Docker Compose — ????????? docker-compose.yml

Эта страница объясняет структуру файла `docker-compose.yml` и основные параметры, используемые при описании сервисов Docker. Здесь разобраны ключевые разделы конфигурации, такие как `services`, `volumes`, `networks`, `environment`, `ports` и другие. Материал рассчитан на начинающих и помогает понять, **как правильно описывать инфраструктуру приложения в Docker Compose**.

??? ????? docker-compose.yml

`docker-compose.yml` — это файл конфигурации, который описывает:

- какие контейнеры нужно запустить
- какие образы использовать
- какие порты открыть
- какие volumes подключить
- какие сети создать

Compose позволяет запускать всё приложение **одной командой**:

```
docker compose up -d
```

??????? ????????? docker-compose.yml

Пример минимального файла:

```
version: "3.9"

services:
  nginx:
```

```
    image: nginx:1.25
    ports:
      - "8080:80"
```

Этот файл создаёт контейнер nginx и открывает порт `8080`.

???????? ???????? docker-compose.yml

Типичная структура файла выглядит так:

```
services:
volumes:
networks:
```

?????? services

Раздел `services` описывает контейнеры проекта.

Пример:

```
services:
  nginx:
    image: nginx:1.25

  php:
    image: php:8.2-fpm
```

В этом примере создаются два сервиса:

- nginx
- php

Каждый сервис становится отдельным контейнером.

???????? image

Определяет Docker-образ, который будет использоваться.

Пример:

```
services:
  nginx:
    image: nginx:1.25
```

Docker скачает образ `nginx:1.25`, если он отсутствует.

???????? build

Используется, если образ нужно **собрать локально из Dockerfile**.

Пример:

```
services:
  app:
    build: .
```

Это означает:

```
собрать образ из Dockerfile текущей директории
```

Можно указать путь:

```
services:
  app:
    build: ./app
```

???????? ports

Используется для проброса портов.

Пример:

```
ports:
  - "8080:80"
```

Это означает:

```
сервер:8080 → контейнер:80
```

После запуска приложение будет доступно:

```
http://server:8080
```

???????? volumes

Подключает volumes или папки сервера.

Пример:

```
volumes:
  - ./src:/var/www/html
```

Это означает:

```
сервер ./src → контейнер /var/www/html
```

????????????? Docker Volumes

Пример с volume:

```
services:
  mysql:
    image: mysql:8
    volumes:
      - mysql_data:/var/lib/mysql
```

```
volumes:
  mysql_data:
```

Volume будет хранить данные базы.

???????? environment

Передаёт переменные окружения контейнеру.

Пример:

```
environment:
  MYSQL_ROOT_PASSWORD: secret
  MYSQL_DATABASE: app_db
```

Контейнер получит переменные:

```
MYSQL_ROOT_PASSWORD
MYSQL_DATABASE
```

????????????? .env ?????

Переменные можно хранить в `.env`.

Пример:

```
DB_PASSWORD=secret
APP_ENV=production
```

И использовать в Compose:

```
environment:
  DB_PASSWORD: ${DB_PASSWORD}
```

???????? depends_on

Определяет зависимости между сервисами.

Пример:

```
services:
  php:
    depends_on:
      - mysql
```

Это означает:

```
сначала запускается mysql
затем php
```

> **Важно:** `depends_on` не гарантирует, что сервис полностью готов к работе.

???????? restart

Определяет политику перезапуска контейнера.

Пример:

```
restart: always
```

Возможные значения:

Значение	Поведение
no	не перезапускать
always	всегда перезапускать
on-failure	только при ошибке
unless-stopped	перезапускать, пока не остановлен вручную

???????? networks

Определяет сети Docker.

Пример:

```
services:
  app:
    networks:
      - backend

networks:
  backend:
```

Контейнер будет подключён к сети `backend`.

?????? ???????? docker-compose.yml

Пример простого веб-приложения:

```
services:

  nginx:
    image: nginx:1.25
    ports:
      - "80:80"
    volumes:
      - ./src:/var/www/html

  php:
    image: php:8.2-fpm
    volumes:
      - ./src:/var/www/html
    depends_on:
      - mysql

  mysql:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: root
```

```
      MYSQL_DATABASE: app
    volumes:
      - mysql_data:/var/lib/mysql

volumes:
  mysql_data:
```

Этот файл запускает:

- nginx
- php
- mysql

???????? docker-compose.yml

Перед запуском полезно проверить конфигурацию:

```
docker compose config
```

Команда покажет итоговую конфигурацию после обработки переменных.

?????? ???????

Запуск всех сервисов:

```
docker compose up -d
```

????????? ???????

```
docker compose down
```

???????? ?????? ? docker-compose.yml

?????? YAML

Файл YAML чувствителен к отступам.

Неправильно:

```
services:
nginx:
image: nginx
```

Правильно:

```
services:
  nginx:
    image: nginx
```

???????? ?????

Если порт уже используется:

```
Bind for 0.0.0.0:80 failed
```

Нужно изменить порт.

?????????? volumes

Если данные не вынесены в volumes, они могут потеряться при пересоздании контейнера.

??????? ???????

```
services     → контейнеры проекта
image        → используемый образ
build        → сборка образа
ports        → проброс портов
volumes      → подключение хранилища
environment  → переменные окружения
depends_on   → зависимости сервисов
restart      → политика перезапуска
networks     → сети контейнеров
```

Revision #1
Created 2026-03-11 08:29:02 UTC by Crimson
Updated 2026-03-11 08:29:25 UTC by Crimson