Engineer

YAML

人が読んで手直しできる設定ファイルの形式

どのようなものか

YAMLは、入れ子になったデータを表すテキスト形式です。括弧や引用符の代わりに、字下げ、コロン、ハイフンで構造を示すため、人が手で書き、レビューする設定によく使われます。名前は、自分自身を含む頭字語の"YAML Ain't Markup Language"です。ファイルの拡張子は `.yaml` または `.yml` です。2024年2月に公開されたRFC 9512は、メディアタイプ `application/yaml` を登録し、推奨する拡張子を `.yaml` としました。

簡素な記法の土台には、小さなデータモデルがあります。キーと値を組にするマッピング、順序付きの一覧であるシーケンス、文字列、数値、真偽値、nullを表すスカラーです。その上に、ファイルを編集する人が欲しい機能を加えています。 `#` で始まるコメント、 `|` で改行を保つか `>` で折り畳むブロックスカラー、 `---` で区切って1ファイルに収める複数の文書、ある値を別の場所でも使えるアンカーとエイリアス（ `&name` と `*name` ）、型を明示するタグです。

最初期のYAMLフレームワークは、2001年にPerlで書かれました。仕様のバージョン1.1は2005年、1.2は2009年に続きました。1.2は、YAMLをJSONの厳密な上位集合にすることを主眼とし、それ以前の暗黙的な型付け規則を多く取り除きました。2021年10月に公開された改訂版1.2.2は、言語を変えず、誤りを直し、表現を明確にしたものです。ただし、古い規則で読むソフトウェアも多く残っています。Ansibleの文書は、PyYAMLとlibyamlがYAML 1.1を実装していると説明しています。

YAMLは、2つの隣接する形式の間に位置します。JSONはその厳密な部分集合で、簡潔で曖昧さがなく、プログラム間の交換向けです。TOMLは、入れ子の少ない設定形式で、すべての文字列を引用符で囲むため、値の型を推測しません。YAMLはどちらよりも多くの機能を持ち、その自由度の高さこそが、問題の起点になります。

概念を判断に生かす

「YAMLとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 構成全体をファイルに記述し、特定の人の記憶だけに頼らず、まったく同じ環境を再現できるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

Linuxサーバーを守る1,000件のRBACルールを作成 お問い合わせ →

なぜ重要なのか

実際に選ぶのは、人が何年も保守するファイルに、どの形式を与えるかです。デプロイの定義、CIのワークフロー、サーバーの設定などが該当します。Ansibleのプレイブックも、多くのインフラツールの設定ファイルもYAMLなので、その種の作業の多くは、すでにツール側で形式が決まっています。残る判断は、どれだけ厳格に書くかです。

YAMLの利点は読みやすさです。レビューする人は波括弧を数えずに差分を読め、値を選んだ理由はコメントで説明でき、複数行のスクリプトや証明書はエスケープせずブロックスカラーに収まります。一方、空白に意味があるため、キーを2スペース深く字下げしただけで、構文解析は通っても別のマッピングに入ることがあります。字下げにタブは一切使えません。引用符のない値は綴りから型を決めますが、その規則は1.1と1.2で異なります。テンプレートを使うと、さらに別の構文が重なります。Ansibleでは、`{{` で始まる値を引用符で囲まないと、パーサーがマッピングと解釈します。

決め手は、ファイルをどう検査するかです。リンターを使い、文字列を引用符で囲む習慣があれば、YAMLはチームで共同編集する設定に使いやすい形式です。どちらもなければ、正しいファイルが、書いた人の意図とは違うことを表しかねません。プログラム間で渡すデータには、基本の選択肢としてはJSONのほうが安全で、入れ子が浅い短い設定ファイルにはTOMLのほうが簡単です。

実際の活用例

当社のインフラリポジトリーは、デプロイ済みの公開サービスと、データベース、バックアップ、メール転送、DNSのロールを含む、ほかの構成要素の設定を扱います。記録された実装は、19のAnsibleプレイブックと34のロール、合計12,065行でした。これは過去の記録であり、現在のリポジトリーの規模ではありません。バックアップとメールの自動化には、設定とは別に、運用を有効にする手順があります。構成を決めるのはフラグではなくデータです。各ホストは階層

別のグループに属し、そのグループのYAML変数が、ホストに割り当てるロールを列挙します。そのため、プロビジョニングの実行には引数を渡さず、全ホストを各グループが宣言する状態にできます。YAMLを、宣言した状態を表す言語として使う例です。成功したと思い込むのではなく、収束したホストが変更ゼロを報告することを確認します。ただし、速度は犠牲になります。変更のたびに、サーバー上のファイルを直接編集するより時間がかかります。また、途中まで適用して止まるほうが、最初から拒否するより悪いため、現在は事前検査のプレイを先に実行します。

約12,000行のYAMLは、人が読む前に機械で読む必要があります。記録されている2026年8月の実装段階では、22のリンターを使うコミットゲートがYAMLを検査しました。ほかの対象はHTML、CSS、JavaScript、Python、Markdown、シェルです。このゲートには警告という扱いがなく、診断が1つでも出るとコミットを止めます。インラインでの抑制も全面的に禁止し、ファイル内の無視コメントや無効化指示を抜け道にしません。YAMLでは、これが重要です。以下の落とし穴のほとんどは構文としては正しく、禁止する規則があって初めて、ホストに届く前に検出できます。

問題が起きるところ

以下の落とし穴には、共通点があります。どれも構文解析は通り、別の意味になります。

国コードがfalseになる。 YAML 1.1の規則では、引用符で囲まない `yes` 、 `no` 、 `on` 、 `off` 、 `y` 、 `n` は、大文字と小文字のいくつかの表記を含め、真偽値です。YAML 1.2のコアスキーマでは、3通りの大文字・小文字表記による `true` と `false` だけを残しました。RFC 9512は、古い構文に従うパーサーが、 `{insecure: n}` を文字のnではなくfalseとして読むおそれを指摘しています。北欧の国を並べると、問題が分かります。

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

別のものとして読まれる可能性のある文字列は引用符で囲み、真偽値は `true` と `false` だけで書き、両方をリンターで徹底します。

バージョン番号から数字が落ちる。 引用符のない数値は変換されるため、`version: 1.10` は浮動小数点数の1.1になり、YAML 1.1の規則では先頭のゼロが8進数を示すため、`010` と書いたコードは8になります。Ansibleの文書が、`"1.0"` のようなバージョンを引用符で囲むよう求めるのも、このためです。識別子、コード、バージョンは、数字を含んでいるだけの文字列なので、引用符で囲みます。

キーが違う親の下に入る。 どのマッピングにキーが属するかは字下げで決まるため、1行を2スペースずらすだけで、エラーなしに設定が別の場所へ移ります。仕様は、同じマッピングのキーがすべて一意であることも求めています。長いブロックの下のほうで同じキーが現れても、レビューでは見落としがちです。そのため、yamllintには、この2種類の間違いにそれぞれ規則があります。ファイルを使う前に実行します。

ファイルの読み込みでコードが動く。 タグは、特定の言語の型を指定できます。それを解釈するローダーは、ファイルで指定されたオブジェクトを構築します。PyYAMLの文書は、安全でないローダーを使う `yaml.load` を `pickle.load` と比較し、オブジェクトの構築やPython関数の呼び出しが可能だと説明しています。ローダーの選択が重要で、現在のバージョンは明示的な指定を求めます。`yaml.safe_load` が認識するのは、基本のYAMLタグだけです。RFC 9512は、デシリアライザーによるコード実行を、初期状態で無効にするよう求めています。通常の設定ファイルには、安全なローダーを使います。

短いファイルがメモリーを埋める。 アンカーとエイリアスがあるため、YAML文書は木ではなくグラフになります。各アンカーが直前のアンカーを2回繰り返すと、1行加えるたびにサイズが倍になり、わずか30行でも、プログラムが結果を通常の木としてたどると、10億を超える文字列を表します。RFC 9512は、この指数関数的なデータ展開をbillion laughs攻撃と呼び、アンカーの再帰の深さを制限する処理系を対策としています。外部からYAMLを受け取る際は、そのような処理系だけを通し、実際に使うパーサーで上限を試します。

参考資料

- YAML 1.2.2仕様 — 引用符のない値の型を決めるコアスキーマを含む、言語そのものの仕様。
- RFC 9512: YAMLメディアタイプ — `application/yaml` の登録と、真偽値、コード実行、エイリアスの展開についてのセキュリティ上の注意。
- YAML仕様の変更 — 1.2.1から1.2.2など、各改訂で変わった点。
- Ansible: YAMLの構文 — Ansibleが使う部分集合と、引用符の落とし穴。
- yamllintの規則 — 真偽値になる値、キーの重複、8進数、字下げの検査。

- PyYAMLの文書 — ライブラリー自身による、`load` と `safe_load` の違いの説明。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「YAMLとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く 連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/yaml/