1v0 Ship v1.0 Faster

Laravel Deployment Checklist for 2025

Laravel Deployment Checklist for 2025

Deploying a Laravel 12 application is more than just copying files to a server. A proper deployment process ensures performance, security, and maintainability in production. This checklist will help you avoid common pitfalls and ship your Laravel apps with confidence in 2025.

1 — Environment Configuration

# .env (production) APP_ENV=production APP_DEBUG=false APP_URL=https://yourdomain.com LOG_CHANNEL=stack DB_CONNECTION=mysql DB_HOST=127.0.0.1 DB_PORT=3306 DB_DATABASE=your_database DB_USERNAME=your_user DB_PASSWORD=your_password

Always disable `APP_DEBUG` in production to avoid exposing sensitive stack traces (see How to Prevent CSRF, XSS, and SQL Injection in Laravel Apps for more security tips).

2 — Cache Config, Routes & Views

```bash
# Optimize config, routes, and views
php artisan config:cache
php artisan route:cache
php artisan view:cache
```
Code language: Bash (bash)

Laravel Starter Kits

This ensures Laravel loads configuration, routes, and views directly from cached files, reducing filesystem lookups. Learn more in 10 Proven Ways to Optimize Laravel for High Traffic.

3 — Queue & Scheduler Setup

```
# Supervisor config for queue workers (example)
[program:laravel-worker]
process_name=%(program_name)s_%(process_num)02d
command=php /var/www/current/artisan queue:work --sleep=3 --tries=3 --max-time=3600
autostart=true
autorestart=true
numprocs=3
redirect_stderr=true
stdout_logfile=/var/www/current/storage/logs/worker.log
```
Code language: Bash (bash)

Queue workers should always be monitored by a process manager like Supervisor. This ensures failed jobs can be retried and workers restart if they crash. For more advanced queue monitoring, see How to Use Laravel Horizon for Queue Monitoring.

4 — File Storage & Symbolic Links

Make sure your `storage` and `bootstrap/cache` folders are writable. Then link `storage/app/public` to `public/storage`.

```
php artisan storage:link
sudo chown -R www-data:www-data /var/www/current/storage
/var/www/current/bootstrap/cache
sudo chmod -R 775 /var/www/current/storage
/var/www/current/bootstrap/cacheCode language: Bash (bash)
```

This ensures user uploads (like images or documents) are accessible through the web server. For a deep dive into secure file handling, check How to Prevent CSRF, XSS, and SQL Injection in Laravel Apps and How to Build a Secure File Upload API in Laravel.

5 — Database Migration & Seeding

```
# Run migrations in production (force required)
php artisan migrate --force

# Optionally seed initial data
php artisan db:seed --forceCode language: Bash (bash)
```

Always run migrations with `--force` in production to apply schema changes without prompts. If you're working with multi-tenant setups, also see Building a Multi-Tenant App in Laravel with Separate Databases for tenant-specific migrations.

6 — Add Security Headers & HTTPS

Use Nginx to enforce HTTPS and add HTTP security headers.

```
server {
```

```nginx
    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;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-Content-Type-Options "nosniff";
    add_header Referrer-Policy "strict-origin-when-cross-origin";
    add_header Content-Security-Policy "default-src 'self'";

    root /var/www/current/public;
    index index.php index.html;
}Code language: Nginx (nginx)
```

Certificates can be managed with Let's Encrypt for free. If you're using AWS or DigitalOcean, see How to Deploy a Laravel 12 App on DigitalOcean or Deploying Laravel on AWS: Complete Guide (2025) for infrastructure-specific instructions.

7 — Deployment Sanity Check UI

Add a simple admin-only endpoint to confirm Nginx headers, HTTPS, and storage access are all functioning correctly.

```php
// routes/web.php
use Illuminate\\Support\\Facades\\Route;

Route::middleware(['auth','can:viewAdmin'])->get('/admin/deployment-
check', function () {
    return view('admin.deployment-check', [
        'phpVersion' => phpversion(),
        'laravelEnv' => app()->environment(),
        'isSecure'   => request()->isSecure(),
        'clientIp'   => request()->ip(),
        'cachePathWritable' =>
is_writable(storage_path('framework/cache')),
    ]);
});
```
Code language: PHP (php)

This creates a page that shows the PHP version, Laravel environment, whether HTTPS is active, the detected client IP (to confirm `real_ip` works), and whether cache directories are writable. Only admins should have access to this page.

```
<!-- resources/views/admin/deployment-check.blade.php -->
@extends('layouts.app')
@section('content')
<div class="container">
  <h1 class="mb-4">Deployment Health Check</h1>
  <ul class="list-group">
    <li class="list-group-item"><strong>PHP Version:</strong> {{
$phpVersion }}</li>
    <li class="list-group-item"><strong>Environment:</strong> {{
$laravelEnv }}</li>
    <li class="list-group-item"><strong>HTTPS Enabled:</strong> {{
$isSecure ? 'Yes' : 'No' }}</li>
    <li class="list-group-item"><strong>Client IP:</strong> {{
$clientIp }}</li>
    <li class="list-group-item"><strong>Cache Writable:</strong> {{
$cachePathWritable ? 'Yes' : 'No' }}</li>
  </ul>
</div>
@endsection
```
Code language: HTML, XML (xml)

This dashboard view provides instant feedback on whether your Laravel app is healthy and properly configured in production.

Wrapping Up

By combining Nginx with Laravel 12, you get a fast and reliable production setup. Key steps include configuring the server block, enabling caching, ensuring HTTPS, tuning PHP-FPM, and monitoring your application. For more advanced scenarios, you can explore containerized setups or automated deployment tools.

What's Next

- Laravel Deployment Checklist for 2025 — a complete pre-launch checklist to avoid common mistakes.
- Optimizing Laravel for AWS Deployment (Step-by-Step) — learn how to scale Laravel with AWS and integrate load balancers.
- Automating Laravel Deployments with Deployer — take your deployments to the next level with zero-downtime automation.