Passage en production Symfony avec Docker

1-Différences développement / production:

Développement	Production
Symfony CLI (symfony serve)	Serveur web (Nginx + PHP-FPM)
Code monté en volume	Code **copié dans l'image**
Debug activé	Debug désactivé
.env.local	.env.prod
phpMyAdmin exposé	❌ phpMyAdmin

*"En production, **on ne développe pas**, on **exécute**."*

2-Conteneurs attendus (prod):

1. **php → PHP-FPM + Symfony**
2. **nginx → serveur web**
3. **db → MySQL**

3-Variables d'environnement production (.env.prod):

```
APP_ENV=prod
APP_DEBUG=0
DATABASE_URL=mysql://symfony:symfony@db:3306/symfony?serverVersion=8.0
```

4-Variables d'environnement production DB (.env.prod.prod):

```
MYSQL_ROOT_PASSWORD: root
MYSQL_DATABASE: symfony
MYSQL_USER: symfony
MYSQL_PASSWORD: symfony
```

5-Dockerfile pour nginx (docker/prod/Dockerfile.nginx)

```
FROM nginx:stable
```

6-Dockerfile multi-stage de production (docker/prod/Dockerfile)

```
# ---------- Étape 1 : build ----------
FROM php:8.2-fpm AS build
RUN apt-get update && apt-get install -y \
    git unzip libzip-dev libpng-dev libonig-dev libxml2-dev \
    && docker-php-ext-install pdo pdo_mysql zip

# Installer Composer
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer
WORKDIR /app

#  Installer les dépendances
COPY composer.json composer.lock ./
RUN composer install --no-dev --optimize-autoloader --no-scripts
# Copier le code source
COPY . .
RUN composer dump-autoload --classmap-authoritative

#  Créer et installer les assets
RUN mkdir -p public/bundles
RUN php bin/console assets:install public --env=prod --no-debug
RUN php bin/console asset-map:compile --env=prod --no-debug

# Gérer le cache
RUN php bin/console cache:clear --env=prod --no-debug
RUN php bin/console cache:warmup --env=prod --no-debug

# ---------- Étape 2 : runtime ----------
FROM php:8.2-fpm
RUN apt-get update && apt-get install -y \
    libzip-dev libpng-dev libonig-dev libxml2-dev \
    && docker-php-ext-install pdo pdo_mysql zip
WORKDIR /var/www/html

# Copier depuis l'étape de build
COPY --from=build /app .

# Vérifier les bundles
RUN ls -la /var/www/html/public/ && \
    (ls -la /var/www/html/public/bundles/ || echo "WARNING: bundles directory not
found")

# Changer les permissions APRÈS la copie
RUN chown -R www-data:www-data var
RUN chmod -R 755 public

USER www-data
EXPOSE 9000
CMD ["php-fpm"]
```

7-Docker Compose de production (docker-compose.prod.yml):

```
services:
  php:
    build:
      context: ./demo
      dockerfile: ../docker/prod/Dockerfile
    container_name: symfony_php_prod
    env_file:
      - ./demo/.env.prod
    volumes:
      - php_public_data:/var/www/html/public
    depends_on:
      - db
  nginx:
    build:
      context: .
      dockerfile: docker/prod/Dockerfile.nginx
    container_name: symfony_nginx_prod
    ports:
      - "80:80"
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
      - php_public_data:/var/www/html/public
    depends_on:
      - php

  db:
    image: mysql:8.0
    container_name: symfony_db_prod
    env_file:
      - ./demo/.env.prod.db
    volumes:
      - db_data:/var/lib/mysql
volumes:
  db_data:
  php_public_data:
```

8-Configuration Nginx (nginx/default.conf)

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

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

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

    location ~* \.(css|js|jpg|jpeg|png|gif|ico|svg|woff|woff2|ttf|eot)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        try_files $uri =404;
    }
}
```

9-Commandes de déploiement

```
docker compose -f docker-compose.prod.yml build
docker compose -f docker-compose.prod.yml up -d
```

Créer le schéma Doctrine

```
docker exec symfony_php_prod php bin/console doctrine:schema:create
```

10-Bonnes pratiques de production

❌ pas de symfony serve
❌ pas de mode debug
✅ images légères
✅ multi-stage build
✅ cache Symfony warmé