????? 6 — ?????? ? ??????????? ?????????

- Docker — Production Deployment Workflow
- Docker — Backup (Резервное копирование данных)

Docker — Production Deployment Workflow

Эта страница описывает рекомендуемый процесс обновления и развёртывания Docker-проектов на production-сервере. Здесь показан безопасный порядок действий при обновлении кода, образов и контейнеров, а также базовые проверки после деплоя. Материал ориентирован на начинающих и помогает выполнять обновления **предсказуемо и без случайных простоев сервисов**.

??? ????? Deployment Workflow

Deployment Workflow — это **последовательность действий**, которую нужно выполнять при обновлении проекта.

Без чёткой процедуры часто возникают проблемы:

- контейнеры запускаются со старыми образами
- сервисы падают после обновления
- изменения не применяются
- ломается база данных
- проект уходит в простой

Поэтому важно использовать **одинаковый и проверенный порядок действий**.

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

Пример директории проекта:

```
/opt/flamy_projects/site1


docker-compose.yml
.env
```

```
Dockerfile
src/
```

Где:

Файл	Назначение
docker-compose.yml	конфигурация сервисов
Dockerfile	инструкция сборки образа
.env	переменные окружения
src	исходный код приложения

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

??? 1 — ??????? ? ?????????? ???????

```
cd /opt/flamy_projects/site1
```

Проверить файлы:

```
ls -la
```

Важно убедиться, что вы находитесь **в правильной директории проекта**.

??? 2 — ???????? ??? ???????

Если используется Git:

```
git pull
```

Команда скачивает последние изменения из репозитория.

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

- обновление приложения
- исправления ошибок
- новые функции

??? 3 — ???????? Docker-??????

Если проект использует готовые образы из registry, нужно загрузить новые версии.

```
docker compose pull
```

Команда скачает новые версии образов, указанных в `docker-compose.yml`.

??? 4 — ???????????? ?????????? (???? ??????????)

Если изменились:

- Dockerfile
- зависимости
- системные пакеты

нужно пересобрать образ.

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

Эта команда:

1. пересоберёт образы
2. обновит контейнеры
3. запустит сервисы

??? 5 — ????????? ????????? ?????????

```
docker compose ps
```

Пример результата:

```
NAME          STATUS
site1_nginx   Up
site1_php     Up
site1_mysql   Up
```

Все сервисы должны иметь статус **Up**.

??? 6 — ????????? ????

После обновления важно убедиться, что сервисы работают корректно.

```
docker compose logs --tail=50
```

В логах не должно быть:

- ошибок запуска
- ошибок подключения к базе
- критических ошибок приложения

??? 7 — ????????? ?????? ??????????

После деплоя рекомендуется проверить:

- доступность сайта
- работу API
- соединение с базой данных

- фоновые задачи (workers, cron)

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

На практике часто используется короткая команда:

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

Она:

1. скачивает новые образы
2. обновляет контейнеры

Это самый быстрый способ обновить сервис.

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

Если нужно полностью пересобрать контейнеры:

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

Это полезно, если:

- изменения не применяются
- образ собран неправильно
- нужно выполнить чистую сборку

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

Иногда требуется полностью остановить и заново запустить сервисы.

```
docker compose down
docker compose up -d
```

Это может помочь при:

- сетевых проблемах
- зависших контейнерах
- некорректном состоянии сервисов

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

Если проблема только в одном сервисе:

```
docker compose restart nginx
```

Это быстрее и безопаснее, чем перезапуск всего проекта.

????? ?????? (Rollback)

Если после обновления возникли проблемы, можно откатиться к предыдущей версии.

Пример:

```
git checkout <previous_commit>
docker compose up -d --build
```

Это восстановит предыдущую рабочую версию приложения.

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

После обновления рекомендуется выполнить несколько проверок.

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

```
docker ps
```

Это покажет все работающие контейнеры.

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

```
docker stats
```

Позволяет убедиться, что сервисы не перегружают сервер.

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

```
docker network ls
```

Если сервисы не взаимодействуют — возможно проблема в сети.

???????? production workflow

Чаще всего используется следующий порядок действий:

```
cd /opt/flamy_projects/site1

git pull
docker compose pull
docker compose up -d --build

docker compose ps
docker compose logs --tail=50
```

Этот процесс:

1. обновляет код
2. обновляет образы
3. пересобирает контейнеры
4. проверяет состояние сервисов

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

??????? ???? pull

Если не выполнить `docker compose pull`, контейнеры могут запускаться со старыми образами.

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

Docker Compose работает с файлом `docker-compose.yml`, поэтому важно находиться в нужной папке.

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

После деплоя всегда нужно проверять логи.

????????????? latest ? production

Лучше указывать фиксированные версии образов:

```
nginx:1.25
```

Это делает деплой более предсказуемым.

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

```
cd /opt/flamy_projects/project_name

git pull

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

```
docker compose ps
docker compose logs --tail=50

docker compose restart <service>
docker compose down && docker compose up -d
```

????

Правильный workflow обновления Docker-проектов позволяет:

- безопасно обновлять сервисы
- быстро диагностировать проблемы
- минимизировать время простоя

Использование чёткой процедуры деплоя помогает поддерживать стабильную работу production-систем и упрощает администрирование Docker-инфраструктуры.

Docker — Backup (????????? ???????????? ??????)

Эта страница описывает способы резервного копирования данных Docker-проектов. Здесь разобраны методы backup контейнерных данных, volumes и баз данных, а также способы восстановления данных. Материал рассчитан на начинающих и помогает понять, **как защитить данные Docker-приложений от потери при сбоях, обновлениях или ошибках администрирования**.

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

Docker-контейнеры можно легко пересоздать, но **данные приложения могут быть потеряны**, если они не вынесены в volumes или не сохраняются отдельно.

Резервное копирование особенно важно для:

- баз данных
- пользовательских файлов
- загрузок
- конфигураций приложения
- логов

Даже небольшая ошибка (например, удаление volume) может привести к потере данных.

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

В Docker-проектах обычно нужно делать backup:

Тип данных	Пример
Базы данных	MySQL, PostgreSQL
Docker volumes	данные контейнеров
Файлы приложения	uploads, storage

Тип данных	Пример
Конфигурации	.env, config

Backup ???? ?????? ?? ??????????

Самый распространённый способ — использовать встроенные утилиты базы данных.

Backup MySQL

```
docker compose exec mysql mysqldump -u root -p database_name > backup.sql
```

Эта команда:

1. подключается к контейнеру MySQL
2. делает дамп базы
3. сохраняет файл `backup.sql`

Backup PostgreSQL

```
docker compose exec postgres pg_dump -U postgres database_name > backup.sql
```

Backup Docker Volume

Если данные хранятся в volume, можно сделать архив.

???????? backup volume

```
docker run --rm \
  -v volume_name:/data \
  -v $(pwd):/backup \
  alpine tar czf /backup/volume_backup.tar.gz /data
```

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

Параметр	Назначение
-v volume_name:/data	подключает volume
-v $(pwd):/backup	папка для сохранения
tar czf	создаёт архив

В результате появится файл:

```
volume_backup.tar.gz
```

????????????? volume

Чтобы восстановить данные из backup:

```
docker run --rm \
  -v volume_name:/data \
  -v $(pwd):/backup \
  alpine tar xzf /backup/volume_backup.tar.gz -C /
```

Данные будут восстановлены в volume.

Backup ?????????? ???????

Иногда нужно сохранить весь Docker-проект.

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

```
tar -czf project_backup_$(date +%Y%m%d).tar.gz /opt/flamy_projects/site1
```

Архив будет содержать:

- docker-compose.yml
- Dockerfile
- исходный код
- конфигурации

Backup ?????????????? ??????

Если приложение хранит файлы в директории:

```
/var/www/storage
```

Можно сделать backup так:

```
tar -czf storage_backup.tar.gz storage/
```

Backup ????? rsync

`rsync` часто используется для копирования backup на другой сервер.

```
rsync -avz /opt/flamy_projects user@backup-server:/backup/docker_projects
```

Это позволяет хранить резервные копии **на отдельном сервере**.

????????????? backup (cron)

Backup можно запускать автоматически.

Пример cron-задачи:

```
crontab -e
```

Добавить строку:

```
0 3 * * * /usr/local/bin/docker_backup.sh
```

Это будет запускать backup каждый день в 03:00.

?????? ????????? backup-???????

```
#!/bin/bash

DATE=$(date +%Y%m%d)

docker compose exec mysql mysqldump -u root -psecret db > backup_$DATE.sql

tar -czf docker_backup_$DATE.tar.gz /opt/flamy_projects/site1
```

Скрипт делает:

- дамп базы
- архив проекта

???????? backup

Очень важно проверять резервные копии.

Проверить архив:

```
tar -tzf backup.tar.gz
```

Проверить SQL-дамп:

```
head backup.sql
```

???????? ?????? ??? backup

Backup ?????? ???????????

Контейнеры легко пересоздаются, поэтому важно сохранять **данные**, а не контейнеры.

?????????? ????????????????? backup

Ручной backup часто забывают делать.

Лучше использовать cron.

???????? backup ?? ??? ?? ???????

Если сервер выйдет из строя, backup тоже может быть потерян.

Лучше хранить копии:

- на другом сервере
- в облаке
- в S3-хранилище

????????????? ?????????? backup

Минимальная стратегия:

- ежедневный backup базы данных
- ежедневный backup volumes
- хранение копий минимум 7 дней

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

```
docker compose exec mysql mysqldump > backup.sql   # backup MySQL
docker compose exec postgres pg_dump > backup.sql  # backup PostgreSQL

docker run -v volume:/data alpine tar czf backup.tar.gz /data   # backup volume

tar -czf project_backup.tar.gz project/            # архив проекта
rsync -avz project backup-server:/backup           # копирование backup
```