MASTERING NGINX

THE DEFINITIVE GUIDE TO ARCHITECTURE, CONFIGURATION, AND PRODUCTION OPERATIONS

- CORE ARCHITECTURE & CONFIGURATION
- LOAD BALANCING & UPSTREAMS
- CACHING & PERFORMANCE
- TLS TERMINATION, SECURITY & HARDENING
- HTTP/2 & HTTP/3 PERFORMANCE
- KUBERNETES INTEGRATION
- PRODUCTION OPERATIONS & OBSERVABILITY

JAMES HUBBARD

DEEP KNOWLEDGE. PROVEN CONFIGURATIONS. PRODUCTION RESULTS.

Mastering NGINX

The Definitive Guide to Architecture, Configuration, and Production Operations

Steve Publications

This book is available at https://leanpub.com/masteringnginx

This version was published on 2026-07-27

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

The Definitive Guide to Architecture, Configuration, and Production Operations

This book is a comprehensive technical reference for intermediate Linux administrators through advanced DevOps and platform engineers who need deep operational knowledge of NGINX. It covers the complete ecosystem from core architecture and configuration fundamentals through load balancing, caching, TLS termination, HTTP/2 and HTTP/3, Kubernetes integration, performance tuning, security hardening, and production operations. Every concept is explained with technical rigor: not just how features work but why they exist, when to use them, their tradeoffs, performance implications, and common pitfalls. Production-ready configuration examples are provided throughout, annotated line by line, so you can understand the reasoning behind each directive.

Introduction: The NGINX Story and Architecture Promise

The Igor Sysoev Problem: Why NGINX Was Created

In 2001, Igor Sysoev, a software engineer at the Russian web portal Rambler, faced a problem that would come to define an era of web infrastructure. Rambler was growing rapidly, and the Apache HTTP Server powering the site was struggling. Not because Apache was poorly designed, but because its fundamental architecture made scaling past a certain threshold prohibitively expensive.

The issue was connection handling. Apache's default MPM (Multi-Processing Module) at the time used a process-per-connection model: each incoming client connection spawned a new operating system process, or in threaded mode, a new thread. Each process consumed significant memory for its stack and data structures. At 10,000 concurrent connections, the server needed thousands of processes, each consuming megabytes of RAM, and the CPU spent more time context-switching between them than actually processing requests.

This was the C10K problem: how to handle 10,000 concurrent connections on a single server without drowning in resource consumption. The name became shorthand for a broader challenge in high-performance networking that affected not just web servers but any system dealing with massive concurrency.

Sysoev's insight was architectural rather than incremental. Instead of optimizing the process-per-connection model, he eliminated it entirely. Starting development in 2002, he designed a server from the ground up around asynchronous, event-driven I/O. The result was NGINX, released as open-source software in October 2004.

By September 2008, NGINX at Rambler was serving 500 million requests per day, handling traffic that would have required dozens of Apache servers with a fraction of the hardware [1,3,9]. The server proved its concept, and the open-source release allowed the broader community to adopt it. What began

as an internal solution to a specific scaling problem became one of the most widely deployed pieces of infrastructure software in the world.

Event-Driven Architecture: The Core Innovation

Understanding NGINX's architecture is not optional background knowledge; it is essential for writing effective configurations and troubleshooting production issues. Every directive you write ultimately affects how the event loop operates, how connections are managed, and how workers interact with the operating system.

At its core, NGINX uses a master-worker process model. When you start the nginx service, a single master process launches first. This master process runs as root (or another privileged user) to bind to low-numbered ports like 80 and 443, reads and validates the configuration file, and then spawns a configured number of worker processes. After spawning workers, the master process steps back and becomes a manager: it monitors worker health, reloads configuration when requested, and handles graceful shutdown. It does not serve any client requests directly.

The actual work happens in the worker processes. Each worker is single-threaded by design, running an event loop that handles all its assigned connections asynchronously. This single-threaded design is deliberate and provides significant advantages: workers do not require locks or synchronization primitives when accessing shared data structures, eliminating race conditions and the performance overhead of mutex contention. The tradeoff is that a blocking operation in one worker affects only that worker's connections, not the entire server, but since NGINX uses non-blocking I/O throughout, blocking operations are rare.

Each worker runs an event loop built on platform-specific mechanisms for efficient I/O readiness notification:

- On Linux, NGINX uses epoll, which allows the kernel to efficiently notify the process when any of thousands of monitored file descriptors become ready for reading or writing.
- On FreeBSD, macOS, and other BSD variants, it uses kqueue, providing similar functionality.
- On Solaris, event ports serve the same purpose.

- Older systems fall back to select or poll, which are less efficient but universally available.

The event loop operates in a tight cycle: call epoll_wait() (or equivalent) to collect all ready events, then process each event by advancing its associated connection through its state machine. A connection moves through states sequentially: accepting, reading request headers, processing the request, sending response headers, sending the response body, keeping alive or closing. Each state transition is driven by I/O readiness events rather than blocking calls. When a worker needs to read data from a socket but none is available, instead of blocking and waiting, it registers interest with epoll and moves on to process other connections. When data arrives, epoll notifies the worker, which then reads the data and advances the connection's state.

This design means a single worker can handle thousands of simultaneous connections using minimal memory. Each connection requires only a small amount of kernel socket buffer space and a user-space data structure tracking its state, typically just a few kilobytes compared to the megabytes required by a dedicated process or thread. With four workers each handling 4,096 connections at worker_connections 4096, a modest server can manage over 16,000 concurrent connections with perhaps half a gigabyte of RAM total.

The architecture also explains why NGINX excels at certain workloads and has limitations in others. It is exceptional for I/O-bound tasks where most connections are idle or waiting on upstream servers, because the event loop never blocks. It is less ideal for CPU-intensive request processing within NGINX itself, because a single-threaded worker cannot parallelize computation across cores, though this limitation is mitigated by having multiple workers, each bound to different CPU cores via worker_cpu_affinity.

What NGINX Is Today: Beyond the Web Server

The NGINX that exists today is substantially broader than the web server Sysoev originally built, though static content serving remains one of its strongest capabilities. Understanding the full scope of what NGINX can do helps in recognizing when it should be part of your architecture and when other tools may be more appropriate.

As a web server, NGINX serves static files with exceptional efficiency using sendfile, direct I/O, and asynchronous operations that minimize CPU usage

and context switches. It handles virtual hosts, URL routing via location blocks, compression with gzip and Brotli, TLS termination, HTTP/2 multiplexing, and HTTP/3 over QUIC. For many use cases, it can serve static content directly without needing a backend application server at all.

As a reverse proxy, NGINX sits in front of application servers and forwards client requests to them while presenting a unified interface to clients. This is arguably its most common production role today. The reverse proxy handles TLS termination so backends do not need to manage certificates, compresses responses, caches content, performs load balancing across multiple backend instances, and provides security controls like rate limiting, IP-based access control, and request filtering.

As a load balancer, NGINX distributes traffic across pools of backend servers using configurable algorithms: weighted round-robin for basic distribution, least connections for uneven workloads, IP hash for session persistence, and custom hash functions for more sophisticated routing. NGINX Plus adds active health checking, least-time balancing based on actual response metrics, random load balancing, and generic consistent hashing.

As an HTTP cache, NGINX can cache responses from upstream servers in memory and on disk, reducing backend load and improving response times. Its proxy cache supports configurable cache zones, granular TTL control per status code and content type, stale serving when backends are unavailable, and cache key customization for precise control over what gets cached.

As an API gateway, NGINX routes requests to different microservices based on URL paths, headers, or other criteria. It handles authentication via JWT validation, OAuth integration patterns, rate limiting per client or endpoint, request transformation, and response manipulation. While not a full-featured API management platform like Kong or Apigee, it provides the core gateway capabilities needed by many organizations.

Beyond HTTP, NGINX supports:

- TCP/UDP load balancing via the stream module for databases, game servers, custom protocols, and DNS forwarding
- Mail proxying via the mail module for SMTP, IMAP, and POP3
- Media streaming for HLS (HTTP Live Streaming) and DASH (Dynamic Adaptive Streaming over HTTP)
- WebSocket proxying for real-time bidirectional communication

- gRPC proxying for modern microservice communication

This breadth of capability comes from NGINX's modular architecture. Core functionality is built into the binary, but additional features can be added via modules compiled statically at build time or loaded dynamically. This modularity allows NGINX to remain lightweight when only basic functionality is needed while supporting complex deployments when required.

Open Source vs NGINX Plus: The Edition Landscape

NGINX exists in two primary editions: the open-source version available under a BSD-like license, and NGINX Plus, a commercial distribution with additional features, vendor support, and longer-term stability guarantees. Both share the same core codebase and event-driven architecture; the differences lie in additional features, support options, and release models.

The open-source NGINX is maintained by F5 Networks following their acquisition of NGINX Inc. It receives regular updates through two branches:

- The stable branch, which receives bug fixes and security patches but no new features during its lifecycle
- The mainline branch, which includes the latest features and improvements in addition to fixes

Most production deployments use the stable branch for predictable behavior, though the mainline branch is often stable enough for production use and provides access to newer features earlier.

NGINX Plus adds several capabilities that are significant for enterprise environments:

- Active health checks that periodically probe backend servers rather than relying solely on passive failure detection
- Dynamic reconfiguration via HTTP API, allowing upstream server changes without reloading the entire configuration
- Advanced load balancing algorithms including least_time (selecting based on actual response times), random distribution, and generic consistent hashing

- Sticky sessions using cookies for session persistence
- Built-in JWT authentication module
- Cache purge API for removing specific entries from the cache
- Enhanced monitoring via comprehensive HTTP API exposing real-time metrics
- Prometheus integration through the NJS module
- Commercial support with guaranteed response times

For many organizations, the open-source version is sufficient, especially when combined with external tools for health checking, configuration management, and monitoring. The decision to use NGINX Plus typically comes down to operational requirements: the need for zero-downtime configuration changes, the value of active health checks in reducing failure detection time, or the requirement for vendor support contracts.

Throughout this book, features unique to NGINX Plus will be clearly labeled as [NGINX Plus only] so you can distinguish between capabilities available in both editions and those requiring a commercial license.

How to Use This Book

This book is organized to take you from foundational understanding through advanced production operations. The chapters build on each other logically, but each chapter is also designed to be useful as a standalone reference once you understand the basics.

The book assumes familiarity with Linux system administration, basic networking concepts (TCP/IP, HTTP protocol), and comfort reading configuration files. It does not assume prior NGINX experience, though readers with existing knowledge will find the depth of coverage useful for filling gaps and understanding the reasoning behind best practices.

Configuration examples are written to be production-ready, not minimal demonstrations. They include appropriate security settings, logging configuration, error handling, and comments explaining non-obvious choices. Where a simpler example would obscure important details, the more complete version is used with explanations for each component.

Cross-references between chapters use the format "(see Chapter N)" to point you to related material without requiring you to re-read content you may

already know. The goal is to provide both a learning path for someone new to NGINX and a reference structure for someone looking up specific topics.

The book covers both conceptual understanding and practical implementation. You will learn why NGINX's architecture works the way it does, how each feature operates internally, what tradeoffs are involved in different configuration choices, and exactly how to configure each feature for production use. The combination of theory and practice is intentional: understanding the underlying mechanisms makes you better at troubleshooting unexpected behavior and making informed decisions when the documentation alone does not provide clear guidance.

Chapter 1: Installation and Environment Setup

Packaging Ecosystems: Official Repositories vs Package Managers

Installing NGINX correctly is the foundation of everything that follows. A poor installation choice can lead to version skew, missing modules, delayed security updates, or configuration incompatibilities that surface only under production load. This chapter covers all installation methods so you can choose the approach that matches your operational requirements.

The first decision is between using NGINX's official repositories and your distribution's packaged version. Both approaches have tradeoffs.

Distribution-packaged NGINX is convenient: it integrates with your system's package manager, receives updates through the normal update cycle, and is often security-reviewed by the distribution maintainers. On Ubuntu and Debian, running apt install nginx installs a working server immediately. On RHEL, CentOS, Rocky Linux, and AlmaLinux, dnf install nginx or yum install nginx does the same. The configuration follows FHS (Filesystem Hierarchy Standard) conventions, systemd service files are provided, and the package is generally well-integrated with the rest of the system.

The problem with distribution packages is version lag. Debian and Ubuntu tend to ship older NGINX versions that may be months or years behind the current stable release. Security patches are backported, but new features and performance improvements arrive slowly. For example, HTTP/3 support (available since NGINX 1.25.0) may not appear in a distribution package until the next major release cycle. If you need specific modules that the distribution has not enabled, you are stuck unless you build from source or switch repositories.

NGINX's official repositories provide more current versions with a choice between stable and mainline branches. The stable branch receives only bug fixes and security patches during its lifecycle, making it suitable for production

environments where stability is paramount. The mainline branch includes the latest features and improvements and is generally stable enough for production use, though some organizations prefer the additional caution of the stable branch.

Setting up the official NGINX repository on Debian and Ubuntu involves importing the signing key and adding the repository:

```
# Import NGINX signing key
curl -fsSL https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
  | sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg > /dev/null

# Add the stable repository (replace codename with your version: jammy, noble,
↪  bookworm, etc.)
echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] \
  https://nginx.org/packages/debian `lsb_release -cs` nginx" \
  | sudo tee /etc/apt/sources.list.d/nginx.list

# Update and install
sudo apt update
sudo apt install nginx
```

For the mainline branch, replace "debian" with "mainline" in the repository URL. The same pattern applies to Ubuntu. On RHEL-based systems, you create a yum/dnf repository file at /etc/yum.repos.d/nginx.repo:

```
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

Replace "centos" with your distribution name (rhel, rocky, alma) and use nginx-mainline instead of nginx-stable for the mainline branch. Then run dnf install nginx.

A critical consideration when using official repositories is module availability. The official NGINX packages include the core modules and commonly used optional modules like HTTP/2, SSL, and real IP, but they may not include every third-party or less common module. If you need a specific module not included in the package, you have three options: use a third-party repository

that provides additional modules (such as the GetPageSpeed APT repository for Debian/Ubuntu, which offers over 100 dynamic modules), build NGINX from source with the required modules, or switch to NGINX Plus if the module is included there.

Building from Source: Control and Customization

Building NGINX from source is necessary when you need modules not available in packages, want to use a specific version of OpenSSL, require performance optimizations for your hardware, or need to minimize the binary by excluding unused modules. It is also useful for understanding what goes into an NGINX build and for creating reproducible builds in containerized environments.

The build process uses a configure script that checks for dependencies, validates options, and generates a Makefile. The full list of options is available via ./configure –help, but the most important ones fall into several categories:

Installation paths control where NGINX and its files are placed:

- –prefix sets the base installation directory (default: /usr/local/nginx)
- –sbin-path specifies the nginx binary location
- –conf-path sets the configuration file path
- –error-log-path, –http-log-path set default log locations
- –pid-path and –lock-path specify where these files are stored

Module selection determines which features are compiled in:

- –with-http_ssl_module enables TLS/SSL support (essential for HTTPS)
- –with-http_v2_module enables HTTP/2 support
- –with-http_v3_module enables HTTP/3 and QUIC support (requires compatible TLS library)
- –with-http_realip_module allows extracting the real client IP from proxy headers
- –with-http_gzip_static_module serves pre-compressed .gz files
- –with-http_sub_module enables response body substitution
- –with-mail and –with-mail_ssl_module enable mail proxy functionality
- –with-stream and –with-stream_ssl_module enable TCP/UDP load balancing

Third-party modules are added with:

- –add-module=/path/to/module for static compilation (module is built into the binary)
- –add-dynamic-module=/path/to/module for dynamic loading (module loaded at runtime via load_module directive, requires –with-compat)

Dependencies must be installed before running configure. The essentials include a C compiler (gcc or clang), make, and development headers for libraries NGINX uses:

- OpenSSL development files for SSL/TLS support
- PCRE or Oniguruma development files for regular expression processing
- zlib development files for gzip compression

A typical production build command looks like this:

```
./configure \
  --prefix=/etc/nginx \
  --sbin-path=/usr/sbin/nginx \
  --modules-path=/usr/lib64/nginx/modules \
  --conf-path=/etc/nginx/nginx.conf \
  --error-log-path=/var/log/nginx/error.log \
  --http-log-path=/var/log/nginx/access.log \
  --pid-path=/run/nginx.pid \
  --lock-path=/run/lock/nginx.lock \
  --user=nginx \
  --group=nginx \
  --with-http_ssl_module \
  --with-http_v2_module \
  --with-http_realip_module \
  --with-http_addition_module \
  --with-http_sub_module \
  --with-http_dav_module \
  --with-http_flv_module \
  --with-http_gunzip_module \
  --with-http_gzip_static_module \
  --with-http_random_index_module \
  --with-http_secure_link_module \
  --with-http_stub_status_module \
  --with-http_auth_request_module \
  --with-http_slice_module \
  --with-threads \
  --with-stream \
```

```
  --with-stream_ssl_module \
  --with-stream_realip_module \
  --with-http_v3_module \
  --with-pcre-jit \
  --with-file-aio \
  --with-compat

make -j$(nproc)
sudo make install
```

The –with-pcre-jit option enables just-in-time compilation for regular expressions, which can significantly improve performance when using regex-heavy location blocks or rewrite rules. The –with-file-aio option enables asynchronous I/O for file operations on Linux, improving static file serving performance. The –with-compat option is required if you plan to use dynamically loaded modules.

After building, verify the compilation with nginx -V, which displays the version and all configure arguments used. This output is valuable for documentation and troubleshooting, as it tells you exactly what capabilities your NGINX binary has.

Building from source requires maintenance: you are responsible for tracking security updates and rebuilding when new versions are released. For this reason, many organizations use the official repositories for most deployments and reserve source builds for specialized cases where specific modules or configurations are required.

Containerized NGINX: Official Images and Variants

Containerization has become the dominant deployment model for NGINX in modern infrastructure, and the official NGINX Docker images provide several variants optimized for different use cases. Understanding the differences between these images helps you choose the right one and avoid common pitfalls.

The official images are published at nginx on Docker Hub and come in four primary families:

- mainline: Based on the mainline NGINX branch with the latest features
- stable: Based on the stable NGINX branch for production stability

- alpine variants (mainline-alpine, stable-alpine): Based on Alpine Linux for minimal image size
- debian variants (the default mainline and stable tags): Based on Debian for broader compatibility

Image sizes vary significantly. A typical nginx:stable-alpine-slim image is approximately 12 MB, while nginx:stable based on Debian is around 190 MB. The size difference comes from Alpine's use of musl libc instead of glibc and its minimal package set. Smaller images reduce pull times, storage costs, and attack surface, but Alpine can introduce compatibility issues with some NGINX modules that expect glibc behavior.

The -perl tag variants include Perl support for the ngx_http_perl_module, which allows embedding Perl code directly in NGINX configuration. This is rarely needed and adds significant image size, so it is used only when specifically required.

A production Dockerfile using the official NGINX image typically follows this pattern:

```
FROM nginx:stable-alpine-slim AS runner

# Remove default config
RUN rm /etc/nginx/conf.d/default.conf

# Copy custom configuration
COPY nginx.conf /etc/nginx/nginx.conf
COPY conf.d/ /etc/nginx/conf.d/

# Copy static content if serving files
COPY html/ /usr/share/nginx/html/

# Create non-root user and set permissions
RUN chown -R nginx:nginx /var/cache/nginx /var/log/nginx /usr/share/nginx/html

# Expose ports
EXPOSE 80 443

# Health check
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD wget -qO- http://localhost:80/health || exit 1

# Run as non-root
USER nginx
CMD ["nginx", "-g", "daemon off;"]
```

Key considerations for containerized NGINX:

Running as non-root is a security best practice. The official images include the nginx user, and you should use USER nginx before the CMD directive. The master process still needs to bind to privileged ports (below 1024), which can be handled by using port mapping in the container runtime (mapping host port 80 to container port 8080) or by granting the CAP_NET_BIND_SERVICE capability.

Configuration should be mounted as volumes rather than baked into the image whenever you need to change configuration without rebuilding. In Kubernetes, ConfigMaps are the standard mechanism for this. However, baking configuration into the image is appropriate when the configuration is immutable and part of the application definition.

Multi-stage builds can reduce final image size by building NGINX with custom modules in one stage and copying only the binary and configuration to a minimal runtime image:

```
FROM nginx:stable-alpine AS builder

# Install build dependencies and compile custom modules
RUN apk add --no-cache gcc musl-dev pcre-dev zlib-dev openssl-dev linux-headers
# ... install and configure custom modules ...
RUN ./configure --add-module=/path/to/module && make && make install

FROM alpine:latest AS runner

# Copy NGINX binary and configuration from builder
COPY --from=builder /usr/local/nginx /usr/local/nginx
COPY nginx.conf /usr/local/nginx/conf/nginx.conf

EXPOSE 80
CMD ["/usr/local/nginx/sbin/nginx", "-g", "daemon off;"]
```

Container orchestration platforms like Kubernetes add additional considerations. The NGINX Ingress Controller is a specialized deployment of NGINX that watches Kubernetes Ingress resources and automatically generates configuration, which is covered in detail in Chapter 18. For standalone container deployments, the patterns above apply directly.

Directory Structure and Binaries

After installation, understanding where NGINX places its files is essential for effective administration. The directory structure varies slightly between installation methods and distributions, but the conventions are consistent enough that you can navigate any NGINX installation with a mental map of the standard layout.

The primary configuration directory is /etc/nginx on most systems when installed via packages or official repositories. This directory contains:

- nginx.conf: The main configuration file, read first at startup
- conf.d/: Directory for additional configuration files included by the main config (commonly used for per-site configurations)
- sites-available/: On Debian/Ubuntu, contains complete site configurations that are not active until linked
- sites-enabled/: On Debian/Ubuntu, contains symbolic links to sites-available entries that should be loaded
- snippets/: Reusable configuration fragments that can be included in multiple places
- modules-enabled/: Dynamic module loading configuration files

Log files default to /var/log/nginx/:

- access.log: Records every HTTP request with details about client, request, response, and timing
- error.log: Records errors, warnings, and debug information depending on configured log level
- Individual site logs may be configured in separate files within this directory

Static content defaults vary:

- /usr/share/nginx/html/: Default document root on most package installations
- /var/www/html/: Sometimes used as the default, particularly on RHEL-based systems

- These are configurable via the root directive and should not be assumed fixed

Runtime files:

- The PID file is typically at /run/nginx.pid or /var/run/nginx.pid, containing the master process ID
- The lock file is at /var/lock/nginx.lock or similar, used to prevent multiple master processes
- Temporary files for large request bodies and proxy responses are stored in directories specified by configuration (often under /var/cache/nginx/)

The nginx binary itself is at /usr/sbin/nginx on package installations or at the path specified during source builds. Running nginx without arguments starts the server using the default configuration. Important command-line options include:

- nginx -t: Test configuration syntax without starting the server
- nginx -T: Test configuration and dump the full effective configuration (including includes)
- nginx -s signal: Send a signal to the running master process (stop, quit, reload, reopen)
- nginx -V: Display version and compile-time options
- nginx -v: Display just the version number

Understanding these paths is practical knowledge used constantly in administration. When troubleshooting, you check logs in /var/log/nginx/. When adding a new site, you create configuration in /etc/nginx/conf.d/ or /etc/nginx/sites-available/. When validating changes, you run nginx -t. These operations are so routine that knowing the layout becomes second nature, but explicit understanding prevents confusion when working on unfamiliar systems.

Default Configuration Anatomy

The default NGINX configuration file provides a template that demonstrates the structure and hierarchy of directives while providing reasonable defaults

for basic web serving. Analyzing it carefully reveals how NGINX organizes configuration and what each section controls.

A typical default nginx.conf from the official packages looks like this:

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
error_log /var/log/nginx/error.log;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
}

http {
    ##
    # Basic Settings
    ##
    sendfile on;
    tcp_nopush on;
    types_hash_max_size 2048;
    server_tokens off;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ##
    # Logging Settings
    ##
    access_log /var/log/nginx/access.log;

    ##
    # Gzip Settings
    ##
    gzip on;

    ##
    # Virtual Host Configs
    ##
    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

Let us examine each section:

The user directive at the top specifies which Unix user and group the worker processes run as. The master process starts as root to bind privileged ports,

then drops privileges to this user before spawning workers. Using www-data (Debian/Ubuntu) or nginx (RHEL-based) is standard; never run workers as root in production.

worker_processes auto tells NGINX to automatically detect the number of available CPU cores and spawn one worker per core. This is generally the right default for most workloads, though specific high-connection scenarios may benefit from manual tuning (see Chapter 3). The auto value was introduced relatively recently; older configurations often use a hardcoded number.

pid specifies where the master process ID is written. This file is used by nginx -s commands to send signals to the running server and by init systems to track the process.

error_log sets the default error log location and level. Without a level specified, it defaults to warn, which captures warnings and errors but not informational messages. For production, warn or error is typical; debug is useful for troubleshooting but generates substantial output.

The include directive loads additional configuration files. The modules-enabled include loads dynamic module configurations if any are installed. Later includes in the http block load site-specific configurations from conf.d and sites-enabled directories. This modular approach allows organizing large deployments without a monolithic configuration file.

The events block controls connection processing settings:

- worker_connections sets the maximum number of simultaneous connections each worker can handle. With auto workers on a 4-core machine and 768 connections per worker, the server can theoretically handle over 3,000 concurrent connections. The Debian default of 768 is conservative; production systems often use higher values (1024, 2048, or more) depending on expected load and available file descriptors.

The http block contains all HTTP-related configuration and is where most of your work will happen:

- sendfile on enables the sendfile system call for efficient static file serving by transferring data directly from disk to network socket without copying through user space
- tcp_nopush on optimizes TCP packet sending by ensuring complete packets are sent together rather than in fragments, reducing overhead

- types_hash_max_size controls the size of the hash table mapping file extensions to MIME types; 2048 is appropriate for most installations
- server_tokens off hides the NGINX version number from response headers and error pages, a basic security hardening measure
- include /etc/nginx/mime.types loads the MIME type mappings
- default_type application/octet-stream sets the fallback content type when the file extension is not recognized
- gzip on enables response compression (though default settings may need tuning for production)

The final includes load site configurations. The conf.d pattern is common across distributions, while sites-available/sites-enabled is a Debian/Ubuntu convention that provides a clear separation between configured sites and active sites. On RHEL-based systems, you typically place all site configurations directly in conf.d.

Understanding this default structure is important because every production configuration you write will be an extension and modification of this template. The hierarchy established here (main context settings -> events block -> http block -> server blocks -> location blocks) defines how directives inherit and override each other, a concept explored in detail in Chapter 2.

Chapter 2: Configuration Syntax, Directives, and Contexts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Configuration Grammar: Blocks, Directives, and Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Directive Contexts: Where Directives Belong

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Configuration Inheritance and Precedence

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Variables: Built-in, Custom, and Computed

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Include Mechanism and Configuration Organization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Configuration Testing and Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 3: Worker Processes and Connection Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Master-Worker Process Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Worker Process Count and CPU Affinity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Event Loop: Epoll, Kqueue, and Event Ports

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Connection Lifecycle States

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Diagram Description: NGINX Connection State Machine

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Keepalive Connections and Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

File Descriptor Limits and System Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 4: Web Serving and Virtual Hosts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Server Blocks and Virtual Hosts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Location Directive: NGINX's Routing Engine

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Serving Static Content Efficiently

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Error Pages and Response Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Request Limits and Resource Protection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Compression: gzip, Brotli, and zstd

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 5: Reverse Proxying and Upstream Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Reverse Proxy Model: Why and How

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Upstream Groups: Structure and Directives

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Load Balancing Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Health Checks: Passive and Active

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Proxy Buffering and Connection Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Timeouts and Retries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 6: Load Balancing Patterns and Advanced Traffic Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Production Load Balancing Topologies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Traffic Splitting and Canary Deployments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Session Persistence Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Geographic and Data Center Routing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Rate Limiting and Request Throttling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Connection Limits and Resource Protection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Media Streaming with NGINX

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Case Study: Scaling NGINX for Black Friday Traffic

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 7: Caching Architecture and Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Caching Fundamentals: When and Why to Cache

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Proxy Cache Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Cache Keys and Granularity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Cache Bypass, Stale Serving, and Upstream Interaction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Cache Purging and Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Memory vs Disk Caching Tradeoffs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Diagram Description: NGINX Cache Lookup Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 8: TLS, SSL, and Transport Security

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

TLS Termination: Architecture and Rationale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Certificate Configuration and Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Protocol Versions and Cipher Suites

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Session Resumption and Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

OCSP Stapling and Certificate Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Advanced TLS: HSTS, MTLS, and Security Headers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 9: HTTP/2, HTTP/3, and QUIC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

HTTP/2: Multiplexing and Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

HTTP/2 Configuration and Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

HTTP/3 and QUIC: The Next Generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Protocol Negotiation and Fallback

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 10: Authentication, Authorization, and Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Basic and Digest Authentication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

IP-Based Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

OAuth and OpenID Connect Integration Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

External Authentication via Subrequests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

JWT Validation and API Security

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Role-Based Access Control Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 11: Request Rewriting, Redirects, and URL Manipulation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Rewrite Directive: Syntax and Behavior

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Internal Redirects vs Client Redirects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Return Directive: Simpler Alternatives

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Argument Manipulation and Query String Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Migrating from Apache .htaccess Rules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

SEO-Friendly Redirects and Canonicalization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 12: Application Integration: FastCGI, uWSGI, SCGI, and gRPC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

FastCGI: PHP and Beyond

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

uWSGI Protocol: Python and Ruby Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

SCGI: A Simpler Alternative

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

gRPC Proxying: Modern Microservices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Application Protocol Selection Guidance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 13: WebSocket and Long-Lived Connection Proxying

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

WebSocket Protocol and Upgrade Mechanism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Configuring WebSocket Proxying

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Long-Polling and Server-Sent Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Scaling WebSocket Connections

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Security Considerations for Persistent Connections

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Troubleshooting WebSocket Issues

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 14: Logging, Monitoring, and Observability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Access and Error Log Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Custom Log Formats

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The stub_status Module and Basic Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX Plus API and Advanced Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Integration with Observability Stacks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Debug Logging and Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Case Study: Building a Production Observability Stack for NGINX

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 15: Stream Proxying, Mail Proxy, and TCP/UDP Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Stream Module: TCP/UDP Load Balancing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

TLS Termination for TCP Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

UDP Proxying and DNS Load Balancing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

The Mail Module: SMTP, IMAP, and POP3 Proxying

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Database Connection Proxying Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 16: Performance Tuning and Kernel Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Worker Process and Event Loop Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Buffer Sizing Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Kernel Parameter Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Advanced Kernel Tuning for Specific Workloads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Filesystem and I/O Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Connection and Timeout Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Load Testing Methodology

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 17: Security Hardening and Defense in Depth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Configuration Hardening Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Mitigating Common Web Attacks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Security Headers Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Vulnerability Management and Patching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Securing the NGINX Process Itself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

WAF Integration and Request Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Case Study: Responding to a DDoS Attack with NGINX Rate Limiting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 18: Containerized Deployments and Kubernetes Ingress

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Docker Deployment Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Kubernetes Ingress Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX Ingress Controller Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Advanced Kubernetes Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Service Mesh Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Diagram Description: Kubernetes Ingress Traffic Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

GitOps and Declarative Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 19: High Availability, Automation, and Production Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

High Availability Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Zero-Downtime Configuration Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Dynamic Reconfiguration (NGINX Plus)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX Plus API: Advanced Usage Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Configuration Management and Automation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

CI/CD Integration for NGINX Configurations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Operational Runbooks and Troubleshooting Flows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 20: Troubleshooting, Debugging, and Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Reading and Interpreting Error Logs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Connection and Timeout Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Upstream and Backend Issues

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

TLS and SSL Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Performance Degradation Diagnosis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Common Configuration Pitfalls and Anti-Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Advanced Debugging Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 21: NGINX Ecosystem Comparison and Technology Selection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs Apache: Architecture and Philosophy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs Apache Feature Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs HAProxy: The Load Balancer Debate

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs HAProxy Feature Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs Envoy: Cloud-Native and Service Mesh

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs Envoy Feature Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs Caddy and Traefik: Modern Alternatives

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX vs Caddy vs Traefik Feature Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Technology Selection Framework

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Chapter 22: Migration Strategies and Modernization Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Migrating from Apache to NGINX

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Upgrading NGINX Versions Safely

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Modernizing Legacy NGINX Configurations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Evolving from Open Source to NGINX Plus

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Decommissioning and Replacement Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Case Study: Migrating from Apache to NGINX for a High-Traffic Media Platform

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Conclusion: The NGINX Mastery Journey

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

RFCs and Standards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Official NGINX Documentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

F5 NGINX Documentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

NGINX Community Blog Articles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Third-Party Resources and Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Historical and Background Sources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.

Performance and Comparison Sources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringnginx.