АНОНИМИЗИРОВАННЫЙ КЕЙС · DEVOPS/SRE-АУДИТ

Laravel-продукт на гибридном облаке: аудит инфраструктуры

Контекст

Продукт	B2B SaaS на Laravel (PHP)
Инфраструктура	5 серверов: 3 в Yandex Cloud, 2 в Hetzner
База данных	MySQL (на отдельной BM в Yandex)
Команда	3 разработчика, DevOps не выделен
Окружения	dev, prod (без staging)
Git	GitLab (self-hosted в Hetzner)
Деплой	ручной, по «инструкции» в голове у старшего разработчика

Запрос заказчика: инфраструктура работает, но каждый релиз — риск. Команда хочет понять, где слабые места и за что браться в первую очередь.

Что нашёл аудит

Критично **1. Нет бэкапов — нигде**

База данных, код приложений, конфигурации — ничего не резервируется. Отказ диска на BM с MySQL или случайный `DROP TABLE` = полная потеря данных продукта и клиентов. Восстановление невозможно в принципе.

Риск: катастрофический. Потеря бизнеса при одном отказе диска.

Критично **2. Нет мониторинга**

Сбор метрик отсутствует. Дежурных нет, алертов нет. О том, что «сервису плохо», команда узнаёт от пользователей — через саппорт или отток.

Риск: инциденты длятся часами и обнаруживаются снаружи. MTTR не измеряется, потому что нечего измерять.

Критично **3. Нет CI/CD при наличии GitLab**

Деплой — ручной скрипт + SSH на сервера + шаги «по памяти» у одного человека. Отката нет: откат = «поднять старую версию и молиться». Воспроизводимости нет — сборка работает у одного и падает у других.

Риск: каждый релиз — лотерея. Уход старшего разработчика = остановка релизов.

Важно **4. Гибрид (Yandex + Hetzner) без единой сетевой модели**

Трафик между площадками идёт через интернет, без VPN. Latency и стоимость egress не контролируются. Граница доверия не определена — где прод, где dev, кто может ходить между ними.

Риск: перерасход на egress, рост latency, неявная attack surface.

Важно **5. dev/prod-разделение есть, но паритета нет**

dev — одна ВМ «всё-в-одном», prod — пять. Версии PHP, расширения, настройки различаются. Баг, который не воспроизводится в dev, ловится только в проде.

Риск: ложное чувство безопасности от «у нас есть dev».

Важно **6. Секреты в `.env`-файлах, распространяются вручную**

Ключи от платёжек, SMTP, сторонних API лежат в `.env` и копируются по чату/почте. Ротации нет. Кто имеет доступ к секретам прод-окружения — неизвестно, список не ведётся.

Риск: утечка при уходе сотрудника или компрометации чата.

Важно **7. Знания об инфре — у одного человека**

Как что связано, как деплоить, где что лежит, как чинить инцидент — знает только старший разработчик. Документации нет, runbook'ов нет.

Риск: bus factor = 1. Болезнь или уход — остановка.

Приятно иметь **8–10. Окружения, зависимости, тесты бэкапов**

Нет staging-окружения (релизы идут напрямую в прод). Версии PHP и Composer-зависимости не обновляются, security-патчи игнорируются, `composer audit` не запущен. Бэкап-тестов нет — потому что нет бэкапов.

Топ-5 рекомендаций

№	ПРОБЛЕМА	РЕКОМЕНДАЦИЯ	ТРУДОЗАТРАТЫ	ОЖИДАЕМЫЙ ЭФФЕКТ
1	Нет бэкапов	Автоматические бэкапы БД (ежечасно + ежедневный full), выгрузка в S3 в другой регион. Регламент restore-test раз в месяц.	2–3 инж.-дня	Устраняет риск полной потери данных
2	Нет мониторинга	Базовый стек: VictoriaMetrics/Prometheus + Grafana + 5–7 критичных алертов (БД, диск, HTTP 5xx, latency p99).	4–6 инж.-дней	Инциденты видны за минуты, а не часы
3	Нет CI/CD	Пайплайн в GitLab CI: линт → тесты → сборка артефакта → деплой по кнопке. Версионированные артефакты, откат одним кликом.	5–8 инж.-дня	Релизы предсказуемы, откат = секунды

№	ПРОБЛЕМА	РЕКОМЕНДАЦИЯ	ТРУДОЗАТРАТЫ	ОЖИДАЕМЫЙ ЭФФЕКТ
4	Секреты в `.env`	Перенос в GitLab CI/CD Variables или Vault. Отзыв доступа при offboarding.	2 инж.-дня	Сокращение attack surface, аудит доступа
5	Знания у одного	Документация инфры + 3 ключевых runbook'а (релиз, откат, инцидент). Схема зависимостей.	3 инж.-дня	bus factor растёт, онбординг быстрее

План на 30 / 60 / 90 дней

Первые 30 дней — критичное и быстрые победы

Бэкапы + восстановление (#1), базовый мониторинг с алертами (#2), перенос секретов (#4). После этого инфра перестаёт быть «чёрным ящиком», который может исчезнуть.

30–60 дней — важное

CI/CD (#3): повторяемые релизы, кнопочный откат. Документация и runbook'и (#5): знания выходят из головы в репозиторий.

60–90 дней — фундамент

Staging-окружение (#8), приведение dev к паритету с prod (#5), регулярные restore-тесты бэкапов, обновление зависимостей (#9).

> **Честно о границах.** Это анонимизированный пример того, что аудит находит и рекомендует на типичном профиле (Laravel, гибрид, команда без DevOps). Здесь нет пост-имплементационных метрик вида «латентность упала на X%» — их неоткуда взять, пока рекомендации не внедрены. Эффекты в таблице — ожидаемые, основанные на практике работы с такими инфраструктурами, а не обещание конкретных цифр. Если вам обещают «сократим косты на 40%» до того, как увидели ваш счёт за облако — это маркетинг, не аудит.
