nginx Web server and proxy

START HERE

`nginx -t`
Test the config

`nginx -s reload`
Apply config, no downtime

`server_name <host>;`
Which host this server answers

`location <path> { }`
Rules for a URL path

`root <dir>;`
Serve files from a folder

`proxy_pass <url>;`
Forward to a backend

`listen 443 ssl;`
Accept HTTPS

`tail -f /var/log/nginx/error.log`
Watch for errors

CONCEPTS

Always run nginx -t before reload: a broken config is rejected and the old one keeps running.

main user, workers, pid
http shared web settings
server one virtual host
location one URL path

Directives are inherited inward; the innermost value wins

listen ip:port
↓ Host header
server_name pick server
↓ URI
location pick best match
↓ handler
root / proxy_pass file or backend

How a request is routed

A directive ends with a semicolon; a block uses braces. A missing semicolon is the classic syntax error.

PROCESS CONTROL

`nginx -t`
Check config syntax

`nginx -T`
Test and dump the full config

`nginx -s reload`
Reload config gracefully

`nginx -s quit`
Graceful shutdown

`nginx -s stop`
Fast shutdown

`nginx -s reopen`
Reopen log files (after rotation)

`nginx -c <file>`
Start with another config file

`nginx -V`
Show version and build options

`nginx -v`
Show version only

`kill -s HUP <pid>`
Reload by signal (pid in nginx.pid)

`worker_processes auto;`
One worker per CPU core

`worker_connections 1024;`
Connections per worker (events block)

Under systemd prefer systemctl reload nginx; the -s forms talk to the master process directly.

SERVER BLOCKS

`listen 80;`
Port to accept connections on

SERVER BLOCKS (cont.)

`listen 443 ssl;`
Accept TLS on this port

`listen 80 default_server;`
Catch requests matching no name

`server_name example.com www.example.com;`
Hosts this server answers

`server_name *.example.com;`
Wildcard host match

`root <dir>;`
Document root; URI is appended

`index index.html index.htm;`
File served for a directory

`include <file>;`
Pull in more config files, globs allowed

Typically include conf.d/.conf; inside http.*

`error_page 404 /404.html;`
Custom page for an error code

`gzip on;`
Compress responses

Add gzip_types for anything beyond text/html.

`error_page 500 502 503 504 /50x.html;`
One page for several codes

`client_max_body_size 10m;`
Max upload size (default 1m, else 413)

LOCATIONS

`location / { }`
Prefix match: the catch-all

`location = /path { }`
Exact match, stops the search

`location ^~ /static/ { }`
Prefix that skips regex checks

`location ~ \.php$ { }`
Case-sensitive regex

`location ~* \.(jpg|png)$ { }`
Case-insensitive regex

`alias <dir>/;`
Replace the location prefix with a path

With a prefix location, keep trailing slashes consistent on both sides; use root when the location matches the end of the path.

`try_files $uri $uri/ =404;`
Try file, directory, then 404

`try_files $uri /index.html;`
SPA fallback to the index page

`return 301 <url>;`
Redirect permanently

LOCATIONS (cont.)

`rewrite <regex> <url> permanent;`
Regex-based redirect

Location matching order

Step	Rule
1	Exact = match ends the search
2	Longest prefix is remembered; ^~ stops here
3	Regex ~ and ~* in file order, first wins
4	No regex hit: first matching predicate $var location (1.31.5+)
5	Nothing else matched: use the remembered prefix

REVERSE PROXY

`proxy_pass http://127.0.0.1:8080;`
Forward requests to a backend

Without a URI part the request path is passed unchanged; with one (even just a trailing /) the matched location prefix is replaced.

`proxy_pass http://<upstream>;`
Forward to a named upstream group

`proxy_set_header Host $host;`
Pass the original host to the backend

The default sends the backend address as Host. A proxy_set_header in a location replaces all inherited ones.

`proxy_set_header X-Real-IP $remote_addr;`
Tell the backend the client IP

`proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;`
Append client IP to the chain

`proxy_set_header X-Forwarded-Proto $scheme;`
Pass http or https

`proxy_http_version 1.1;`
Default since 1.29.7; set it on older builds

`proxy_read_timeout 60s;`
Wait between reads from the backend

`upstream <upstream> { server <host:port>; }`
Define a backend pool

Keep-alive to backends is on by default since 1.29.7.

`fastcgi_pass <host:port>;`
Hand off to a FastCGI backend

TLS

`ssl_certificate <file>;`
Certificate (leaf first, then chain)

`ssl_certificate_key <file>;`
Private key, PEM

TLS (cont.)

`ssl_protocols TLSv1.2 TLSv1.3;`
Allowed TLS versions (the default)

`ssl_session_cache shared:SSL:10m;`
Share TLS sessions across workers

`http2 on;`
Enable HTTP/2 in the server (1.25.1+)

`return 301 https://$host$request_uri;`
Redirect HTTP to HTTPS

ssl on; is gone since 1.25.1: enable TLS with the ssl flag on listen.

LOGS AND VARIABLES

`access_log /var/log/nginx/access.log;`
Request log location

`error_log /var/log/nginx/error.log warn;`
Error log and minimum level

`$host`
Request host name

`$request_uri`
Original URI with query string

`$uri`
Current normalized URI, no query

`$remote_addr`
Client IP address

RECIPES

Static site with HTTPS

```
server {
  listen 443 ssl;
  http2 on;
  server_name example.com;
  ssl_certificate     /etc/nginx/certs/example.com.crt;
  ssl_certificate_key /etc/nginx/certs/example.com.key;
  root /var/www/example;
  location / { try_files $uri $uri/ =404; }
}
```

Redirect HTTP to HTTPS

```
server {
  listen 80;
  server_name example.com;
  return 301 https://$host$request_uri;
}
```

Reverse proxy an app

```
location / {
  proxy_pass http://127.0.0.1:3000;
  proxy_set_header Host $host;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-Proto $scheme;
}
```