

beeline cloud ×  Deckhouse

Kubernetes своими руками: цена инженерного энтузиазма

Спикеры:

Антон Четин

Андрей Радыгин

О чем поговорим

- 1 Про DIY-подход
- 2 Ключевые вызовы: от первого кластера до хаоса
- 3 Скрытые издержки DIY K8s
- 4 Когда «из коробки» = дешевле и надёжнее
- 5 Engineer lock-in vs vendor lock-in: мифы

Для кого актуально

- ✓ Техлиды: DevOps/Infra/SRE TL, Head of Dev, CISO
- ✓ Компании: старт или рост K8s-инфраструктуры
- ✓ Решающие: строить самим или взять готовое

💡 Disclaimer:

Не для вас, если: вы ищете технический tutorial по настройке kube-apiserver или хотите сравнить 10 дистрибутивов «в лоб». Этот вебинар — про стратегию, экономику и операционную эффективность.

Вам будет интересно, если:

- ✓ Планируете запуск инфраструктуры с Kubernetes
- ✓ Начинаете внедрение микросервисов
- ✓ Хотите понять перспективы развития своего Kubernetes
- ✓ Ощущаете проблемы в эксплуатации инфраструктуры на Kubernetes

Вебинар поможет вам:

- ✓ Оценить особенности и перспективы DIY
- ✓ Поможет построить собственную ИТ-стратегию
- ✓ Понять ценность готовых платформ

Данные из реальной практики

100+ аудитов у компаний с 3 до 100 кластерами

Типовой сценарий, который разберем:

- ✓ Kubernetes работает
- ✓ Всё вокруг — система, которую никто не проектировал
- ✓ Недели на доступ разработчикам
- ✓ Месяцы на обновления версий
- ✓ Годы на «нормализацию» безопасности

Первый кластер: эйфория

01

Первый кластер: эйфория

Kubernetes — старт,
а не финиш

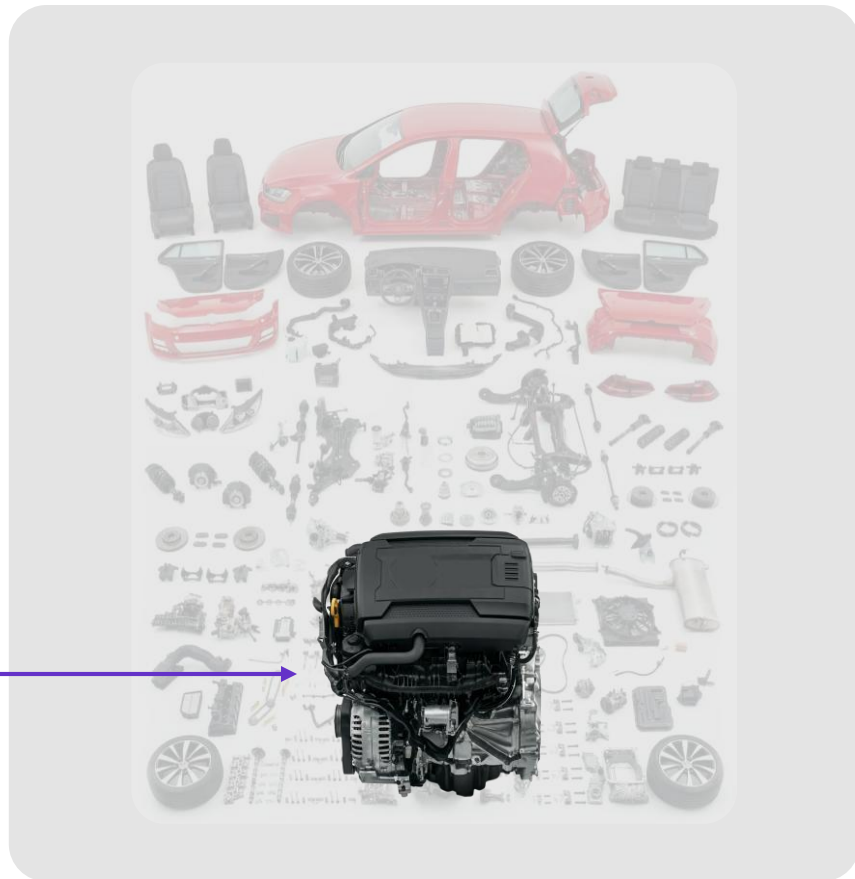


Одного Kubernetes недостаточно!

В состав Kubernetes **не входят**
многие компоненты, необходимые
для запуска продуктивных нагрузок

Две трети организаций имеют
более пятнадцати отдельных подсистем
в составе внутреннего Kubernetes-стека *

Kubernetes —
ЭТО ТОЛЬКО ДВИГАТЕЛЬ →



* [The Spectro Cloud 2025 State of Production Kubernetes](#)

Первый кластер: эйфория

Managed-K8s — только control-plane:

- ✓ быстрый старт, SLA провайдера
- ✓ закрыты базовые вопросы

НО, нужны:

- ✓ мониторинг
- ✓ резервное копирование
- ✓ безопасность
- ✓ управлению сетью
- ✓ сертификатами и многое другое

Первый кластер: эйфория

Ванильный Kubernetes

- ✓ полный контроль
- ✓ гибкость

НО:

- ✓ Еще больше компонентов
- ✓ Критическая зависимость от людей
- ✓ Time-to-Market месяцы
- ✓ Отсутствие соответствия требованиям регулятора
- ✓ Скрытые расходы
- ✓ Фокус не на бизнесе

Первый кластер: эйфория

Эффект масштаба = нелинейный рост сложности

Закономерность роста

Неоднородность накапливается:

- ✓ Разные версии
- ✓ Ручное вмешательство
- ✓ Разная конфигурация

А если несколько провайдеров:

- ✓ Разная автоматизация
- ✓ Разное масштабирование
- ✓ Разные подходы к доступу

Закономерность роста

Чтобы кластеры не болели:

- ✓ Нужен продуктовый подход:
 - ✓ Не «решить сейчас», а «поддерживать 3-5-7 лет»

Self-service: автоматизация или технический долг?

Self-service: автоматизация или технический долг?

Рост разработки → рост потребности в self-service

Self-service: автоматизация или технический долг?

Рост разработки → рост потребности в self-service

«Сделаем шаблоны и скрипты, и все дела»

Self-service: автоматизация или технический долг?

Рост разработки → рост потребности в self-service

«Сделаем шаблоны и скрипты, и все дела»

Через несколько недель: база готова, PR → проверка → деплой

Profit? Но есть нюанс...

Вопросы, которые появляются через 3-6-9 месяцев

Сколько времени сейчас занимает создание окружения?

...какой у нас TTM?

Вопросы, которые появляются через 3-6-9 месяцев

Сколько времени сейчас занимает создание окружения?

Сколько изменений внесли в шаблон за 3 месяца?

...и все ли шаблоны актуальны?

Вопросы, которые появляются через 3-6-9 месяцев

Сколько времени сейчас занимает создание окружения?

Сколько изменений внесли в шаблон за 3 месяца?

Кто владеет этим кодом?

Вопросы, которые появляются через 3-6-9 месяцев

Сколько времени сейчас занимает создание окружения?

Сколько изменений внесли в шаблон за 3 месяца?

Кто владеет этим кодом?

Что происходит, когда надо обновить все шаблоны?

Вопросы, которые появляются через 3-6-9 месяцев

Сколько времени сейчас занимает создание окружения?

Сколько изменений внесли в шаблон за 3 месяца?

Кто владеет этим кодом?

Что происходит, когда надо обновить все шаблоны?

Почему в dev и staging разные политики безопасности?

Self-made = технический долг

Самодельные self-service решения становятся техническим долгом

Self-made = технический долг

Самодельные self-service решения становятся техническим долгом

Работают, но:

- Требуют постоянного внимания
- Привязаны к узким специалистам
- Хрупки при изменениях
- Разработчики в очереди — просто теперь она в Git
- Вопросы: сколько инвестируете в поддержку?

DKP: баланс вместо груза на плечах

Управление проектами из коробки:

- Квоты, доступы, политики безопасности
- Разрешенные реестры
- Сетевые политики по умолчанию
- Готовые шаблоны + безопасность «из коробки»
- Создание проекта: UI / CLI / IaC

Главное: при обновлении платформы/k8s/компонентов: ничего не ломается

**Day-2: когда кластеров 5,
а ощущение что 300**

03

Day-2: когда кластеров 5, а ощущение что 300

Разные версии — это нормально и неизбежно

Day-2: когда кластеров 5, а ощущение что 300

Разные версии — это нормально и неизбежно

Dev, staging, prod — каждый под свою задачу

Day-2: когда кластеров 5, а ощущение что 300

Разные версии — это нормально и неизбежно

Dev, staging, prod — каждый под свою задачу

Со временем версии расходятся:

- Dev: новый CNI «на тест»
- Staging: обновили control plane
- Prod: «пока не убедимся»
- Проблема: «На staging работает. На prod — нет».

Где теряется время?

Управление версиями вручную:

- Таблица совместимости
- Квартальные проверки обновлений

Где теряется время?

Управление версиями вручную:

- Таблица совместимости
- Квартальные проверки обновлений
- Проверка работы нового CNI
- Тестирование ingress-контроллера

Где теряется время?

Управление версиями вручную:

- Таблица совместимости
- Квартальные проверки обновлений
- Проверка работы нового CNI
- Тестирование ingress-контроллера
- Многое из этого — в ночном окне
- а еще: e2e тесты, откаты, breaking changes...

Где теряется время?

Управление версиями вручную:

- Таблица совместимости
- Квартальные проверки обновлений
- Проверка работы нового CNI
- Тестирование ingress-контроллера
- Многое из этого — в ночном окне
- а еще: e2e тесты, откаты, breaking changes...

Обновления превращаются в логистику: кто, когда, на каком кластере

Где теряется время?

Управление версиями вручную:

- Таблица совместимости
- Квартальные проверки обновлений
- Проверка работы нового CNI
- Тестирование ingress-контроллера
- Многое из этого — в ночном окне
- а еще: e2e тесты, откаты, breaking changes...

Обновления превращаются в логистику: кто, когда, на каком кластере

В K8s нет координации версий → Постоянный ручной режим

В итоге

Расхождение версий



Скрытые несовместимости

В итоге

Расхождение версий



Скрытые несовместимости

Ваша devops/sre команда:

- Отслеживает релизы
- Проверяет матрицу совместимости
- Поднимает тестовые стенды
- Пишет инструкции
- Чинит, когда ломается

Это операционная нагрузка, растущая нелинейно!

DKP: контроль версий без рутины

Все части платформы — единый продукт с контролем версий

DKP: контроль версий без рутины

Все части платформы — единый продукт с контролем версий

Все обновления проверяются вендором на:

- Обратную совместимость
- Отсутствие breaking changes

DKP: контроль версий без рутины

Все части платформы — единый продукт с контролем версий

Все обновления проверяются вендором на:

- Обратную совместимость
- Отсутствие breaking changes

Релизные каналы

DKP: контроль версий без рутины

Все части платформы — единый продукт с контролем версий

Все обновления проверяются вендором на:

- Обратную совместимость
- Отсутствие breaking changes

Релизные каналы

Предсказуемость обновлений

DKP: контроль версий без рутины

Все части платформы — единый продукт с контролем версий

Все обновления проверяются вендором на:

- Обратную совместимость
- Отсутствие breaking changes

Релизные каналы

Предсказуемость обновлений

Автоматизация: DKP сам предлагает обновления по выбранной политике

Компонентный зоопарк

04

Компонентный зоопарк

Больше масштаб — больше разных компонентов

Компонентный зоопарк

Больше масштаб — больше разных компонентов

Production-кластер на vanilla = 20+ Open Source компонентов:

- CNI, CSI, ingress, CoreDNS
- Trivy, OPA, Falco
- Prometheus, Grafana, Loki
- Istio, Cert-manager, Velero...

Компонентный зоопарк

Больше масштаб — больше разных компонентов

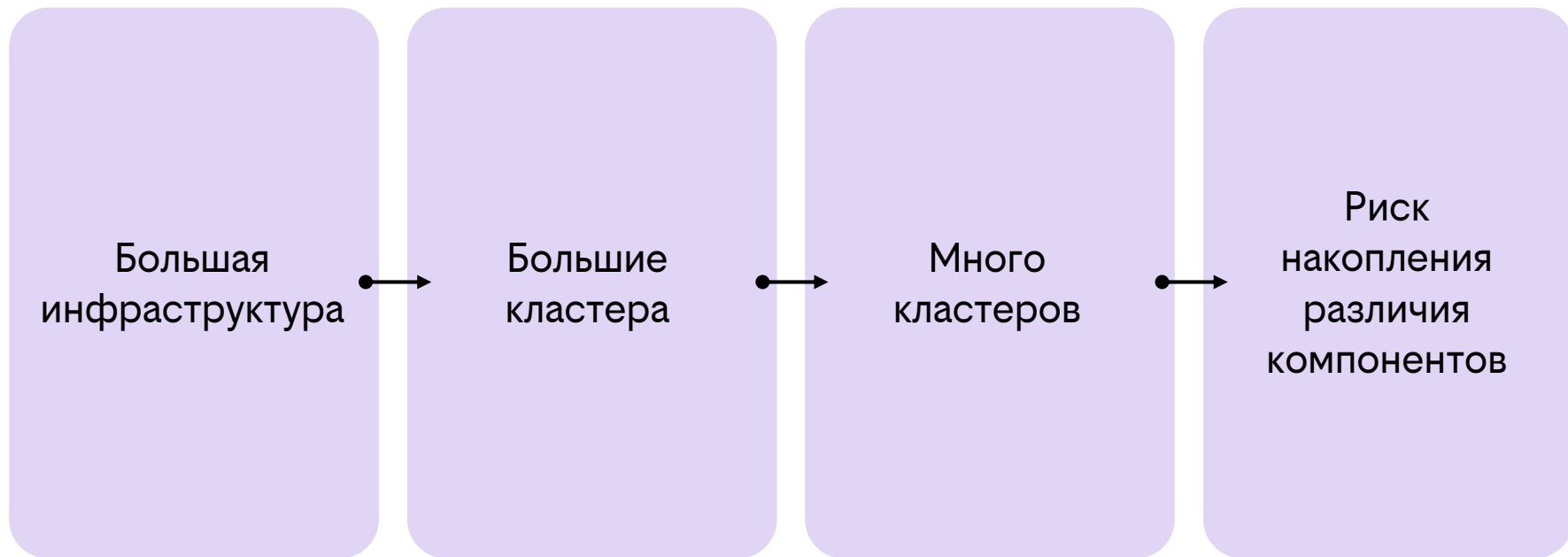
Production-кластер на vanilla = 20+ Open Source компонентов:

- CNI, CSI, ingress, CoreDNS
- Trivy, OPA, Falco
- Prometheus, Grafana, Loki
- Istio, Cert-manager, Velero...

У каждого компонента:

- Свой CRD
- Свои уязвимости
- Своя конфигурация
- Свой мониторинг
- Свой релизный цикл
- Своя аутентификация/авторизация

Компонентный зоопарк



DKP: компоненты как цельный организм

В Deckhouse все компоненты — части единой платформы

DKP: компоненты как цельный организм

В Deckhouse все компоненты — части единой платформы

Хотите Service Mesh? Включаете

- Istio + Kiali + Jaeger уже настроены

DKP: компоненты как цельный организм

В Deckhouse все компоненты — части единой платформы

Хотите Service Mesh? Включаете

- Istio + Kiali + Jaeger уже настроены

Нужна безопасность? Подключите модуль и задайте политики!

DKP: компоненты как цельный организм

В Deckhouse все компоненты — части единой платформы

Хотите Service Mesh? Включаете

- Istio + Kiali + Jaeger уже настроены

Нужна безопасность? Подключите модуль и задайте политики!

Нужны managed-сервисы: просто поставьте галочку

DKP: компоненты как цельный организм

В Deckhouse все компоненты — части единой платформы

Хотите Service Mesh? Включаете

- Istio + Kiali + Jaeger уже настроены

Нужна безопасность? Подключите модуль и задайте политики!

Нужны managed-сервисы: просто поставьте галочку

Из коробки

Централизованное
управление

Корректное обновление
в рамках релиза

Протестированные
конфигурации

Важно: версии всех компонентов синхронизированы и протестированы

Момент истины

В какой-то момент вы понимаете, что пытаетесь построить платформу, а не просто используете Kubernetes

С готовой платформой вы:

- Не собираете «пазл» — вы включаете функцию
- Не нужно выбирать версии
- Не нужно проверять совместимость
- Не нужно писать костыли
- 90% компонентов — distroless-образы

Безопасность: иллюзия контроля

05

Безопасность: иллюзия контроля

Безопасность — это не установка open-source компонента. Это постоянный процесс.



Поставить ИБ инструменты — недостаточно

Пара недель

Внедрить пару политик

Начинается с: «нужны политики по версиям образов»



Поставить ИБ инструменты — недостаточно

An iceberg floating in water, used as a metaphor for the visibility of security tool deployment. The small tip above the water represents the visible part of the effort, while the much larger part below the water represents the hidden, more significant effort.

Пара недель

Внедрить пару политик

Начинается с: «нужны политики по версиям образов»

Сотни человеко-дней

Реактивная реализация ИБ

Поставить ИБ инструменты — недостаточно

Пара недель



Внедрить пару политик

Начинается с: «нужны политики по версиям образов»

Реактивная реализация ИБ

- «Убедиться, что нет привилегированных контейнеров»

Вовлечены

- Команда QA

Поставить ИБ инструменты — недостаточно

Пара недель

Внедрить пару политик

Начинается с: «нужны политики по версиям образов»

RBAC

PSS

Network Policies

Audit/Siem

Реактивная реализация ИБ

- «Убедиться, что нет привилегированных контейнеров»
- «Как контролируем доступ по API?»

Вовлечены

- Команда QA
- Команда DevSecops

Поставить ИБ инструменты — недостаточно

Пара недель

Внедрить пару политик

Начинается с: «нужны политики по версиям образов»

RBAC

PSS

Network Policies

Audit/Siem

CIS Benchmark

Secet Injection

Реактивная реализация ИБ

- «Убедитесь, что нет привилегированных контейнеров»
- «Как контролируем доступ по API?»
- «Есть ли аудит всех изменений?»

Вовлечены

- Команда QA
- Команда DevSecops
- Команда SRE

Поставить ИБ инструменты — недостаточно

Пара недель

Внедрить пару политик

Начинается с: «нужны политики по версиям образов»

RBAC

PSS

Network Policies

Audit/Siem

CIS Benchmark

Secet Injection

Distroless

Runtime audit

Реактивная реализация ИБ

- «Убедитесь, что нет привилегированных контейнеров»
- «Как контролируем доступ по API?»
- «Есть ли аудит всех изменений?»
- «Соответствуем ли CIS-бенчмаркам?»
- «Как реагируем на CVE?» (и реагируем ли вообще)

Вовлечены

- Команда QA
- Команда DevSecops
- Команда SRE
- Команда Разработки
- **Команда ИБ**

Какие риски по ИБ потенциально несёт Open Source

Из 1 067 проанализированных репозиториев*:

91 %

содержит компоненты, которые на 10 и более версий отстают от актуальной

84 %

содержат уязвимости

74 %

содержат критические уязвимости

49 %

содержат компоненты, которые не развиваются более двух лет

14 %

содержат уязвимости старше 10 лет (средний возраст уязвимостей — 2,8 года)

* По данным исследования [Open Source Security and Risk Analysis Report](#) от SYNOPSIS за 2024 год

Циклическая работа с безопасностью

Со временем:

- Политики устаревают
- Новые кластеры настраиваются не так
- Кто-то отключает проверку «временно» — забывает включить
- Аудит логи никто не смотрит

Циклическая работа с безопасностью

Со временем:

- Политики устаревают
- Новые кластеры настраиваются не так
- Кто-то отключает проверку «временно» — забывает включить
- Аудит логи никто не смотрит

Безопасность — замкнутый круг:

- Нарушение \leftrightarrow Исправление
- Через недели: новое нарушение
- Отсутствие best practices

Циклическая работа с безопасностью

Со временем:

- Политики устаревают
- Новые кластеры настраиваются не так
- Кто-то отключает проверку «временно» — забывает включить
- Аудит логи никто не смотрит

Безопасность — замкнутый круг:

- Нарушение \longleftrightarrow Исправление
- Через недели: новое нарушение
- Отсутствие best practices

Проблема: реактивная работа вместо предотвращения

Вызовы при использовании Kubernetes

Безопасность

67 %

Откладывали или замедляли развёртывания из-за проблем с безопасностью Kubernetes

42 %

Безопасность — одна из главных проблем, связанных с использованием Kubernetes

46 %

Теряли прибыль или клиентов из-за инцидентов, связанных с безопасностью

30 %

Уязвимости — главная проблема окружений Kubernetes

Исследование VMware [The State of Kubernetes 2023](#)

Реализация лучших практик безопасности



Безопасная конфигурация кластера Kubernetes «из коробки»

Минимальные привилегии компонентов, преднастроенная ролевая модель, сквозная идентификация объектов в системе аудита, интеграция с внешними службами каталогов



Сканирование образов на уязвимости

Сканирование на этапе конвейера CI/CD и в кластере Kubernetes.
Запрет на запуск образов с уязвимостями



Сетевая безопасность

Централизованное управление сетевыми политиками, контроль за всеми входящими и исходящими соединениями, визуализация сетевых взаимодействий

Реализация лучших практик безопасности

Контроль над запускаемыми приложениями

Встроенная реализация Pod Security Standards [🔗](#), готовый набор рекомендуемых политик с возможностью расширения

Аудит и регистрация событий безопасности

Готовый набор правил для определения событий безопасности с возможностью расширения, гибкая фильтрация событий, отправляемых в SIEM-систему, уведомление сотрудников ИБ об инцидентах безопасности

Соответствие стандартам Kubernetes CIS Benchmark

Мониторинг: от метрик к ответу

06

Мониторинг: от метрик к ответу

Сбор данных — это не диагностика и не мониторинг

Мониторинг: от метрик к ответу

Сбор данных — это не диагностика и не мониторинг

Мониторинг должен помогать в:

- Предотвращении аварий
- Оперативной диагностике и поиске root cause
- Быстрой деэскалации и устранении первопричины

Данные есть, но они не «разговаривают»

Данные есть, алерты есть, графики есть

НО:

- Нет единого контекста
- Нет корреляции между метриками, логами и событиями

Данные есть, но они не «разговаривают»

Данные есть, алерты есть, графики есть

НО:

- Нет единого контекста
- Нет корреляции между метриками, логами и событиями

Нужно мысленно склеивать:

- «Тут падает под»
- «Тут росло потребление памяти»
- «Тут были изменения в конфиге»

Данные есть, но они не «разговаривают»

А еще

- Повышенная когнитивная нагрузка при стрессе
- Требуется квалификация и глубокое понимание k8s
- Нелинейная инфраструктура

DKP: мониторинг как единый инструмент

Предупреждение сбоев до их возникновения:

300 предустановленных алертов, 9 уровней критичности. Система отслеживает ранние признаки и шлёт уведомления нужной команде

Диагностика за 30 секунд:

60+ готовых дашбордов автоматически развёртываются при установке DKP, позволяя быстро локализовать проблему

Готовый мониторинг пользовательских приложений:

Метрики CPU, RAM, диск, сеть, Pod'ы, Ingress и др. собираются автоматически

Централизованная работа с логами:

Автоматический сбор, хранение и визуализация в дашбордах

Интеграция с внешними системами:

Отправка метрик и алертов в удобные каналы оповещения

Результат

1

Сокращение MTTD
(Mean Time to Detect)

2

Сокращение MTTR
(Mean Time to Resolve)

3

Меньше стресса,
больше точности

4

Предупреждение
вместо реагирования

Vendor-lock vs Engineer-lock

07

Смена команды: знания не передаются

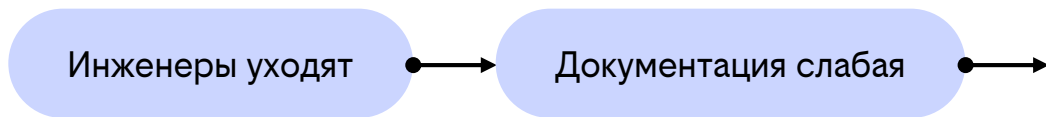
Развитие событий:

Инженеры уходят



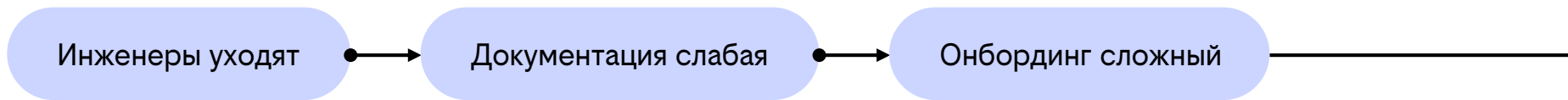
Смена команды: знания не передаются

Развитие событий:



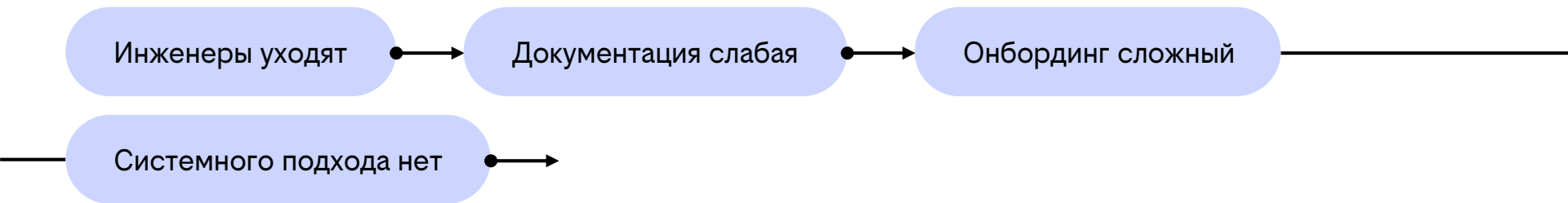
Смена команды: знания не передаются

Развитие событий:



Смена команды: знания не передаются

Развитие событий:



Смена команды: знания не передаются

Развитие событий:



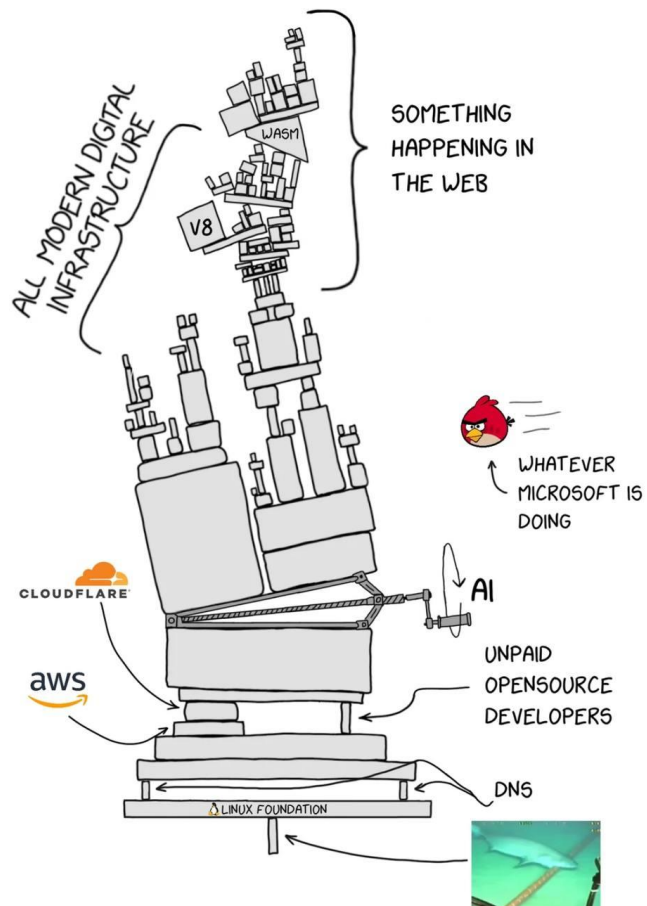
Смена команды: знания не передаются

Развитие событий:



Смена команды: знания не передаются

Результат



Engineer lock-in vs Vendor lock-in

Нужно понимать особенности

Vendor lock-in:

- ✓ Решение под ключ
- ✓ Есть документация, поддержка
- ✓ Миграция тяжелая, но возможна

Engineer lock-in:

- ✓ Только один человек понимает логику
- ✓ Нет документации, комментарии "#If it works, don't touch it"
- ✓ Дефицит кадров
- ✓ Большие накладные расходы (время/деньги/люди)

На чем фокусироваться: это выбор

Можете инвестировать в:

- Создание шаблонов
- Управление версиями
- Контроль безопасности
- Диагностику после сбоев

На чем фокусироваться: это выбор

Можете инвестировать в:

- Создание шаблонов
- Управление версиями
- Контроль безопасности
- Диагностику после сбоев

Или взять готовую основу и направить ресурсы на:

- Developer Experience
- Оптимизацию затрат
- Стратегию безопасности
- Борьбу на рынке

Совокупная стоимость владения DKP за 5 лет по сравнению с самостоятельной поддержкой инфраструктуры

Математическая модель

На **27 %**

дешевле обходится
использование Deckhouse
Kubernetes Platform за 5 лет

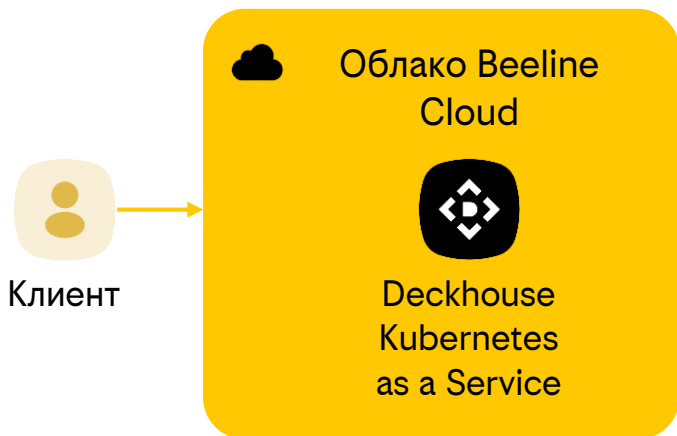
Вебинар «ТСО для СТО: считаем совокупную стоимость владения для Kubernetes-платформ»,
октябрь 2025 года

	DIY	DKP EE	
Трудозатраты (млн руб.)	136,9	44,2	-67 %
Потери произв. труда (млн руб.)	180,0	75,1	-58 %
Затраты на инфраструктуру (млн руб.)	114,6	117,1	+2 %
Потери выручки из-за несозданных фич (млн руб.)	411,8	205,9	-50 %
Лицензии и ТП DKP (млн руб.)	0	173,3	
Итого	843,3	615,7	-27 %

Демонстрация

08

Deckhouse Kubernetes as a Service от Beeline Cloud



Managed Kubernetes

Beeline Cloud полностью управляет системными компонентами, обновлениями, безопасностью

На базе Deckhouse Kubernetes Platform

Все модули уже интегрированы:
сеть, хранилище, мониторинг, балансировщики, политики безопасности

Работает на инфраструктуре Beeline Cloud

Гарантированные ресурсы, SLA, интеграция с VPC, Load Balancers

Процесс подключения и использования

01

Архитектурное
планирование

Анализ требований, SLA

02

Создание кластера

15–30 минут, доступ через
kubecfg/веб-консоль

03

Go Live

Запуск, документация,
onboarding-сессия

04

Бесплатное
пилотирование

05

Поддержка миграции
workload'ов

Помощь в адаптации
манифестов

06

Поддержка и развитие

Регулярные
патчи/обновления,
консультации по best
practices

beeline cloud ×  Deckhouse

Что дальше?

Тестирование: 30 дней бесплатно

