Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Docker без DevOps: GitLab, Nextcloud и Wiki в офисе

Как мы собираем и эксплуатируем внутренние сервисы на Docker Engine 29 и Compose

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Внутренние сервисы клиента — файловое облако, репозиторий кода, база знаний — ставятся на один сервер пакетами, и первый же апгрейд ломает соседа: GitLab тянет свою PostgreSQL, Nextcloud требует другую версию PHP. Дальше — откат к копии трёхнедельной давности. Мы переводим такие сервисы в контейнеры: у каждого свои зависимости, фиксированный тег, данные на хосте.

Почему это важно бизнесу

- Конфликт зависимостей на общем сервере кладёт сразу несколько сервисов — простой всего офиса, а не одного отдела.
- Откат к копии трёхнедельной давности означает потерю документов, задач и истории кода за этот период.
- Ручное обновление сервиса занимает у админа часы, обновление контейнера — минуты: это прямая экономия часов обслуживания.
- Свои Nextcloud и Wiki убирают рабочие документы из мессенджеров и личных облаков — данные остаются у компании.
- Переносимость стека: смена хостинга или железа делается копированием compose-файла и каталога данных.

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

29.7.2

актуальный Docker Engine; на новых установках хранилище образов — containerd

docs.docker.com, release notes 29.x, 08.2026

34.0.3

тег Nextcloud вместо latest: фиксируем мажор и обновляем осознанно по release notes

docs.nextcloud.com, ветка 34

256m

shm_size контейнера gitlab-ce из офиц. compose-примера, без него omnibus нестабилен

docs.gitlab.com, install in Docker

512M

дефолт PHP_MEMORY_LIMIT и PHP_UPLOAD_LIMIT образа Nextcloud; поднимаем под тяжёлые сканы

образ nextcloud, значения по умолчанию

60 с

heartbeat Uptime Kuma 2.5 по каждому сервису, retries 2 — одиночный лаг не даёт алерт

наш стандарт

10m x3

ротация json-file логов в daemon.json: без неё логи контейнеров съедают корневой раздел

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Базовая линия хоста: Docker Engine, лимиты, логи

Что настраиваем

Один узел Ubuntu 24.04 LTS или Debian 13, роль docker-host, от 8 ГБ RAM, в парке клиента — сервисный сервер

Как мы это делаем

1. Ставим docker-ce и docker-compose-plugin из официального репозитория Docker: пакет docker.io из дистрибутива отстаёт и не даёт актуальный docker compose.
2. В /etc/docker/daemon.json задаём json-file с max-size 10m, max-file 3 и live-restore: ротации по умолчанию нет, лог одного контейнера заполняет корень.
3. Каждому сервису в compose задаём restart: unless-stopped, deploy.resources.limits (memory, pids) и healthcheck с interval, timeout, retries, start_period.
4. Связки приложение-БД поднимаем через depends_on с condition: service_healthy, чтобы приложение не стартовало раньше готовой базы и не писало битую схему.
5. Публикуем порты только на 127.0.0.1, наружу выпускаем через reverse-proxy: правила Docker в nftables/iptables идут мимо ufw и открывают порт молча.

РЕЗУЛЬТАТ

Хост становится воспроизводимым: перенос на другое железо — это копирование compose-файлов и каталогов данных. Падение одного контейнера не задевает соседей, лимиты памяти не дают одному сервису выесть RAM всего узла, а логи перестают быть причиной аварии по месту на диске.

КЛЮЧЕВОЙ НЮАНС

Docker Engine 29.0 включает containerd-хранилище образов только на НОВЫХ установках: при апгрейде остаётся старое. Переезд на containerd планируем отдельным окном с бэкапом /var/lib/docker.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Nextcloud 34: приложение, MariaDB и Redis одним стеком

Что настраиваем

Четыре контейнера на docker-хосте: app (apache), db, redis и отдельный cron; данные — bind mount /opt/nextcloud

Как мы это делаем

1. Фиксируем nextcloud:34-apache и mariadb:11.8: мануал требует InnoDB и уровень изоляции READ COMMITTED, рекомендованные ветки — MariaDB 11.8 или PostgreSQL 18.
2. Задаём переменные образа: NEXTCLOUD_TRUSTED_DOMAINS, MYSQL_HOST/MYSQL_USER/MYSQL_PASSWORD/MYSQL_DATABASE, REDIS_HOST для блокировок файлов и кэша сессий.
3. Каталоги /opt/nextcloud/html и /data монтируем bind mount'ом и заранее выставляем владельца под www-data образа — иначе первый старт падает на правах.
4. Фоновые задачи выносим в контейнер с /cron.sh и переключаем режим на Cron вместо AJAX: без этого не отрабатывают превью, чистка корзины и индексация.
5. PHP тюним по мануалу: opcache.save_comments=1, revalidate_freq=60, JIT 1255 при буфере 8M, pm.max_children = RAM под PHP делить на 50-100 МБ воркера.

РЕЗУЛЬТАТ

Клиент получает облако с синхронизацией, ссылками и календарями, где файлы лежат на его сервере. Обновление сводится к смене тега и docker compose pull/up -d, а не к часовой возне с PHP и Apache. Redis снимает конфликты блокировок при одновременной работе десятков клиентов.

КЛЮЧЕВОЙ НЮАНС

Мануал Nextcloud 34 помечает PHP 8.2 устаревшим и рекомендует 8.5, из БД — MariaDB 11.8 или PostgreSQL 18. Совместимость версии БД проверяем ДО апгрейда мажора: откат из дампа обходится дороже.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Git, Wiki и TLS: Gitea или GitLab CE, BookStack, Traefik

Что настраиваем

Второй контур того же хоста: reverse-proxy, git-сервис, база знаний и монитор доступности

Как мы это делаем

1. Traefik v3.7 ставим единой точкой на 80/443: certificatesresolvers.<name>.acme.tlschallenge=true, storage acme.json, права 600 на файле хранилища.
2. Git подбираем по размеру команды: до 10 разработчиков — Gitea 1.27 (по докам хватает 2 ядер и 1 ГБ RAM), нужен встроенный CI/CD — gitlab-ce с shm_size 256m.
3. SSH GitLab публикуем как 2424:22 и синхронно ставим gitlab_rails['gitlab_shell_ssh_port'] = 2424, иначе ссылки clone врут, а порт 22 хоста уже занят.
4. BookStack 26.05 разворачиваем на PHP 8.2+ и MySQL 8.0 / MariaDB 10.6 с обязательным расширением zip; вход подключаем к AD клиента через LDAP.
5. Бэкап: дамп БД штатной утилитой внутри контейнера плюс rsync каталогов /opt/*; в Uptime Kuma 2.5 заводим HTTP-монитор на каждый сервис с алертом в Telegram.

РЕЗУЛЬТАТ

Сертификаты выпускаются и продлеваются сами, все сервисы доступны по HTTPS на своих поддоменах. Код и регламенты уходят из личных папок в системы с правами и историей правок, а падение любого контейнера видно раньше, чем о нём напишет пользователь.

КЛЮЧЕВОЙ НЮАНС

Выбор git-сервиса — это бюджет RAM: базовая линия GitLab по докам — 8 vCPU и 16 ГБ, нижняя граница 8 ГБ, тогда как Gitea обслуживает ту же команду на 1-2 ГБ.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Тег latest в compose-файле**
  Однажды приезжает мажор с миграцией БД без отката. Фиксируем мажорный тег (nextcloud:34-apache, gitea:1.27) и обновляемся осознанно.

- ✕ **Данные внутри контейнера**
  Пересоздание контейнера стирает всё. Состояние держим в bind mount /opt/<сервис>, конфиги — в git, секреты — в .env вне репозитория.

- ✕ **Порт 22 отдан GitLab**
  Проброс 22:22 конфликтует с SSH хоста и режет доступ к серверу. Выносим git-SSH на 2424 и дублируем номер в gitlab_shell_ssh_port.

- ✕ **Права на bind mount**
  Каталог создан от root, а процесс в контейнере работает под своим UID и падает на записи. Владельца выставляем до первого старта стека.

- ✕ **Логи без ротации**
  json-file по умолчанию растёт бесконечно и однажды забивает корень. Ротацию задаём глобально в daemon.json, а не по контейнерам.

- ✕ **Публикация портов мимо firewall**
  Docker пишет собственные правила и обходит ufw: порт «закрыт» в фаерволе, но доступен из интернета. Биндим на 127.0.0.1 и пускаем через proxy.

- ✕ **Копирование файлов БД на живую**
  Снимок каталога работающей MariaDB даёт нецелостную базу. Бэкап делаем логическим дампом из контейнера либо с остановкой сервиса.

- ✕ **docker system prune -a вслепую**
  Команда сносит неиспользуемые образы и тома, включая нужные при остановленном стеке. Чистим точечно и только после проверки бэкапа.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Docker CE и compose-plugin из репозитория Docker, а не пакет docker.io
- Данные в bind mount /opt/<сервис>, ежедневный дамп БД на отдельное хранилище
- restart: unless-stopped и ротация json-file логов 10m x3 в daemon.json
- Мажорные теги образов вместо latest во всех compose-файлах

НОРМАЛЬНО

- Traefik v3.7 с ACME, редирект HTTP-HTTPS, acme.json с правами 600
- healthcheck у каждого сервиса и depends_on с condition: service_healthy
- Лимиты deploy.resources.limits по памяти и pids на каждый контейнер
- Uptime Kuma: HTTP-монитор на сервис, интервал 60 с, retries 2, Telegram

ХОРОШО

- Тестовое восстановление всего стека на запасном хосте раз в квартал
- Вход в Nextcloud, BookStack и Gitea через LDAP/AD, а не локальные пароли
- Порты только на 127.0.0.1, наружу — исключительно через reverse-proxy
- Регламент обновлений: release notes мажора, окно работ, откат из дампа

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Docker поставлен из репозитория Docker, и docker compose version показывает v2 или новее?
- [ ] Все данные сервисов лежат в bind mount на хосте, а не в анонимных томах контейнеров?
- [ ] В compose зафиксированы мажорные теги образов, а не latest?
- [ ] В daemon.json настроена ротация json-file логов (max-size, max-file)?
- [ ] У каждого контейнера есть healthcheck, а зависимости идут через condition: service_healthy?
- [ ] SSH GitLab разведён с портом 22 хоста и продублирован в gitlab_shell_ssh_port?
- [ ] Бэкап баз делается дампом, а не копированием файлов БД работающей СУБД?
- [ ] Восстановление стека на чистом хосте проверяли за последние три месяца?
- [ ] Опубликованные порты закрыты снаружи с учётом того, что Docker пишет правила мимо ufw?
- [ ] Выпуск и продление TLS-сертификатов автоматизированы, а срок контролируется монитором?

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).

- Разворачиваем docker-хост под ключ: движок, лимиты, ротация логов, доступы, регламент обновлений
- Собираем стек Nextcloud с БД и Redis, переносим файлы с сетевых шар, подключаем вход через AD
- Ставим Gitea или GitLab CE по размеру команды, выносим git-SSH, настраиваем бэкап репозиториев
- Поднимаем BookStack или Wiki.js, переносим регламенты из Word, раздаём права по отделам
- Настраиваем Traefik с автосертификатами и Uptime Kuma с алертами, обучаем вашего админа

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 release notes, ветка 29.x (docs.docker.com — 29.7.2)

02 Compose file reference: services, deploy, healthcheck, depends_on (docs.docker.com — 2026)

03 Nextcloud Admin Manual: System requirements, Server tuning (docs.nextcloud.com — 34)

04 GitLab Docs: Install GitLab in a Docker container, Requirements (docs.gitlab.com — 2026)

05 Traefik Docs: certificate resolvers, ACME (TLS challenge) (doc.traefik.io — v3.7)

06 BookStack Docs: Installation, требования к PHP и БД (bookstackapp.com — 26.05)

07 Gitea Docs: installation, системные требования (docs.gitea.com — 1.27)

08 Шаблон compose-стека и чек-лист приёмки docker-хоста ITfresh (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.