1v0 Ship v1.0 Faster

How to Deploy Laravel to Shared Hosting (Step by Step)

How to Deploy Laravel to Shared Hosting (Step by Step)

Not every project requires cloud infrastructure like AWS or DigitalOcean. Sometimes, you just need to put a Laravel 12 app on traditional **shared hosting**. While shared hosting environments are limited (no root access, limited SSH, restricted PHP extensions), you can still successfully deploy Laravel apps with a few adjustments. In this guide, we'll cover a step-by-step process to deploy Laravel on shared hosting the right way.

1 — Prepare Your Laravel App Locally

Before uploading to your hosting server, you should optimize and clean your Laravel project.

```
# On your local machine
composer install --optimize-autoloader --no-dev
npm run build

php artisan config:cache
php artisan route:cache
php artisan view:cacheCode language: Bash (bash)
```

This installs only production dependencies, compiles frontend assets (via Vite), and caches configuration, routes, and views for speed. By doing this locally, you avoid heavy Composer or NPM installs on your shared host (where resources are often limited).

2 — Adjust the Public Directory

Shared hosting usually points the web root to `public_html/`. Laravel, however, expects `public/` as its entry point. To fix this, move your project and update paths.

```
# Folder structure after moving
/my-app
  /app
  /bootstrap
  /config
  /database
  /public (renamed or moved to /public_html)
  /resources
  /routes
  /vendor
  .env
  artisanCode language: Bash (bash)
```

Move everything *except* the `public/` folder outside of `public_html/`. Then, copy the contents of `public/` into `public_html/`. This keeps your sensitive files out of the web root while still serving assets and `index.php`.

3 — Update index.php Paths

Because `public/index.php` is now inside `public_html/`, you need to update the paths to point to your Laravel root.

```
// public_html/index.php

require __DIR__.'/../vendor/autoload.php';

$app = require_once __DIR__.'/../bootstrap/app.php';Code language: PHP (php)
```

The `../` ensures Laravel loads from the parent directory (where `vendor/` and `bootstrap/` are stored). Without this, your app will break on shared hosting.

4 — Configure .env for Shared Hosting

Edit your `.env` file to match the hosting provider's database credentials and environment settings.

APP_ENV=production APP_KEY=base64:xxxxxx APP_DEBUG=false APP_URL=https://yourdomain.com DB_CONNECTION=mysql DB_HOST=127.0.0.1 DB_PORT=3306 DB_DATABASE=your_db DB_USERNAME=your_user DB_PASSWORD=your_pass

Most shared hosts provide database info in cPanel (MySQL Databases section). Always set `APP_DEBUG=false` in production to avoid leaking sensitive details.

5 — Upload Your Project

Zip your project locally (excluding `node_modules`) and upload via FTP or cPanel's File Manager. Then extract the zip on the server.

```
# Example of excluding node_modules
zip -r app.zip . -x "node_modules/*"Code language: Bash (bash)
```

Extract the archive so the Laravel core files are one level above `public_html`. Ensure `storage` and `bootstrap/cache` directories are writable (set permissions to 775).

6 — Configure .htaccess

Most shared hosting environments run Apache. Add or edit `public_html/.htaccess` for Laravel's routing.

```
<IfModule mod_rewrite.c>
    <IfModule mod_negotiation.c>
        Options -MultiViews -Indexes
    </IfModule>

    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]
</IfModule>Code language: Apache (apache)
```

This routes all non-file requests through `index.php`, which boots Laravel. Without it, URLs like `/dashboard` will 404.

7 — Run Migrations and Storage Link

If you have SSH access, run migrations and link the storage folder. If not, you may need to use a route-based trick for one-time setup.

```
# If SSH available
php artisan migrate --force
php artisan storage:linkCode language: Bash (bash)
```

`--force` ensures migrations run in production. `storage:link` makes uploaded files accessible from `/storage` in the browser. On hosts without SSH, you can temporarily create a route/controller to call `Artisan::call('migrate')` and then remove it for security.

8 — Deployment Check UI

Since shared hosting often limits server monitoring, you can build a minimal admin-only status page to confirm database and storage health.

```
// routes/web.php
Route::middleware(['auth', 'can:viewAdmin'])->get('/status', function
() {
    return view('admin.shared-status', [
        'db' => DB::connection()->getDatabaseName(),
        'storageWritable' => is_writable(storage_path('app')),
    ]);
});Code language: PHP (php)
```

This route checks whether the database connection works and if the storage directory is writable.

```
<!-- resources/views/admin/shared-status.blade.php -->
@extends('layouts.app')
```

```
@section('content')
<div class="container">
  <h1 class="mb-4">Shared Hosting Status</h1>
  <ul class="list-group">
    <li class="list-group-item">Database: {{ $db ? $db : 'Not
connected' }}</li>
    <li class="list-group-item">Storage Writable: {{ $storageWritable
? 'Yes' : 'No' }}</li>
  </ul>
</div>
@endsectionCode language: HTML, XML (xml)
```

This simple page reassures you that your Laravel app is properly running on shared hosting, without needing advanced server access.

Wrapping Up

Deploying a Laravel 12 app on shared hosting requires some manual adjustments: moving the `public` folder into `public_html`, fixing `index.php` paths, setting the correct `.env`, and configuring `.htaccess`. Once migrations and storage links are in place, your project will run smoothly—even in limited environments. While shared hosting isn't ideal for large-scale apps, it's a cost-effective solution for small projects and MVPs.

What's Next

- Laravel & Nginx: Best Practices for Production (#56) — applies if your shared host uses Nginx instead of Apache.

- Laravel Deployment Checklist for 2025 (#58) — a final checklist before going live.
- Automating Laravel Deployments with Deployer (#59) — for when you move beyond shared hosting.