Engineer

nginx

a web server and reverse proxy

What it is

nginx is an open-source program that accepts HTTP connections and, for each request, either answers it from files on disk or hands it to another server and relays the reply. The name is pronounced “engine x”, and the project’s own pages also write it NGINX. Igor Sysoev wrote it, and the first public version, 0.1.0, came out on 4 October 2004. It is free to use under a 2-clause BSD licence, and F5, Inc. sells enterprise distributions, support and training.

The project lists six roles: HTTP server, reverse proxy, content cache, load balancer, TCP and UDP proxy, and mail proxy. On a web host it usually holds one or both of two jobs: serving static files from disk, and standing in front of an application server, where it terminates TLS and forwards each request inward.

When it starts, nginx runs a single master process and a number of worker processes. The master reads and checks the configuration and looks after the workers. The workers run as an unprivileged user and do all of the request handling. Each one uses an event-based loop, built on epoll on Linux and kqueue on FreeBSD, so a handful of workers can keep thousands of connections open at once. The worker_processes directive sets how many there are, either as a fixed number or as auto for one per CPU core, and the project’s feature list puts 10,000 idle keep-alive connections at about 2.5 MB of memory.

The configuration is a text file of directives. A simple directive is a name and its parameters, ending in a semicolon. A block directive wraps other directives in braces, and a block that can hold directives is called a context. The events and http contexts sit at the top level. Inside http, each server block describes one virtual server, picked by the address and port it listens on and then by the host name the client asked for. Inside a server block, each location block says what to do with a family of request paths.

Use the concept in a decision

Put “What is nginx?” to work in a real decision. We can help relate the concept to the system you are building or buying. Day-to-day care of servers, mail, accounts and updates, so the things everybody relies on stay boring.

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. Ran 10 PostgreSQL and MS SQL servers on Ubuntu — 10 database servers deployed and kept healthy

Ran 10 PostgreSQL and MS SQL servers on Ubuntu Get in touch →

Why it matters

The web server is the one program every visitor touches, so its behaviour under load and its defaults reach everyone. Two properties make the case for nginx.

The first is how cheaply it carries slow connections. Response buffering from a proxied server is on by default: nginx takes the backend’s answer as fast as the backend can produce it, spilling to a temporary file when it outgrows memory, and then feeds it to the client at whatever pace the client manages. The application server is free for its next request while a phone on a weak signal is still downloading the last one.

The second is that configuration changes go out without dropping requests. Configuration edits take effect only on a reload or a restart. On a reload the master checks the new configuration and tries to apply it. When that works, new workers start with it, while the old ones stop accepting connections, finish the requests they hold and exit. When it fails, the master rolls back and keeps serving with the configuration it already had. The project also lists upgrading the binary itself without interrupting clients.

The price is paid in the configuration language, whose rules on precedence and inheritance are where most surprises live. For a request path, nginx first finds the longest matching prefix location and remembers it, then tries the regular-expression locations in the order they appear in the file and takes the first that matches. The remembered prefix is used when the regular expressions all fail. A location marked `=` matches the path exactly and ends the search at once, and `^~` on the longest prefix tells nginx to skip the regular expressions. Some directives, add_header and proxy_set_header among them, are inherited from the enclosing level only if the current level defines none of its own.

Certificates long came from an ACME client running beside nginx. The project now also offers ngx_http_acme_module, shipped separately from the core build.

What it looks like in practice

Two pieces of hosting work from the founder's earlier roles involve nginx, and in both it shared the servers with Apache rather than replacing it.

The first was forty client websites that had to be quick and always reachable, which ran on Ubuntu Linux servers where Apache and nginx were both in use. The job was writing their configuration, tuning them for speed and looking after them while real visitors were on them, and all forty stayed highly available. Two web servers on one fleet means every rule in two dialects: an Apache virtual host is an nginx server block, and a fix found for one must be translated for the other.

The second ran from 2015 to 2022. An ACME workflow took over certificates for a hundred Docker applications, issuing each one, renewing it before expiry and installing it on the Ubuntu hosts with no person in the loop, and Apache and nginx were again both in use on those hosts. On the nginx side, a renewed file on disk is half of such a change. A certificate named by a plain path is read when nginx loads its configuration, and the documentation notes that only a file name built from variables makes it load the certificate again on each TLS handshake. A renewal that writes new files therefore reaches visitors only after a reload, so any automated renewal for nginx has to end with one.

Where it goes wrong

A header set for the whole site vanishes on one path. A location that has any add_header of its own inherits none from the server or http level, so one cache header added to a single location silently drops every security header above it. proxy_set_header follows the same rule. add_header also applies only to success and redirect codes such as 200, 301 and 304 unless given `always`, so error pages go out bare. Keep shared headers in one file pulled in with include at every level that sets its own, or from 1.29.3 set `add_header_inherit merge;`, add `always` where error responses need them, and check each path with `curl -sI`.

Requests for one site are answered by another. nginx matches the address and port first, then the Host header against server_name. When the name is unknown to every server block on that port, the request goes to the default server, which is the first one listed unless another is marked default_server. A

typo in a new site's server_name quietly serves an old neighbour's pages. Declare an explicit default server that answers unknown names with `return 444`, which closes the connection, so a mistake shows up as an error rather than as the wrong site.

Uploads fail with 413 Request Entity Too Large. client_max_body_size defaults to 1 MB, and nginx rejects a larger request body before the application sees it. Raise the limit in the location that accepts uploads rather than for the whole server, and keep it in step with the application's own limit.

The backend builds links to the wrong host or receives the wrong path. Unless told otherwise, nginx sends the proxied server a Host header made from the address in proxy_pass, such as `127.0.0.1:8080`, so the application generates redirects and absolute links to that address. Set `proxy_set_header Host $host;` in the proxying location. Paths change with the form of proxy_pass: written with a URI, even a single `/`, it replaces the part of the path that matched the location, while written without one it passes the path through unchanged, so test a nested path after any change.

Clients still get the old configuration or the old certificate. An edited file, or a certificate renewed overnight, has no effect on a running nginx until a reload, and a reload of a broken configuration leaves the old one in place. Test before every reload, then read the error log and check what clients actually receive:

```
nginx -t && nginx -s reload
openssl s_client -connect example.com:443 \
  -servername example.com </dev/null |
  openssl x509 -noout -enddate
```

Further reading

- nginx documentation — the project's index of guides and per-module directive references.
- Beginner's Guide — master and worker processes, reloading, and the configuration's structure.
- How nginx processes a request — how a server block is chosen by address, port and Host header.
- Module ngx_http_core_module — location matching order, server_name and client_max_body_size.

- Module ngx_http_proxy_module — proxy_pass, header defaults, timeouts and response buffering.
- Module ngx_http_headers_module — add_header, its response codes and its inheritance rule.

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 nginx?” 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/nginx/