1v0 Ship v1.0 Faster

How to Deploy a Laravel 12 App on DigitalOcean

How to Deploy a Laravel 12 App on DigitalOcean

DigitalOcean makes it fast to get a production-ready Laravel app online using Droplets (VMs), Managed Databases, and Spaces (S3-compatible storage). In this step-by-step guide, you'll provision a Droplet, install Nginx + PHP-FPM, configure env and services, secure the server with UFW + Let's Encrypt, wire Redis queues with Horizon, add a health check endpoint for your monitors, and ship with a zero-downtime deployment script.

1 – Create Your Droplet & Prerequisites

In the DigitalOcean dashboard, create an Ubuntu LTS Droplet (2 vCPU / 4GB RAM is a comfortable baseline), add your SSH key, and attach a floating IP if you want stable addressing. Optionally create a Managed MySQL/PostgreSQL database and a Managed Redis (or run Redis locally to start). Point your domain's A record at the Droplet IP.

2 – Install Nginx, PHP-FPM, and Extensions

SSH into the Droplet and install the web stack. We'll use PHP 8.3 for best performance and features.

```
# on Ubuntu
sudo apt update
sudo apt install -y nginx software-properties-common
```

```
sudo add-apt-repository ppa:ondrej/php -y
sudo apt update
sudo apt install -y php8.3-fpm php8.3-cli php8.3-mysql php8.3-xml
php8.3-mbstring php8.3-curl php8.3-zip php8.3-bcmath unzip git
sudo systemctl enable --now nginx php8.3-fpmCode language: Bash (bash)
```

This installs Nginx and PHP-FPM with common Laravel extensions and enables both services on boot so a restart won't bring the app down.

3 – Create the Nginx Server Block

Point Nginx to Laravel's `public` folder and pass PHP requests to PHP-FPM's Unix socket.

```
# /etc/nginx/sites-available/laravel.conf
server {
    listen 80;
    server_name your-domain.com www.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 snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_read_timeout 60s;
    }

    location ~* \.(?:css|js|jpg|jpeg|png|gif|webp|ico|woff2?)$ {
        expires 7d;
```

```
        access_log off;
    }
}Code language: Nginx (nginx)
```

The `root` uses a *current* symlink, which we'll switch atomically during deploys. Static assets get 7-day caching; PHP routes go to `index.php`.

```
sudo ln -s /etc/nginx/sites-available/laravel.conf /etc/nginx/sites-enabled/
sudo nginx -t && sudo systemctl reload nginxCode language: Bash (bash)
```

This enables the site, lints the config, and reloads Nginx without downtime. For deeper tuning and hardening, see Laravel & Nginx: Best Practices for Production (Article #56).

4 – Clone App & Prepare Directory Layout

We'll use a releases pattern under `/var/www` for zero-downtime.

```
sudo mkdir -p /var/www/releases /var/www/shared
sudo chown -R $USER:www-data /var/www
cd /var/www/releases
# first release
git clone https://github.com/you/your-laravel-app.git 20250828-000001
cd 20250828-000001
composer install --no-dev --optimize-autoloader
cp .env.example /var/www/shared/.env
php artisan key:generate --force --env=production --no-interactionCode language: Bash (bash)
```

The `shared` directory holds persistent files like `.env`, storage, etc. Each release gets its own folder; a `current` symlink will point to the active one.

```
# link shared storage into the release
```

```
rm -rf storage
ln -s /var/www/shared/storage storage
# make current point to the release
ln -sfn /var/www/releases/20250828-000001 /var/www/current
sudo chown -R www-data:www-data /var/www/shared
/var/www/current/storage
php artisan storage:link
php artisan config:cache && php artisan route:cache && php artisan
view:cacheCode language: Bash (bash)
```

We symlink `storage` so logs/uploads persist across deploys. Caching config/routes/views speeds up boot (see Article #41: *10 Proven Ways to Optimize Laravel for High Traffic*).

5 – Configure .env (Managed DB, Redis, Mail)

If you created a Managed DB/Redis in DigitalOcean, paste the connection strings into `/var/www/shared/.env`. Otherwise, point to your Droplet's local services.

/var/www/shared/.env (example) APP_ENV=production APP_DEBUG=false APP_URL=https://your-domain.com DB_CONNECTION=mysql DB_HOST=your-managed-mysql-do-user-1234.c.db.ondigitalocean.com DB_PORT=25060 DB_DATABASE=app DB_USERNAME=doadmin DB_PASSWORD=super-secret MYSQL_ATTR_SSL_CA=/etc/ssl/certs/ca-certificates.crt CACHE_DRIVER=redis SESSION_DRIVER=redis REDIS_HOST=your-redis-do-1234.db.ondigitalocean.com REDIS_PASSWORD=redis-secret REDIS_PORT=25061 QUEUE_CONNECTION=redis MAIL_MAILER=log

Managed DB/Redis often require SSL and nonstandard ports. Keep `APP_DEBUG=false` to avoid exposing stack traces in production.

6 – Optional: Store Files on DigitalOcean Spaces

Spaces is S3-compatible, so the standard S3 driver works. Great for offloading user uploads and serving via CDN.

```
// config/filesystems.php (snippet)
'disks' => [
  'spaces' => [
    'driver' => 's3',
    'key'    => env('SPACES_KEY'),
    'secret' => env('SPACES_SECRET'),
    'endpoint' => env('SPACES_ENDPOINT',
'https://nyc3.digitaloceanspaces.com'),
    'region' => env('SPACES_REGION', 'nyc3'),
    'bucket' => env('SPACES_BUCKET'),
    'url'    => env('SPACES_CDN_URL'), // e.g., CDN endpoint
    'visibility' => 'public',
  ],
]Code language: PHP (php)
```

Set `FILESYSTEM_DISK=spaces` and configure `SPACES_*` vars. The optional `SPACES_CDN_URL` makes `Storage::url()` return CDN-hosted URLs automatically.

7 – Queues with Horizon (Redis)

Use Horizon for live metrics, auto-balancing, and easier on-call. See Article #45 for the full dashboard walkthrough.

```
composer require laravel/horizon
php artisan horizon:install
php artisan migrateCode language: Bash (bash)
```

This adds Horizon's config and tables. Horizon exposes a dashboard at `/horizon` for job throughput, failures, and retries.

```
# /etc/systemd/system/horizon.service
[Unit]
Description=Laravel Horizon
After=network.target
[Service]
User=www-data
WorkingDirectory=/var/www/current
ExecStart=/usr/bin/php artisan horizon
Restart=always
RestartSec=3
[Install]
WantedBy=multi-user.targetCode language: TOML, also INI (ini)
```

Running Horizon under systemd keeps it alive across crashes and deploys. For queue fundamentals, check Article #42: *How to Use Laravel Queues for Faster Performance*.

8 – Free SSL with Let's Encrypt

Use Certbot's Nginx plugin to get and auto-renew TLS certificates.

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d your-domain.com -d www.your-domain.com
# renewals are installed to crontab/systemd timers automaticallyCode language: Bash (bash)
```

Certbot edits your Nginx config to redirect HTTP→HTTPS and configures auto-renewal,

keeping your TLS valid without manual steps.

9 – Basic Firewall (UFW)

Lock down the Droplet so only SSH and web ports are exposed.

```
sudo ufw allow OpenSSH
sudo ufw allow "Nginx Full"
sudo ufw --force enable
sudo ufw statusCode language: Bash (bash)
```

This opens ports 22, 80, and 443 while blocking everything else by default. Pair with SSH keys (no password auth) for secure access.

10 – Health Check Endpoint (UI)

Add a lightweight endpoint for monitors (or a load balancer later) to validate DB/Redis connectivity quickly.

```
// routes/web.php
Route::get('/health', function () {
    try {
        DB::connection()->getPdo();
        Cache::put('healthcheck', now(), 5);
        return response()->json(['status' => 'ok', 'time' => now()],
200);
    } catch (\Throwable $e) {
```

```
        return response()->json(['status' => 'fail'], 500);
    }
});Code language: PHP (php)
```

The route touches both DB and cache. If either fails, your monitoring will catch it. Keep the logic minimal so checks are fast and reliable.

11 – Zero-Downtime Deployment Script

Use an atomic symlink swap after warming the release (composer/artisan/migrations). This pattern keeps users online during updates.

```
# /usr/local/bin/deploy.sh
set -euo pipefail
APP=/var/www
REL=$APP/releases
NEW=$REL/$(date +%Y%m%d%H%M%S)

mkdir -p "$NEW"
# Upload or fetch the build (choose one)
# rsync -az --delete ./build/ "$NEW/"
git clone --depth=1 https://github.com/you/your-laravel-app.git "$NEW"

cd "$NEW"
ln -sfn /var/www/shared/.env .env
rm -rf storage && ln -s /var/www/shared/storage storage

composer install --no-dev --optimize-autoloader
php artisan config:cache
php artisan route:cache
php artisan view:cache
php artisan migrate --force
```

```
ln -sfn "$NEW" "$APP/current"
sudo systemctl reload php8.3-fpm
sudo systemctl reload nginx
sudo systemctl restart horizon || trueCode language: Bash (bash)
```

The script clones or syncs your build into a timestamped folder, warms caches, applies migrations, then flips `current`. Finally it reloads PHP-FPM/Nginx and restarts Horizon.

Automate this from GitHub Actions for push-button deploys—see CI/CD for Laravel Projects with GitHub Actions (Article #54).

Wrapping Up

You now have a secure, optimized Laravel deployment on DigitalOcean: Nginx + PHP-FPM on a Droplet, Managed DB/Redis, Spaces for files, HTTPS via Let's Encrypt, UFW firewall, Horizon for queues, a health endpoint, and zero-downtime releases. As traffic grows, you can scale vertically (bigger Droplet) or horizontally (multiple Droplets behind a load balancer), and complement with the performance strategies from Article #41.

What's Next

- CI/CD for Laravel Projects with GitHub Actions — automate builds, tests, and zero-downtime deploys.
- Laravel & Nginx: Best Practices for Production — tune timeouts, compression, and buffers.
- Laravel Deployment Checklist for 2025 — run this pre-launch audit before every release.

- Optimizing Laravel for High Concurrency with Octane — supercharge RPS when you need it.