

Nginx Quick Reference

Server blocks, proxying, SSL, load balancing, logging

Installation

Install by OS

Ubuntu / Debian	sudo apt install nginx
RHEL / CentOS	sudo dnf install nginx
macOS	brew install nginx
Alpine	apk add nginx
Docker	docker run -p 80:80 nginx

Service Management

sudo systemctl start nginx	Start Nginx
sudo systemctl stop nginx	Stop Nginx
sudo systemctl reload nginx	Reload config (no downtime)
sudo systemctl enable nginx	Enable on boot
nginx -t	Test config syntax
nginx -T	Test and dump full config
nginx -s reload	Signal running process to reload

Basic Config

File Locations

/etc/nginx/nginx.conf	Main configuration file
/etc/nginx/conf.d/	Drop-in site configs (*.conf)
/etc/nginx/sites-available/	Available site configs (Debian)
/etc/nginx/sites-enabled/	Symlinks to active configs
/var/log/nginx/	Access and error logs
/var/www/html/	Default document root

Minimal Config

```
server {
    listen 80;
    server_name example.com;
    root /var/www/mysite;
    index index.html;
}
```

Config Structure

http { }	HTTP server settings (top level)
server { }	Virtual host definition
location { }	URI matching block
upstream { }	Backend server group
events { }	Connection handling settings

Server Blocks

Name-Based Virtual Hosts

```
server {
    listen 80;
    server_name site-a.com;
    root /var/www/site-a;
}
server {
    listen 80;
    server_name site-b.com;
    root /var/www/site-b;
}
```

Default & Catch-All

```
server {
    listen 80 default_server;
    server_name _;
    return 444; # drop connection
}
```

HTTPS Redirect

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

Location Blocks

Match Priority (high to low)

= /path	Exact match (highest priority)
^~ /path	Prefix match, skip regex
~ regex	Case-sensitive regex
~* regex	Case-insensitive regex
/path	Prefix match (lowest priority)

Location Examples

```
location = / {
    # exact root only
}
location /api/ {
    proxy_pass http://backend;
}
location ~* \.(jpg|png|gif)$ {
    expires 30d;
}
```

try_files

```
location / {
    try_files $uri $uri/ /index.html;
}
```

Try file, then directory, then fallback -- essential for SPAs

Reverse Proxy

Basic Proxy

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

WebSocket Proxy

```
location /ws/ {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
```

Proxy Directives

proxy_pass	Backend URL
proxy_set_header	Pass custom headers to backend
proxy_read_timeout	Timeout for backend response (default 60s)
proxy_buffering off	Disable response buffering
proxy_redirect	Rewrite Location headers from backend

SSL / TLS

HTTPS Server

```
server {
    listen 443 ssl;
    server_name example.com;

    ssl_certificate /etc/ssl/certs/example.crt;
    ssl_certificate_key /etc/ssl/private/example.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
}
```

Let's Encrypt with Certbot

```
sudo certbot --nginx -d example.com
sudo certbot renew --dry-run
```

SSL Best Practices

ssl_protocols TLSv1.2 TLSv1.3	Disable old TLS versions
ssl_prefer_server_ciphers on	Server chooses cipher
ssl_session_cache shared:SSL:10m	Session reuse for performance
add_header Strict-Transport-Security	HSTS header
ssl_stapling on	OCSP stapling for faster handshake

Load Balancing

Upstream Block

```
upstream backend {
    server 10.0.0.1:3000;
    server 10.0.0.2:3000;
    server 10.0.0.3:3000;
}
server {
    location / {
        proxy_pass http://backend;
    }
}
```

Load Balancing Methods

(default)	Round-robin
least_conn	Fewest active connections
ip_hash	Client IP sticky sessions
hash \$request_uri	Consistent hash by URI

Server Options

weight=3	Send 3x more traffic
max_fails=3	Failures before marking down
fail_timeout=30s	Time to mark server as down
backup	Use only when others are down
down	Mark server as permanently offline

Static Files & Caching

Serve Static Files

```
location /static/ {
    alias /var/www/assets/;
    expires 30d;
    add_header Cache-Control "public, immutable";
}
```

Nginx Quick Reference

Gzip Compression

```
gzip on;
gzip_types text/plain text/css
        application/json application/javascript;
gzip_min_length 1000;
gzip_comp_level 5;
```

Caching Directives

expires 30d	Set Expires and Cache-Control max-age
expires off	Disable expires header
etag on	Enable ETag header (default)
sendfile on	Efficient file serving via kernel
tcp_nopush on	Optimize packet sending

Logging

Log Configuration

```
access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log warn;

# Custom log format
log_format main '$remote_addr - $status '
                '$request' $body_bytes_sent';
access_log /var/log/nginx/access.log main;
```

Error Log Levels

debug	Verbose (requires --with-debug)
info	Informational
notice	Normal but notable
warn	Warnings
error	Errors (default)
crit	Critical issues

Conditional Logging

```
map $status $loggable {
    ~^[23] 0;
    default 1;
}
access_log /var/log/nginx/access.log combined if=$loggable;
```

Skip logging 2xx/3xx responses to reduce log volume

Security

Rate Limiting

```
limit_req_zone $binary_remote_addr
zone=api:10m rate=10r/s;

location /api/ {
    limit_req zone=api burst=20 nodelay;
}
```

Access Control

```
location /admin/ {
    allow 192.168.1.0/24;
    deny all;
}
```

Security Headers

X-Frame-Options DENY	Prevent clickjacking
X-Content-Type-Options nosniff	Prevent MIME sniffing
X-XSS-Protection "1; mode=block"	XSS filter (legacy browsers)
Content-Security-Policy	Control resource loading sources
Referrer-Policy no-referrer	Control referrer information

Common Patterns

SPA (Single-Page App)

```
location / {
    root /var/www/app;
    try_files $uri $uri/ /index.html;
}
```

CORS Headers

```
location /api/ {
    add_header Access-Control-Allow-Origin *;
    add_header Access-Control-Allow-Methods
        "GET, POST, PUT, DELETE, OPTIONS";
    if ($request_method = OPTIONS) {
        return 204;
    }
    proxy_pass http://backend;
}
```

Useful Variables

\$host	Request Host header
\$uri	Current URI (normalized)
\$request_uri	Original URI with query string
\$remote_addr	Client IP address
\$scheme	http or https
\$args	Query string parameters
\$status	Response status code