

NGINX クイックリファレンス

サーバーブロック、プロキシ、SSL、ロードバランシング、ロギング

インストール

OS ごとのインストール

```
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`
```

サービス管理

```
sudo systemctl start nginx `Nginx を起動`
sudo systemctl stop nginx `Nginx を停止`
sudo systemctl reload nginx `設定を再読み込み (ダウタイムなし)`
```

```
sudo systemctl enable nginx `起動時に有効化`
nginx -t `設定構文をテスト`
nginx -T `テストして全設定をダンプ`
nginx -s reload `実行中のプロセスに再読み込みシグナルを送信`
```

基本設定

ファイルの場所

```
/etc/nginx/nginx.conf `メイン設定ファイル`
/etc/nginx/conf.d/ `サイト設定のドロップイン (.conf)`
```

```
/etc/nginx/sites-available/ `利用可能なサイト設定 (Debian)`
```

```
/etc/nginx/sites-enabled/ `有効な設定へのシンボリックリンク`
```

```
/var/log/nginx/ `アクセスログとエラーログ`
/var/www/html/ `デフォルトのドキュメントルート`
```

最小構成

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

設定の構造

```
http { # HTTP サーバー設定 (トップレベル)
server { # バージャルホストの定義
location { # URL マッチングブロック
upstream { # バックエンドサーバーグループ
events { # 接続処理の設定
```

サーバーブロック

名前ベースのバーチャルホスト

```
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;
}
```

デフォルトとキャッチオール

```
server {
    listen 80 default_server;
    server_name _;
    return 444; # 接続を切断
}
```

HTTPS へのリダイレクト

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

location ブロック

マッチの優先順位 (高→低)

```
= /path `完全一致 (最高優先度)`
^~ /path `プレフィックス一致、正規表現をスキップ`
~ regex `大文字小文字区別の正規表現`
~* regex `大文字小文字区別なしの正規表現`
/path `プレフィックス一致 (最低優先度)`
```

location の例

```
location = / {
    # ルートのみ完全一致
}
location /api/ {
    proxy_pass http://backend;
}
location ~* \.(jpg|png|gif)$ {
    expires 30d;
}
```

try_files

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

ファイル、ディレクトリ、フォールバックの順に試行 – SPA に必須

リバースプロキシ

基本プロキシ

```
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 プロキシ

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

プロキシディレクティブ

```
proxy_pass `バックエンド URL`
proxy_set_header `カスタムヘッダーをバックエンドに転送`
proxy_read_timeout `バックエンド応答のタイムアウト (デフォルト 60 秒)`
proxy_buffering off `レスポンスバッファリングを無効化`
proxy_redirect `バックエンドからの Location ヘッダーを書き換え`
```

SSL/TLS

HTTPS サーバー

```
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;
}
```

Certbot による Let's Encrypt

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

SSL のベストプラクティス

```
ssl_protocols TLSv1.2 TLSv1.3 `古い TLS バージョンを無効化`
```

```
ssl_prefer_server_ciphers on `サーバーが暗号スイートを選択`
```

```
ssl_session_cache shared:SSL:10m `パフォーマンスのためのセッション再利用`
```

```
add_header Strict-Transport-Security `HSTS ヘッダー`
ssl_stapling on `高速ハンドシェイクのための OCSP ステージング`
```

ロードバランシング

upstream ブロック

```
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;
    }
}
```

ロードバランシング方式

```
(デフォルト) `ラウンドロビン`
least_conn `アクティブ接続数が最も少ない`
ip_hash `クライアント IP によるステッキーセッション`
hash $request_uri `URI による一貫性ハッシュ`
```

サーバーオプション

```
weight=3 `3 倍のトラフィックを送信`
max_fails=3 `ダウンと判定するまでの失敗回数`
fail_timeout=30s `サーバーをダウンと判定する時間`
backup `他のサーバーがダウンした場合のみ使用`
down `サーバーを永続的にオフラインとしてマーク`
```

静的ファイルとキャッシュ

静的ファイルの配信

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

Gzip 圧縮

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

キャッシュディレクティブ

```
expires 30d `Expires と Cache-Control max-age を設定`
expires off `expires ヘッダーを無効化`
etag on `ETag ヘッダーを有効化 (デフォルト)`
sendfile on `カーネル経由の効率的なファイル配信`
tcp_nopush on `パケット送信の最適化`
```

ロギング

ログ設定

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

# カスタムログフォーマット
log_format main '$remote_addr - $status '
'$request' "$body_bytes_sent";
access_log /var/log/nginx/access.log main;
```

エラーログのレベル

```
debug `詳細 (--with-debug が必要)`
info `情報`
```

notice `通常だが注目すべき`

warn `警告`

error `エラー (デフォルト)`

crit `重大な問題`

条件付きロギング

```
map $status $loggable {
    ~{2} 0;
    default 1;
}
access_log /var/log/nginx/access.log combined if=$loggable;
```

2xx/3xx レスポンスのロギングをスキップしてログ量を削減

セキュリティ

レートリミット

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

アクセス制御

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

セキュリティヘッダー

```
X-Frame-Options DENY `クリックジャッキングを防止`
```

```
X-Content-Type-Options nosniff `MIME スニッフィングを防止`
```

```
X-XSS-Protection "1; mode=block" `XSS フィルター (レガシーブラウザ)`
```

```
Content-Security-Policy `リソース読み込み元を制御`
```

```
Referrer-Policy no-referrer `リファラー情報を制御`
```

よくあるバグ

SPA (シングルページアプリケーション)

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

CORS ヘッダー

```
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;
}
```

便利な変数

```
$host `リクエストの Host ヘッダー`
$uri `現在の URI (正規化済み)`
$request_uri `クエリ文字列付きの元の URI`
$remote_addr `クライアント IP アドレス`
$scheme `http または https`
$args `クエリ文字列パラメータ`
$status `レスポンスのステータスコード`
```