Dockerizing a Laravel App with PHP, MySQL, and Nginx

Here's a complete guide to dockerizing a Laravel application using Docker Compose with PHP, MySQL, and Nginx.

1. Project Structure

First, let's create the following project structure:

```
your-laravel-app/
├── docker/
│   ├── nginx/
│   │   └── default.conf
│   └── php/
│       └── Dockerfile
├── src/ (your Laravel application files)
├── docker-compose.yml
└── .env
```

2. Docker Compose File

Create a `docker-compose.yml` file:

```yaml
version: '3.8'

services:
  app:
    build:
      context: .
      dockerfile: docker/php/Dockerfile
    container_name: laravel_app
    restart: unless-stopped
    working_dir: /var/www
    volumes:
      - ./src:/var/www
    environment:
      - DB_HOST=mysql
      - DB_DATABASE=${DB_DATABASE}
      - DB_USERNAME=${DB_USERNAME}
      - DB_PASSWORD=${DB_PASSWORD}
    depends_on:
      - mysql

  webserver:
    image: nginx:alpine
```

PROF

```
    container_name: laravel_webserver
    restart: unless-stopped
    ports:
      - "8000:80"
    volumes:
      - ./src:/var/www
      - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf

  mysql:
    image: mysql:8.0
    container_name: laravel_mysql
    restart: unless-stopped
    environment:
      - MYSQL_DATABASE=${DB_DATABASE}
      - MYSQL_USER=${DB_USERNAME}
      - MYSQL_PASSWORD=${DB_PASSWORD}
      - MYSQL_ROOT_PASSWORD=${DB_PASSWORD}
    volumes:
      - mysql_data:/var/lib/mysql
    ports:
      - "3306:3306"

volumes:
  mysql_data:
    driver: local
```

3. PHP Dockerfile

Create `docker/php/Dockerfile`:

```
FROM php:8.2-fpm-alpine

WORKDIR /var/www

RUN apk update && apk add \
    build-base \
    vim \
    curl \
    php82 \
    php82-fpm \
    php82-common \
    php82-pdo \
    php82-pdo_mysql \
    php82-mysqli \
    php82-mcrypt \
    php82-mbstring \
    php82-xml \
    php82-openssl \
    php82-json \
    php82-phar \
```

PROF

```
    php82-zip \
    php82-gd \
    php82-dom \
    php82-session \
    php82-zlib

RUN docker-php-ext-install pdo pdo_mysql mysqli

RUN apk add --no-cache zip libzip-dev
RUN docker-php-ext-configure zip
RUN docker-php-ext-install zip

RUN apk add --no-cache jpeg-dev libpng-dev freetype-dev
RUN docker-php-ext-configure gd --with-freetype --with-jpeg
RUN docker-php-ext-install gd

RUN curl -sS https://getcomposer.org/installer | php -- --install-
dir=/usr/local/bin --filename=composer

RUN addgroup -g 1000 -S www && \
    adduser -u 1000 -S www -G www

USER www

COPY --chown=www:www . /var/www

EXPOSE 9000
CMD ["php-fpm"]
```

4. Nginx Configuration

Create `docker/nginx/default.conf`:

PROF

```
server {
    listen 80;
    index index.php index.html;
    server_name localhost;
    error_log  /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
    root /var/www/public;

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

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass app:9000;
        fastcgi_index index.php;
```

```
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

5. Environment Variables

Create or update your `.env` file (both in root and src directory):

```
DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=laravel_user
DB_PASSWORD=secret
```

6. Build and Run the Containers

1. First, make sure you have Docker installed and running on your system.
2. Build and start the containers:

```
docker-compose up -d --build
```

3. Install Laravel dependencies (run inside the app container):

```
docker-compose exec app composer install
```

4. Generate application key:

```
docker-compose exec app php artisan key:generate
```

5. Run migrations (if needed):

```
docker-compose exec app php artisan migrate
```

7. Access the Application

Your Laravel application should now be accessible at:

- Web: http://localhost:8000
- MySQL: localhost:3306 (use credentials from .env)

Additional Commands

- To stop the containers:

```
docker-compose down
```

- To stop and remove volumes (including database data):

```
docker-compose down -v
```

- To view logs:

```
docker-compose logs -f
```

This setup provides a complete development environment for Laravel with PHP 8.2, MySQL 8.0, and Nginx. You can customize the versions and configurations as needed for your specific project requirements.