

Docker Compose – środowisko developerskie PHP od zera

 Piotr Wiśniewski  03.10.2026  386 wyświetleń  3 komentarzy

 Dostępne w językach:

 PL

 EN

[🏠 Start](#) / [Blog](#) / [DevOps](#)

/ Docker Compose – środowisko developerskie PHP od zera

[← Wróć do bloga](#)



Pobierz PDF

Dlaczego Docker?

"U mnie działa" to historia. Docker gwarantuje, że środowisko deweloperskie = środowisko stagingowe = produkcja.

Korzyści:

- **izolacja** – każdy projekt ma własne wersje PHP, PostgreSQL, Node
- **onboarding** – nowy developer: `git clone + docker compose up -d` = gotowe
- **reproducibility** – zawsze ten sam wynik, niezależnie od maszyny

Plik compose.yaml

```
services:

###> php-fpm ###
app:
  build:
    context: .
    dockerfile: docker/php/Dockerfile
  volumes:
    - ../var/www/html:cached
  depends_on:
    database:
      condition: service_healthy
  environment:
    APP_ENV: dev

###> nginx ###
nginx:
  image: nginx:1.25-alpine
  ports:
    - "8080:80"
  volumes:
    - ../var/www/html:ro
    - ./docker/nginx/default.conf:/etc/nginx/conf:ro
  depends_on:
    - app

###> postgresql ###
database:
  image: postgres:16-alpine
  environment:
    POSTGRES_DB: ${POSTGRES_DB:-app}
    POSTGRES_USER: ${POSTGRES_USER:-app}
    POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:-secret}
  volumes:
    - db_data:/var/lib/postgresql/data
  healthcheck:
    test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER} -d ${POSTGRES_DB}"]
    interval: 10s
```

```
        timeout: 5s
        retries: 5

###> mailpit (przechwytuje maile na dewie) ###
mailer:
  image: axllent/mailpit:latest
  ports:
    - "8025:8025"    # UI
    - "1025:1025"    # SMTP

volumes:
  db_data:
```

Dockerfile dla PHP

```
FROM php:8.3-fpm-alpine

RUN apk add --no-cache \
    git \
    unzip \
    libpq-dev \
    && docker-php-ext-install \
    pdo pdo_pgsql \
    opcache

COPY --from=composer:2 /usr/bin/composer /usr/bin/composer

WORKDIR /var/www/html

COPY . .

RUN composer install --no-dev --optimize-autoloader
```

Codzienne komendy

```
# start w tle
docker compose up -d

# zatrzymanie
docker compose down

# logi (na żywo)
docker compose logs -f app

# wejście do kontenera
docker compose exec app bash

# uruchomienie komend Symfony
docker compose exec app php bin/console cache:clear
docker compose exec app php bin/console doctrine:migrations:migrate

# rebuild po zmianie Dockerfile
docker compose build --no-cache app
```

Nginx — konfiguracja dla Symfony

```
server {
    listen 80;
    root /var/www/html/public;

    location / {
        try_files $uri /index.php$is_args$args;
    }

    location ~ ^/index\.php(/|$) {
        fastcgi_pass app:9000;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        fastcgi_param DOCUMENT_ROOT $realpath_root;
        internal;
    }
}
```

```
location ~ /\.php$ {  
    return 404;  
}  
}
```

Healthcheck – po co?

Bez `healthcheck` kontenery mogą startować w złej kolejności:

app startuje → próbuje połączyć się z bazą → baza :

Z `condition: service_healthy` w `depends_on`, app poczeka na `database`. Zawsze dodawaj healthcheck do bazy danych.

Tagi:

#PHP

#Docker

#Linux

Udostępnij:

 Facebook

 X

 LinkedIn

 Kopiuż link



Komentarze (3)

A

Anna Nowak 21.10.2026, 14:27

Healthcheck dla bazy to must-have. Bez niego migracje startują za wcześnie.

J

Jan Kowalski 21.10.2026, 14:27

Jak masz zoptymalizowany Dockerfile? Mój rebuild po każdej zmianie trwa 3 min.

P

Piotr Wiśniewski 21.10.2026, 14:27

Przenieś COPY composer.json przed RUN composer install – cache warstw pomoże.

[Zaloguj się](#), aby dodać komentarz.

FactoryCode

Projekt demonstracyjny zbudowany na Symfony 8, PHP 8.4 i PostgreSQL 16.
Tłumaczenia zarządzane przez panel admina z AI.

NAWIGACJA

-  Start
-  Blog
-  Projekty
-  Kontakt
-  Polityka prywatności

KONTAKT

-  kontakt@factorycode.pl
-  GitHub