Laravel Production Deployment with Docker, PostgreSQL, and Nginx

Source: https://chat.deepseek.com/a/chat/s/dad474cc-10e5-49ef-8868-d7ace3a17d49

Overview

This guide will walk you through deploying a Laravel application in production mode using Docker and Docker Compose on a DigitalOcean Droplet running Ubuntu.

Prerequisites

- A DigitalOcean account
- Basic knowledge of Linux commands
- A Laravel project ready for deployment
- Domain name (optional but recommended)

Step 1: Set Up DigitalOcean Droplet

1.1 Create a New Droplet

1. Log in to your DigitalOcean dashboard
2. Click "Create" → "Droplets"
3. Choose:
    - **Image**: Ubuntu 22.04 LTS
    - **Plan**: Basic (choose based on your needs)
    - **CPU**: Regular Intel/AMD (start with 1GB RAM)
    - **Datacenter**: Choose closest to your users
    - **Authentication**: SSH keys (recommended) or password
4. Click "Create Droplet"

1.2 Access Your Droplet

Code:

```
ssh root@your_droplet_ip
```

Step 2: Server Initial Setup

2.1 Update System Packages

bash:

```
apt update && apt upgrade -y
```

2.2 Create a Non-root User (Security Best Practice)

bash:

```
adduser deployer
usermod -aG sudo deployer
```

2.3 Set Up Firewall

bash:

```
ufw allow OpenSSH
ufw allow 80
ufw allow 443
ufw enable
```

2.4 Switch to Deployer User

bash:

```
su - deployer
```

Step 3: Install Docker and Docker Compose

3.1 Install Docker

```
bash:
# Add Docker's official GPG key
sudo apt install apt-transport-https ca-certificates curl software-properti
es-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmo
r -o /usr/share/keyrings/docker-archive-keyring.gpg

# Add Docker repository
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.
gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io

# Add user to docker group
sudo usermod -aG docker $USER
newgrp docker
```

3.2 Install Docker Compose

```
bash:
sudo curl -L "https://github.com/docker/compose/releases/download/v2.20.0/d
ocker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

3.3 Verify Installation

```
bash:
docker --version
```

```
docker-compose --version
```

Step 4: Prepare Your Laravel Project

4.1 Project Structure

Create the following structure in your local project:

```text
your-laravel-project/
├── docker/
│   ├── nginx/
│   │   └── default.conf
│   └── php/
│       └── Dockerfile
├── docker-compose.yml
├── .env.production
└── (your laravel files)
```

4.2 Create Dockerfile for PHP-FPM

Create `docker/php/Dockerfile`:

```dockerfile
FROM php:8.2-fpm

# Install system dependencies
RUN apt-get update && apt-get install -y \
    git \
    curl \
    libpng-dev \
    libonig-dev \
    libxml2-dev \
    zip \
    unzip \
    libpq-dev
```

```
# Clear cache
RUN apt-get clean && rm -rf /var/lib/apt/lists/*

# Install PHP extensions
RUN docker-php-ext-install pdo_pgsql pgsql mbstring exif pcntl bcmath gd

# Get latest Composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

# Create user for laravel application
RUN groupadd -g 1000 www
RUN useradd -u 1000 -ms /bin/bash: -g www www

# Set working directory
WORKDIR /var/www

# Copy existing application directory contents
COPY . /var/www

# Copy existing application directory permissions
COPY --chown=www:www . /var/www

# Change current user to www
USER www

# Expose port 9000 and start php-fpm server
EXPOSE 9000
CMD ["php-fpm"]
```

4.3 Create Nginx Configuration

Create `docker/nginx/default.conf`:

```nginx
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;
    root /var/www/public;
    index index.php index.html index.htm;

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

    location ~ \.php$ {
        fastcgi_pass app:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }

    error_log /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
}
```

4.4 Create Docker Compose File

Create `docker-compose.yml`:

```yaml
version: '3.8'

services:
  # PostgreSQL Database
  db:
    image: postgres:15
    container_name: laravel_db
    restart: unless-stopped
    environment:
      POSTGRES_DB: ${DB_DATABASE}
      POSTGRES_USER: ${DB_USERNAME}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - laravel_network

  # PHP-FPM Service
  app:
    build:
      context: .
      dockerfile: docker/php/Dockerfile
    container_name: laravel_app
    restart: unless-stopped
    working_dir: /var/www
    volumes:
      - .:/var/www
    networks:
      - laravel_network
    depends_on:
      - db

  # Nginx Service
  webserver:
    image: nginx:alpine
    container_name: laravel_webserver
    restart: unless-stopped
    ports:
      - "80:80"
    volumes:
      - .:/var/www
      - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
    networks:
      - laravel_network
    depends_on:
      - app

volumes:
  postgres_data:

networks:
  laravel_network:
    driver: bridge
```

4.5 Create Production Environment File

Create `.env.production`:

```env
APP_NAME="Your App Name"
APP_ENV=production
APP_KEY=base64:your_app_key_here
APP_DEBUG=false
APP_URL=http://your-domain.com

DB_CONNECTION=pgsql
DB_HOST=db
DB_PORT=5432
DB_DATABASE=laravel_production
DB_USERNAME=laravel_user
DB_PASSWORD=your_secure_password_here

CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
```

Step 5: Prepare Laravel for Production

5.1 Generate Application Key

```bash
# Locally, in your project directory
php artisan key:generate
```

5.2 Update Laravel Configuration

Ensure your `config/database.php` has PostgreSQL configuration.

5.3 Optimize for Production

```
bash:
# Locally, before deployment
php artisan config:cache
php artisan route:cache
php artisan view:cache
```

Step 6: Deploy to DigitalOcean Droplet

6.1 Transfer Files to Droplet

```
bash:
# From your local machine
scp -r your-laravel-project/ deployer@your_droplet_ip:/home/deployer/
```

6.2 Set Up Project on Droplet

```
bash:
# On the droplet
cd /home/deployer/your-laravel-project

# Copy production environment file
cp .env.production .env

# Set proper permissions
sudo chown -R deployer:deployer /home/deployer/your-laravel-project
sudo chmod -R 755 /home/deployer/your-laravel-project
sudo chmod -R 775 /home/deployer/your-laravel-project/storage
sudo chmod -R 775 /home/deployer/your-laravel-project/bootstrap/cache
```

6.3 Build and Start Containers

```
bash:
docker-compose up -d --build
```

6.4 Run Laravel Setup Commands

```bash
bash:
# Install PHP dependencies
docker-compose exec app composer install --no-dev --optimize-autoloader

# Generate application key (if not set)
docker-compose exec app php artisan key:generate

# Run database migrations
docker-compose exec app php artisan migrate --force

# Cache configuration
docker-compose exec app php artisan config:cache
docker-compose exec app php artisan route:cache
docker-compose exec app php artisan view:cache
```

Step 7: SSL Certificate with Let's Encrypt

7.1 Install Certbot

```bash
bash:
sudo apt install certbot python3-certbot-nginx -y
```

7.2 Get SSL Certificate

```bash
bash:
sudo certbot --nginx -d your-domain.com -d www.your-domain.com
```

7.3 Auto-renewal Setup

```bash
bash:
sudo crontab -e
# Add this line:
0 12 * * * /usr/bin/certbot renew --quiet
```

Step 8: Monitoring and Maintenance

8.1 Check Container Status

```
bash:
docker-compose ps
docker logs laravel_app
docker logs laravel_webserver
```

8.2 Backup Strategy

Create backup script `/home/deployer/backup.sh`:

```
bash:
#!/bin/bash:
DATE=$(date +%Y%m%d_%H%M%S)
docker-compose exec db pg_dump -U laravel_user laravel_production > /home/d
eployer/backups/backup_$DATE.sql
find /home/deployer/backups -name "*.sql" -mtime +7 -delete
```

Step 9: Troubleshooting Common Issues

9.1 Check Logs

```
bash:
docker-compose logs
docker-compose logs app
docker-compose logs webserver
docker-compose logs db
```

9.2 Database Connection Issues

```
bash:
```

```
# Test database connection
docker-compose exec db psql -U laravel_user -d laravel_production
```

9.3 Permission Issues

```
bash:
docker-compose exec app chown -R www:www /var/www/storage
docker-compose exec app chown -R www:www /var/www/bootstrap/cache
```

Step 10: Update Deployment

10.1 Update Process

```
bash:
# Pull latest code
git pull origin main

# Rebuild containers
docker-compose up -d --build

# Run migrations (if any)
docker-compose exec app php artisan migrate --force

# Clear and re-cache
docker-compose exec app php artisan config:cache
docker-compose exec app php artisan route:cache
docker-compose exec app php artisan view:cache
```

Security Considerations

1. **Keep software updated**: Regularly update Ubuntu, Docker, and containers
2. **Use strong passwords**: Especially for database and application keys
3. **Regular backups**: Implement automated backup procedures

4. **Monitor logs**: Set up log monitoring for suspicious activities
5. **Firewall configuration**: Only expose necessary ports
6. **SSL/TLS**: Always use HTTPS in production

Maintenance Tasks

- Regularly update Docker images
- Monitor disk space and logs
- Test backups regularly
- Update Laravel and dependencies
- Review security patches