich zwischen 1.1 und 1.2. Vorlagen legen eine zweite Syntax darüber. In Ansible muss ein mit `{{` beginnender Wert in Anführungszeichen stehen, sonst liest ihn der Parser als Zuordnung.

Den Ausschlag gibt, wie die Datei geprüft wird. Mit einem Linter und der Gewohnheit, Zeichenfolgen in Anführungszeichen zu setzen, ist YAML ein angenehmes Format für gemeinsam bearbeitete Konfigurationen. Fehlt beides,

kann eine gültige Datei etwas anderes aussagen als von ihrem Autor beabsichtigt. Für Daten zwischen Programmen ist JSON die sicherere Standardwahl. Für eine kurze Einstellungsdatei mit wenigen Verschachtelungsebenen ist TOML einfacher.

Wie es in der Praxis aussieht

Das Infrastruktur-Repository des Unternehmens umfasst öffentlich betriebene Dienste und die Konfiguration weiterer Komponenten, darunter Datenbank-, Backup-, Mailtransport- und DNS-Rollen. Die dokumentierte Implementierung umfasste 19 Ansible-Playbooks und 34 Rollen mit insgesamt 12.065 Zeilen. Das ist eine historische Zählung, nicht die aktuelle Repository-Größe. Die Backup- und Mailautomatisierung hat einen von ihrer Konfiguration getrennten betrieblichen Aktivierungsschritt. Die Zusammensetzung steckt in Daten statt in Schaltern. Jeder Host gehört zu einer Gruppe einer Infrastrukturebene. Deren YAML-Variablen listen die Rollen des Hosts auf. Ein Provisionierungslauf braucht daher keine Argumente, um alle Hosts in den von ihrer Gruppe deklarierten Zustand zu bringen. Hier dient YAML als Sprache des deklarierten Zustands. Der Erfolg wird geprüft, nicht vorausgesetzt: Ein bereits in diesen Zustand gebrachter Host muss null Änderungen melden. YAML kostet dabei Geschwindigkeit. Jede Änderung dauert länger als das direkte Bearbeiten einer Datei auf dem Server. Weil ein nur halb angewendeter Lauf schlimmer ist als ein abgewiesener, steht inzwischen ein Play für Vorabprüfungen davor.

Zwölftausend Zeilen YAML sollte zuerst eine Maschine lesen und erst dann ein Mensch. Im dokumentierten Implementierungsstand vom August 2026 gehörte YAML zu den Sprachen einer verbindlichen Qualitätsprüfung mit 22 Lintern vor jedem Commit, neben HTML, CSS, JavaScript, Python, Markdown und Shell. Eine Warnstufe gibt es nicht: Jede Diagnose verhindert den Commit. Unterdrückungen direkt in der Datei sind grundsätzlich verboten; weder ein Ignorierkommentar noch eine Deaktivierungsanweisung ist ein Ausweg. Für YAML zählt das besonders, weil die meisten folgenden Fallen syntaktisch völlig gültig sind. Nur eine Regel, die sie zurückweist, fängt sie ab, bevor sie einen Host erreichen.

Wo es schiefgeht

Die folgenden Fallen haben eines gemeinsam: Die Datei lässt sich fehlerfrei einlesen und bedeutet etwas anderes.

Ein Ländercode wird zum Wahrheitswert falsch. Nach den Regeln von YAML 1.1 sind die nicht in Anführungszeichen gesetzten Werte `yes`, `no`, `on`, `off`, `y` und `n` in mehreren Groß- und Kleinschreibungen boolesch. Das Core-Schema von YAML 1.2 behält nur `true` und `false`, in drei Schreibweisen. RFC 9512 warnt, dass ein Parser mit älterer Syntax `{insecure: n}` als falsch statt als den Buchstaben n lesen kann. Eine Liste nordischer Länder zeigt das Problem:

```
countries:
    - DK
    - SE
    - NO # YAML 1.1 reads this as false
```

Setzen Sie jede Zeichenfolge, die anders gelesen werden könnte, in Anführungszeichen. Schreiben Sie boolesche Werte nur als `true` und `false` und lassen Sie einen Linter beides durchsetzen.

Eine Versionsnummer verliert eine Ziffer. Nicht in Anführungszeichen gesetzte Zahlen werden umgewandelt. So kommt `version: 1.10` als Gleitkommazahl 1,1 an. Nach den Regeln von 1.1 markiert eine führende Null zudem eine Oktalzahl: Ein als `010` geschriebener Code wird zu 8. Genau deshalb fordert Ansible seine Benutzer auf, eine Version wie `"1.0"` in Anführungszeichen zu setzen. Tun Sie das bei Kennungen, Codes und Versionen: Es sind Zeichenfolgen, die zufällig Ziffern enthalten.

Ein Schlüssel landet unter dem falschen Elternknoten. Die Einrückung legt fest, zu welcher Zuordnung ein Schlüssel gehört. Eine um zwei Leerzeichen verschobene Zeile verlegt eine Einstellung deshalb ohne Fehler an eine andere Stelle. Die Spezifikation verlangt außerdem, dass jeder Schlüssel einer Zuordnung eindeutig ist. Eine weit unten in einem langen Block wiederholte Angabe ist bei der Prüfung leicht zu übersehen. yamllint hat daher für beide Fehler eine Regel. Führen Sie es aus, bevor die Datei verwendet wird.

Das Laden einer Datei führt Code aus. Tags können sprachspezifische Typen benennen. Ein Loader, der sie berücksichtigt, erzeugt jedes von der Datei verlangte Objekt. Die PyYAML-Dokumentation vergleicht das Laden mit `yaml.load` (bei Verwendung eines unsicheren Loaders) mit `pickle.load`: Dabei können Objekte erzeugt und Python-Funktionen aufgerufen werden. Die Wahl des Loaders ist entscheidend; aktuelle Versionen verlangen sie ausdrücklich.

`yaml.safe_load` erkennt nur grundlegende YAML-Tags. RFC 9512 verlangt, dass die Codeausführung in Deserialisierern standardmäßig abgeschaltet ist. Verwenden Sie für gewöhnliche Konfigurationsdateien den sicheren Loader.

Eine kurze Datei füllt den Speicher. Anker und Aliasse machen ein YAML-Dokument zu einem Graphen statt zu einem Baum. Wiederholt jeder Anker den vorherigen zweimal, verdoppelt jede zusätzliche Zeile die Größe. Dreißig kurze Zeilen beschreiben so mehr als eine Milliarde Zeichenfolgen, sobald ein Programm das Ergebnis als einfachen Baum durchläuft. RFC 9512 nennt dies exponentielle Datenausweitung, den Billion-Laughs-Angriff, und fordert dafür Prozessoren, die die Rekursionstiefe von Ankern begrenzen. Nehmen Sie YAML von außen nur über einen solchen Prozessor an und testen Sie die Grenze mit dem tatsächlich verwendeten Parser.

Weiterführende Lektüre

- YAML-Spezifikation 1.2.2 — die Sprache selbst, einschließlich des Core-Schemas zur Typisierung nicht in Anführungszeichen gesetzter Werte.
- RFC 9512: YAML Media Type — die Registrierung von `application/yaml` mit Sicherheitshinweisen zu booleschen Werten, Codeausführung und Aliasexpansion.
- Änderungen der YAML-Spezifikation — die Änderungen jeder Revision, von 1.2.1 bis 1.2.2.
- Ansible: YAML-Syntax — die von Ansible verwendete Teilmenge und ihre Fallen bei Anführungszeichen.
- yamllint-Regeln — Prüfungen auf boolesch interpretierte Werte, doppelte Schlüssel, Oktalzahlen und Einrückungen.
- PyYAML-Dokumentation — der Unterschied zwischen `load` und `safe_load` in den Worten der Bibliothek.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist YAML?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/yaml/