Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Docker на Linux: наш продакшн-стандарт внедрения

Engine 29.7, Compose v5.5, тома, healthcheck, лимиты и ротация логов на узлах клиентов

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Клиент держит 8–15 сервисов на одном VPS: версии PHP, Python и libc конфликтуют, обновление одного ломает соседний, откат — ручная переустановка на полдня. Закрываем это контейнеризацией: окружение — в образе, состояние — в named volume, порядок старта — в Compose по healthcheck. Иначе релиз остаётся лотереей, а восстановление после сбоя — часы простоя бухгалтерии и склада.

Почему это важно бизнесу

- Простой основного сервиса на 4 часа для офиса из 40 мест — это сорванные отгрузки и вставшая касса, а не «неудобство ИТ».
- Откат релиза сводится к запуску предыдущего тега образа — минуты вместо ночной переустановки сервера с нуля.
- Плотность 10–15 сервисов на одном VPS убирает счёт за 3–4 отдельные виртуалки и их обслуживание.
- Подрядчик получает compose-файл и работает в своём контуре, не имея доступа к продакшн-хосту клиента.
- Уход разработчика перестаёт быть катастрофой: сборка описана Dockerfile и воспроизводима без него.

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

29.7.2

Ветка Docker Engine, которую ставим из download.docker.com, а не пакетом docker.io

Docker Engine 29 release notes, 05.08.2026

v5.5.0

Compose-плагин: сверка digest без лишнего пересоздания контейнеров, pull_policy refresh

Compose release notes v5.5.0, 17.08.2026

20m x 5

Ротация драйвера local по умолчанию: 100 МБ логов на контейнер, файлы сжимаются

docs.docker.com, logging drivers: local

10 с

Пауза docker stop между SIGTERM и SIGKILL на Linux — окно для graceful shutdown

docs.docker.com, docker container stop

65536

Минимум subordinate UID/GID в /etc/subuid и /etc/subgid для rootless-режима

docs.docker.com, rootless mode

80%

Порог заполнения раздела data-root: триггерим docker system prune и алерт дежурному

наш стандарт приёмки узла

РЕАЛИЗАЦИЯ 1 ИЗ 3

Базовый узел: Engine 29 из upstream-репозитория

Что настраиваем

Один Linux-хост клиента (Ubuntu 22.04/24.04 LTS или Debian 12), роль — сервер приложений на 10–15 контейнеров

Как мы это делаем

1. Ключ в /etc/apt/keyrings/docker.asc, репозиторий download.docker.com, пакеты docker-ce, docker-ce-cli, containerd.io, docker-buildx-plugin, docker-compose-plugin
2. В daemon.json задаём data-root на отдельный раздел (/data/docker) и live-restore: true, чтобы рестарт демона не гасил контейнеры
3. Логи: log-driver local — он ротирует из коробки (max-size 20m, max-file 5, сжатие); json-file по умолчанию пишет без предела
4. docker info: Storage Driver overlay2, Cgroup Version 2, containerd image store — в Engine 29 он по умолчанию, но не на демонах с userns-remap
5. Членство в группе docker выдаём поимённо и равняем к root на хосте; фиксируем состав в паспорте доступов

РЕЗУЛЬТАТ

Узел принимается по чек-листу за один заход: обновления идут из upstream-репозитория, логи не растут бесконечно, /var не забивается, а рестарт docker.service после apt upgrade не роняет прод клиента.

КЛЮЧЕВОЙ НЮАНС

В Engine 29 cgroup v1 объявлен устаревшим (поддержка минимум до мая 2029), а Docker Content Trust вырезан из CLI — старые ядра и скрипты подписи образов ловим до миграции, а не после.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Данные и сеть: тома, именованные сети, лимиты

Что настраиваем

Стек «БД + приложение + reverse proxy» на узле клиента; всё состояние вынесено за пределы контейнеров

Как мы это делаем

1. Named volume под каждую БД (docker volume create pgdata), bind mount — только для конфигов и статики в режиме :ro
2. Отдельная сеть docker network create app-net: сервисы резолвят друг друга по имени, порты БД наружу не публикуем
3. Бэкап тома снимаем одноразовым контейнером с примонтированным томом через pg_dump/tar, а не копированием /var/lib/docker/volumes
4. Лимиты в compose: deploy.resources.limits.cpus и memory, pids_limit — чтобы один сервис не съел узел целиком
5. tmpfs-монт под секреты и временные кеши, чтобы они не оседали в слое записи контейнера

РЕЗУЛЬТАТ

Падение или пересборка контейнера перестаёт означать потерю данных: том живёт отдельно, восстановление БД — это docker compose up с тем же volume. Публикация 5432 наружу закрыта, обмен идёт по внутренней сети по DNS-именам сервисов.

КЛЮЧЕВОЙ НЮАНС

Bind mount без :ro на проде — самая частая причина, когда контейнер молча переписывает хостовые файлы; и любой bind наследует UID процесса контейнера, а не владельца каталога.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Compose-стек: healthcheck, порядок старта, обновление

Что настраиваем

docker-compose.yml в git-репозитории клиента, разворот через docker compose up -d на целевом узле

Как мы это делаем

1. В Dockerfile фиксируем базовый тег (python:3.12-slim, а не latest), ставим зависимости до COPY кода и задаём USER appuser
2. HEALTHCHECK с --interval 30s, --timeout 5s, --retries 3, широким --start-period и --start-interval 5s: до готовности проба чаще
3. depends_on с condition: service_healthy вместо голого списка; разовые миграции — в pre_start-хук сервиса, а не отдельным контейнером
4. restart: unless-stopped на всех сервисах плюс .dockerignore, чтобы .git и node_modules не уезжали в контекст сборки
5. Обновление: docker compose pull && docker compose up -d, откат — прежний тег образа в файле; latest в проде не держим

РЕЗУЛЬТАТ

Стек поднимается детерминированно: приложение не стучится в ещё не готовую БД, гонок при старте нет, обновление и откат — две команды из git-репозитория, а не ручные шаги по памяти дежурного.

КЛЮЧЕВОЙ НЮАНС

depends_on без condition гарантирует только порядок запуска, но не готовность; а start-period для БД нужен большой — иначе первые провалы пробы уводят контейнер в unhealthy-рестарт.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Пакет docker.io из репозитория дистрибутива**
  Отстаёт на мажорные ветки, нет buildx и compose-плагина. Ставим docker-ce и docker-ce-cli из download.docker.com.

- ✕ **latest в продакшне**
  Тег плавающий: после pull неизвестно, что приехало, и откатиться некуда. Фиксируем мажор-минор, критичное — по digest.

- ✕ **json-file без ротации**
  Драйвер по умолчанию пишет лог без предела и выедает корень. Переводим на log-driver local: ротация 20m x 5 из коробки.

- ✕ **root внутри контейнера**
  Уязвимость приложения даёт UID 0. Добавляем USER appuser, --security-opt no-new-privileges, --cap-drop ALL и точечный --cap-add.

- ✕ **docker system prune -af --volumes**
  Ключ --volumes сносит и данные БД, если том не подключён к живому контейнеру. Чистим образы отдельно, тома — только по списку.

- ✕ **Данные внутри контейнера**
  Слой записи умирает вместе с контейнером. Всё состояние — в named volume; проверяем docker inspect по секции Mounts перед сдачей узла.

- ✕ **Группа docker выдана «всем админам»**
  Членство равно root на хосте: через bind mount корня получают всю ФС. Выдаём поимённо, остальным — sudo-обёртки на нужные команды.

- ✕ **Публикация портов БД наружу**
  -p 5432:5432 открывает порт мимо ufw через цепочки DOCKER в iptables. Либо не публикуем вовсе, либо -p 127.0.0.1:5432:5432.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Docker CE из download.docker.com вместо пакета docker.io дистрибутива
- log-driver local с max-size 20m и max-file 5 в /etc/docker/daemon.json
- Все данные — в named volumes, фиксированные теги образов вместо latest
- restart: unless-stopped и docker compose up -d вместо ручных docker run

НОРМАЛЬНО

- data-root на отдельном разделе и live-restore: true в daemon.json
- HEALTHCHECK в каждом образе и depends_on: condition: service_healthy
- USER appuser в Dockerfile, no-new-privileges и cap-drop ALL по умолчанию
- Бэкап томов по расписанию отдельным контейнером и проверка восстановления

ХОРОШО

- Rootless-режим или userns-remap с 65536 subordinate UID в /etc/subuid
- Приватный реестр, фиксация образов по digest и скан CVE до выката на прод
- Лимиты deploy.resources.limits и pids_limit на каждом сервисе стека
- Мониторинг health-статуса и заполнения data-root с алертом на 80%

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Engine поставлен из download.docker.com пакетами docker-ce и docker-ce-cli, а не docker.io?
- [ ] В /etc/docker/daemon.json задана ротация логов, а data-root вынесен с системного раздела?
- [ ] Все базы и пользовательские загрузки лежат в named volumes, а не в слое записи контейнера?
- [ ] У каждого образа зафиксирован тег версии, latest в compose-файлах отсутствует?
- [ ] Есть HEALTHCHECK и depends_on с condition: service_healthy для зависимых сервисов?
- [ ] Контейнеры запускаются не под root: в Dockerfile указан USER, лишние capability сняты?
- [ ] Порты БД и внутренних сервисов не опубликованы на хост через -p (или только на 127.0.0.1)?
- [ ] Бэкап томов делается и хотя бы раз проверен восстановлением на тестовом узле?
- [ ] Список членов группы docker согласован и совпадает с реальным составом администраторов?
- [ ] Compose-файлы лежат в git, а разворот воспроизводится с нуля одной командой?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Ставим и настраиваем Docker Engine 29 на Ubuntu LTS и Debian 12: daemon.json, ротация логов, data-root, доступы
- Упаковываем ваши приложения в образы: Dockerfile, .dockerignore, multi-stage, non-root USER, HEALTHCHECK
- Собираем compose-стек с томами, внутренними сетями, лимитами ресурсов и порядком старта по healthcheck
- Настраиваем бэкап томов с проверкой восстановления, мониторинг health и заполнения data-root
- Переносим legacy-сервисы с EOL-дистрибутивов в контейнеры без переписывания кода приложения

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Docker Engine version 29 release notes (docs.docker.com — 29.7.2)

02 Configure logging drivers: local file logging driver (docs.docker.com — 2026)

03 Install Docker Engine on Ubuntu/Debian, daemon.json reference (docs.docker.com — 2026)

04 Dockerfile reference: USER, HEALTHCHECK, multi-stage (docs.docker.com — 2026)

05 Compose file reference: depends_on, pre_start, deploy (docs.docker.com — v5.5.0)

06 Docker Engine security: rootless mode, userns-remap, seccomp (docs.docker.com — 2026)

07 Manage data in Docker: volumes, bind mounts, tmpfs (docs.docker.com — 2026)

08 Чек-лист приёмки Docker-узла ITfresh (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.