Production Deployment Guide: Laravel + Docker + PostgreSQL + Nginx on a DigitalOcean Ubuntu Droplet

Goal

Deploy a Laravel application to a production Ubuntu VPS (DigitalOcean Droplet) using Docker and Docker Compose, with PostgreSQL as the database and Nginx as the web server, served via PHP-FPM. This guide is beginner-friendly and includes copy/paste commands.

Source: https://chatgpt.com/g/g-p-68d0af7a431c8191927747e5f0a56bb0-laravel/c/68d84f01-ec00-8323-a57a-def5fb0b93cc

Prerequisites

• A DigitalOcean droplet (Ubuntu 22.04 LTS or 24.04 LTS), 1–2 GB RAM minimum. • A domain name you control (optional but recommended for HTTPS). • A Laravel project in a Git repository (or zipped). • A non-root user with sudo (we create it below).

1) Create and Secure Your Droplet

```
# SSH into the droplet as root (replace with your droplet IP)
ssh root@YOUR_DROPLET_IP

# Create a non-root sudo user (replace 'deploy' with your preferred user)
adduser deploy
usermod -aG sudo deploy

# Copy your SSH key to the new user (from your local machine)
ssh-copy-id deploy@YOUR_DROPLET_IP

# Enable basic firewall
ufw allow OpenSSH
ufw allow 80
ufw allow 443
ufw enable
```

2) Install Docker Engine & Docker Compose plugin

We'll use Docker's official repository. Run the following as your sudo user (not root):

```
# Become your non-root user if you are still root
su - deploy

# Required packages
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-release

# Add Docker's official GPG key & repo
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.g
echo   "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]   https://download.
sudo mv /var/tmp/docker.list.tmp /etc/apt/sources.list.d/docker.list
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Install Docker Engine and the Compose CLI plugin
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# Let your user run docker without sudo (log out/in after this)
sudo usermod -aG docker $USER

# Verify
docker --version
docker compose version
```

3) Get Your Laravel App onto the Server

```
# From your home directory
cd ~
# Option A: Clone from Git
git clone https://github.com/you/your-laravel-app.git app
# Option B: Upload a zip and unzip it here, then rename to 'app'
# unzip your-archive.zip -d ./ && mv your-archive app

cd app
cp .env.example .env
# Set APP_ENV=production, APP_DEBUG=false and APP_URL=https://yourdomain.com in .env
# Set database vars to use Postgres service name 'db' (see docker-compose below)
```

4) Project Files You'll Add

Add these files to your repository (or create them on the server).

A) docker-compose.yml

```
services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    image: yourname/laravel-app:prod
    restart: unless-stopped
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    depends_on:
      - db

  nginx:
    image: nginx:1.27-alpine
    restart: unless-stopped
    ports:
      - "80:80"
      # Uncomment after you obtain certificates:
      # - "443:443"
    volumes:
      - ./:/var/www/html
      - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf:ro
      # For Let's Encrypt HTTP challenge and cert files (added later):
      - ./docker/certbot/www:/var/www/certbot
      - ./docker/certbot/conf:/etc/letsencrypt

  db:
    image: postgres:16-alpine
    restart: unless-stopped
    environment:
      POSTGRES_DB: laravel
      POSTGRES_USER: laravel
      POSTGRES_PASSWORD: strong-password-change-me
    volumes:
      - ./docker/postgres/data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U $$POSTGRES_USER"]
      interval: 10s
      timeout: 5s
      retries: 5

# Optional one-off builder for assets (run when needed):
  node:
    image: node:20-alpine
    working_dir: /var/www/html
```

```
    volumes:
      - ./:/var/www/html
    entrypoint: ["sh","-c"]
    command: ["npm ci && npm run build"]

# Optional: certbot (run on demand to get/renew certs)
  certbot:
    image: certbot/certbot
    volumes:
      - ./docker/certbot/conf:/etc/letsencrypt
      - ./docker/certbot/www:/var/www/certbot
    entrypoint: ["/bin/sh","-c"]
    command: ["trap exit TERM; while :; do sleep 6h & wait $${!}; certbot renew; done"]

networks:
  default:
    driver: bridge
```

B) Dockerfile (PHP-FPM + Composer)

```
# docker/Dockerfile or just Dockerfile at project root
FROM php:8.3-fpm-alpine

# System deps
RUN apk add --no-cache bash git unzip libpq icu-dev oniguruma-dev libzip-dev zlib-dev

# PHP extensions
RUN docker-php-ext-install intl mbstring zip pdo_pgsql opcache

# Opcache recommended settings
RUN {   echo 'opcache.enable=1';   echo 'opcache.enable_cli=1';   echo 'opcache.memory_consumption=128';

# Install Composer
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

WORKDIR /var/www/html

# Install PHP dependencies (optimized) – uses cache if composer files unchanged
COPY composer.json composer.lock /var/www/html/
RUN composer install --no-dev --prefer-dist --optimize-autoloader --no-interaction

# Copy the rest of the app
COPY . /var/www/html

# Ensure storage is writable
RUN chown -R www-data:www-data storage bootstrap/cache &&     chmod -R ug+rwx storage bootstrap/cache

USER www-data

CMD ["php-fpm"]
```

C) Nginx vhost: docker/nginx/default.conf

```
server {
    listen 80;
    server_name _;
    root /var/www/html/public;
    index index.php;

    # For Let's Encrypt HTTP-01 challenge
    location ^~ /.well-known/acme-challenge/ {
        root /var/www/certbot;
        default_type "text/plain";
        try_files $uri =404;
```

```
    }

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass app:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
        fastcgi_read_timeout 60s;
    }

    location ~* \.(?:css|js|jpg|jpeg|gif|png|svg|webp|ico|ttf|otf|woff|woff2)$ {
        expires 7d;
        access_log off;
        add_header Cache-Control "public, max-age=604800";
    }

    client_max_body_size 20M;
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
}
```

5) Configure Laravel for Production

```
# Copy .env and set production flags
cp .env.example .env
# Edit .env:
APP_ENV=production
APP_DEBUG=false
APP_URL=https://yourdomain.com
LOG_CHANNEL=stderr

# DB for Docker (service name 'db' from compose)
DB_CONNECTION=pgsql
DB_HOST=db
DB_PORT=5432
DB_DATABASE=laravel
DB_USERNAME=laravel
DB_PASSWORD=strong-password-change-me
```

6) Build & Start Containers

```
# From the project root (where docker-compose.yml lives)
docker compose pull
docker compose build
docker compose up -d

# Install PHP deps (if not done in Dockerfile) and generate app key
docker compose exec app composer install --no-dev --prefer-dist --optimize-autoloader
docker compose exec app php artisan key:generate

# Build frontend assets (if you use Vite)
docker compose run --rm node

# Run database migrations (and seeders if desired)
docker compose exec app php artisan migrate --force
# docker compose exec app php artisan db:seed --force

# Optimize Laravel for production
```

```
docker compose exec app php artisan config:cache
docker compose exec app php artisan route:cache
docker compose exec app php artisan view:cache
```

7) Point Your Domain (Optional, for HTTPS)

Create an A record for your domain (e.g., example.com) pointing to your droplet's public IP. Wait for DNS to propagate (usually minutes).

8) Obtain and Enable HTTPS (Let's Encrypt)

We'll use a certbot container with the webroot method. Replace yourdomain.com and, if needed, add -d www.yourdomain.com.

```
# Create folders for certbot volumes
mkdir -p docker/certbot/www docker/certbot/conf

# Ensure nginx is serving HTTP (port 80) and the /.well-known path
docker compose up -d nginx

# Obtain certificate
docker compose run --rm certbot certbot certonly --webroot   -w /var/www/certbot -d yourdomain.com --agree

# After success, create docker/nginx/default-ssl.conf with TLS server block,
# expose 443 in docker-compose under nginx, then:
docker compose up -d --force-recreate nginx

# Test renewal
docker compose run --rm certbot certbot renew --dry-run
```

9) Useful Maintenance Commands

```
# Check logs
docker compose logs -f nginx
docker compose logs -f app
docker compose logs -f db

# Restart services after config changes
docker compose up -d --force-recreate app nginx

# Update code and rebuild
git pull
docker compose build app
docker compose up -d

# Clear caches (if needed) and rebuild
docker compose exec app php artisan cache:clear
docker compose exec app php artisan config:cache
docker compose exec app php artisan route:cache
docker compose exec app php artisan view:cache

# Database backup (simple example)
docker compose exec -T db pg_dump -U laravel -d laravel > backup_$(date +%F).sql
```

Appendix A — Minimal File Tree

```
app/                 # Your Laravel app
docker-compose.yml
Dockerfile
docker/
  nginx/
    default.conf        # HTTP config (and add default-ssl.conf for HTTPS)
  certbot/
    www/                # ACME webroot
    conf/               # Let's Encrypt cert files
  postgres/
    data/               # Persistent PG data (volume)
```

Appendix B — Example .env (Highlights)

```
APP_NAME="Laravel App"
APP_ENV=production
APP_KEY=base64:GENERATED_BY_KEY_GENERATE
APP_DEBUG=false
APP_URL=https://yourdomain.com

LOG_CHANNEL=stderr

DB_CONNECTION=pgsql
DB_HOST=db
DB_PORT=5432
DB_DATABASE=laravel
DB_USERNAME=laravel
DB_PASSWORD=strong-password-change-me

SESSION_DRIVER=cookie
SESSION_SECURE_COOKIE=true
```

Appendix C — One-time Commands Cheat Sheet

```
# First-time setup
docker compose build
docker compose up -d
docker compose exec app composer install --no-dev --prefer-dist --optimize-autoloader
docker compose exec app php artisan key:generate
docker compose exec app php artisan migrate --force
docker compose run --rm node   # if you use Vite
docker compose exec app php artisan config:cache route:cache view:cache
```