Laravel & Nginx: Best Practices for Production

Laravel & Nginx: Best Practices for Production

Caddy might be the new kid on the block, but **Nginx** remains the most widely used web server for Laravel in production. It's stable, efficient, and battle-tested at scale. To run a Laravel 12 application smoothly in production with Nginx, you'll need the right configuration, caching rules, and security headers. In this guide, we'll walk through best practices to deploy Laravel with Nginx.

1 — Install Nginx & PHP-FPM

Most Linux servers don't ship with Nginx preinstalled. You'll need Nginx plus PHP-FPM to handle dynamic PHP requests efficiently.

```
sudo apt update
sudo apt install -y nginx php8.3-fpm php8.3-mysql unzip curlCode language: Bash (bash)
```

We install the `nginx` server, PHP 8.3 FPM for executing Laravel, and supporting packages. If you're deploying on a provider like DigitalOcean or AWS, see How to Deploy a Laravel 12 App on DigitalOcean and Deploying Laravel on AWS: Complete Guide (2025) for cloud-specific steps.

2 — Nginx Server Block for Laravel

Create a new config file `/etc/nginx/sites-available/laravel.conf` and point it to the `public/` directory. This ensures only public assets are directly accessible.

```
server {
    listen 80;
    server_name your-domain.com;
    root /var/www/current/public;

    index index.php index.html;

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

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass unix:/var/run/php/php8.3-fpm.sock;
        fastcgi_param SCRIPT_FILENAME
$realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }
}Code language: Nginx (nginx)
```

Here, `root` points to `public/`, and all requests fall back to `index.php` if no file is found. The `fastcgi_pass` points to the PHP-FPM socket (adjust version accordingly).

3 — Enable Compression

Gzip (or Brotli if supported) should be enabled to reduce file sizes. Add this to your `nginx.conf` under the `http` block.

```
gzip on;
gzip_comp_level 5;
gzip_min_length 1024;
gzip_proxied any;
gzip_types text/plain text/css application/javascript application/json
application/xml image/svg+xml;
gzip_vary on;Code language: Nginx (nginx)
```

This ensures text-based assets are compressed before sending to the client, saving bandwidth and improving speed. For further performance tuning, check 10 Proven Ways to Optimize Laravel for High Traffic.

4 — Handling HTTPS

Use Let's Encrypt certificates or integrate with your hosting provider's SSL. With Nginx, configure your `server` block for HTTPS:

```
server {
    listen 443 ssl http2;
    server_name your-domain.com;

    ssl_certificate /etc/letsencrypt/live/your-
domain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/your-
domain.com/privkey.pem;

    include snippets/ssl-params.conf;
```

```
    root /var/www/current/public;
    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }
}Code language: Nginx (nginx)
```

With this, Caddy would have automated TLS by default, but in Nginx you typically configure Let's Encrypt with `certbot`. If you're deploying on a cloud provider like AWS, offload TLS termination to an ALB or load balancer (Deploying Laravel on AWS: Complete Guide (2025)).

5 — Optimize Nginx & PHP-FPM

Set proper buffer and worker configurations in `nginx.conf` and tune PHP-FPM for concurrent requests.

```
# /etc/nginx/nginx.conf (http { ... })
worker_processes auto;
worker_connections 1024;
client_max_body_size 20M;

# PHP upstream for TCP (if not using socket)
upstream php {
    server 127.0.0.1:9000;
}Code language: Nginx (nginx)
```

Here, `worker_processes auto;` lets Nginx use available CPU cores. The `client_max_body_size` ensures large uploads don't get rejected prematurely.

```
; /etc/php/8.3/fpm/pool.d/www.conf
pm = dynamic
```

```
pm.max_children = 20
pm.start_servers = 4
pm.min_spare_servers = 4
pm.max_spare_servers = 8Code language: TOML, also INI (ini)
```

These PHP-FPM settings control concurrency. Adjust `pm.max_children` based on available memory and expected traffic. For very high traffic, consider Optimizing Laravel for High Concurrency with Octane.

6 — Add a Health Check Route

Monitoring your Laravel + Nginx deployment ensures uptime. Add a simple route for your load balancer or uptime monitor.

```
// routes/web.php
Route::get('/health', function () {
    return response()->json([
        'status' => 'ok',
        'app' => config('app.name'),
        'time' => now(),
    ]);
});Code language: PHP (php)
```

This `/health` endpoint is lightweight and lets Nginx or a monitoring tool confirm your Laravel app is alive. You can extend it with database or cache checks for better observability.

Wrapping Up

Deploying Laravel 12 with Nginx is reliable and production-friendly when configured correctly. Focus on secure server blocks, efficient PHP-FPM tuning, caching, and TLS. For large-scale scenarios, you may want to explore Optimizing Laravel for High Concurrency with Octane. If you're still on shared hosting, revisit How to Deploy Laravel to Shared Hosting (Step by Step) to compare approaches.

What's Next

- Step-by-Step CI/CD Pipeline Setup for Laravel 12 on GitHub Actions — automate deployments with testing & caching.
- Laravel Deployment Checklist for 2025 — essential production readiness checklist.
- Optimizing Laravel for AWS Deployment (Step-by-Step) — integrate Nginx with AWS services.