Budgora Laravel + Docker + PostgreSQL Deployment Guide

Overview

This document describes the exact steps you followed to deploy your Laravel + PostgreSQL application (Budgora) on a DigitalOcean droplet using Docker, Nginx, HTTPS (Let's Encrypt), and Vite-built front-end assets. You can reuse these steps to deploy the same repository on a new droplet.

Prerequisites

- DigitalOcean droplet (Ubuntu)
- Domain pointing to droplet IP (example: budgora.xyz)
- Your Laravel repository URL (Git)
- SSH key pair for secure server access

1. Create droplet and user

1. Create a new droplet on DigitalOcean (Ubuntu).
2. SSH as root:
   ssh root@DROPLET_IP
3. Create a non-root user:
   adduser mohammad
   usermod -aG sudo mohammad
4. Set up SSH key auth for mohammad:
   - Create /home/mohammad/.ssh/authorized_keys
   - Paste your public key
   - chmod 700 ~/.ssh
   - chmod 600 ~/.ssh/authorized_keys

2. Secure SSH and move to port 22022

1. Edit /etc/ssh/sshd_config:
   - Replace or add:
     Port 22022
   - Add user-specific block to disable password for mohammad:
     Match User mohammad
       PasswordAuthentication no
       PubkeyAuthentication yes
2. Allow new SSH port and web ports via UFW:
   sudo ufw allow 22022/tcp
   sudo ufw allow 80/tcp
   sudo ufw allow 443/tcp
   sudo ufw enable
3. Test config and restart SSH:
   sudo sshd -t
   sudo systemctl restart ssh
4. Reconnect as:
   ssh -p 22022 mohammad@DROPLET_IP

3. Install Docker and Docker Compose

1. Update packages:
   sudo apt update
2. Install docker and docker-compose:
   sudo apt install -y docker.io docker-compose
3. Enable docker on boot:
   sudo systemctl enable docker
4. Allow mohammad to use docker without sudo:
   sudo usermod -aG docker mohammad
   (Then log out and log back in.)

4. Create project directory structure
---
1. Create base directory:
   sudo mkdir -p /opt/budgora
   sudo chown -R mohammad:mohammad /opt/budgora
2. Inside /opt/budgora create:
   - webroot/ (Laravel app)
   - docker-compose.yml
   - nginx.conf
   - php.Dockerfile (custom PHP image with PostgreSQL driver)

5. Clone the Laravel repository
---
1. Ensure mohammad owns webroot:
   sudo chown -R mohammad:mohammad /opt/budgora/webroot
2. Clone repo:
   cd /opt/budgora/webroot
   git clone REPO_URL .

6. PHP Dockerfile with PostgreSQL driver

Create /opt/budgora/php.Dockerfile with:

```
FROM php:8.2-fpm-alpine
RUN apk add --no-cache postgresql-dev
RUN docker-php-ext-install pdo pdo_pgsql
```

7. docker-compose.yml

In /opt/budgora/docker-compose.yml define three services:
- php (built from php.Dockerfile)
- nginx (nginx:alpine)
- db (postgres:16-alpine)

Example:

```
version: '3.8'

services:
  php:
    build:
      context: .
      dockerfile: php.Dockerfile
```

```
    container_name: budgora_php
    restart: unless-stopped
    volumes:
      - ./webroot:/var/www/html
    depends_on:
      - db

  nginx:
    image: nginx:alpine
    container_name: budgora_nginx
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - php
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf:ro
      - ./webroot:/var/www/html
      - /etc/letsencrypt:/etc/letsencrypt:ro

  db:
    image: postgres:16-alpine
    container_name: budgora_db
    restart: unless-stopped
    environment:
      POSTGRES_DB: budgora_db
      POSTGRES_USER: budgora_user
      POSTGRES_PASSWORD: StrongPass123
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

8. Nginx configuration for HTTPS Laravel

Create /opt/budgora/nginx.conf:

```
server {
    listen 80;
    server_name budgora.xyz www.budgora.xyz;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name budgora.xyz www.budgora.xyz;

    root /var/www/html/public;
    index index.php index.html;

    ssl_certificate /etc/letsencrypt/live/budgora.xyz/fullchain.pem;
```

```
    ssl_certificate_key /etc/letsencrypt/live/budgora.xyz/privkey.pem;

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

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass php:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }
}
```

9. Obtain SSL certificates with Certbot
---
1. Stop nginx containers if running:
   cd /opt/budgora
   docker-compose down
2. Run certbot (standalone):
   sudo certbot certonly --standalone -d budgora.xyz -d www.budgora.xyz
3. After success, the certs will be at:
   /etc/letsencrypt/live/budgora.xyz/

10. Laravel .env configuration
---
1. Copy example env:
   cd /opt/budgora/webroot
   cp .env.example .env
2. Edit .env:
   APP_URL=https://budgora.xyz
   DB_CONNECTION=pgsql
   DB_HOST=db
   DB_PORT=5432
   DB_DATABASE=budgora_db
   DB_USERNAME=budgora_user
   DB_PASSWORD=StrongPass123
   CACHE_DRIVER=file
   SESSION_DRIVER=file
3. Generate app key:
   cd /opt/budgora
   docker-compose up -d
   docker-compose exec php php /var/www/html/artisan key:generate

11. Synchronize PostgreSQL user password

If needed, inside db container:
   cd /opt/budgora
   docker-compose exec db sh
   psql -U postgres
   ALTER USER budgora_user WITH PASSWORD 'StrongPass123';
   \q
   exit

12. Run migrations and seeders

```
cd /opt/budgora
docker-compose exec php php /var/www/html/artisan migrate:fresh --seed
```

13. Install Node and build Vite assets
---
1. Install nvm and Node 22:
```
cd /opt/budgora/webroot
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.2/install.sh | bash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"
nvm install 22
nvm use 22
```
2. Add swap (to avoid build being killed):
```
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```
3. Install dependencies and build:
```
cd /opt/budgora/webroot
rm -rf node_modules
npm install
npm run build
```

14. Final container restart

```
cd /opt/budgora
docker-compose down
docker-compose up -d
```

At this point:
- Nginx serves the Laravel app at https://budgora.xyz
- PHP-FPM uses PostgreSQL as DB
- Front-end assets are built by Vite and served from public/build