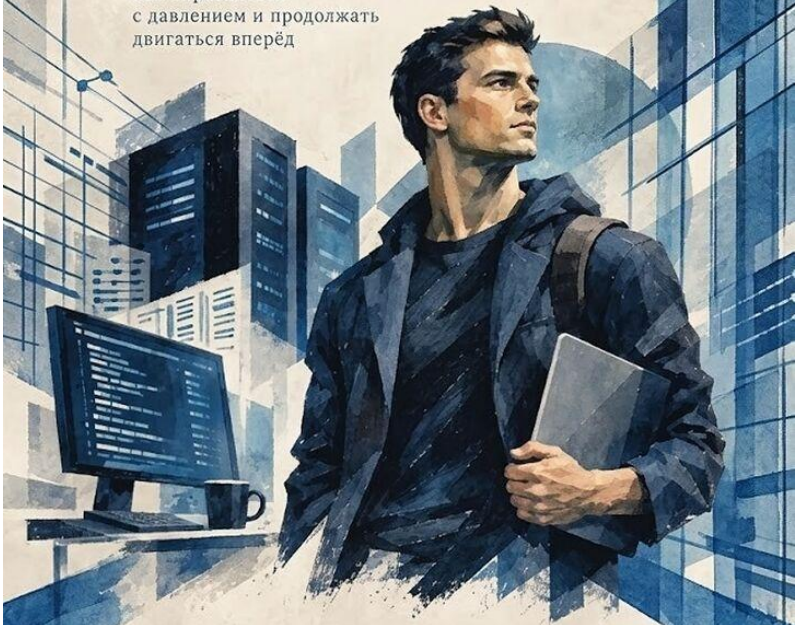
18+

Алексей Коробов

НЕ СЛОМАЙ ПРОД И СЕБЯ ТОЖЕ

как справляться
с давлением и продолжать
двигаться вперёд



НАУЧПОП • IT • ПСИХОЛОГИЯ

Алексей Коробов

Не сломай прод и себя тоже

https://litres.ru/74835137
ISBN 9785007164313

Аннотация

Вам скажут: учи Linux, Docker, Kubernetes, облака, сети и еще сорок страшных слов. Но не рассказывают, каково это на самом деле. Как сорок семь раз подряд падает сборка, а на сорок восьмой ты орёшь от счастья в пустом ночном офисе.

Почему всё ломается по пятницам, почему за DevOps платят такие деньги, как не утонуть в океане технологий и не сгореть на старте.

Семь простых правил, переведённых на язык терминалов, ночных авралов и набитых шишек.

Содержание

Посвящение 5
От автора 6
Пролог 8
Для кого эта книга — и для кого нет 13
Семь правил. Один путь 15
Интерлюдия · Между правилами 18
Правило 1. Составь чёткое видение 21
Что такое DevOps 24
Две компании. Один день. Разные вселенные 31
Почему за это платят 44
Так это профессия или философия? 51
Зачем это тебе — вопрос важнее всех остальных 53
Основные мысли главы 56
Упражнение · Между правилами 57
Правило 2. Не разменивайся 60
Почему DevOps — это не «немного сисадмин плюс немного разработчик» 64
Что реально нужно знать, а что просто шум 67
Дорожная карта без паники 70
Конец ознакомительного фрагмента. 73

Не сломай прод и себя тоже

Алексей Евгеньевич Коробов

Дизайнер обложки Алексей Евгеньевич Коробов

© Алексей Евгеньевич Коробов, 2026
© Алексей Евгеньевич Коробов, дизайн обложки, 2026

ISBN 978-5-0071-6431-3
Создано в интеллектуальной издательской системе Ridero

Посвящение

Моему сыну Алексею

Ты появился на свет и тихо перевернул всё, что я считал в жизни главным, расставив наконец по местам. Рядом с тобой я понял, зачем на самом деле стоит расти, и стал сильнее, чем считал возможным. Ты открыл мне горизонты, о которых я и не догадывался, и дал силы до них идти.

От автора

Это не учебник. Учебников по нашему делу написаны горы, толстых и умных, и они прекрасно справляются со своей работой. У этой книги работа другая.

Представь, что ты пришёл в новую профессию, растерянный, с сотней открытых вкладок и тихим страхом, что отстал навсегда. И вот сел рядом человек, который всё это уже прошёл. От самой нижней ступеньки, от духоты подвальной серверной, до спокойного старшего, к которому приходят за советом. Сел и говорит: давай по-честному, без галстука и умных слов, я расскажу, как тут всё устроено на самом деле. Вот эта книга и есть тот самый разговор.

Здесь не будет академических определений, от которых вязнет в зубах. Не будет обещаний золотых гор и зарплаты мечты за месяц. И не будет пафоса успешного успеха, которым нынче торгуют на каждом углу. Будет другое. Живые истории из окопов, набитые шишки, ночные авралы и та странная тихая радость, ради которой во всё это и стоит ввязываться.

Из этой книги ты узнаешь:

— что такое DevOps на самом деле и почему за него платят такие деньги;

— с чего начинать, чтобы не утонуть, и что можно пока не учить вовсе;

— как пережить первые сто дней, когда хочется всё бросить, и не сгореть на старте;

— как рассказать о своей пользе и продать себя, даже когда опыта ещё нет;

— почему всё ломается, как учиться на поломках и перестать их бояться;

— и, может быть, самое важное: что ты во всём этом не один.

Книга собрана вокруг семи простых правил. Не я их придумал, я лишь перевёл на язык терминалов, серверов и живой работы то, что давно знакомо людям, добившимся своего в самых разных делах. Каждое правило это глава и одна честная история за ней. Между главами короткие передышки, байки у костра, можешь читать их, а можешь пропускать.

Если ты только входишь в профессию, пусть эта книга будет тебе картой и фонариком разом. Если ты уже давно внутри и устал делать вид, что всё под контролем, узнаешь в этих историях свои бессонные ночи и, надеюсь, почувствуешь дружеское плечо рядом. А мне больше ничего и не надо.

Ну что, садись поудобнее. Начнём.

Пролог

Пятница, 17:59. Всё сломано. Добро пожаловать.

«Не масштабируй хаос — сначала исправь его».

Так говорил мой старший наставник, чьё имя стёрлось из памяти. Урок остался.

Бывает ли самосаботаж изящнее, чем потратить три дня на автоматизацию того, что делается за десять минут, и в самом финале обнаружить, что эти десять минут были сломаны с самого начала?

Спрашиваю за друга.

Друг это я.

Сейчас будет история. Одна из тех, что у нас травят с банкой пива в руке и нервным смешком. Нервным, потому что сначала смешно, а потом вдруг становится не очень. История о том, как вполне вменяемый человек с лучшими на свете намерениями взял маленькую проблему и превратил её в большую. Идеально автоматизированную. Катастрофу.

* * *

Пятница. 17:59.

Половина команды уже мысленно жарит шашлык. Про-

дакш[1] работает. Билд[2] собирается. Тишина. И именно эта тишина должна была меня насторожить, но в пятницу вечером инстинкт самосохранения уходит домой раньше всех остальных.

В тот день кто-то раскопал старый скрипт[3] сборки. Древний. Написанный руками человека, который давно уволился: он ушёл, а скрипт остался, как проклятие в наследство. Этот скрипт отличался одной особенностью. Примерно каждый десятый запуск он падал[4]. Без причины. Без логики. Просто падал, и всё. Причину не знал никто. Все давно привыкли и выработали ритуал: упал, перезапусти, сходи за кофе, на второй раз соберётся. Так и жили. Годами. С падающим скриптом и крепким кофе.

---

[1] **Продакшн (прод, «боевой» сервер) —** Та самая, настоящая версия системы, которой прямо сейчас пользуются живые люди. Противоположность тренировочным копиям. «Уронить прод» — значит сломать то, что видят реальные клиенты.

[2] **Билд (сборка) —** Процесс превращения написанного программистами кода в готовую, способную запуститься программу. «Собрать билд» — значит сложить из исходных кусочков работающее приложение.

[3] **Скрипт —** Небольшая программа для одного конкретного действия или короткой цепочки действий. Здесь — самодельная программка, которая по команде собирала приложение.

[4] **Упал (про программу, скрипт, сервер) —** на профессиональном жаргоне «упало» значит «перестало работать», аварийно остановилось, сломалось на ходу. Программа не «свалилась» откуда-то — она запустилась, наткнулась на ошибку и прекратила работу, не доделав дело. «Сайт упал» — перестал открываться; «скрипт упал» — не довёл сборку до конца и остановился на полпути.

И тут у нас возникла Идея.

Именно так, с большой буквы. Потому что плохие идеи всегда приходят с большой буквы, в сиянии и под фанфары. Хорошие приходят тихо и в рабочей одежде. А плохие всегда как на парад.

Идея была не «давай наконец разберёмся, почему эта штука падает». Нет, что ты. Скучно. Копаться неохота. Идея была другая, блестящая, достойная презентации: «Давай завернём весь этот хаос в красивый, современный, воспроизводимый конвейер! С контейнерами. С автозапуском. С мониторингом. По-взрослому. Чтобы не стыдно было людям показать».

Три дня. Три дня работы. Красивейший YAML[5], выверенный по каждому пробелу. Рецепт сборки на сорок строк, которым хотелось любоваться, как акварелью. Мы откинулись на стульях, оглядели сделанное, и нас распирала гордость честных мастеров.

Результат?

Фантомные падения никуда не делись.

Просто теперь они воспроизводились сами. Автоматически. В сто раз быстрее, чем раньше. Аккуратно, до миллисекунды, записанные в логи[6]. И, для финального штриха, кра-

[5] **YAML —** Простой текстовый формат для записи настроек. Печально знаменит строгими отступами пробелами: один лишний пробел в начале строки способен сломать всё.

[6] **Логи —** автоматический дневник программы: строчки, которые она сама записывает по ходу работы — что и когда сделала, что пошло не так и где именно.

сиво нарисованные на дашборде[7], где каждое падение горело укоризненным красным огоньком.

Раньше скрипт ронялся тихо, стыдливо, где-то в тёмном углу, и об этом знали два человека. Теперь он падал триумфально. В прямом эфире. С графиками и подсветкой.

Мы не починили баг. Мы возвели его в степень. Отлили в бронзе. Мы масштабировали его до состояния произведения искусства, и торжественно повесили в самом освещённом зале галереи.

* * *

Ты, наверное, ждёшь морали. Что-нибудь мудрое в духе «семь раз отмерь, один раз автоматизируй». Мораль будет, но позже, она заслужила отдельную главу. А пока я хочу спросить тебя о другом.

Знаешь, что спасло бы нас в тот вечер?

Не инструмент. Не ещё одна умная программа. DevOps.

Только сразу уберём недоразумение, пока оно не пустило корни. DevOps — это не Kubernetes, не Docker, не Jenkins, не Ansible и не ещё сорок страшных слов, от которых у новичка начинается лёгкая паника. (Не пугайся их, кстати. Большинство из них тебе пока не нужны. А те немногие, что нужны, мы разберём спокойно и по одному, без этой вот стены ло-

[7] **Дашборд —** Экран-панель, где в одном месте собраны графики и показатели состояния системы. Та самая «приборная доска», только для серверов.

готипов.)

DevOps вообще не про слова. Это способ думать. Это культура, в которой кто-нибудь обязательно, обязательно поднял бы руку и спросил занудным тихим голосом: «Стоп. А зачем мы автоматизируем то, что сломано? Может, сперва починим, а уж потом будем любоваться?».

В тот вечер такого человека рядом не нашлось. Точнее, он был. Но молчал. Потому что был самым младшим в комнате, робел и думал, что умные взрослые знают лучше. Это, к сожалению, был я.

Так вот, эту книгу написал тот самый молчавший. Просто спустя много лет. Теперь я и есть тот зануда, который поднимает руку и задаёт неудобный вопрос раньше, чем красиво полыхнёт. Этому, в сущности, и научила меня профессия. И этому я хочу научить тебя, чтобы тебе не пришлось, как мне, покупать каждый такой урок ценой собственного испорченного вечера.

Вот об этом и книга.

* * *

Для кого эта книга — и для кого нет

Скажу честно сразу, чтобы никто не потратил вечер впустую. Эта книга не для тех, кто уже с азартом спорит, что лучше, ArgoCD или Flux, и небрежно роняет в разговоре слово «GitOps». Если ты понял предыдущее предложение, и оно тебя не напугало, спокойно закрой книгу, она тебе мала. Тебе дорога на профильные конференции и в глубокие воды форумов, там тебе будут рады.

Эта книга для других.

Для того, кто слышит слово «DevOps» из каждого утюга, видел вакансии с зарплатами, от которых перехватывает дыхание, и думает по ночам одну и ту же мысль: «А вдруг это моё? Только с чего начать, непонятно. А спросить стыдно, вокруг все вроде бы умные, один я не в теме».

Для того, кто устал от работы, которую честнее всего описать словом «никак», и хочет наконец понять, правда ли можно войти в IT без волшебного диплома и десяти лет стажа за спиной.

Для того, кто уже в IT, тестировщиком, сисадмином, на поддержке, и чувствует, что упёрся в потолок и буксует на месте, а куда расти, не видно.

И просто для любопытного. Потому что DevOps это не только про серверы и провода. Это про то, как вообще устроены лучшие команды на свете. А это, согласись, интересно

даже тому, кто за всю жизнь не откроет ни одного терминала.

* * *

Семь правил. Один путь

Арнольд Шварценеггер[8] однажды написал книгу о семи правилах жизни. И, надо сказать, право он на это имел: человек приехал в Америку без денег и почти без английского, а стал чемпионом мира по бодибилдингу, голливудской звездой и губернатором Калифорнии. Когда такой человек делится правилами, к нему стоит прислушаться, даже если ты просто хочешь научиться спокойно катить код в продакшн.

Я не Арнольд. Даже не претендую. Но его семь правил, к моему собственному изумлению, описывают дорогу в DevOps точнее любого учебника. Поэтому книга выстроена на них: переосмысленных, переведённых с языка тренажёрного зала на язык терминалов, ночных инцидентов и тех самых пятничных деплоев. Вот они:

Семь глав. Одна дорога. А между ними кое-где короткие байки из жизни, передышки у того самого костра, с пивом и нервным смешком. С таких баек, если честно, в этой профессии всё и началось.

1. Составь чёткое видение: «Зачем тебе это вообще надо?»

2. Не разменивайся: фокус или смерть.

---

[8] **Арнольд Шварценеггер (род. 1947)** — австрийско-американский актёр, предприниматель, культурист и политик, многократный победитель конкурса «Мистер Олимпия» и бывший губернатор штата Калифорния.

3. Паши как проклятый: первые сто дней в DevOps.
4. Продавай: DevOps — это продукт, учись его продавать.
5. Переключайся: хватит читать. Начни ломать.
6. Закрой рот, открой глаза: прод — лучший учитель.
7. Бей зеркала: DevOps — это не про тебя.

* * *

Да, и про того парня с идеально автоматизированными ошибками. Помни его?

Причину фантомных падений в конце концов нашли. Спустя ещё неделю и некоторое количество седых волос. Виновата была опечатка. В переменной окружения. Один символ. Одна-единственная неверная буква, затаившаяся в настройке, в которую три года не заглядывал никто. Потому что, ну ты уже знаешь эту волшебную формулу, «оно же както работает».

Вот, если совсем коротко, и весь DevOps. Это когда ты смотришь в ту сторону, куда три года никто не смотрел. Не дожидаясь, пока оно упадёт. Красиво, на дашборде, в прямом эфире, с подсветкой.

И знаешь, с годами я понял, что это правило шире серверов. Что в ту сторону, куда давно никто не смотрит, тихо ломается не только прод. Так же тихо, незаметно, по одной неверной букве в день, ломается и человек, который про себя забыл. Поэтому книга, как ни странно, для технической

темы, получилась сразу про двоих. Про то, как не сломать прод. И про то, как при этом не сломать себя.

Об этом мы дальше и поговорим. Спокойно, по-честному, без галстука.

Ну всё. Хватит предисловий. Начнём.

Интерлюдия · Между правилами
Тот самый коммит в пятницу

У каждого инженера есть свой пятничный коммит[9]. Тот самый. О котором потом рассказывают новичкам вполголоса, как страшилку у костра. Есть он и у меня. Расскажу, раз уж мы договорились быть честными друг с другом.

Пятница. Без десяти шесть. Я уже мысленно надел куртку и стоял одной ногой в выходных. И тут заметил в коде мелочь. Совсем крошечную. Опечатку в тексте, которую видел пользователь. Не та буква. Дел на десять секунд. Одна строчка.

Я знал правило. Его знают все. В пятницу вечером в прод не катят. Это не суеверие. Это народная мудрость, выстраданная поколениями, как и «не свисти в доме — денег не будет». За этим правилом стоят тысячи испорченных выходных и остывших шашлыков. Я всё это знал.

Но правка выглядела такой безобидной. А я так уверился в своей правоте. Голосом, которым судьба обычно заманивает людей в ловушку, я сказал себе: «Да что тут может сломаться? Одна буква».

---

[9] **Коммит (Commit)** — зафиксированное изменение в системе контроля версий, сохраняющее текущее состояние проекта и позволяющее отслеживать историю разработки.

Я даже не стал гонять тесты. Тесты[10] нужны неуверенным в себе, а я был уверен. Залил изменение. Захлопнул ноутбук. Поехал домой с чувством человека, который сделал миру маленькое добро.

Прод[11] не упал.

В этом и была вся изощрённость пытки. Он не упал сразу. Быстрое падение было бы слишком милосердным. Он держался. Весь вечер держался. Я успел доехать, поужинать, расслабиться, поверить, что пронесло. А в одиннадцатом часу телефон засветился в темноте.

Оказалось, моя «одна буква» жила в файле, который при сохранении подтянул за собой обновление совсем другой, дальней части системы, той, к которой я даже не прикасался и о существовании которой почти забыл.

Эффект бабочки в чистом виде. Я поправил опечатку в подписи к кнопке, и через цепочку из пяти невидимых звеньев уронил оформление заказов. Кнопка стала идеальной. Заказы перестали оформляться.

Отката[12] на ту версию настроено не было. Так что я сидел в семейных трусах перед ноутбуком, в одиннадцать вечера

---

[10] **Тесты (в ИТ)** — автоматические или ручные проверки, предназначенные для выявления ошибок и подтверждения корректной работы программы.

[11] **Прод (Production)** — рабочая среда, в которой приложение доступно конечным пользователям и выполняет реальные задачи.

[12] **Откат** — возврат системы, программы или изменений к предыдущему рабочему состоянию после неудачного обновления, ошибки или сбоя. Используется для восстановления стабильной работы и отмены нежелательных изменений.

в пятницу, и сорок минут искал, как одна буква могла обрушить то, что к ней не имеет отношения.

Нашёл. Починил. Руки слегка потряхивало — от той особой злости на самого себя, не от страха, которая знакома каждому, кто хоть раз был абсолютно уверен и абсолютно неправ.

Я не буду навязывать тебе мораль. Ты сам заведёшь себе личное правило про пятницу. И, боюсь, заведёшь его ровно той же ценой, что и я. Так уж это работает: чужой обжёгшийся палец ничему не учит, пока не обожжёшь свой. Скажу только одно, тихо. С тех пор, когда внутренний голос говорит мне: «Да что тут может сломаться», я слышу в нём не уверенность. Я слышу звонок телефона в темноте. И иду гонять тесты.

Правило 1. Составь чёткое видение *«Зачем тебе это вообще надо?»*

«Видение без действия — это мечта. Действие без видения — это кошмар».

Так звучит японская пословица, которую все цитируют в LinkedIn.

Серверная[13] находилась в подвале, и летом там было под тридцать.

Кондиционер, рассчитанный на стойки, а не на людей, гнал холод в железо и мной не интересовался совершенно. Я сидел на корточках перед вскрытым корпусом, зажав в зубах фонарик, как пират зажимает зубами нож, и менял блок питания, сдохший в самый неподходящий момент. Других моментов они, впрочем, себе не выбирают. Мне было слегка за двадцать, на мне болтался чужой халат на размер больше с биркой «техник», и я ещё не знал слова, ради которого ты открыл эту книгу.

Его для меня тогда не существовало. Где-то оно уже жило, как раз в те годы на какой-то конференции в далёкой Бельгии двое усталых инженеров впервые произнесли его вслух, но до подвала большого банка, где я менял блоки питания и

[13] **Серверная** — специализированное помещение, в котором размещаются серверы, сетевое оборудование и другие элементы ИТ-инфраструктуры.

таскал коробки тяжелее собственных амбиций, оно дойти не успело. Я торчал самым нижним винтиком в самой нижней службе с длинным казённым названием про обслуживание электронного оборудования. Если что-то в здании жужжало, грелось и время от времени умирало, это было по мою душу.

И, знаешь, мне нравилось.

Мне нравилось, что у железа есть характер. Что сервер, который все считали мёртвым, можно вернуть к жизни, если не паниковать и думать по порядку. Нравился старший, немногословный человек, который однажды посмотрел, как я наугад дёргаю провода, отобрал у меня отвёртку и сказал фразу, которую я тащу за собой через всю карьеру: «Сначала пойми, почему сломалось. Чинить будешь потом». Он не знал, что цитирует будущую философию целой индустрии. Он просто не любил чинить дважды.

Потом я стал сисадмином[14], тем самым, которого все зовут, когда «интернет не работает», и кого никто не зовёт, когда работает. Потом инженером. Потом ведущим инженером. Потом человеком, к которому приходят, когда непонятно вообще ничего и нужно, чтобы кто-то спокойно сказал, что делать. Дорога заняла годы и петляла как угодно, только не по прямой: с тупиками, с ночными авралами, с парой увольнений «по собственному», с проектами, которыми гор-

[14] **Сисадмин** — разговорное название системного администратора, специалиста, отвечающего за настройку, поддержку и бесперебойную работу компьютерных систем, серверов и сетей.

жусь, и проектами, о которых лучше молчать.

Рассказываю я это не для того, чтобы произвести впечатление. Скорее наоборот. Я хочу, чтобы ты с первой страницы знал обо мне одну вещь: я начинал снизу. С отвёртки, с коробок, с подвала, в котором летом плюс тридцать. Без диплома «DevOps-инженера»: его, к слову, не существует и сегодня. Без связей, без правильного вуза, без человека, который провёл бы меня за руку. Если эту дорогу прошёл я, по ней пройдёшь и ты. Это не лозунг с мотивационного плаката. Это просто статистика, в которой я сам одна из точек, причём не самая одарённая.

Но, прежде чем мы возьмём в руки первый инструмент, я задам вопрос важнее всех инструментов, вместе взятых. Не «что учить». Не «сколько платят». А зачем тебе это.

Пока у тебя нет ясной картинки того, куда ты идёшь, любое движение остаётся суетой, даже если ты при этом очень быстро перебираешь ногами. Поэтому отсюда начинаем и мы. А чтобы тебе было из чего складывать своё видение, давай сперва по-честному разберёмся, что такое DevOps на самом деле — без глянца тренинговых центров и без буллшита из вакансий.

* * *

Что такое DevOps

Когда я впервые услышал это слово, я решил, что это должность. Логика выглядела железной: есть программисты, есть сисадмины, а есть вот эти, новенькие, — девопсы. Сидят где-то посередине, едят пиццу и деплоят[15] в продакшн. Примерно так это слово понимают почти все за пределами IT. И, что куда обиднее, добрая половина людей внутри.

Один мой знакомый, назову его Серёгой, потому что его и вправду зовут Серёга, работал проджект-менеджером в маленькой веб-студии. Однажды клиент прислал письмо в две строки: «Нам нужно внедрить DevOps». Серёга, человек обстоятельный, ответил так, как отвечают на любую задачу: «Хорошо, пришли ТЗ».

Клиент прислал. Две страницы. Серёга прочитал. Перечитал. Потом снял трубку.

— Слушайте, — осторожно начал он, — а вы понимаете, что DevOps — это не программа, которую можно установить?

— Ну как же, — искренне удивился клиент. — Нам вон в соседней компании установили. Всё отлично работает.

Это не анекдот, который я где-то услышал. Это реальный разговор, и я до сих пор не уверен, что Серёга его пережил

---

[15] **Деплой (Deploy) —** процесс развёртывания новой версии приложения или сервиса на сервере и ввода её в эксплуатацию.

без потерь для веры в человечество.

Так вот, запомни раз и навсегда. DevOps не программа. И не должность, хотя должности с таким названием существуют, и к ним мы ещё вернёмся, когда дело дойдёт до твоего первого оффера. DevOps, по сути, способ устроить работу так, чтобы программы рождались быстро, выкатывались без валидола и не падали каждую пятницу в семь вечера, когда все уже мысленно на шашлыках.

Само слово склеено из двух: Development, разработка[16], и Operations, эксплуатация[17]. Двух племён, которые исторически друг друга недолюбливали, мягко говоря. Я знаю это не из учебника. Я был по обе стороны баррикады и могу описать каждую изнутри.

Эксплуатация хочет ровно обратного. Чтобы не трогали. Чтобы всё стояло как стоит, и желательно вечно. И у неё, поверь, есть на это веская причина. Сейчас расскажу. История из жизни, из моей, к сожалению.

Пятница. Без пяти шесть. Молодой разработчик, назовём его Костей — допиливает крошечную правку. Одну строчку. Ну что может пойти не так от одной несчастной строчки? Тесты он, разумеется, не гоняет: тесты нужны неуверенным в себе. Заливает. Захлопывает ноутбук. Едет на дачу. Жарить

---

[16] **Разработка (Development)** — среда, в которой создаётся, изменяется и тестируется программное обеспечение до его выпуска для пользователей.

[17] **Эксплуатация** — использование и сопровождение программного обеспечения или ИТ-инфраструктуры после ввода в рабочую среду.

мясо, смотреть на огонь, быть счастливым человеком. Телефон ставит на беззвучный. Выходные — это святое.

В три часа ночи падает прод.

Звонит телефон.

Но не у Кости. У Кости беззвучный, костёр и чистая совесть.

Телефон звонит у дежурного инженера. То есть у меня. Потому что дежурным в ту ночь был, конечно же, я.

Я сажусь в кровати. Темно. Не понимаю ничего. На экране алерты[18], как новогодняя гирлянда, только злая. Открываю логи. Ищу. Сорок минут ищу, на что именно сегодня обиделся прод. И нахожу. Одну строчку. Ту самую. Костину.

Утром Костя проснётся отдохнувшим, румяным, пахнущим дымком. Откроет рабочий чат. Увидит там двести сообщений капслоком. И напишет, абсолютно искренне: «Ой, а у меня же работало».

Вот по этой фразе «у меня работало» и проходит граница. Линия фронта длиной в целую индустрию.

И вот что важно понять. Костя не злодей. Он не хотел зла, он хотел как лучше, хотел движения. И я не зануда, тормозящий прогресс, я просто хотел поспать. Мы оба были правы. Виноват был не человек. Виновата была стена между нами. Глухая бетонная стена, по одну сторону которой пишут, а по другую расхлёбывают, и сквозь которую не проходит ни од-

---

[18] **Алерт** — автоматическое тревожное уведомление: система сама сигналит, что что-то пошло не так или вот-вот пойдёт. От англ. alert, «тревога».

но слово, кроме ночного звонка.

И вот в эту минуту в историю входит DevOps. Без пафоса. Без презентации на сорок слайдов. Он просто оглядывает обе стороны и тихо произносит: «Ребята. А может, хватит воевать? Может, „часто“ и „надёжно“ вовсе не противоположности? Может, если выстроить процесс по-человечески, выкатывать получится и быстро, и так, чтобы дежурный спал?».

Звучит банально. Знаю. Но в своё время это перевернуло всё. И многие компании, поверь, переживают этот переворот до сих пор со скрипом, с сопротивлением и лёгким запахом гари.

Так что же он предлагает на деле, этот DevOps?

Если совсем коротко: сломать стену. Ту самую. Между Костей и мной.

Чтобы тот, кто пишет код[19], и тот, кого этот код будит ночью, перестали быть двумя враждующими племенами. Чтобы стали одной командой с общей судьбой. Чтобы Костя видел свою строчку — и то, что с ней случится в три часа ночи. Чтобы я, дежурный, знал о его правке заранее и успевал сказать слово, а не получал её как сюрприз с того света. Одно общее окружение[20]. Один общий пейджер[21]. Одна общая

[19] **Код** — текст команд, который человек пишет для компьютера на специальном языке. По сути, пошаговый рецепт: что и в каком порядке машина должна сделать. Программа — это и есть код, который запустили.

[20] **Окружение** (среда) — все условия, в которых работает программа: операционная система, версии вспомогательных частей, настройки. Грубо говоря,

ответственность. И, как следствие, один общий интерес — сделать хорошо.

Вот, в сущности, и весь DevOps.

Всё остальное (Docker, пайплайны, облака, Kubernetes и прочие страшные слова из вакансий) лишь инструменты, которыми эту стену разбирают по кирпичику. Способы. Вот как эту простую и очень человеческую идею воплощают на практике: разговаривай друг с другом и отвечай за результат вместе.

У этой идеи есть почти официальный день рождения, и за ним стоит пара живых людей, а не безликая «индустрия». В 2008 году инженер по имени Эндрю Клей Шафер заявил на одной agile[22]-конференции маленькое обсуждение про, как он это назвал, гибкую инфраструктуру. Тему сочли такой неинтересной, что на встречу почти никто не пришёл, и, по легенде, сам Шафер, решив, что провалился, на неё едва не забил. Но пришёл один человек. Патрик Дебуа. Тот самый, которого годами выматывала вечная вражда на стыке разра-

---

«комната», в которой программа живёт. В разных комнатах мебель стоит по-разному — поэтому и рождается знаменитое «у меня работает, а у тебя нет».

[21] **Пейджер** — здесь в переносном смысле: дежурный телефон, который будит инженера среди ночи, когда что-то сломалось. (Когда-то пейджер был маленьким устройством для срочных сообщений; слово осталось как символ «тревожного вызова».)

[22] **Agile (от англ. agile — «гибкий»)** — гибкая методология управления разработкой, основанная на быстрой поставке результата, тесном взаимодействии с заказчиком и готовности к изменениям.

ботки и эксплуатации, где каждый релиз[23] маленькая война, за общий результат не отвечает никто. Виноватым просто оказывается тот, кто стоял ближе. Эти двое нашли друг друга, разговорились, и оказалось, что болит у них одно и то же.

А через год, в 2009-м, Дебуа собрал в бельгийском Генте конференцию для таких же измученных. Думал, как её назвать покороче, и склеил два враждующих слова в одно. Developers[24] и operations[25]. DevOps. Так у боли впервые появилось имя. Потом из имени выросло движение. Потом целая индустрия с конференциями, зарплатами и вот этой книгой в придачу.

Хотя, честно сказать, сама практика жила задолго до названия. В крошечных командах, в стартапах, там, где просто не было лишних людей, чтобы делить их на «тех, кто пишет» и «тех, кто запускает». Один несчастный делал всё сам. И поневоле учился делать это хорошо. Я, кстати, был как раз таким несчастным. И до сих пор за это судьбе благодарен.

А почему об этом слове кричат на каждом углу именно сейчас? Потому что сменился сам темп жизни.

---

[23] **Релиз** — выпуск новой версии программы для пользователей; момент, когда сделанные изменения официально становятся доступны всем.

[24] **Developers (разработчики)** — специалисты, которые создают и изменяют программный продукт: пишут код, добавляют новые функции и исправляют ошибки.

[25] **Operations (эксплуатация)** — специалисты, которые отвечают за работу продукта после выпуска: серверы, инфраструктуру, стабильность, доступность и восстановление после сбоев.

Двадцать лет назад программу обновляли раз в полгода. Люди шли в магазин и покупали коробку с диском. Притаился внутри баг, что ж, ждали следующей коробки. Сегодня приложение в твоём телефоне обновляется по нескольку раз в день. А крупнейшие интернет-компании выкатывают изменения в продакшн буквально каждые несколько секунд. Даже не каждый день и не каждый час — пока ты читал это предложение. Конкурент выпускает новую фичу раз в неделю, а ты раз в квартал? Тогда через год ты будешь разглядывать его в зеркале заднего вида. Если он вообще согласится взять тебя в это зеркало.

Скорость стала оружием. А DevOps, по сути, про то, как мчаться и не разбиться. Про скорость без потери надёжности. Вот за это и платят такие деньги.

И вот почему, между нами, я довольно спокойно отношусь к модным разговорам, что «ИИ скоро заменит инженеров». Часть рутины заменит, наверняка. И слава богу, руки развяжет. Но объяснять заказчику в три часа ночи, почему лёг прод, держать удар под давлением, брать на себя решение, за которое потом отвечать, — пока что приходится живому человеку. Инструменты меняются каждые пять лет. Ответственность не меняется никогда.

* * *

Две компании. Один день. Разные вселенные

Сейчас по-хорошему здесь должны появиться графики. Стрелочки вверх. Метрики. Какие-нибудь серьёзные исследования. И обязательно квадранты Гартнера, чтобы всё выглядело совсем убедительно. Но, если честно, от таких вещей глаза обычно начинают закрываться раньше, чем приходит понимание. Поэтому давай проще. Давай проживём один обычный рабочий день в компании без DevOps. А потом ещё один, в компании, где DevOps стал частью культуры. Без графиков. Без умных слов. Просто посмотрим, как проходят их будни.

Потому что за годы работы я насмотрелся на оба варианта столько, что могу пересказать каждый почти по минутам. Где начинается паника. Где появляются герои. Где кто-то в очередной раз произносит: «А кто вообще это менял?».

И самое интересное вот что. На бумаге эти компании могут выглядеть одинаково. Те же разработчики. Те же серверы. Те же дедлайны. Иногда даже те же инструменты. А живут они почему-то совершенно по-разному.

Давай посмотрим почему. И, поверь, оба этих дня я видел вблизи столько раз, что мог бы пересказывать их во сне.

Вселенная А.

Компания называется красиво: «Технологии Будущего». Полсотни человек, восемь лет на рынке, амбиции до Луны. DevOps здесь нет. Зато есть дядя Валера.

Дяде Валере под шестьдесят. И руки у него, как у человека, который тридцать лет что-то чинил. Крупные, в мелких белых шрамах. С той траурной каёмкой под ногтями, которую не берёт никакое мыло, потому что это уже часть биографии, не грязь.

Он пришёл в компанию, когда вся она помещалась в двух комнатах и одном кредите. И помнит каждый сервер[26] по характеру, не по адресу в системе: этот капризный, этот тянет за двоих, а вон тот лучше лишний раз не трогать, у него «настроение».

Пароли Валера хранит в толстом блокноте под клавиатурой. И это, как ни смешно, самая защищённая система во всей конторе: блокнот в единственном экземпляре, а ключ к нему в одной немолодой голове.

На двух мониторах у Валеры всегда одно и то же. На левом бегут логи. На правом — «Косынка». Не спеши называть это ленью. Это философия, выстраданная годами.

Валера давно понял простую вещь: здесь паникуют всегда, так устроена здешняя атмосфера. И научился не входить в неё заранее. Как опытный пловец ложится на волну, а не дёр-

[26] **Сервер** — мощный компьютер, который всегда включён и обслуживает других: хранит сайт, отвечает на запросы пользователей. «Боевой» сервер работает для реальных клиентов, «тестовый» — его тренировочная копия, где можно безопасно проверять изменения.

гается в ней. Косынка его медитация. Его островок порядка в мире, где порядка нет нигде.

Девять утра.

Артём, разработчик, открывает ноутбук и заглядывает в трекер задач[27]. Там сорок семь задач «в работе». Из них он помнит хорошо если двенадцать. Остальные завели коллеги, менеджеры и кто-то по имени «Система», чьи действия расшифровке не поддаются.

Артём берётся за первую попавшуюся, пишет код. Код работает у него на ноутбуке. Запомни эту деталь, она ещё выстрелит, как ружьё в пьесе.

Половина двенадцатого.

Тестировщица Маша находит баг в фиче, которую Артём сдал три недели назад. Артём смотрит и произносит фразу, древнейшую в истории профессии. Её, кажется, выбивали ещё на глиняных табличках: «У меня всё работает».

У Маши не работает. И оба правы.

Это называется «расхождение окружений», и любой порядочный DevOps решает эту проблему одной из самых первых. Но в «Технологиях Будущего» о таком не слышали.

Двадцать минут переписки выясняют страшное: на тесто-

---

[27] **Трекер задач** — программа-список, где команда заводит и отслеживает рабочие задачи: кто что делает, на каком этапе, что готово. Цифровая доска с наклейками-делами.

вом сервере стоит другая версия[28] библиотеки[29]. Кто её поставил, когда и зачем — тайна, покрытая мраком и косынкой.

Артём идёт вниз. К Валере.

— Валер, там на тесте библиотека не та.

— Знаю, — говорит Валера, не отрываясь от карт. — Маша во вторник просила обновить. Я обновил.

— Но теперь у меня тест не проходит.

— Ну так и обнови у себя.

Соломоново решение. Артём уходит обновлять. И это съест у него полтора часа, потому что на его машине вспыхнет конфликт зависимостей[30]. Это когда одна библиотека требует вторую версии 2.1, а третья требует ту же вторую, но версии 3.4. И обе абсолютно уверены в своей правоте. Победителей в этой войне не бывает. Бывают только вкладки Stack Overflow[31] и тихое страдание.

---

[28] **Версия** — конкретный «выпуск» программы или её части, обычно с номером (например, 2.1 и 3.4). Чем больше номер, тем новее. Разные версии могут вести себя по-разному и не всегда ладят между собой.

[29] **Библиотека** — готовый кусок чужого кода, который подключают к своей программе, чтобы не писать всё с нуля. Как готовые ингредиенты: не выращиваешь пшеницу сам, а берёшь муку.

[30] **Конфликт зависимостей** — ситуация, когда разные части программы требуют разные, несовместимые версии одного и того же вспомогательного компонента. Все по-своему правы, а ужиться вместе не могут — и программа не запускается.

[31] **Stack Overflow** — крупнейший сайт-форум, где программисты задают вопросы и отвечают друг другу. Народная «скорая помощь» для всех, у кого что-

Половина пятого.

В «Технологиях Будущего» раз в две недели случается деплой. Это событие отмечено в общем календаре красным. Не праздник. Штормовое предупреждение. Все знают, что сегодня будет плохо. Вопрос только в том, насколько.

Деплой Валера катит руками. По инструкции, которую написал три года назад и с тех пор ни разу не открывал. И так помнит.

Первый заход падает. Валера сорок минут смотрит в логи — спокойно, без суеты, как смотрят на знакомого зверя. И находит причину. Опечатка в конфиге[32]. Одна буква в слове localhost[33]. Сервер не смог найти сам себя, обиделся и лёг.

Второй заход проходит. Без четверти семь разработчики расходятся по домам с лицами людей, переживших небольшую, но настоящую катастрофу.

Четверть двенадцатого ночи.

Звонит телефон Валеры. Сайт упал. И не из-за деплоя. Просто так, от старости и недосмотра: на сервере кончилось

то не работает.

[32] **Конфиг** — файл с настройками программы (от «конфигурация»): адреса, режимы, пароли. Одна неверная буква здесь — и программа может не запуститься.

[33] **localhost** — служебное имя, которым компьютер обозначает сам себя («этот же компьютер»). Ошибся в написании — и программа в буквальном смысле не может найти саму себя.

место на диске.

Никто не следил. Мониторинга нет. Узнали от пользователя, который написал в соцсети компании коротко и ёмко: «Эй, у вас сайт не пашет».

Валера встаёт, одевается, едет в офис на такси. В час ночи.

И вот что важно понять: это не геройство. Это болезнь. Системная. Которую почему-то принято называть красивым словом «преданность».

Половина третьего ночи.

Сайт поднят. Валера едет домой. Спать те несколько часов, что у него остались. А завтра в девять он снова сядет за два монитора. Логи и «Косынка». И будет прав. Потому что человек, который в час ночи едет чинить чужой диск, заслужил право на пасьянс до конца своих дней.

Вселенная Б.

Компания называется «Пульс». Людей примерно столько же. Только атмосфера другая. И ещё здесь нет дяди Валеры. Есть Катя.

Самое забавное, что за весь день ты её почти не увидишь. Она не бегает по коридорам с криками «всё пропало». Не собирает вокруг себя толпу. Не устраивает героических спасений в последний момент. В какой-то момент ты вообще начнёшь сомневаться, работает ли она сегодня. Работает. Просто у неё всё происходит сильно раньше того момента, ко-

гда остальные замечают проблему. Пока разработчики спокойно выкатывают обновления, где-то незаметно отрабатывают проверки. Пока менеджер рассказывает клиенту про новую функцию, где-то уже собирается следующая версия. Пока вся компания занимается своей работой, Катя делает так, чтобы эта работа не спотыкалась о технику каждые полчаса. К обеду ты вдруг поймаешь себя на странной мысли. За весь день никто ни разу не произнёс фразу: «Срочно собираемся в переговорке, всё горит». Именно поэтому о Кате почти не говорят.

Девять утра.

Денис, разработчик, открывает ноутбук. А там его уже ждёт сообщение от бота мониторинга: ночью нагрузка на один из серверов поднималась до семидесяти восьми процентов, причина установлена — плановый бэкап, всё в норме, спи спокойно. Денис кивает и идёт за кофе.

О серверах он не думает. Не оттого, что ему наплевать. А оттого, что система устроена так: случись что серьёзное, он узнал бы первым. Причём от самой системы, а не от рассерженного пользователя. Это называется «наблюдаемость»[34]. Здоровая система умеет рассказывать о себе сама. Не дожидаясь, пока кто-то заметит беду по жалобе в интернете.

---

[34] **Наблюдаемость** — свойство системы «рассказывать о себе»: показывать изнутри, что с ней происходит и почему, чтобы проблему было видно сразу, а не постфактум по жалобам. Здоровый организм, который сам сообщает, где болит.

Денис открывает задачу, пишет код и тут же, не отходя от кассы, пишет к нему тесты[35]. Здесь так заведено. И не ради бюрократии — просто это банально быстрее, чем потом ловить те же баги руками бедной Маши из соседней вселенной.

Рабочая среда поднимается у Дениса одной короткой командой. И содержит ровно те же версии всего, что стоят на тестовом и боевом серверах.

В «Пульсе» фраза «у меня работает» не вызывает желания кого-нибудь придушить. Потому что «у меня» и «на сервере» здесь означают буквально одно и то же. Звучит до зевоты скучно. На деле — экономит каждому по нескольку часов жизни в неделю. А из этих часов, если задуматься, жизнь и состоит.

Денис заканчивает фичу и просит влить её в общую ветку[36].

Через три минуты — пока он наливает второй кофе — в его запрос[37] само собой прилетает целое досье. Все автотесты прогнаны и зелёные. Посчитано, какую долю кода они

---

[35] **Тесты (автотесты)** — маленькие проверки, которые автоматически прогоняют программу и убеждаются, что она работает правильно. «Авто» — значит без участия человека, при каждом изменении.

[36] **Ветка** — отдельная «дорожка», на которой программист спокойно делает своё изменение, не мешая общему коду. Когда работа готова и проверена, ветку «вливают» в общую. Как черновик, который потом сводят в общий документ.

[37] **Запрос на слияние / на правку (англ. pull request)** — официальная просьба влить своё изменение в общий код. Это контрольная точка: прежде чем влить, изменение проверяют автоматика и живой коллега.

покрывают[38]. Статический анализатор[39] придрался к одному лишнему пробелу. И уже назначен человек, который посмотрит код свежим глазом.

Никто пальцем не пошевелил — всё сделала настроенная заранее машинерия[40]. Коллега бросает один дельный комментарий, Денис правит, код уходит в дело.

Два часа дня.

Деплой в «Пульсе» происходит прямо сейчас, пока ты читаешь. И происходил в 11:34. И будет ещё в 16:20. Каждый раз, когда код попадает в общую ветку, сама собой запускается длинная заботливая цепочка. Собрать. Прогнать тесты. Проверить на дыры в безопасности. Упаковать. Выкатить на репетиционный стенд. Прогнать ещё раз. И, если всюду горит зелёный, — тихо выложить на боевой.

Весь путь от строчки кода до живого продукта занимает двенадцать минут. Денис в это время обедает. Катя тоже обедает. Она не стоит над душой у каждого деплоя. Она постро-

---

[38] **Покрытие кода** — мера того, какую часть программы реально проверяют тесты. Чем выше покрытие, тем меньше «слепых зон», где ошибка может спрятаться.

[39] **Статический анализатор** — программа, которая читает чужой код и заранее указывает на подозрительные места и неаккуратности, даже не запуская его. Строгий корректор для программистов.

[40] **Машинерия** — здесь образно: вся совокупность настроенных автоматических механизмов, которые делают работу сами. Не один скрипт, а целый отлаженный «механизм из механизмов».

ила эту цепочку один раз, несколько месяцев назад, и с тех пор та работает за неё. Вот это и есть автоматизация[41]. Не «скриптик», который делает одно движение». А живой механизм, который сам, без нянек, проводит изменение весь путь — надёжно и одинаково. В сотый раз так же, как в первый.

Четыре часа дня.

На боевом сервере начинает потихоньку расти время ответа одного из запросов. Пользователи ещё ничего не чувствуют. Но система уже знает. И уже сообщает: прилетает алерт, что задержка подросла.

Катя глядит на дашборд, в три клика находит виновника. Свежий деплой принёс запрос к базе[42] без индекса[43]. На маленьких данных незаметно, а под нынешней нагрузкой уже кусается.

Катя пишет Денису. Тот говорит: «Ой». Добавляет индекс, заводит запрос на правку. Через четырнадцать минут

---

[41] **Автоматизация** — передача рутинной ручной работы программам, чтобы она делалась сама — быстро и одинаково каждый раз. Человек настраивает один раз, машина повторяет всегда.

[42] **Запрос к базе данных** — обращение программы к хранилищу данных, чтобы что-то найти, добавить или изменить. Например: «покажи все заказы за сегодня».

[43] **Индекс (в базе данных)** — служебный «указатель», который помогает базе находить нужное быстро, не перебирая всё подряд. Как алфавитный указатель в конце толстой книги: без него пришлось бы листать каждую страницу. Без индекса под большой нагрузкой запрос работает медленно — именно это и случилось у Кати.

починка на боевом, задержка вернулась к норме, пользователи так ничего и не заметили. На весь разбор ушло двадцать минут. И ни единого крика. Никто не искал виноватого. Нашли, починили, коротко записали в постмортем: что случилось, почему, что сделали и как сделать, чтобы не повторилось.

В «Технологиях Будущего» точно такой же сбой нашли бы часа через два по жалобам. Чинили бы до полуночи. А виноватым по итогам назначили бы Артёма.

Шесть вечера.

Денис закрывает ноутбук. За день он выкатил новую фичу, поймал и убил баг на боевом. При этом успел и пообедать, и уйти домой к семье вовремя. Катя уходит тоже. И ночью её телефон будет молчать. Если что-то всё же стрясётся, сперва сработает автоматическое восстановление. А уж потом разбудят дежурного по графику. И дежурство это ходит по кругу, чтобы никто не тащил ночную ношу в одиночку до выгорания.

Завтра будет ещё несколько деплоев. Таких же тихих и незаметных, как сегодняшние. Это не магия. Это инженерия. И разница тут принципиальная.

Что мы с тобой только что увидели?

Два рабочих дня. Две компании почти одного размера. Одна индустрия, один город.

В первой хаос, который держится исключительно на том, что есть живой человек, готовый в час ночи ехать чинить диск. И вот мысль, ради которой я всё это затеял: дядя Валера не злодей и не бездарь. Наоборот. Он сильный специалист. Попробуй-ка сам в полночь по логам найти, отчего лёг сервер. Он хороший профессионал в плохой системе. И таких Валер я знал десятки. Больше скажу: я сам едва не стал одним из них. Незаменимым. Выжатым. Гордым своей незаменимостью, как медалью, которая на самом деле кандалы. Проблема не в людях. Проблема в том, как выстроены (а вернее, не выстроены) процессы вокруг этих людей.

Во второй вселенной система работает независимо от того, кто сегодня в офисе. Кто слёг с температурой. Кто улетел в отпуск. И разница между двумя мирами вовсе не в инструментах. Серверы, базы, система контроля версий есть и там, и там, железо примерно одинаковое. Разница в том, как всё это связано между собой. И кто отвечает за итог.

Запомни принцип, который стоит дороже половины этой книги. Хорошая система делает правильное действие лёгким, а неправильное делает трудным. Плохая устроена наоборот.

У Валеры задеплоить мимо всех проверок легко: просто крикни вниз. А задеплоить с проверками невозможно, потому что проверок нет.

У Кати всё наоборот. Проскочить мимо проверок физически нельзя, пайплайн не пустит. Зато сделать правильно

так же просто, как нажать одну кнопку.

Вот и весь секрет. **DevOps заключается в искусстве строить такие системы, в которых правильный путь оказывается путём наименьшего сопротивления. Не контролировать людей. Не назначать виноватых. Не требовать героизма. А сделать так, чтобы хорошо работать было просто.**

И сразу, чтобы ты не обольщался.

Большинство компаний, где тебе выпадет работать, живут не во вселенной А и не во вселенной Б. А где-то между Валерой и Катей. Есть автоматизация, но дырявая. Есть мониторинг, но настроенный криво и орущий по пустякам, отчего его все давно отключили. Есть честные попытки навести порядок, но старые привычки сидят за соседним столом и тихо саботируют любое новшество фразой «мы так всегда делали». Это нормально. Это и есть реальность.

И именно поэтому человек, который умеет двигать компанию от вселенной А к вселенной Б (медленно, по сантиметру, через сопротивление и усталость) стоит дорого. Это и есть работа DevOps-инженера в его лучшем издании. **Не «нажимать кнопки в облаке». А менять то, как целая компания доставляет пользу живым людям.**

* * *

Почему за это платят

Вот теперь, когда мы прожили оба дня и познакомились с Валерой лично, можно спокойно поговорить о деньгах. Спокойно, потому что бизнес существо предельно прагматичное. Он не покупает философию, не умиляется красивым словам и не платит за модность. Он платит за результат, который можно потрогать, а лучше — пересчитать. И вся прелесть DevOps в том, что его результат пересчитывается в деньги пугающе легко. Иногда в ту же ночь.

Помни ночной звонок Валере и сайт, упавший оттого, что на диске банально кончилось место? А теперь представь, что это случилось не в глухой вторник с его полутора посетителями на сайте. Это случилось в первую минуту большой распродажи, когда туда разом ломанулась вся страна. В компаниях, которые я наблюдал вблизи, одна такая ночь стоила нескольких миллионов рублей выручки за два-три часа белого экрана. Плюс репутация, которую в рубли перевести трудно, а ощущается она ещё дороже: ушедший покупатель редко возвращается, чтобы дать второй шанс. Вот за предотвращение таких ночей и платят. И платят, если присмотреться, за три довольно простые вещи.

Платят за скорость.

Есть такой авторитетный ежегодный отчёт о состоянии DevOps в мире, его делает исследовательская группа

DORA[44] (и мы к нему ещё не раз вернёмся). Из года в год он показывает одно и то же: лучшие команды выкатывают код в продакшн по нескольку раз в день, средние — раз в неделю, отстающие — раз в месяц и реже.

Что это значит на языке бизнеса? Простую и беспощадную арифметику. Пока твоя команда таскает одну фичу из спринта в спринт, конкурент успевает выкатить её, проверить на живых людях три варианта, оставить лучший и взяться за следующую. Отставание в скорости оборачивается отставанием на рынке, в выручке, в самой жизни компании. Бизнес чувствует это кожей задолго до того, как научится объяснять словами.

Платят за надёжность.

По известной и многократно проверенной оценке, добавление каких-то ста миллисекунд к загрузке страницы роняло продажи примерно на один процент. Эту цифру, сто миллисекунд против процента, в индустрии повторяют не первый десяток лет, и она крепкая.

А дальше начинается арифметика, от которой кружится голова. Представь сам. Если перевести этот процент в деньги такой махины, как Amazon, то одна секунда задержки на загрузке оборачивается, по ходовым оценкам, суммой под миллиард, а то и за миллиард долларов в год недополучен-

---

[44] **DORA (DevOps Research and Assessment)** — исследовательская группа, изучающая эффективность IT-команд и DevOps-практик. Известна своими метриками скорости и надёжности разработки.

ной выручки. Точную цифру тут тебе не назовёт никто, она кочует из статьи в статью в разных вариантах, так что считай это прикидкой на салфетке, а не строкой из годового отчёта. Но даже как прикидка она оглушает.

Миллиард. За секунду, которую ни один живой человек даже не успел бы заметить.

Ты, конечно, не Amazon. Но и твой простой в час пик считается во вполне реальных деньгах, возможно, просто с меньшим числом нулей.

А разница между мирами тут вот какая. У зрелых команд время возвращения сервиса к жизни после сбоя измеряется минутами, у отстающих — днями, а порой и неделями. Одна ночь у Валеры против двадцати Катиных минут. **Посчитай эту разницу применительно к своему бизнесу, и ты получишь готовое, неоспоримое обоснование зарплаты инженера.**

И, наконец, **платят за спокойствие**.

Звучит как нечто нематериальное, а материальнее некуда. Валера, который один знает все пароли и держит всю инфраструктуру в собственной голове, — это бомба замедленного действия с очень обаятельным детонатором, а не актив компании.

Что будет с конторой, если он уйдёт на пенсию? Если просто сляжет с гриппом в самую горячую неделю года? DevOps в том числе про то, чтобы знание об инфраструктуре жило в коде, в документации, в автоматических процессах, которые

можно прочитать, повторить и передать, а не в одной усталой голове. У этого есть имя — «инфраструктура как код» (и мы разберём её подробно дальше).

Бизнес, который однажды обжёгся, платит большие деньги именно за то, чтобы больше никогда не зависеть от незаменимых людей. Потому что незаменимый человек — самый дорогой и самый непредсказуемый ресурс на свете.

Если тебе хочется цифр, пожалуйста. Возьмём один конкретный, уже не новый, но очень показательный замер. В отчёте DORA за 2019 год команды с высокой DevOps-зрелостью сравнили с отстающими по нескольким простым параметрам. Результат вышел почти неприличный. Зрелые команды выкатывали изменения в продакшн примерно в двести восемь раз чаще. Поднимали упавший сервис после сбоя примерно в две тысячи шестьсот раз быстрее. И при этом, что особенно обидно для отстающих, ломали в среднем в семь раз реже. То есть катили чаще и при этом падали меньше.

И нет, две с лишним тысячи раз по скорости восстановления не опечатка и не фигура речи. Это сухой результат опроса реальных компаний. Просто вдумайся в эту пропасть. Там, где у одних минуты, у других недели.

Скажу честно и про осторожность с такими цифрами, чтобы ты не принял их за священное писание. Год от года конкретные множители скачут будь здоров. В одни годы разрыв

оказывался даже резче этого, в другие методику замера пересматривали и считали иначе. Молиться на точное число тут не надо. Но за все годы наблюдений, а их уже больше десяти, главный вывод не дрогнул ни разу.

Скорость и надёжность не враги, как привыкли думать. Это две стороны одной зрелости. Команды, которые катят чаще, падают реже и встают быстрее. Вот и вся «магия», за которую, в сущности, и платят такие деньги.

За годы я для себя поделил все компании на три типа по их отношению к DevOps. Классификация ненаучная, чисто из наблюдений, но за её точность ручаюсь.

Первый тип я про себя зову «у нас есть свой Валера». Здесь либо ещё не налетели на большую боль, либо налетели, но так и не поняли, обо что. Деплой тут раз в квартал, потому что «лишний раз не трогаем, авось не упадёт». Слово «мониторинг» означает, что Валера время от времени поглядывает на сервер. О проблемах узнают от пользователей. Молодому инженеру в такой конторе уготована ровно одна дорога: стать новым Валерой. А это, при всём уважении к Валере, карьерный тупик с видом на пенсию. Идти сюда стоит, только если тебе по нраву роль одинокого первопроходца в джунглях без карты, компаса и надежды на эвакуацию.

Второй тип — «мы внедряем DevOps». Здесь уже дозрели до мысли, что менять что-то придётся. Купили модный трекер, поставили сервер сборки, гордо вписали в годовую стратегию слово Agile, завели дашборд с метриками. Прав-

да, что делать с этими метриками, пока не очень понимают, и смотрят на них, как на красивые обои. Переходный период самый хаотичный из всех. Старые процессы ещё живы и брыкаются, новые ещё не прижились. А за соседним столом сидит человек, который произносит: «Мы так всегда делали», — с лицом первооткрывателя закона природы. Зато именно здесь можно вырасти стремительно. Если ты приходишь с ясной головой и внятными идеями, тебя, скорее всего, услышат, потому что больше слушать особо некого. Как доносить такие идеи, чтобы их не похоронили на первом же совещании, — отдельный большой разговор, к нему мы обязательно вернёмся.

И третий тип — «DevOps? Да, это просто то, как мы работаем». Мечта. **Здесь никто не произносит слово «DevOps» по пять раз на дню. Ровно так же, как здоровый человек не сообщает каждые пять минут, что он дышит.** Команды самостоятельны, деплой — рутина, а не подвиг, инциденты разбирают спокойно и без охоты на ведьм. Именно в таких местах люди вырастают в настоящих специалистов, и именно отсюда они потом расходятся по миру и меняют другие компании. Попасть сюда с улицы, без опыта, тяжело, здесь чаще берут уже состоявшихся. Но держи эту картинку перед глазами как цель. Туда стоит идти.

«Хорошо, хорошо», — скажешь ты, — «А в деньгах-то сколько?».

Ты об этом думал. Все думают. И стыдиться тут нечего. Я

и сам в своё время думал в первую очередь об этом.

Отвечу честно и сразу с оговоркой: рынок живой, цифры на нём текут, так что относись к ним как к порядку величин, а не как к обещанию из рекламы.

На момент, когда я пишу эти строки, джуниор в DevOps в России стартует где-то около сотни тысяч рублей в месяц, миддл уходит в район двух-трёх сотен, у сеньора верхней планки, считай, нет. А в международных командах счёт идёт в валюте и по своей, отдельной арифметике. Но важнее любых цифр вот что, и я прошу запомнить это крепче, чем суммы. **DevOps не та профессия, где платят за факт владения инструментом, за галочку в резюме. Платят за решённые проблемы и за снятую с компании боль. Поэтому правильный вопрос звучит не «сколько я буду получать», а «какие проблемы я умею решать и насколько дорого они обходятся бизнесу без меня».**

Чем дороже проблема, тем выше твоя цена. В DevOps это видно особенно ясно, потому что цена простоя тут — живые деньги, которые кто-то прямо сейчас не заработал, а не абстракция.

* * *

Так это профессия или философия?

И то, и другое сразу стоит уложить в голове прямо сейчас, чтобы потом не разочаровываться и не спорить в комментариях.

Как философия, DevOps — это про культуру. Про то, что разработчик держит в голове эксплуатацию, эксплуатация понимает разработчика, скучная рутина отдаётся машинам, а за общий результат отвечают сообща, а не показывают пальцем.

Как профессия, DevOps означает конкретного живого человека, который строит инфраструктуру, налаживает те самые цепочки сборки и выкатки и следит, чтобы приложение держалось на ногах в любую погоду.

Вся загвоздка в том, что в разных компаниях под словами «DevOps-инженер» прячется совершенно разное содержимое. Где-то это прокачанный сисадмин, обложившийся контейнерами. Где-то специалист, который перестраивает культуру всего IT-департамента. Где-то кентавр, наполовину разработчик, наполовину инфраструктурщик. **Поэтому, увидев заветную вакансию, первым делом задавай себе вопрос «что они вообще понимают под этим словом», а не «сколько платят». Ответ либо обрадует тебя до глубины души, либо убережёт от ошибки длиной в целый год.**

А главное, что я прошу унести из этой части: инструменты приходят и уходят, как мода на брюки.

Сегодняшний контейнер завтра сменится другим, нынешний грозный оркестратор[45] однажды станет музейным экспонатом, и над ним будут посмеиваться джуниоры. **А вот человек, который умеет думать системно и спокойно жить на стыке разработки и эксплуатации, нужен будет всегда. На него мода не проходит.**

* * *

[45] **Оркестратор (Orchestrator)** — система, которая автоматически управляет запуском, взаимодействием, масштабированием и восстановлением приложений и сервисов. В контейнерной среде оркестратор распределяет нагрузку и поддерживает работоспособность приложений.

Зачем это тебе — вопрос важнее всех остальных

Вот мы вернулись к тому, с чего начали. К видению. К твоему личному «зачем».

Я повидал людей, которые шли в DevOps, потому что «все говорят, там деньги». Они честно вызубрили нужные инструменты, прошли сертификацию по облаку, завели красивый профиль на GitHub с аккуратными репозиториями и при этом глубоко, по-настоящему тосковали. Потому что делали всё это, как дрессированные тюлени: технично, на результат и совершенно не понимая зачем.

А ещё я повидал других. Тех, кому было до зуда любопытно, как эта махина вообще работает. Как тысячи запросов в секунду не валят сервер. Как приложение само разворачивается в облаке за минуты. Как десять человек умудряются каждый день катить новое без хаоса, крика и валидола. Эти росли быстро и счастливо. Их не нужно было заставлять, их нужно было разве что иногда отгонять от компьютера, чтобы поели.

Скажу и про себя, раз уж мы тут разоткровенничались.

Меня в этой профессии держат не деньги. Деньги приятны, врать не стану, я не из тех, кто красиво кривит губы при их упоминании. Но держит меня не это. Держит один тихий момент, ради которого, в сущности, всё.

Когда сложная система, которую ты сам, по кирпичику, собрал и выстрадал, в час пиковой нагрузки просто работает. Молча. Ровно. Без единого звонка.

Где-то там, в большую ночь большой распродажи, дежурный пьёт кофе и скучает, потому что скучать ему позволил ты заранее подумавший обо всём, что может пойти не так, и сделавший так, чтобы оно не пошло.

В этом чувстве нет фейерверков и аплодисментов. Оно тихое, почти незаметное, как сама хорошая работа. Но ради него я когда-то таскал коробки по тому подвалу с плюс тридцатью. И, оглядываясь, не жалею ни об одной из них.

Поэтому прямо сейчас, до того, как мы перевернём страницу и шагнём к Правилу 2, задай себе один честный вопрос. Настоящий, не тот, на который удобно ответить: «Зачем мне DevOps?».

Не «потому что платят» и не «потому что знакомая сказала, что это перспективно», а по-честному, наедине с собой.

Тебе в самом деле интересно копаться в том, как устроены сложные системы?

Тебя по-настоящему бесит, когда вокруг бардак и никто не понимает, что происходит?

Тебе хочется оказаться внутри компаний, которые двигают мир вперёд, а не наблюдают за ним с обочины?

Если хотя бы на один вопрос ты ответил «да», думаю, дальше нам по пути. А если самый честный твой ответ по-

ка звучит как «не знаю, просто хочу зарабатывать», тоже не закрывай её, ради бога. Деньги — абсолютно нормальное, взрослое топливо. **Просто тогда читай дальше особенно внимательно: мы разберём по косточкам, что именно нужно делать, чтобы в DevOps зарабатывать всерьёз, а не просто числиться в штатном расписании.**

Одно совсем маленькое задание. На самом деле самое важное во всей этой главе, важнее любых Валер и любых миллиардов Amazon. Попробуй взять лист бумаги. Мне всегда казалось, что на бумаге такие вещи формулируются честнее. И сформулируй своё видение одной-двумя фразами с ответами на вопрос: «Зачем тебе DevOps и где ты хочешь оказаться через пару лет». Не красиво, не для соцсетей. А правдиво пусть коряво, пусть стыдно перечитывать. Это не формальность для галочки. Это та самая точка на горизонте, без которой всё дальнейшее движение, сколько бы ты ни учил, превратится в бег по кругу.

* * *

Основные мысли главы

DevOps не программа, которую устанавливают в соседней компании, не должность с пиццей и не магия. Это культура совместной работы и набор практик, которые позволяют рождать и выкатывать софт быстро и надёжно одновременно.

Мир ускорился, и DevOps, спокойный, выстраданный ответ бизнеса на этот вызов; платят здесь за снятую боль, не за зазубренные инструменты: за скорость, за надёжность, за то самое спокойствие, с которым кто-то спит, пока система работает.

Хороший DevOps незаметен: деплой, которого никто не заметил, — это победа, не провал. А проблема, если уж она случилась, почти никогда не в людях — она в том, как выстроены процессы вокруг них. Помни об этом, когда в следующий раз захочется назначить виноватого.

Следующий вопрос, который встанет перед тобой в полный рост: «Как не утонуть в океане того, что предстоит выучить?».

От себя, по праву человека, который сам когда-то тонул в этом океане, добавлю: думай масштабно, но об одном большом. Об этом Правило 2.

Упражнение · Между правилами Выход из клуба зрителей

Упражнение, которое все хотят пропустить

Прежде чем мы пойдём дальше, я попрошу тебя сделать одну вещь. Маленькую. Скучную на первый взгляд. И при этом самую важную во всей первой главе.

Я знаю, как это бывает. Сейчас ты дочитаешь абзац, кивнёшь, скажешь себе «сделаю потом» и перелистнёшь страницу. Так делают девять человек из десяти. И ровно девять из десяти потом годами учат то одно, то другое, выгорают, бросают и не понимают, что пошло не так. А не так пошло вот что: они так и не ответили себе на вопрос «зачем». Не нарисовали точку на горизонте. И поэтому шли куда угодно, то есть в никуда.

Поэтому я прошу тебя стать тем десятым. Это займёт двадцать минут. Это бесплатно. И это, возможно, сэкономит тебе год жизни.

Упражнение: твоё видение на одну страницу

Возьми лист бумаги и просто начни писать. Не подбирай красивые слова и не шлифуй формулировки. Это не сочинение и не резюме, свободным текстом. Это разговор с самим собой. Опорные вопросы, чтобы было от чего оттолкнуться:

— Зачем мне DevOps по-настоящему? Не «потому что платят», а что именно меня туда тянет.

— Что я почувствую в тот день, когда у меня получится?

— Где я хочу оказаться через два года? Опиши один обычный рабочий день из этого будущего. Во сколько ты встаёшь? Над чем работаешь? С кем? Что тебя в этом дне радует? Чем он отличается от твоего сегодняшнего дня?

— От чего я ухожу? Что в нынешней работе или жизни меня не устраивает настолько, что я готов учиться по вечерам после работы. Это твоё топливо. Знать его в лицо полезно.

— Сколько я готов вложить? Честно: сколько часов в неделю я реально могу выделить? Не в мечтах. В жизни, где есть семья, усталость и сериалы. Пять часов? Десять? Любая честная цифра лучше героической выдуманной.

— Как я пойму, что движусь? Придумай один-единственный признак, по которому через три месяца ты увидишь: да, я не стою на месте. Не «выучу всё», а что-то маленькое и проверяемое.

Правил оформления нет. Можно списком, можно сплошным текстом, можно коряво. Единственное требование — честность. Это письмо никто, кроме тебя, не прочитает, так что врать тут некому и незачем.

Когда закончи не выбрасывай. Сфотографируй, спрячь в ящик, заведи для него отдельную заметку. Мы вернёмся к этой странице в конце книги, в эпилоге, и ты сам удивишься,

как много из написанного сбудется, и как изменится то, что казалось главным.

А теперь предостережение, без которого упражнение не работает. Не пиши чужое видение. Не списывай у блогера с ютуба, который красиво рассказывает про «удалёнку с зарплатой в долларах и кофе на балконе в Лиссабоне». Может, это и твоя мечта. А может, чужая, которую тебе аккуратно продали. Через тяжёлые вечера тебя будет тащить только то видение, которое действительно твоё. Даже если оно скромнее лиссабонского балкона.

Написал? Спрятал? Тогда, и только тогда, переворачиваем страницу. Потому что дальше начинается самое трудное и самое интересное: что со всем этим делать.

Правило 2. Не разменивайся
«Фокус или смерть»

«Я не боюсь того, кто отработал десять тысяч ударов по одному разу. Я боюсь того, кто отработал один удар десять тысяч раз».
— Брюс Ли

Хочу познакомить тебя с Никитой.

С самим Никитой я познакомился много позже всей этой истории, и при каких обстоятельствах, расскажу в самом конце книги, это отдельный сюжет. А пока скажу так. Эту свою историю он мне однажды рассказал сам, целиком, от первой до последней детали. Поэтому я и ручаюсь за подробности. Имя я, правда, изменил, по понятной причине: дальше пойдёт речь о его провалах, а человек он живой и до сих пор работает. Но всё, что с ним было, было на самом деле.

И знаешь, что в этой истории самое важное? Никита, конечно, один. А вот история его на тысячу человек одна. Я потом сотни раз встречал её на форумах, где входящие в профессию по ночам пишут незнакомцам то, в чём стыдно признаться вслух. Меняются имена, города, года. Не меняется ничего больше. Так что Никита тут вроде как один за всех. Может, и за тебя тоже, кто знает.

Никите было двадцать шесть. Днём он работал на работе,

которую не любил. Что-то про таблицы и отчёты, неважно. А по вечерам решил войти в DevOps. Решил твёрдо, всерьёз, с горящими глазами. Как решают раз в жизни. Открыл вечером ноутбук, набрал в поиске «дорожная карта DevOps» и нажал ввод.

Лучше бы он этого не делал.

На него обрушилась лавина. Linux, сети, Docker, Kubernetes, облака, для надёжности сразу три штуки, Terraform, Ansible, Jenkins, Prometheus, Grafana, ArgoCD, Helm, Vault, CI/CD, мониторинг, языки, базы, и ещё слова, каждое из которых тянуло за собой следующие сорок. Никита смотрел на это всё и чувствовал, как внутри тихо холодеет. Так много. Так бесконечно много. А он один, и вечер всего один, и в нём после нелюбимой работы сил часа на полтора. Никита посчитал. Получилось двадцать три названия. Двадцать три. Он сглотнул, открыл первую вкладку, начал.

И тогда Никита совершил роковую ошибку, в которой, между нами, нет ни капли его вины. Он решил успеть всё. Сразу. Параллельно.

В одной курс по Linux, в другой видео про Kubernetes, в третьей чья-то статья про облака, в четвёртой документация, в которой он уже забыл, как очутился. Он метался между вкладками, как белка в колесе. Час почитал про одно, бросил, схватился за другое, испугался третьего, краем глаза заглянул на четвёртое. Каждый вечер. Месяцами.

Восемьдесят процентов своего времени Никита тратил на

борьбу с непослушными отступами в YAML[46], капризном формате, где один лишний пробел рушит всё. И только двадцать процентов на что-то, отдалённо похожее на дело. Он прыгал. С Linux на Kubernetes, потому что Kubernetes моднее. С Kubernetes на Terraform, потому что в вакансии было написано Terraform. С Terraform обратно на Linux, потому что выяснилось, что без Linux ничего не работает. А его-то он толком и не знал.

Прошло полгода. Полгода вечеров. Полгода «потом досплю». И знаешь, что Никита умел к концу этого полугодия? Ничего. То есть он умел произнести сорок умных слов и даже примерно объяснить, что они значат. Но если бы его попросили взять простое приложение и довести его от строчки кода до работающего сайта в интернете (целиком, сам, от начала до конца) он не смог бы. Ни одного раза. Ни одного законченного дела за шесть месяцев.

И вот однажды вечером Никита сидел перед своими вкладками, и его накрыло. Та самая тихая, вязкая мысль, которая накрывала до него тысячи людей: «Это не для меня. Я тупой. Все вокруг уже умеют, один я топчусь на месте. Надо бросать».

Так вот. Запомни, пожалуйста, это крепко, потому что это правда, проверенная на тысячах судеб. Никита не тупой. И

[46] **YAML —** Простой текстовый формат для записи настроек, который человеку читать удобнее, чем машинный код. Его главная особенность и главная боль новичка — строгие отступы пробелами: один лишний пробел в начале строки может сломать всё.

ты не тупой. Просто Никита совершил одну-единственную ошибку, самую частую и самую дорогую ошибку всех, кто входит в эту профессию. Он попытался съесть океан. Целиком. За один присест. А океан так не едят.

И вот что особенно обидно. Никита не был ленивым. Совсем наоборот. Он пахал. Он жертвовал сном, отменял встречи с друзьями, чувствовал укол вины каждый раз, когда вместо урока смотрел сериал. По часам, вложенным в дело, Никита заслуживал медаль. Беда была в направлении усилий, не в их количестве. Он грёб изо всех сил, грёб сразу в сорок сторон. А лодка, которую тянут в разные стороны, стоит на месте и только трещит по швам. Это, если хочешь, самый жестокий вид неудачи: когда ты искренне выложился, а на выходе — ноль. После такого и опускаются руки. Не от лени. От несправедливости.

Через полгода такой каторги он знал понемногу обо всём и толком ничего. Мог с умным видом ввернуть в разговоре «Kubernetes», но не мог поднять простейший контейнер своими руками. Сорок дверей, в каждую заглянул, ни в одну не вошёл.

* * *

Почему DevOps — это не «немного сисадмин плюс немного разработчик»

Есть стойкое заблуждение, и Никита пал его жертвой одним из первых. Звучит оно так. DevOps — это когда ты немножко сисадмин, немножко программист, немножко сетевик, немножко спец по облакам, немножко по безопасности — словом, по чуть-чуть отовсюду. Этакий швейцарский нож. Мастер на все руки.

Красивая картинка. И вредная.

Потому что «мастер на все руки» на практике слишком часто означает «толком не умеет ничего». Человек-нож с сорока лезвиями, каждое из которых заточено на четверть. Он знает обо всём по верхам и не может вонзиться вглубь ни в одном месте. А вглубь — это как раз там, где деньги, уважение и спокойный сон.

Давай честно. DevOps действительно профессия широкая, тут заблуждение не врёт. Хороший инженер и правда касается многого: и серверов, и кода, и сетей, и облаков. Но широта эта приходит после глубины и поверх неё — не сразу и не вместо неё. **Сначала ты в чём-то одном становишься по-настоящему уверен. Встаёшь на твёрдую ногу. И только с этой твёрдой ноги начинаешь дотягиваться до соседних областей из любопытства и по необходимости, не из паники.**

Помни, в прошлой главе я говорил, что DevOps — это прежде всего способ думать, а инструменты лишь детали реализации? Вот ключ ко всему второму правилу. Способ думать всего один. Инструментов, которыми он воплощается, — сотни, и они меняются каждые несколько лет. **Если ты гонишься за инструментами, ты гонишься за горизонтом: сколько ни беги, он всё там же. А если ты выращиваешь внутри себя способ думать, он остаётся с тобой навсегда, на каком бы модном инструменте ни писали вакансии в этом сезоне.**

Я сам прошёл это на своей шкуре. Я не стал инженером, потому что нахватался всего по чуть-чуть.

Я стал инженером не потому, что знал всё. Я стал инженером потому, что когда-то, в той самой подвальной серверной, по-настоящему, до самого дна разобрался в одной вещи — в том, как устроены машины, на которых всё крутится.

Глубоко. Скучно. Дотошно.

Это была моя опора.

Потом появились Docker, облака, контейнеры, оркестраторы и ещё десятки новых слов. Но каждую новую вещь я ставил на уже готовый фундамент. Мне не приходилось строить дом в воздухе.

И вот это я хочу, чтобы ты запомнил. Сначала нужна одна твёрдая нога. Один фундамент. Одна область, в которой ты стоишь уверенно. Всё остальное достраивается потом.

Стив Джобс однажды сказал коротко хлестко. Фокус —

это умение говорить «нет». Не «нет» плохому. С плохим легко. **Фокус — это говорить «нет» сотне хороших, заманчивых, блестящих вещей, чтобы осталось время на ту единственную, которая сейчас важнее всех.** Никита говорил «да» всем двадцати трём логотипам сразу. И именно поэтому не сказал настоящего, глубокого «да» ни одному.

* * *

Что реально нужно знать, а что просто шум

Хорошая новость, ради которой стоило открывать эту главу. Из двадцати с лишним логотипов на той стене для старта тебе нужны буквально пять. Остальное пока шум. Не мусор, нет — многое из этого прекрасно и однажды тебе пригодится. Но именно пока, на входе, это шум, который только мешает расслышать главное.

Чтобы отличать сигнал от шума, держи в голове один простой принцип. Инструменты вторичны по отношению к понятиям. За каждым модным названием стоит простая идея, которая жила задолго до него и переживёт его. Понятие «хранить историю изменений кода» и есть идея; Git всего лишь сегодняшний инструмент для неё. Понятие «упаковать программу вместе с её окружением» — идея; Docker — её нынешнее воплощение. Поймёшь идею — любой инструмент под неё освоишь за выходные. Выучи инструмент, не поняв идею, и останешься Никитой с двадцатью тремя вкладками, который умеет нажимать кнопки и не понимает, что под ними.

Поэтому простой фильтр на ближайший год. Спрашивай себя про каждую новую блестящую штуку: «Это фундаментальное понятие или конкретный инструмент поверх него?». Если понятие, изучай, оно навсегда. Если инструмент, отло-

жи все остальные, оставив по одному инструменту на каждое понятие. Один инструмент контроля версий (Git). Один для контейнеров (Docker). И так далее. Вторые, третьи и десятые варианты подождут, пока у тебя не появится работа, где они реально понадобятся. На работе ты их освоишь за неделю, и ещё и денег за это получи.

Сент-Экзюпери — писатель, летавший на самолётах между книгами, — оставил фразу, которую стоит повесить над столом всякому, кто учится: *«Совершенство достигается не тогда, когда нечего добавить, а тогда, когда нечего убрать».* Твоя программа обучения на первый год должна уложиться в короткий маршрут из пяти шагов, не в список из сорока пунктов. **Всё лишнее нужно убрать. Безжалостно. Это и есть фокус.**

И ещё одно, чтобы снять груз с души. Если ты откроешь любую вакансию DevOps, там будет перечислено двадцать технологий, и от этого списка хочется закрыть ноутбук и пойти спать. Не пугайся. Эти списки редко отражают то, что действительно требуется человеку в первый рабочий день. Чаще это просто список пожеланий. Я нанимал людей и видел это по другую сторону стола. Новую технологию можно освоить за несколько недель. Инструмент можно заменить. Название в вакансии сменится уже через год. Гораздо сложнее научить человека понимать основы, видеть систему целиком и находить решение там, где нет готовой инструкции. Реально на старте от тебя ждут уверенной базы и горящих

глаз. Поэтому на собеседовании инженер с крепкой базой почти всегда выигрывает у человека, который знает по верхам сорок модных слов. Логотипы впечатляют до первого вопроса. Основа остаётся с тобой на всю карьеру.

* * *

Дорожная карта без паники

А теперь, тот самый маршрут. Пять остановок. Не сорок. Проходить строго по порядку, потому что каждая следующая стоит на плечах предыдущей и без неё повисает в воздухе. И никакой гонки. Следующая остановка открывается тогда, когда на нынешней ты чувствуешь себя как дома — а не тогда, когда просто «надоело».

Linux. Фундамент фундамента.

Почти весь интернет, все эти серверы, на которых живут сайты и приложения, работают на Linux. Не научились разговаривать с Linux? Ты как повар, не умеющий пользоваться плитой. Учиться надо не «всему Linux» (это снова океан), а конкретному: уверенно жить в терминале, двигаться по папкам, читать и править файлы, разбираться с правами доступа, ставить программы, смотреть логи, понимать, что за процессы крутятся на машине.

Маленькая цель, по которой ты поймёшь, что готов дальше. Возьми дешёвый виртуальный сервер в любом облаке за копейки в месяц, зайди на него из терминала и подними на нём простой сайт-визитку, который открывается в браузере по адресу.

Сделал без подсказок? Поздравляю, у тебя под ногами появилась твёрдая земля.

Git. Как только ты начнёшь писать хоть что-то, тебе по-

надобится хранить историю: кто что менял, когда и зачем, и как вернуться назад, когда всё сломалось. Этим занимается система контроля версий, и сегодня её зовут Git. Это машина времени для твоего кода и твоих настроек, и работать без неё в профессии так же немыслимо, как водителю без руля. Тебе не нужно знать «весь Git», в нём есть тёмные углы, куда даже сеньоры лазают со шпаргалкой. Нужно немногое: сохранять изменения, заводить отдельные дорожки для своей работы, сливать их обратно, выкладывать всё это на GitHub (большую общую площадку, где код хранят и показывают миру).

Маленькая цель. Положи код вчерашнего сайта-визитки на GitHub так, чтобы его можно было дать ссылкой другому человеку. Заодно это первый кирпич твоего будущего портфолио, но об этом в правиле про первую работу.

Docker. Помни боль Артёма и Маши из прошлой главы: «у меня работает, а у тебя нет»? Docker служит лекарством от той самой боли. Он берёт твою программу и упаковывает её вместе со всем окружением в стандартную коробку-контейнер, которая одинаково запускается где угодно: на твоём ноутбуке, на сервере, на Луне. С Docker заканчивается эпоха «а на моей машине всё было нормально». Учить, опять же, не «весь Docker», а суть: что такое образ и что такое контейнер, как написать простой рецепт сборки своей коробки, как её запустить.

Маленькая цель. Упакуй свой сайт-визитку в контейнер и запусти его одной командой. В тот момент, когда это полу-

чится, ты физически почувствуешь, как абстрактное слово из вакансий превращается в инструмент в твоих руках. Это сильное чувство. Ради него многое затевается.

CI/CD. Вот здесь, на четвёртой остановке, разрозненные навыки наконец сплетаются в то, что и делает DevOps собой. CI/CD и есть та самая заботливая автоматическая цепочка из истории про Катю и «Пульс». Ты меняешь код, а дальше машина сама его проверяет, собирает, тестирует и выкатывает, без ручных плясок и ночных подвигов. Это сердце профессии. Здесь «программа обучения» и «реальная работа» сливаются. Ты строишь конвейер, который работает за тебя. Начать проще, чем кажется. На том же GitHub есть встроенный механизм, который умеет запускать твою цепочку сам.

Маленькая цель. Первая, которой не стыдно хвастаться. Сделай так, чтобы при каждом твоём изменении кода сайт сам, без единого твоего касания, пересобирался в контейнер и выкатывался на сервер. Ты нажал «сохранить», и через минуту в интернете обновилась живая страница. Вот теперь ты делаешь то, за что платят деньги.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, купив полную легальную версию на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.