Engineer

Configuration

the settings that decide how a system behaves

What it is

Configuration is everything a program reads from outside its own source code to decide how to run in one particular place: the port it listens on, the database it connects to, the accounts it lets in, the features it switches on. In conversation it is nearly always shortened to config, and the files that carry it are config files. NIST, in SP 800-128, its guide to security-focused configuration management, defines it as “the possible conditions, parameters, and specifications with which an information system or system component can be described or arranged”. The word to hold on to is described: a configuration is a description, and the system may or may not agree with it.

A program seldom takes its configuration from one place. Defaults are compiled into the binary, a main file sits on disk with a directory of drop-in fragments beside it, and environment variables and command-line flags arrive at start. The program merges these into one set of values, with a rule for what happens when two sources set the same key, and that rule belongs to each program. The OpenSSH server keeps the first value it reads for most keywords, while other tools let a later file override an earlier one. The same files read in a different order produce a different system.

The formats vary: INI-style lines, TOML, YAML, JSON and formats private to one program. TOML 1.0.0 aims to be “a minimal configuration file format”, while YAML 1.2.2 is a general data language that config files borrow, and each has its own rules for turning text into values.

Use the concept in a decision

Put “What is configuration?” 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

Configuration is how one build serves many places. The same program runs on a laptop, a staging host and production, and what differs between them is its configuration. The Twelve-Factor App, a methodology for deployable services, draws the line there: config is “everything that is likely to vary between deploys”, such as handles to backing services, credentials for outside services and the hostname of each deploy. It requires strict separation of config from code, and it offers a test worth applying anywhere: whether “the codebase could be made open source at any moment, without compromising any credentials”.

The same text marks the opposite edge. Internal wiring that stays the same on every deploy, such as a web framework’s table of routes, falls outside its definition and is “best done in the code”. A value that is the same on every deploy is code dressed as a setting, and a file only gives it another place to be wrong.

Configuration is also where a system states its security: who may log in, which ports answer, what a firewall lets through. A host is exactly as hardened as the values it is actually using, and between a file and those values sit the parser, the precedence rules, the compiled defaults, the program version and the environment. Any of them can change the outcome while the file stays the same, so the dependable habit is to ask the program what it is using and treat that answer as the configuration.

What it looks like in practice

This site is configured in one TOML file, hugo.toml, which sets its languages, taxonomies and mounts, among much else. One table in hugo.toml shows how much a few lines can decide:

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

Each list is a lookup order: a date written in the page itself, then the date of the last commit that touched the file, then the file's modification time. A flag near the top of the file, `enableGitInfo = true`, lets Hugo read the git history at all. The comment above the table explains the first entry: the literal front-matter field has to be named, or a `lastmod` written in a page is silently ignored while the site builds as usual.

That table is the fix described in repairing a sitemap in which most pages shared one timestamp. The sitemap had told search engines that 172 of the site's 176 pages last changed at the same instant: with the file's modification time as the source of the date, each run of the content export, which writes out every file again, restamped all of them at once. The fallback chain moved the source to the version history. At that recorded implementation stage, a re-sync after a month of content edits touched eight files of 186.

On a server, the same lesson runs through hardening SSH with three stages of validation. An automation role had written the SSH daemon's configuration for months when an access report in September 2026 read the daemon's resolved settings and found two of them disagreeing with the role's file. The operating system's own defaults sat above the drop-in, and for those two settings the first occurrence wins, so the fix was a filename prefix that sorts ahead of the vendor's. Around it the role now validates in three stages: the candidate file is syntax-checked before install, the assembled configuration is checked after, and the resolved output is read back with 24 directives asserted. Two directives were left out on purpose, because the daemon no longer implements them.

Three SSH brute-force protections that never worked shows the failure from the other side. In August 2026 a review set each protection's configuration beside the running host. All three were written down, all three printed in every report, and none had done its job since the day the host was built. The jail's configuration still named port 22, a value left behind when the initial hardening moved the daemon to 1986. The rate limit's own line was right, and the order of the firewall's rules undid it: a broader rule listed above it caught the same traffic first. The ban action's configuration called a packet-filter binary that the package system recommends without requiring, and on this host that program never resolved. Two rules came out of the review: every security control is backed by an assertion that tests it, never by a comment that describes it, and a firewall is checked by where each rule sits in the order, not by whether it is listed.

Where it goes wrong

Each failure below is a file that is correct as text and wrong as behaviour, and each has a check that reads the behaviour.

- The wrong source wins. The OpenSSH manual for its server configuration says: “Unless noted otherwise, for each keyword, the first obtained value will be used.” A vendor default read before your drop-in therefore beats it, and a tool where the last value wins turns the same layout around. Read the precedence rule in the program’s own manual, then ask the program: `sshd -t` checks that a configuration is valid, and `sshd -T` prints the effective one.
- A TOML key lands in the wrong table. In TOML 1.0.0 the pairs under a `[table]` header belong to it “until the next header or EOF”, and indentation is ignored. A key added at the bottom of a file joins the last table even when it is written flush left. Keep tables at the end and plain keys above them. Defining a key or a table twice is invalid, so a duplicate is an error to fix, never a way to override.
- A plain YAML value changes type. Under the core schema, which YAML 1.2.2 recommends as the default, an unquoted value that matches one of its patterns becomes a number, a boolean or null: `1.10` is the float 1.1, `0o17` is the integer 15, `0123` comes back as 123, `FALSE` is a boolean and `~` is null. Quote every value that has to stay text, version strings and identifiers with leading zeros first. The spec also counts repeated mapping keys among the ways a load ends incomplete, so a key written twice is a defect.
- A setting reads as active and does nothing. A rule shadowed by an earlier one, a directive a newer release has dropped, a port that the service has since left: each stays in the file and looks deliberate. The jail on port 22 and the rate limit beneath a broader rule both printed as healthy. A check that reads each file alone passes them, so assert order and presence against the running system.
- A secret travels with the file. A password in a configuration file goes wherever the file goes: the repository, the backups, the bug report. Twelve-Factor keeps config in environment variables partly because, unlike config files, “there is little chance of them being checked into the code repo accidentally”.

Further reading

- NIST CSRC Glossary: configuration — each definition NIST publications give the term, with its source document.
- TOML v1.0.0 — the complete specification, short enough to read in one sitting.
- YAML Ain’t Markup Language, version 1.2.2 — the specification; chapter 10 holds the core schema that decides what a plain value becomes.
- The Twelve-Factor App: Config — the case for keeping deploy-specific values out of the code, with the open-source test.

- sshd_config(5) — the OpenSSH server's configuration manual, with its first-value rule, Include and Match.
- sshd(8) — the daemon's manual, where `-t` checks a configuration and `-T` prints the effective one.

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 configuration?" 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/configuration/