Compose — ?????????? ?????????

Эта страница объясняет базовые команды `docker compose`, которые используются для запуска, остановки, пересборки и перезапуска сервисов в проекте. Материал рассчитан на начинающих: здесь разобрано, что делает каждая команда, в каких случаях её использовать и на что обратить внимание, чтобы случайно не сломать рабочее окружение.

??? ????? Docker Compose

`Docker Compose` — это инструмент для управления несколькими контейнерами как единым приложением. Обычно в проекте есть файл `docker-compose.yml`, где описано:

- какие сервисы нужно запускать
- какие образы использовать
- какие порты пробрасывать
- какие папки подключать как volumes
- какие переменные окружения передавать контейнерам

Например, один проект может состоять из:

- `nginx`
- `php`
- `mysql`
- `redis`

Вместо того чтобы запускать каждый контейнер отдельно длинными командами `docker run`, Compose позволяет поднять всё одной командой.

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

Обычно работа с Compose ведётся из директории проекта, где лежит `docker-compose.yml`.

Пример:

```
cd /opt/flamy_projects/project_name
```

Проверить, что файл действительно есть в текущей папке:

```
ls -la
```

Если нужно найти compose-файлы во всей папке проектов:

```
find /opt/flamy_projects -type f -name "docker-compose*.yml"
```

> **Примечание:** Большинство команд `docker compose` нужно выполнять именно в той директории, где находится `docker-compose.yml`. Иначе Docker не поймёт, с каким проектом работать.

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

Чаще всего цикл работы выглядит так:

1. перейти в папку проекта
2. запустить сервисы
3. проверить, что контейнеры работают
4. посмотреть логи при ошибках
5. при необходимости пересобрать или перезапустить сервисы

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

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

```
docker compose up -d
```

Эта команда:

- читает `docker-compose.yml`
- создаёт нужные контейнеры
- запускает их
- оставляет терминал свободным для дальнейшей работы

Флаг `-d` означает **detached mode**, то есть запуск в фоне.

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

Подходит почти для любого обычного старта проекта:

- после перезагрузки сервера

- после первого клонирования проекта
- после остановки контейнеров
- после мелких изменений конфигурации

??????

```
cd /opt/flamy_projects/site1
docker compose up -d
```

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

Если образа ещё нет локально, Docker попытается:

- скачать его из реестра
- либо собрать локально, если это описано в конфигурации

> **Для новичков:** `up` не всегда означает просто «включить». Эта команда может не только запускать, но и создавать контейнеры заново, если это требуется по конфигурации.

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

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

```
docker compose down
```

Команда останавливает и удаляет:

- контейнеры проекта
- созданные Compose-сети

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

Подходит, если нужно:

- полностью выключить проект
- освободить ресурсы
- заново поднять окружение
- сбросить сетевые проблемы внутри проекта

??? ??????? **?? ???????**

По умолчанию `docker compose down` **не удаляет volumes**. Это важно, потому что именно в volumes часто лежат данные:

- база данных
- загруженные файлы
- кэш приложения
- постоянные данные сервисов

> **Важно:** Для новичков это одна из самых полезных вещей: обычный `down` не стирает данные сам по себе. Но всё равно всегда нужно понимать, где именно проект хранит данные — в volumes, bind mounts или внутри контейнера.

??????

```
docker compose down
```

После этого проект можно снова поднять:

```
docker compose up -d
```

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

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

```
docker compose up -d --force-recreate
```

Эта команда заставляет Docker пересоздать контейнеры, даже если он считает, что ничего не поменялось.

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

Полезно, если:

- контейнер ведёт себя странно
- изменились переменные окружения
- конфигурация применена не так, как ожидалось
- нужен «чистый» перезапуск контейнера без удаления volumes

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

- контейнеры будут пересозданы

- volumes останутся на месте
- данные в volumes не пропадут

> **Пояснение:** Контейнер и volume — это не одно и то же. Контейнер — это «запущенная оболочка приложения». Volume — это место, где могут лежать постоянные данные. Поэтому пересоздание контейнера не обязательно означает потерю данных.

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

Пересборка нужна тогда, когда меняется не только запуск контейнера, а именно его образ. Обычно это происходит, если вы изменили:

- `Dockerfile`
- системные пакеты
- зависимости приложения
- шаги сборки

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

```
docker compose build --no-cache
```

Docker при сборке старается использовать кэш, чтобы не выполнять одинаковые шаги повторно. Это ускоряет работу, но иногда мешает, если нужен действительно «чистый» rebuild.

Флаг `--no-cache` говорит: **не использовать ранее сохранённые слои**, а собрать всё заново.

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

Подходит, если:

- изменения в Dockerfile «не подхватываются»
- обновились зависимости
- есть подозрение на устаревший build cache
- нужно убедиться, что образ собран полностью с нуля

??????

```
docker compose build --no-cache
docker compose up -d
```

> **Примечание:** Эта операция может занять заметно больше времени, чем обычная сборка.

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

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

Команда делает два действия сразу:

1. при необходимости пересобирает образ
2. запускает контейнеры

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

Это один из самых частых сценариев после изменения:

- `Dockerfile`
- зависимостей
- конфигурации сборки

??????

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

??? ?????????? ?? `build --no-cache`

- `docker compose up -d --build` может использовать кэш
- `docker compose build --no-cache` собирает всё полностью заново

Если нужно просто обновить контейнер после обычных изменений — чаще хватает:

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

Если есть сомнения в корректности кэша — используйте:

```
docker compose build --no-cache
docker compose up -d
```

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

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

```
docker compose pull
```

Команда скачивает новые версии образов, которые указаны в `docker-compose.yml`.

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

Подходит, если проект использует готовые образы из registry, например:

- `nginx:latest`
- `redis:7`
- `mysql:8`

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

`docker compose pull` **только скачивает образы**, но **не перезапускает контейнеры автоматически**.

Чтобы применить обновление, обычно выполняют:

```
docker compose pull && docker compose up -d
```

??????

```
cd /opt/flamy_projects/site1
docker compose pull
docker compose up -d
```

> **Важно:** На production-серверах обновление образов нужно делать осторожно. Новая версия образа может изменить поведение приложения или сломать совместимость.

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

Иногда контейнер не нужно пересобирать или пересоздавать — достаточно просто перезапустить процесс.

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

```
docker compose restart <service>
```

Пример:

```
docker compose restart nginx
```

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

Подходит, если:

- сервис завис
- нужно перечитать конфигурацию
- произошёл временный сбой
- после некоторых изменений достаточно обычного рестарта

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

```
docker compose restart
```

Перезапускает все сервисы, описанные в compose-файле.

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

Подходит, если:

- нужно быстро перезапустить весь проект
- нет уверенности, какой именно сервис работает некорректно
- нужно «освежить» всё окружение без удаления контейнеров

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

```
docker compose stop <service> && docker compose start <service>
```

Пример:

```
docker compose stop php && docker compose start php
```

????? ??? ?????, ???? ???? `restart`

Иногда удобнее разделить действия на две части:

- сначала корректно остановить контейнер

- затем отдельно запустить снова

Это бывает полезно при диагностике или когда нужно убедиться, что сервис полностью остановился.

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

```
docker compose up -d <service>
```

Пример:

```
docker compose up -d nginx
```

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

Полезно, если:

- нужно поднять только один сервис
- вы тестируете конкретный контейнер
- не хотите трогать всё окружение

> **Примечание:** Если сервис зависит от других, Compose может поднять и их тоже.

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

?????????? ???????? compose-??????

```
docker compose config
```

Эта команда показывает итоговую конфигурацию после обработки:

- `docker-compose.yml`
- override-файлов
- переменных окружения
- подстановок из `.env`

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

Очень полезно перед запуском, если:

- не подставляются env-переменные
- есть сомнения в корректности конфигурации
- нужно понять, что Docker реально «видит»

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

Иногда в YAML всё выглядит правильно, но итоговая конфигурация получается другой. `docker compose config` помогает увидеть конечный результат до запуска.

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

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

```
docker compose up -d
```

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

```
docker compose pull && docker compose up -d
```

????? ????????? Dockerfile ??? ????????????

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

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

```
docker compose build --no-cache
docker compose up -d
```

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

```
docker compose restart nginx
```

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

```
docker compose down
```

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

????????: ????? ?????? ????????? ??????

Используйте:

```
docker compose up -d
```

????????: ????????? Dockerfile

Используйте:

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

????????: ????? ????? ???? ???????, ????? ?????? rebuild

Используйте:

```
docker compose build --no-cache
docker compose up -d
```

????????: ????? ??????? ?????? ?????? ?? registry

Используйте:

```
docker compose pull && docker compose up -d
```

????????: ?????? ?????? ????? ????????????

Используйте:

```
docker compose restart <service>
```

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

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

Если выполнить `docker compose up -d` не там, где лежит `docker-compose.yml`, можно получить ошибку о том, что конфигурационный файл не найден.

???????? ????? `restart`, `up`, `build` ? `down`

Кратко:

- `restart` — просто перезапуск уже существующих контейнеров
- `up` — поднять проект
- `up --build` — поднять проект с пересборкой
- `down` — остановить и удалить контейнеры проекта

????????, ??? `pull` ??? ??????? ?????????? ??????????

`docker compose pull` только скачивает новый образ. Чтобы контейнеры начали работать на новой версии, нужен ещё запуск:

```
docker compose up -d
```

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

```
docker compose up -d                    # Запуск проекта в фоне
docker compose down                     # Остановка и удаление контейнеров проекта
docker compose up -d --force-recreate   # Пересоздать контейнеры принудительно
docker compose build --no-cache         # Полная пересборка без кэша
docker compose up -d --build            # Пересборка и запуск
docker compose pull                     # Скачать свежие образы
docker compose restart <service>        # Перезапуск одного сервиса
docker compose restart                  # Перезапуск всех сервисов
docker compose stop <service> && docker compose start <service>  # Остановить и запустить
сервис отдельно
docker compose up -d <service>          # Запуск одного сервиса
docker compose config                   # Проверка итоговой конфигурации
```

Revision #5
Created 2026-03-11 07:20:56 UTC by Crimson
Updated 2026-03-13 20:12:41 UTC by Crimson