1v0 Ship v1.0 Faster

How to Set Up Laravel with Caddy for Performance & HTTPS

How to Set Up Laravel with Caddy for Performance & HTTPS

Caddy is a modern web server that makes deploying Laravel 12 projects easier and faster. Unlike Nginx or Apache, Caddy has automatic HTTPS via Let's Encrypt built-in, simple configuration, and excellent support for HTTP/2 and HTTP/3 (QUIC). In this step-by-step guide, we'll set up Laravel behind Caddy, configure PHP-FPM, enable caching & compression, and add best practices for production.

1 — Install Caddy & PHP-FPM

On Ubuntu, you can install Caddy from the official repository and PHP-FPM for Laravel.

```
# Install Caddy
sudo apt install -y debian-keyring debian-archive-keyring apt-
transport-https
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' |
sudo gpg --dearmor -o /usr/share/keyrings/caddy.gpg
curl -1sLf
'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo
tee /etc/apt/sources.list.d/caddy-stable.list
sudo apt update
sudo apt install caddy -y

# Install PHP-FPM 8.3
sudo apt install php8.3-fpm php8.3-cli php8.3-mysql unzip -yCode
language: Bash (bash)
```

This installs both Caddy and PHP-FPM. Caddy handles HTTP/HTTPS while PHP-FPM executes PHP code for Laravel.

2 — Basic Caddyfile for Laravel

Caddy uses a single `Caddyfile` for configuration. Point the root to Laravel's `public/` folder, enable PHP, and set caching rules.

```
# /etc/caddy/Caddyfile
your-domain.com {
    root * /var/www/current/public

    # PHP-FPM
    php_fastcgi unix//run/php/php8.3-fpm.sock

    # Clean URLs
    try_files {path} {path}/ /index.php?{query}

    # Static file caching
    @static {
        path_regexp static
\.(?:ico|css|js|gif|jpg|jpeg|png|svg|woff2?)$
    }
    header @static Cache-Control max-age=604800

    # Security headers
    header {
        X-Frame-Options SAMEORIGIN
        X-Content-Type-Options nosniff
        Referrer-Policy "strict-origin-when-cross-origin"
    }

    encode gzip zstd
```

```
    file_server
}Code language: Nginx (nginx)
```

This simple Caddyfile enables automatic TLS (no extra steps needed for Let's Encrypt), serves static assets with caching, enables gzip/zstd compression, and configures PHP-FPM socket handling. Caddy automatically handles certificate renewals.

3 — Zero-Downtime Deploys with Symlinks

Just like Nginx, you can use a `current/` symlink for zero-downtime deployments. Upload new code to `releases/2025xxxx/` and update the symlink.

```
# Example deployment script
cd /var/www/releases
mkdir 20250828
unzip /tmp/app.zip -d 20250828
ln -sfn /var/www/releases/20250828 /var/www/current
php /var/www/current/artisan migrate --force
php /var/www/current/artisan config:cache
php /var/www/current/artisan route:cacheCode language: Bash (bash)
```

The `ln -sfn` command atomically switches the `current` symlink to the new release. No downtime—requests are served by the old code until the new one is ready. For a more advanced and automated approach, check out Automating Laravel Deployments with Deployer .

4 — Handling Queues & Scheduler

Shared hosting often won't give you Supervisor. With Caddy on VPS, you can configure Supervisor for workers and cron for scheduled tasks.

```
# /etc/supervisor/conf.d/laravel-worker.conf
[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.logCode language:
Bash (bash)
```

This keeps 3 queue workers always alive. Pair with Laravel Horizon for UI monitoring. For scheduled tasks, add a cron entry to call `artisan schedule:run` every minute.

5 — UI: TLS & Header Checker

Caddy's automatic TLS is great, but you might want a simple Laravel admin page to confirm HTTPS and headers are applied correctly.

```
// routes/web.php
Route::middleware(['auth','can:viewAdmin'])->get('/caddy-status',
function () {
    return view('admin.caddy-status', [
        'scheme' => request()->getScheme(),
        'headers' => [
            'X-Frame-Options' => request()->header('X-Frame-Options'),
            'X-Content-Type-Options' => request()->header('X-Content-
```

```
Type-Options'),
            'Referrer-Policy' => request()->header('Referrer-Policy'),
            'Content-Encoding' => request()->header('Content-
Encoding'),
        ]
    ]);
});Code language: PHP (php)
```

This route passes key header and scheme info to a Blade view for inspection. You'll see if the request is truly over HTTPS and whether compression headers (gzip/zstd) are active.

```
<!-- resources/views/admin/caddy-status.blade.php -->
@extends('layouts.app')
@section('content')
<div class="container">
  <h1 class="mb-4">Caddy Status</h1>
  <p><strong>Scheme:</strong> {{ $scheme }}</p>
  <ul class="list-group">
    @foreach($headers as $key => $value)
      <li class="list-group-item"><strong>{{ $key }}:</strong> {{
$value ?? '—' }}</li>
    @endforeach
  </ul>
</div>
@endsectionCode language: HTML, XML (xml)
```

This UI confirms your Caddy TLS and security headers are correctly applied, helping you quickly verify production readiness.

Wrapping Up

Caddy is an excellent alternative to Nginx for Laravel deployments: it's easy to configure, has automatic HTTPS, and supports HTTP/2 and HTTP/3 out of the box. By combining Caddy

with PHP-FPM, caching, compression, Supervisor for queues, and good deployment practices, you'll have a modern, fast, and secure Laravel production setup.

What's Next

- Laravel & Nginx: Best Practices for Production — compare Nginx vs Caddy approaches.
- Optimizing Laravel for AWS Deployment (Step-by-Step) — learn how to scale containers and servers.
- Laravel Deployment Checklist for 2025 — don't go live without reviewing this list.