Engineer

YAML

a format for settings a person can still read

What it is

YAML is a text format for nested data that marks structure with indentation, colons and dashes instead of brackets and quotes, which is why it is the usual choice for configuration that people write and review by hand. The name is a recursive acronym, “YAML Ain’t Markup Language”. Files end in `.yaml` or `.yml`, and RFC 9512, published in February 2024, registered the media type `application/yaml` and named `.yaml` the preferred extension.

Underneath the light punctuation sits a small data model. There are mappings, which pair keys with values, sequences, which are ordered lists, and scalars: strings, numbers, booleans and null. Around that model the language adds what a person editing a file wants: comments that start with `#`, block scalars that keep line breaks with `|` or fold them with `>`, several documents in one file separated by `---`, anchors and aliases (`&name` and `*name`) that let one value be reused elsewhere, and tags that name a type outright.

The earliest YAML framework was a Perl one, written in 2001. Version 1.1 of the specification followed in 2005, and version 1.2 in 2009, with a primary focus on making YAML a strict superset of JSON and with many of the older implicit typing rules removed. Revision 1.2.2, published in October 2021, corrected errors and clarified wording without changing the language. Plenty of software still reads the older rules, though: Ansible’s documentation notes that PyYAML and libyaml implement YAML 1.1.

So YAML sits between two neighbours. JSON is its strict subset, terse and unambiguous, made for programs to exchange. TOML is a flatter configuration format in which every string is quoted, so a value’s type is never guessed. YAML offers more than either, and the extra freedom is exactly where its trouble comes from.

Use the concept in a decision

Put “What is YAML?” to work in a real decision. We can help relate the concept to the system you are building or buying. The whole setup written down as files, so it can be recreated exactly and never lives only in one person's head.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Wrote 1,000 RBAC rules to secure Linux servers — 1,000 access rules written to close a security gap

Wrote 1,000 RBAC rules to secure Linux servers Get in touch →

Why it matters

The real choice is what format to give a file that people will maintain for years: deployment descriptions, CI workflows, server configuration. Ansible playbooks are YAML, and so are the configuration files of many other infrastructure tools, so for much of that work the decision has already been made upstream, and what is left to decide is how strictly to write it.

The case for YAML is readability. A reviewer can read a diff of it without counting braces, a comment can explain why a value is what it is, and a multi-line script or certificate fits in a block scalar without escaping. Against that, whitespace carries meaning, so a key indented two spaces too far can land in a different mapping while the file still parses. Tabs are forbidden for indentation altogether. Unquoted values are typed by their spelling, and the spelling rules differ between 1.1 and 1.2. Templating adds a second syntax on top: in Ansible, a value that begins with `{{` must be quoted, or the parser takes it for a mapping.

What tips the decision is how the file will be checked. YAML with a linter and a habit of quoting strings is a comfortable format for configuration a team edits together. YAML without either is a format in which a valid file can say something other than what its author meant. For data passed between programs, JSON is the safer default, and for a short settings file with few levels of nesting TOML is simpler.

What it looks like in practice

The company’s infrastructure repository covers deployed public services and configuration for other components, including database, backup, mail transport and DNS roles. Its recorded implementation comprised 19 Ansible playbooks and 34 roles, 12,065 lines in all, a historical count rather than the current repository size. Backup and mail automation have an operational activation step separate

from their configuration. Composition is data rather than flags. Each host belongs to a tier group, and the YAML variables of that group list the roles the host gets, so a provisioning run needs no arguments to bring every host to the state its group declares. That is YAML used as the language of declared state, and its success is checked rather than assumed, because a converged host has to report zero changes. The YAML costs speed: every change is slower than editing a file on the server would be, and because a converge that half-applies is worse than one that refuses, a preflight play now runs in front of it.

Twelve thousand lines of YAML need a machine reading them before a person does. At the documented August 2026 implementation stage, YAML was one of the languages a 22-linter commit gate checked, alongside HTML, CSS, JavaScript, Python, Markdown and shell. The gate has no warning tier, so every diagnostic stops the commit, and inline suppressions are prohibited outright: an ignore comment or a disable directive in the file is not an escape route. For YAML that matters, because most of the traps below are perfectly valid syntax, and only a rule that refuses them catches them before they reach a host.

Where it goes wrong

The traps below share one trait: the file parses cleanly and means something else.

A country code turns into false. Under YAML 1.1 rules, unquoted `yes`, `no`, `on`, `off`, `y` and `n`, in several capitalisations, are booleans. The YAML 1.2 core schema keeps only `true` and `false`, in three capitalisations, and RFC 9512 warns that a parser following the older syntax can read `{insecure: n}` as false rather than as the letter n. A list of Nordic countries shows the problem:

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

Quote any string that could be read as something else, write booleans only as `true` and `false`, and let a linter enforce both.

A version number loses a digit. Unquoted numbers are converted, so `version: 1.10` arrives as the float 1.1, and under 1.1 rules a leading zero marks an octal number, so a code written as `010` becomes 8. Ansible's documentation tells its users to quote a version such as `"1.0"` for exactly this reason. Quote identifiers, codes and versions, which are strings that happen to contain digits.

A key ends up under the wrong parent. Indentation decides which mapping a key belongs to, so one line shifted by two spaces moves a setting somewhere else without any error. The specification also requires every key in a mapping to be unique, and a repeated key far down a long block is easy to miss in review, so yamllint carries a rule for each of the two mistakes. Run it before the file is used.

Loading a file runs code. Tags can name language-specific types, and a loader that honours them will construct whatever object the file asks for. PyYAML's documentation compares `yaml.load` with an unsafe loader to `pickle.load`: it can construct objects and invoke Python functions. The loader choice matters; current versions require it explicitly. `yaml.safe_load` recognises only basic YAML tags. RFC 9512 asks for code execution in deserializers to be off by default. Use the safe loader for ordinary configuration files.

A short file fills the memory. Anchors and aliases make a YAML document a graph rather than a tree. When each anchor repeats the one before it twice, every added line doubles the size, so thirty short lines describe more than a billion strings once a program walks the result as a plain tree. RFC 9512 names this exponential data expansion, the billion laughs attack, and answers it with processors that limit anchor recursion depth. Accept YAML from outside only through such a processor, and test the limit against the parser actually in use.

Further reading

- YAML 1.2.2 specification — the language itself, including the core schema that decides how unquoted values are typed.
- RFC 9512: YAML Media Type — the `application/yaml` registration, with security notes on booleans, code execution and alias expansion.
- YAML specification changes — what each revision changed, from 1.2.1 to 1.2.2.
- Ansible: YAML Syntax — the subset Ansible uses, with its quoting gotchas.
- yamllint rules — the checks for truthy values, duplicate keys, octal numbers and indentation.

- PyYAML documentation — the difference between `load` and `safe_load`, in the library's own words.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how "What is YAML?" applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

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

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/yaml/