ŚCIĄGA LARAVEL

Nginx dla Laravel

Publiczny `public/`, TLS, PHP-FPM, reverse proxy dla WebSocketów

Zasady

- Nginx wystawia **wyłącznie** `public/`.
- Ścieżki aplikacji → `index.php`; ukryte pliki i arbitralne PHP → `deny`.
- nginx ≥ 1.25.1: `http2 on;` (nie `listen … http2`).
- Zawsze `nginx -t` przed `reload`.
- TLS wyżej (LB/proxy) → skonfiguruj trusted proxies w Laravelu.

Minimalny bezpieczny host

```
server {
    listen 443 ssl;
    http2 on;
    server_name example.com;
    root /var/www/current/public;      # tylko public/
    index index.php;

    location / { try_files $uri $uri/ /index.php?$query_string; }

    location ~ \.php$ {
        try_files $uri =404;           # nie wykonuj nieistniejących skryptów
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        fastcgi_pass unix:/run/php/php8.5-fpm.sock;
    }

    location ~ /\.(?!well-known).* { deny all; }
}
```

`$realpath_root` rozwiązuje symlink wydania przed przekazaniem skryptu do PHP-FPM — kluczowe przy atomowych deployach.

Port 80 → tylko redirect

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

server {
    listen 443 ssl;
    http2 on;
    server_name example.com;
    root /var/www/current/public;
    client_max_body_size 20m;
    add_header X-Content-Type-Options nosniff always;

    location / { try_files $uri $uri/ /index.php?$query_string; }
    location ~ \.php$ { try_files $uri =404; include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        fastcgi_pass unix:/run/php/php8.5-fpm.sock; }
    location ~ /\.(?!well-known).* { deny all; }
}
```

TLS, redirecty i wartości FastCGI trzymaj w **przejrzanych** plikach `include`.

Reverse proxy (Reverb)

```
location /app/ {
    proxy_pass http://reverb:8080;
    proxy_http_version 1.1;                # WebSocket wymaga HTTP/1.1
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_read_timeout 60s;
}
```

Cache i limitowanie

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

location = /login {
    limit_req zone=login burst=10 nodelay;
    try_files $uri $uri/ /index.php?$query_string;
}
```

Cache'uj agresywnie **tylko** assety z hashem w URL. Nigdy zalogowany HTML ani odpowiedzi autoryzacji bez jawnego klucza i invalidacji.

`limit_req` na loginie **nie** zastępuje autoryzacji ani weryfikacji podpisu webhooka.

Weryfikacja

```
$ sudo nginx -t
$ sudo systemctl reload nginx
$ curl -I https://example.com/health
```

Kiedy tego NIE używać

- Shared hosting lub platforma zarządzająca Nginxem → wspierana integracja, nie ta konfiguracja.
- Build obrazów Docker, lokalny cache i sam mechanizm wdrożenia to osobne tematy — inne tryby awarii i rollback.

PODSUMOWANIE

Ściąga w skrócie

- → Serwuj tylko `public/`, `http2 on;`
- → `try_files $uri =404` w bloku PHP
- → `$realpath_root` dla atomowych deployów
- → Port 80 tylko `return 301`
- → WebSocket: `proxy_http_version 1.1` + upgrade
- → Cache tylko hashowane assety; `nginx -t` przed reload

Przydało się?

Zapisz tę ściągę na później.

Pełny artykuł: dominik-dev.pl

Udostępnij • Skomentuj