Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Kubernetes для малого бизнеса: когда нужен, а когда нет

Архитектура, k3s, версии и обновления: наш разбор для компаний до 50 рабочих мест

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

К нам приходят с тремя болями: релиз — ночное окно и остановка работы, сервис живёт на одном сервере и падает вместе с железом, деплой делается руками по SSH. Kubernetes закрывает все три, но приносит свой слой отказа: kube-apiserver и etcd, kubelet и containerd на узлах, CNI, ingress, RBAC и апгрейд минора раз в год. Под один-два сервиса без дежурной смены он не нужен.

Почему это важно бизнесу

- Обновление без rolling update — это окно 30-60 минут и простой отдела продаж или приёмного отделения.
- Один сервер: отказ диска или БП кладёт сервис до ручного восстановления из бэкапа, это часы.
- Ручной деплой по SSH не воспроизводится: откат держится на памяти конкретного инженера.
- Кластер сам становится объектом обслуживания: минорный релиз живёт около 14 месяцев, апгрейд обязателен.
- Кластер без мониторинга и бэкапа etcd опаснее одного сервера: потеря кворума роняет весь контур сразу.

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

3 релиза

Одновременно поддерживаются только три последние минорные версии Kubernetes

kubernetes.io/releases

~14 мес

Жизнь минорного релиза: 12 месяцев патчей плюс 2 месяца maintenance mode

kubernetes.io/releases/patch-releases

1.35-1.37

Окно поддержки на сентябрь 2026; ветка 1.34 уходит в EOL 27.10.2026

kubernetes.io/releases

2 CPU / 2 ГБ

Минимум под server-узел k3s; агенту достаточно 1 ядра и 512 МБ RAM

docs.k3s.io/installation/requirements

3 узла

Минимум для HA k3s на встроенном etcd: число server-узлов должно быть нечётным

docs.k3s.io/datastore/ha-embedded

00:00/12:00

Штатное расписание снапшотов etcd в k3s, по умолчанию хранится 5 последних

docs.k3s.io/cli/etcd-snapshot

РЕАЛИЗАЦИЯ 1 ИЗ 3

Когда мы отговорили от кластера: клиника на 35 рабочих мест

Что настраиваем

Медцентр: МИС на одном сервере плюс сайт записи, релизы раз в квартал, своей команды разработки нет

Как мы это делаем

1. Считаем нагрузку: два приложения, пик 40 запросов в секунду, состояние целиком в PostgreSQL — горизонтальное масштабирование не требуется.
2. Считаем обслуживание: кластер добавляет апгрейд минорной версии раз в год, бэкап датастора, обновление CNI и ingress — это отдельная компетенция.
3. Вместо K8s разворачиваем две VM с Docker Compose, nginx-фронт с health-check и переключением, реплику PostgreSQL и восстановление по расписанию.
4. Деплой переводим на git-тег и один скрипт: сборка образа, docker compose up -d с новым тегом, откат возвратом на предыдущий тег.

РЕЗУЛЬТАТ

Обновление приложения перестало требовать ночного окна: переключение фронта занимает секунды. Отказ основной VM закрывается переездом на вторую за 10-15 минут. Клиент не платит за компетенцию, которая не нужна ему каждый день, и не получает новый класс аварий вида «развалился etcd».

КЛЮЧЕВОЙ НЮАНС

Kubernetes оправдан, когда сервисов больше пяти, релизы чаще раза в неделю и нужна автоматическая пересборка при отказе узла. Иначе выигрыш съедается стоимостью эксплуатации кластера.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Когда кластер оправдан: торговая компания, k3s на трёх узлах

Что настраиваем

Оптовая торговля, 45 рабочих мест: витрина, API обмена с 1С, очередь, воркеры обработки заказов, релизы 2-3 раза в неделю

Как мы это делаем

1. Ставим три server-узла k3s на встроенном etcd: число серверов нечётное ради кворума, флаги --cluster-cidr и --service-cidr одинаковые на всех.
2. Точка входа — Traefik, который k3s ставит по умолчанию; для новых маршрутов закладываем Gateway API: ingress-nginx выведен из поддержки 24.03.2026.
3. На каждый Deployment задаём resources.requests и limits, readinessProbe и startupProbe, PodDisruptionBudget с minAvailable, чтобы drain не снял все реплики.
4. Состояние выносим наружу: PostgreSQL на отдельной VM, PVC local-path только под кэш; снапшоты etcd (00:00 и 12:00, хранится 5) копируем в S3-хранилище.
5. Обновляем окнами: kubectl drain --ignore-daemonsets, апгрейд узла, kubectl uncordon; kubelet держим не старше kube-apiserver более чем на 3 минорные версии.

РЕЗУЛЬТАТ

Релиз идёт rolling update без окна: старые поды снимаются только после успешного readinessProbe у новых. Плановая перезагрузка узла проходит в рабочее время. Ошибочный образ ловится на readiness и не уводит трафик, откат — kubectl rollout undo за секунды.

КЛЮЧЕВОЙ НЮАНС

Кластер даёт эффект только вместе с дисциплиной: заданные лимиты, пробы, PodDisruptionBudget и проверенное восстановление etcd. Без этого получается тот же одиночный сервер, только с YAML сверху.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Ремонт заброшенного кластера: версия за пределами поддержки

Что настраиваем

Производство, 50 мест: кластер поставил подрядчик и ушёл, минорная версия не обновлялась больше года

Как мы это делаем

1. Фиксируем факт: ветка вне окна поддержки, патчи безопасности не выходят, перескакивать через минорные версии нельзя — только по одной.
2. Перед апгрейдом снимаем снапшот etcd командой k3s etcd-snapshot save и проверяем восстановление на отдельном стенде, а не в проде.
3. Идём по одной минорной версии: сначала control plane (kube-apiserver), затем kubelet и kube-proxy на узлах, в пределах version skew.
4. Параллельно вычищаем устаревшие apiVersion в манифестах и добавляем недостающие пробы и лимиты, которых подрядчик не закладывал.

РЕЗУЛЬТАТ

Кластер вернулся в поддерживаемое окно версий, обновления безопасности снова применяются штатно. Восстановление etcd стало проверенной процедурой с известным временем, а не гипотезой. Манифесты приведены к текущим apiVersion, апгрейды перестали ломать деплой.

КЛЮЧЕВОЙ НЮАНС

Апгрейд Kubernetes — не разовый проект, а годовой цикл: релиз живёт около 14 месяцев. Если у клиента нет ресурса на один апгрейд в год, кластер закончится тем же брошенным состоянием.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Кластер там, где хватает двух VM**
  Один stateless-сервис и релиз раз в квартал не окупают etcd, CNI, ingress и ежегодный апгрейд минорной версии.
- ✕ **Нет requests и limits**
  Без requests планировщик не резервирует ресурсы и переполняет узел; без limits один под выедает CPU и память соседей.
- ✕ **Пробы не настроены или скопированы**
  Один эндпоинт в livenessProbe и readinessProbe даёт цикл рестартов под нагрузкой; долгий старт закрывается startupProbe.
- ✕ **Бэкап etcd не проверялся**
  Снапшот есть, восстановление не отрабатывали. Реальное время подъёма кластера выясняется в момент аварии.
- ✕ **Два server-узла в HA**
  Чётное число серверов не добавляет отказоустойчивости: потеря одного из двух ломает кворум etcd и кладёт control plane.
- ✕ **Состояние внутри кластера**
  База на PVC локального провижнера привязывает под к узлу: узел выпал — данные недоступны, планировщик не перенесёт под.
- ✕ **Ставка на ingress-nginx в новом проекте**
  Проект выведен из поддержки 24.03.2026: новых релизов и исправлений CVE не будет. Новые маршруты делаем на Gateway API.
- ✕ **Нет PodDisruptionBudget**
  При drain узла добровольное выселение снимает все реплики сервиса разом и даёт тот самый даунтайм.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Честно ответить, нужен ли кластер: меньше 3 сервисов и редкие релизы — не нужен
- Один k3s server (датастор SQLite), базу данных вынести за пределы кластера
- resources.requests/limits и readinessProbe на каждом Deployment без исключений
- Бэкап каталога /var/lib/rancher/k3s/server и токена по расписанию, копия вне узла

НОРМАЛЬНО

- 3 server-узла k3s на встроенном etcd, число серверов нечётное ради кворума
- Traefik или Gateway API с TLS: единая точка входа и понятные маршруты
- PodDisruptionBudget и startupProbe для сервисов с долгим прогревом
- Регламент апгрейда: одна минорная версия за раз, в пределах version skew

ХОРОШО

- Манифесты в git, деплой только через пайплайн, ручной kubectl apply запрещён
- Метрики и алерты по узлам, подам, рестартам и задержке записи в etcd
- Ежеквартальные учения: восстановление кластера из снапшота etcd на стенде
- Разделение прав через RBAC и отдельные namespace под окружения

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Версия кластера внутри поддерживаемого окна: три последние минорные ветки
- [ ] У каждого контейнера заданы resources.requests и limits по CPU и памяти
- [ ] readinessProbe отделён от livenessProbe, долгий старт закрыт startupProbe
- [ ] PodDisruptionBudget задан для всех сервисов, участвующих в rolling update
- [ ] Снапшоты etcd (или копия SQLite-датастора k3s) делаются по расписанию, копия вне узла
- [ ] Восстановление из снапшота отработано на стенде, время подъёма измерено
- [ ] Состояние (БД, файлы) вынесено из кластера или лежит на сетевом хранилище
- [ ] Точка входа: ingress или Gateway API с валидным TLS и автопродлением
- [ ] Число server-узлов нечётное, кворум etcd сохраняется при отказе одного
- [ ] Есть регламент апгрейда на год: кто, когда и в каком окне обновляет узлы

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).

- Считаем окупаемость до внедрения: где хватит двух VM, а где кластер действительно нужен
- Разворачиваем k3s или kubeadm-кластер под ключ: сеть, ingress, TLS, хранилище, RBAC
- Приводим манифесты в порядок: лимиты, пробы, PDB, деплой через пайплайн вместо ручных команд
- Ведём годовой цикл апгрейдов и проверяем восстановление etcd на стенде, а не в проде
- Подключаем мониторинг узлов, подов и etcd с алертами и дежурной реакцией

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 Releases: поддерживаемые версии 1.35-1.37 и даты EOL (kubernetes.io — 2026)

02 Patch Releases: 12 месяцев патчей плюс maintenance mode (kubernetes.io — 2026)

03 Version Skew Policy: разрыв kube-apiserver, kubelet, kube-proxy (kubernetes.io — 2026)

04 Gateway API: замена Ingress и статус проекта Ingress NGINX (kubernetes.io — 2026)

05 Configure Liveness, Readiness and Startup Probes (kubernetes.io — 2026)

06 K3s Requirements: минимальные ресурсы server и agent, v1.34 (docs.k3s.io — 2026)

07 K3s High Availability with Embedded etcd, v1.34 (docs.k3s.io — 2026)

08 K3s etcd-snapshot: расписание, хранение, восстановление (docs.k3s.io — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.