How to Deploy Laravel 12 on cPanel Hosting

How to Deploy Laravel 12 on cPanel Hosting

Not every project requires a VPS or cloud infrastructure. Many developers still rely on **cPanel hosting** for simplicity and cost-effectiveness. Deploying a Laravel 12 application on cPanel comes with its own challenges: shared servers, no SSH access in some cases, and the need to configure the `public/` folder properly. In this guide, we'll walk through a step-by-step approach to get your Laravel app running on a cPanel server with HTTPS, correct file permissions, and a clean URL structure.

1 — Prepare Your Laravel Project

Before uploading your application to cPanel, make sure you've run composer and built your frontend assets locally:

```bash
# From your local machine
composer install --optimize-autoloader --no-dev
npm install && npm run build
php artisan config:cache
php artisan route:cache
php artisan view:cache
php artisan migrate --force
```
Code language: Bash (bash)

This ensures your dependencies are installed, caches are built for performance, and your migrations are ready. Running `--no-dev` keeps unnecessary dev packages out of production.

2 — Upload to cPanel

Most cPanel providers give you FTP or File Manager access. You should upload your entire Laravel project, but place the `public/` folder's contents into the `public_html` directory.

```
# Example structure after upload
/home/username/
  ├── laravel-app/        # all Laravel files here
  │   ├── app/
  │   ├── bootstrap/
  │   ├── config/
  │   └── ...
  └── public_html/        # cPanel web root
      ├── index.php       # replaced with Laravel's public/index.php
      ├── css/
      ├── js/
      └── ...Code language: Bash (bash)
```

Instead of uploading the whole `public/` folder, move its contents into `public_html`. Then, edit `public/index.php` to point to the correct paths for `autoload.php` and `app.php`.

3 — Update Index.php Paths

After moving files, update the paths inside `public_html/index.php` so they point to the correct directory:

```
// public_html/index.php
```

```php
require __DIR__.'/../laravel-app/vendor/autoload.php';
$app = require_once __DIR__.'/../laravel-app/bootstrap/app.php';
```
Code language: PHP (php)

By default, `index.php` looks for `../vendor/autoload.php`. Since we moved files into `laravel-app/`, update the paths accordingly. Without this, Laravel won't boot in production.

4 — Configure .htaccess for Routing

Laravel routes everything through `public/index.php`. Update the `.htaccess` in `public_html/` to handle this properly:

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

    RewriteEngine On

    # Redirect Trailing Slashes...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^(.*)/$ /$1 [L,R=301]

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

This ensures that all requests are funneled through Laravel's front controller, while still serving static assets directly. For a similar setup on Nginx instead of cPanel/Apache, check Laravel & Nginx: Best Practices for Production.

5 — Configure Cron for Scheduler

Laravel’s task scheduler requires a cron job. You can set this in the cPanel > Cron Jobs section:

```bash
* * * * * php /home/username/laravel-app/artisan schedule:run >> /dev/null 2>&1
```
Code language: Bash (bash)

This runs the Laravel scheduler every minute, which then executes scheduled tasks as defined in `app/Console/Kernel.php`. For a production-grade queue and scheduling setup, see How to Use Laravel Queues for Faster Performance.

6 — Secure Your Application

cPanel hosting often comes with Apache and free SSL (via AutoSSL or Let’s Encrypt). Make sure you force HTTPS and add security headers in your `.htaccess` file.

```
# Force HTTPS
RewriteEngine On
RewriteCond %{HTTPS} !=on
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

# Security headers
<IfModule mod_headers.c>
    Header set X-Frame-Options "SAMEORIGIN"
    Header set X-Content-Type-Options "nosniff"
    Header set Referrer-Policy "strict-origin-when-cross-origin"
```

```
</IfModule>
```
Code language: Apache (apache)

These rules ensure all traffic is encrypted and common browser security headers are applied. For more on security, check [How to Prevent CSRF, XSS, and SQL Injection in Laravel Apps](#).

7 — Final Pre-Launch Checks

Before going live, walk through this checklist:

- ▯ `APP_ENV=production` and `APP_DEBUG=false`
- ▯ Run `php artisan config:cache`, `route:cache`, `view:cache`
- ▯ Ensure storage & bootstrap/cache are writable
- ▯ Public files are inside `public_html/`
- ▯ `.env` is in `/laravel-app/` (not exposed)
- ▯ Database migrations and seeders run with `--force`
- ▯ Cron job for scheduler is configured
- ▯ HTTPS and security headers are enforced

Once complete, your Laravel app is safely deployed to your cPanel hosting environment.

What's Next

- [Laravel Deployment Checklist for 2025](#) — follow a complete step-by-step checklist before going live.
- [Laravel & Nginx: Best Practices for Production](#) — see how to configure Nginx for high performance and security.

- Automating Laravel Deployments with Deployer — learn how to set up zero-downtime deployments.