ANNA

Evolutionary Infrastructure: Reload

ANNA

@aatarasoff

@aatarasoff

@aatarasoff

@aatarasoff

Минздрав предупреждает

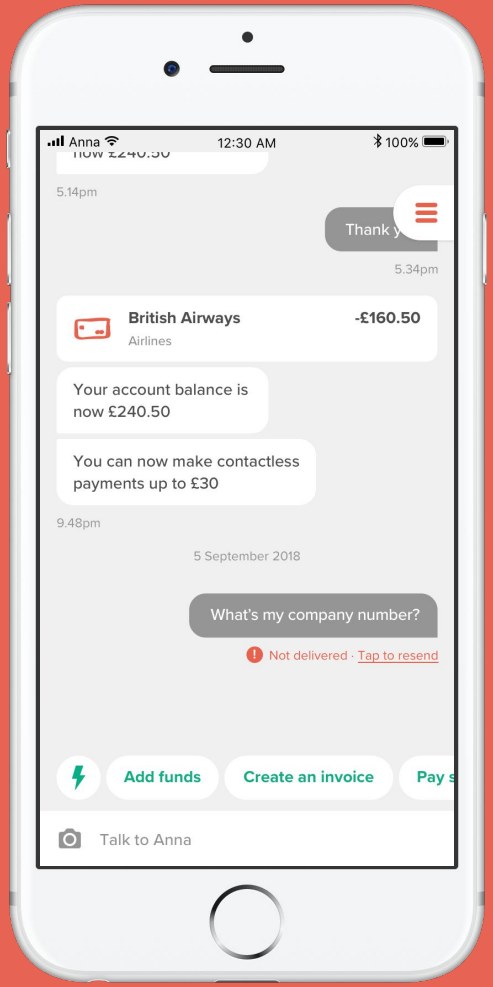
Мнение докладчика может не совпадать с официальной позицией его работодателя, коллег или других специалистов.

Все представленные в докладе сведения, примеры, выводы и другую информацию вы можете использовать на свой страх и риск. За все ваши действия ответственность несёте только вы сами.

Что такое ANNA?

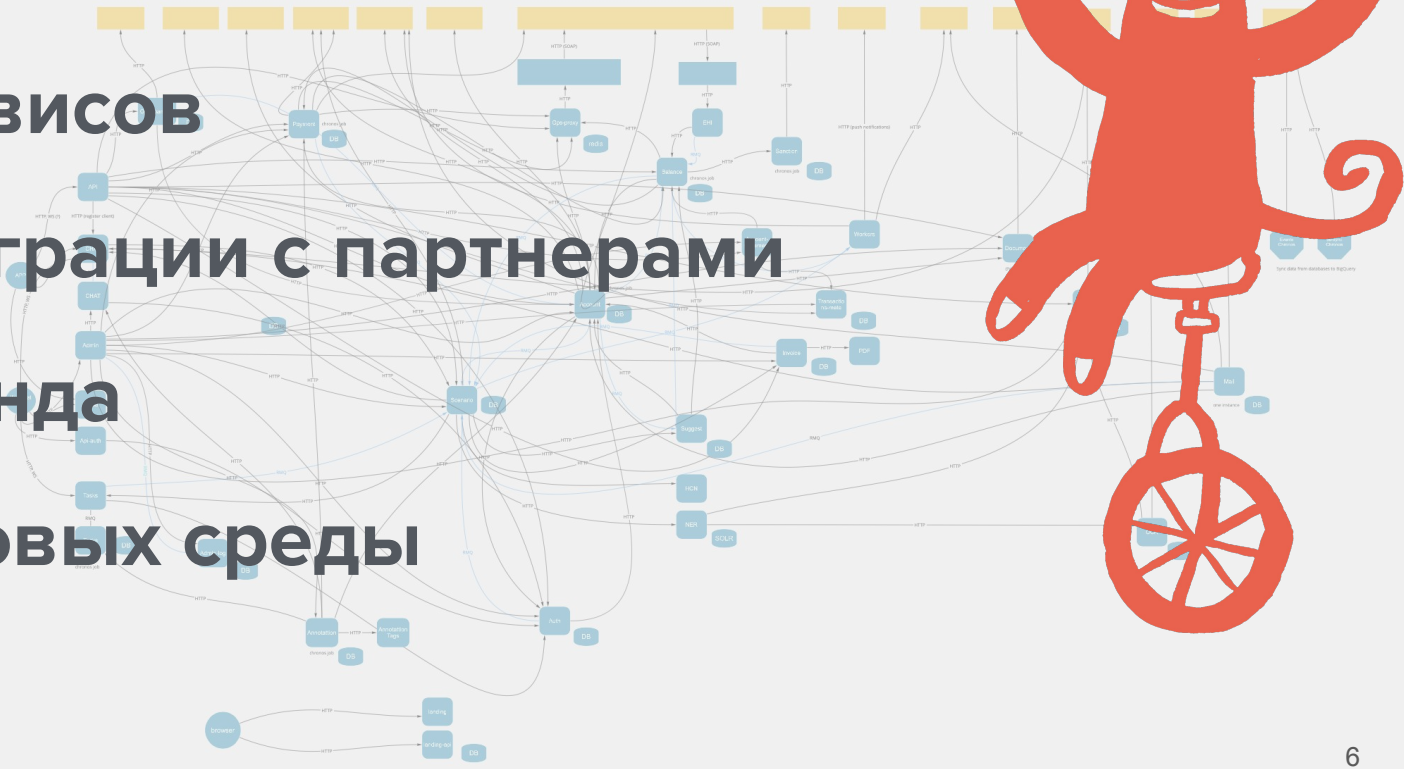
ANNA stands for Absolutely No Nonsense Admin. Because that’s what we do.

- Стартап в UK
- ~~Банк~~ для бизнеса
- Mobile-First
- AI Assistant



When We Were Young

- 10 сервисов
- 2 интеграции с партнерами
- 1 команда
- 4 тестовых среды



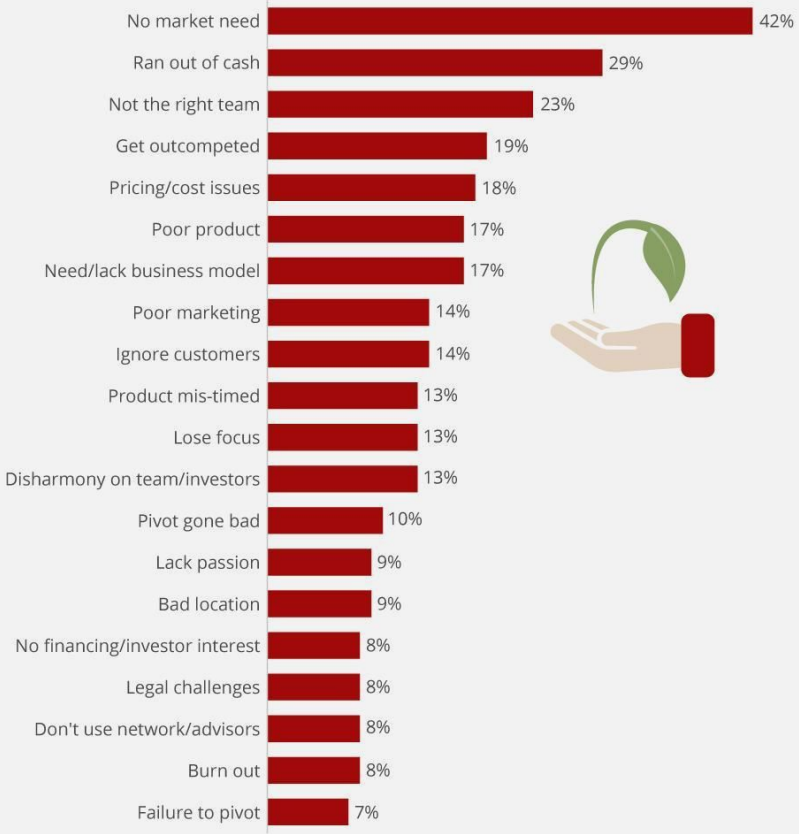
We Are Still Young, But...

- 10 -> 150+ сервисов
- 2 -> 10+ интеграций с партнерами
- 1 -> 7 независимых команд
- 4 -> 30+ тестовых сред

Пролог: инфраструктура и эволюция

The Top Reasons Startups Fail

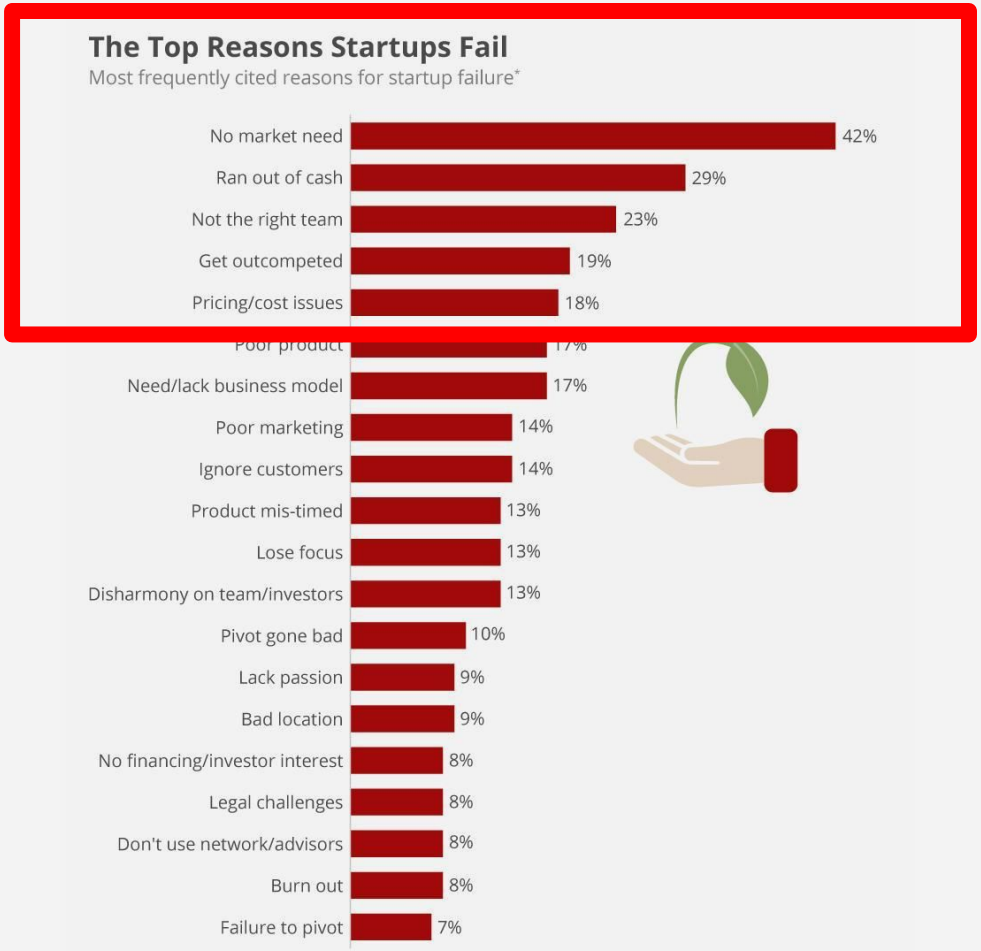
Most frequently cited reasons for startup failure*



* Based on an analysis of 101 startup post-mortems

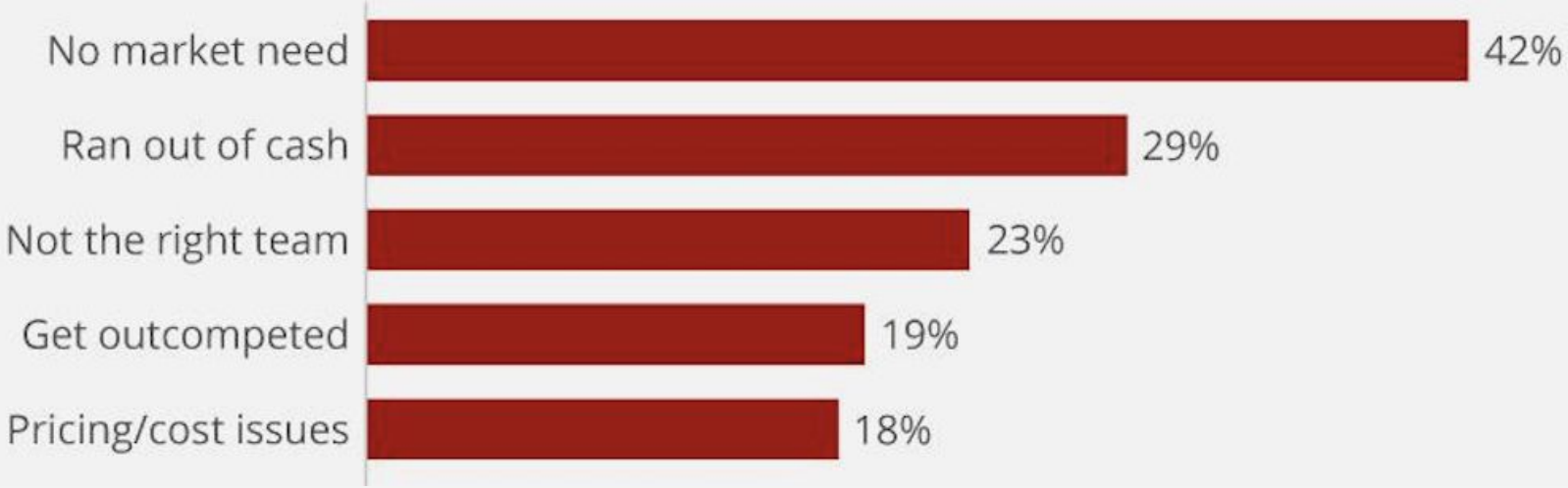
Source: CB Insights

Forbes statista


The Top Reasons Startups Fail
Most frequently cited reasons for startup failure*
No market need 42%
Ran out of cash 29%
Not the right team 23%
Get outcompeted 19%
Pricing/cost issues 18%
Poor product 17%
Need/lack business model 17%
Poor marketing 14%
Ignore customers 14%
Product mis-timed 13%
Lose focus 13%
Disharmony on team/investors 13%
Pivot gone bad 10%
Lack passion 9%
Bad location 9%
No financing/investor interest 8%
Legal challenges 8%
Don't use network/advisors 8%
Burn out 8%
Failure to pivot 7%
* Based on an analysis of 101 startup post-mortems
@StatistaCharts
Source: CB Insights
Forbes
statista

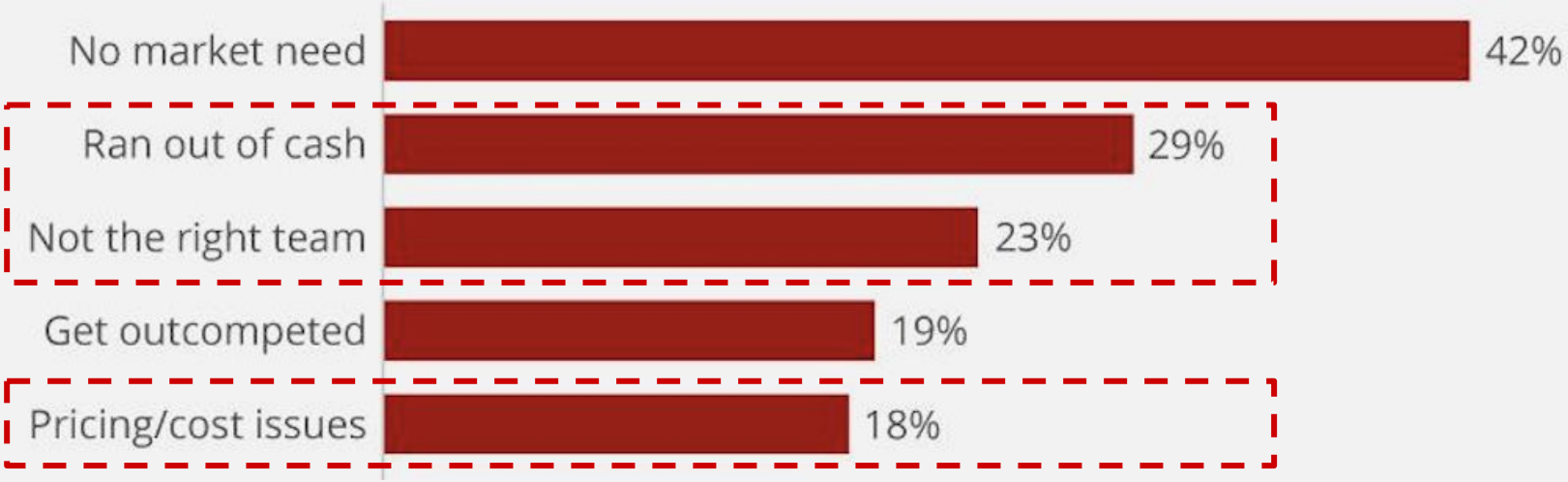
The Top Reasons Startups Fail

Most frequently cited reasons for startup failure*



The Top Reasons Startups Fail

Most frequently cited reasons for startup failure*



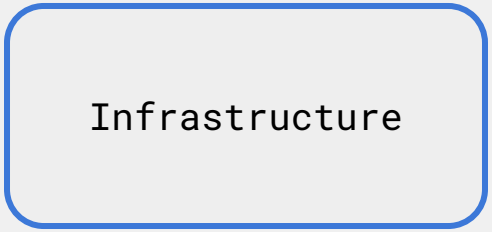
Инфраструктура —

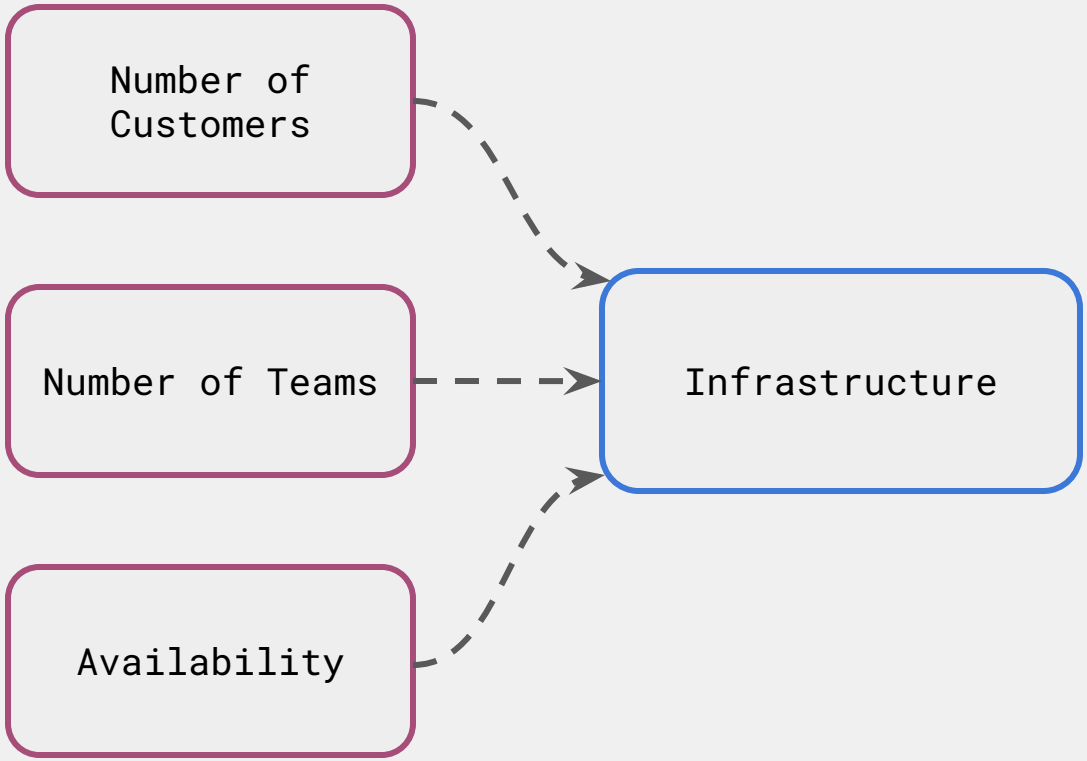
Инфраструктура — всё, что не является программным кодом приложений

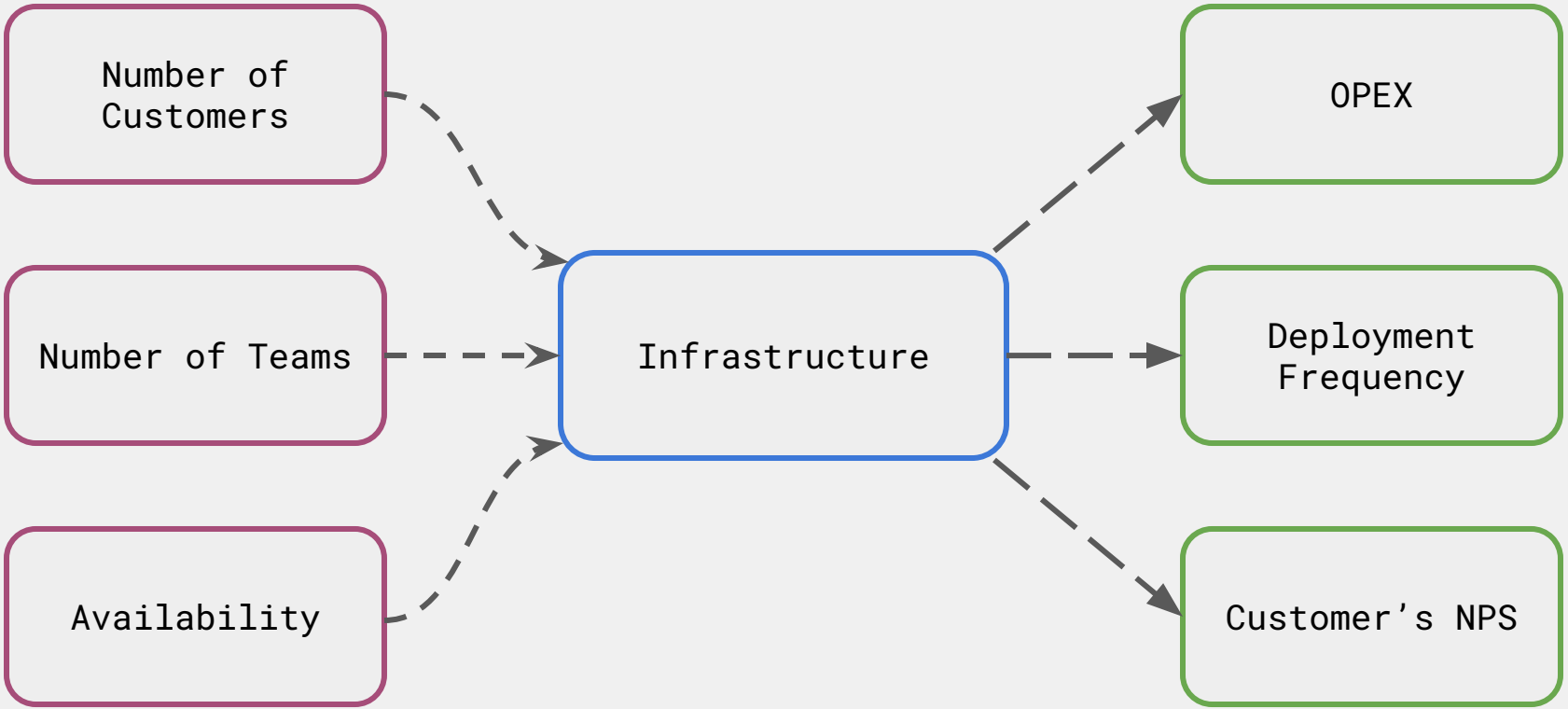
И это тоже инфраструктура

- Кластеры развертывания ПО
- Базы данных
- Подсистема логирования
- Подсистема мониторинга
- Система запуска и анализа тестов
- CI / CD

  (включая конфигурацию пайплайнов)


Infrastructure


Number of Customers
Number of Teams
Availability
Infrastructure


Number of Customers
Number of Teams
Availability
Infrastructure
OPEX
Deployment Frequency
Customer’s NPS

**Выживает не самый сильный и умный,
а тот, кто лучше всех приспосабливается к изменениям**

Чарльз Дарвин

Эволюционная инфраструктура та, которая умеет приспосабливаться под требования бизнеса и команд

(чтобы стартап не склеил ласты)

IMHO

Инфраструктура как живой организм

Внешние “раздражители”

Внутренние “раздражители”

Внешние “раздражители”

Внутренние “раздражители”

Наследственность

Изменчивость

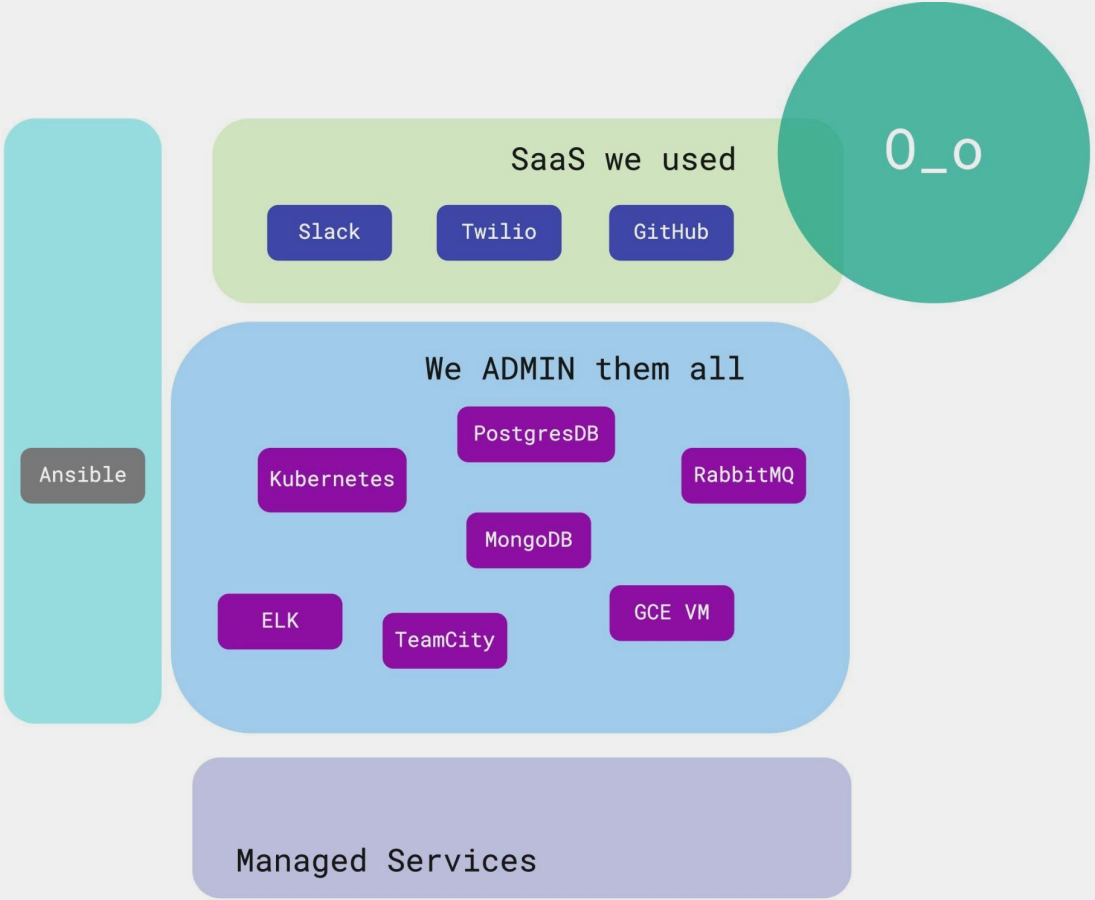
Внешние “раздражители”

Внутренние “раздражители”

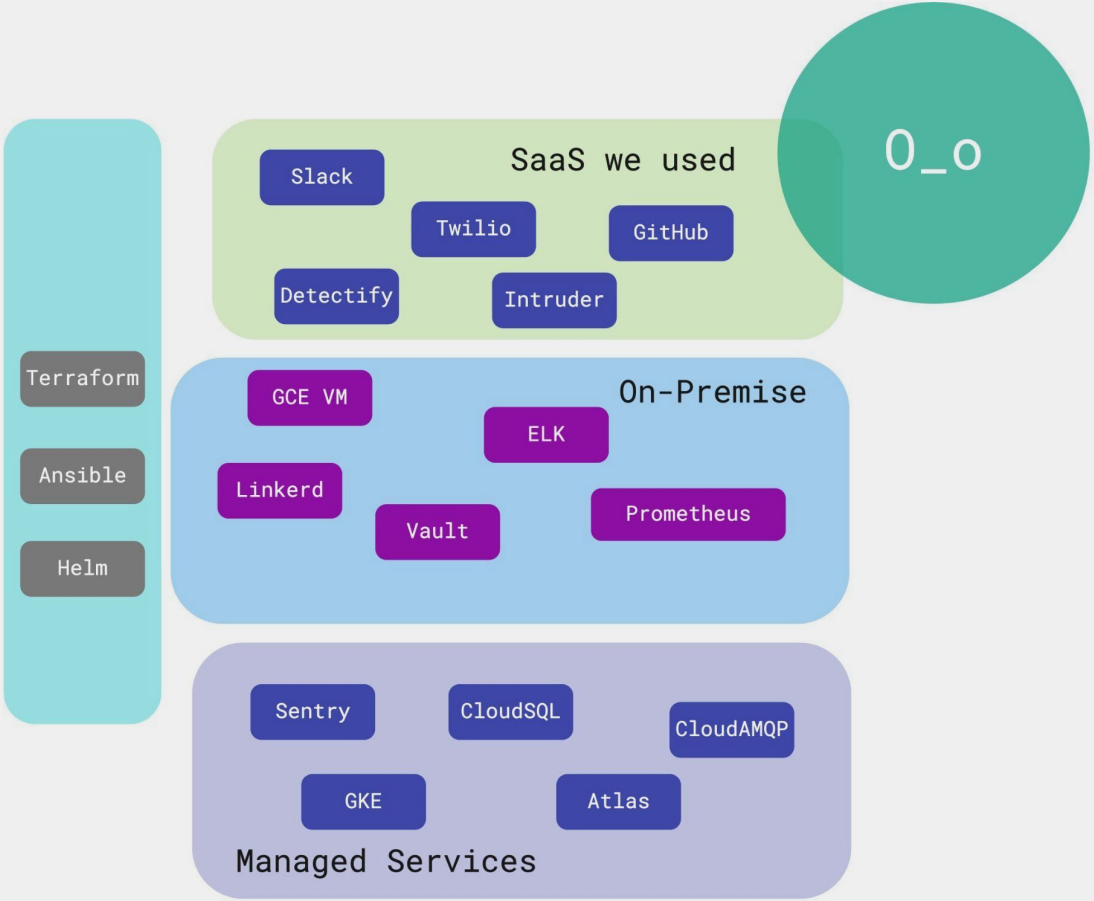
Наш единорог изменил цвет

Но это всё ещё единорог

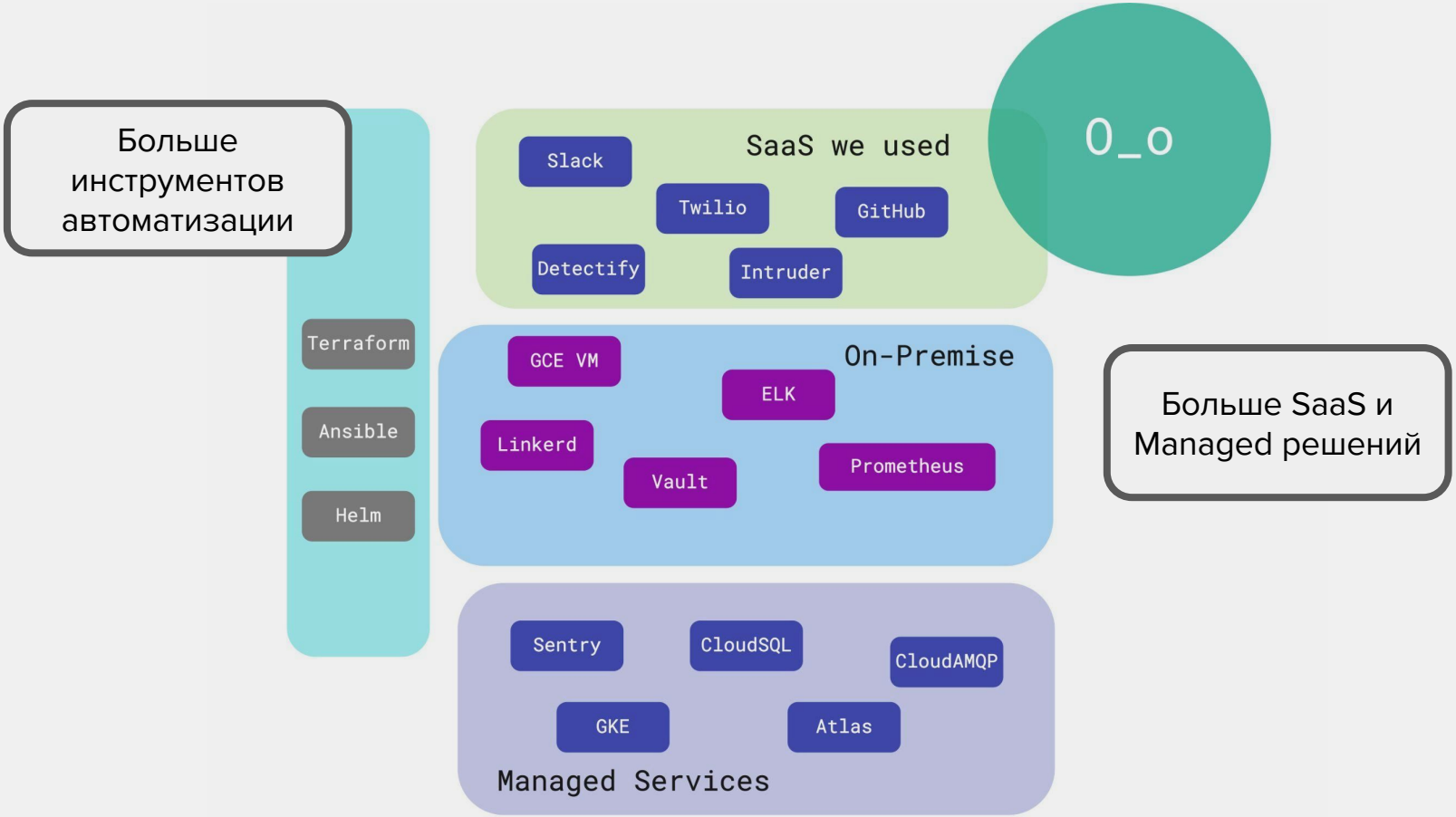
Знакомьтесь - ANNAzaur

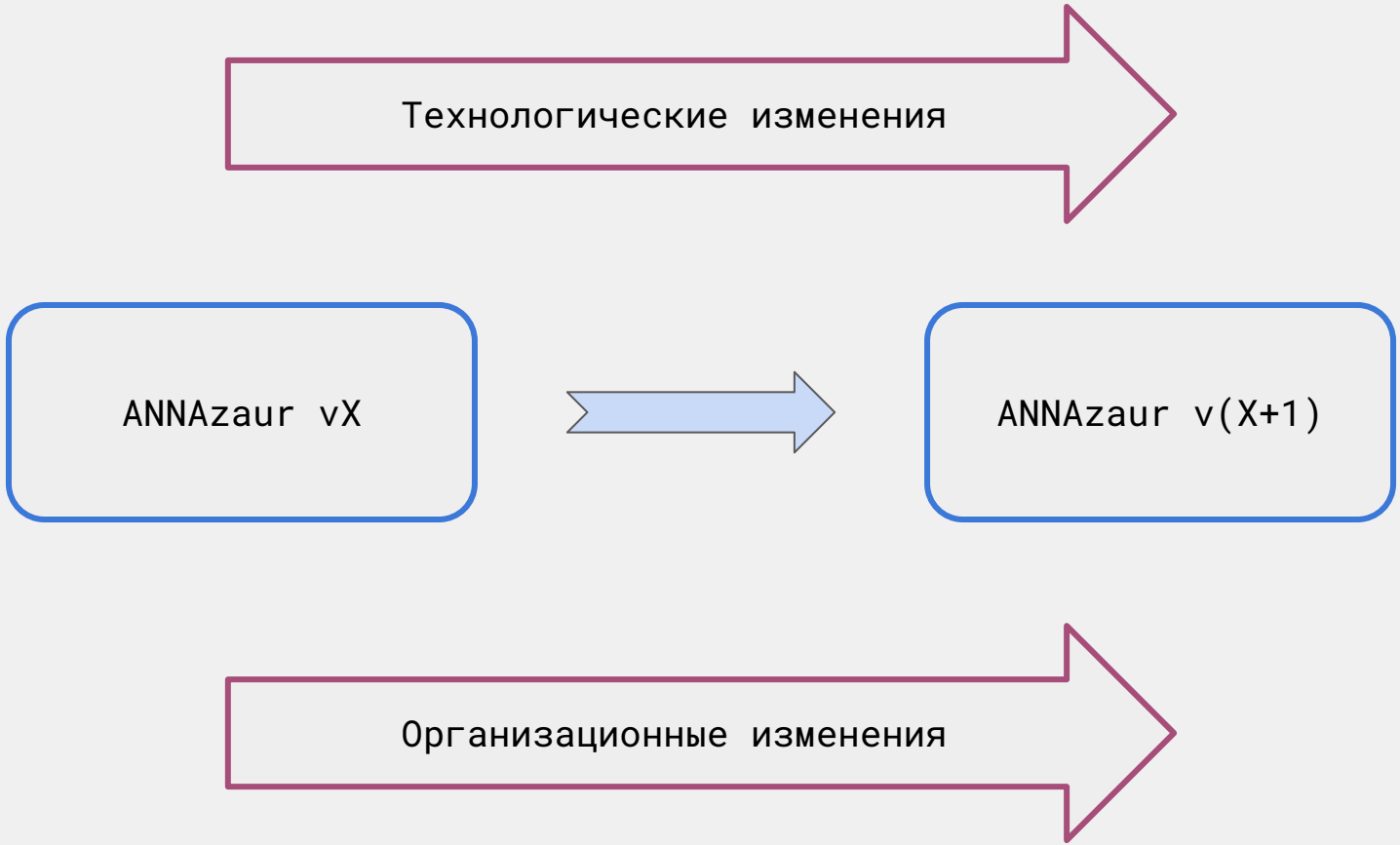


ANNAzaur сейчас



ANNAzaur сейчас




Технологические изменения
ANNAzaur vX
ANNAzaur v(X+1)
Организационные изменения

“Простейшие” операции

- **Добавление элемента**
  - **не было X -> стало X**

“Простейшие” операции

- **Добавление элемента**
  - **не было X -> стало X**
- **Замена элемента**
  - **было X -> стало Y**

“Простейшие” операции

- **Добавление элемента**
  - **не было X -> стало X**
- **Замена элемента**
  - **было X -> стало Y**
- **Модификация элемента**
  - **было X -> стало X***

“Простейшие” операции

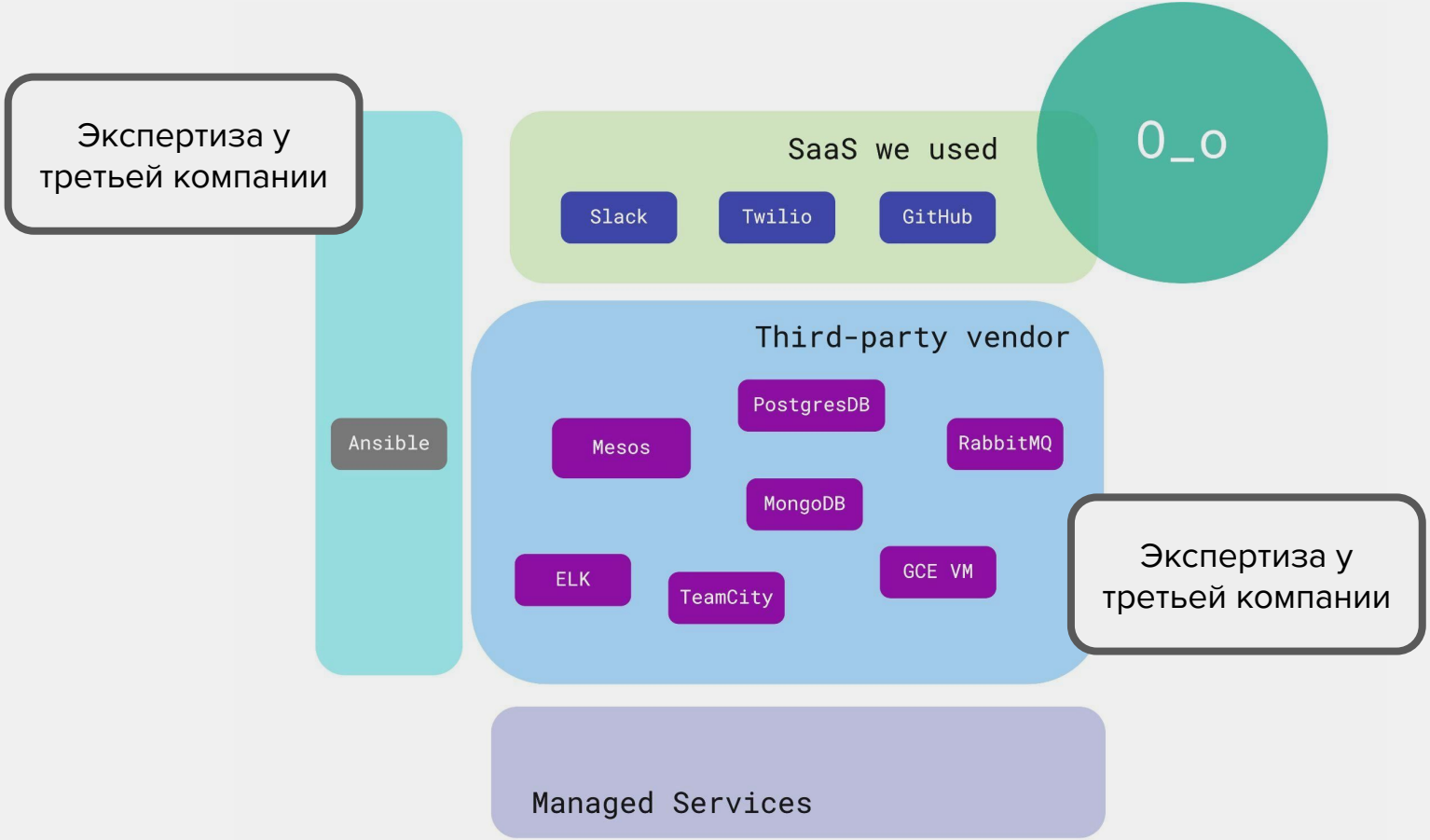
- **Добавление элемента**
  - **не было X -> стало X**
- **Замена элемента**
  - **было X -> стало Y**
- **Модификация элемента**
  - **было X -> стало X***
- **Удаление элемента**
  - **было X -> нет X**

Кто делает эти операции?

1. Генная инженерия

Когда-то я пришёл в команду

Организационный аспект



Проблемы отсутствия экспертизы

- “Колодцы”
  - Мы ставим задачу, но не можем решить
  - Они могут решить задачу, но не могут поставить
- Вдолгую не кажется выигрышной стратегией

Команда SRE (платформы)

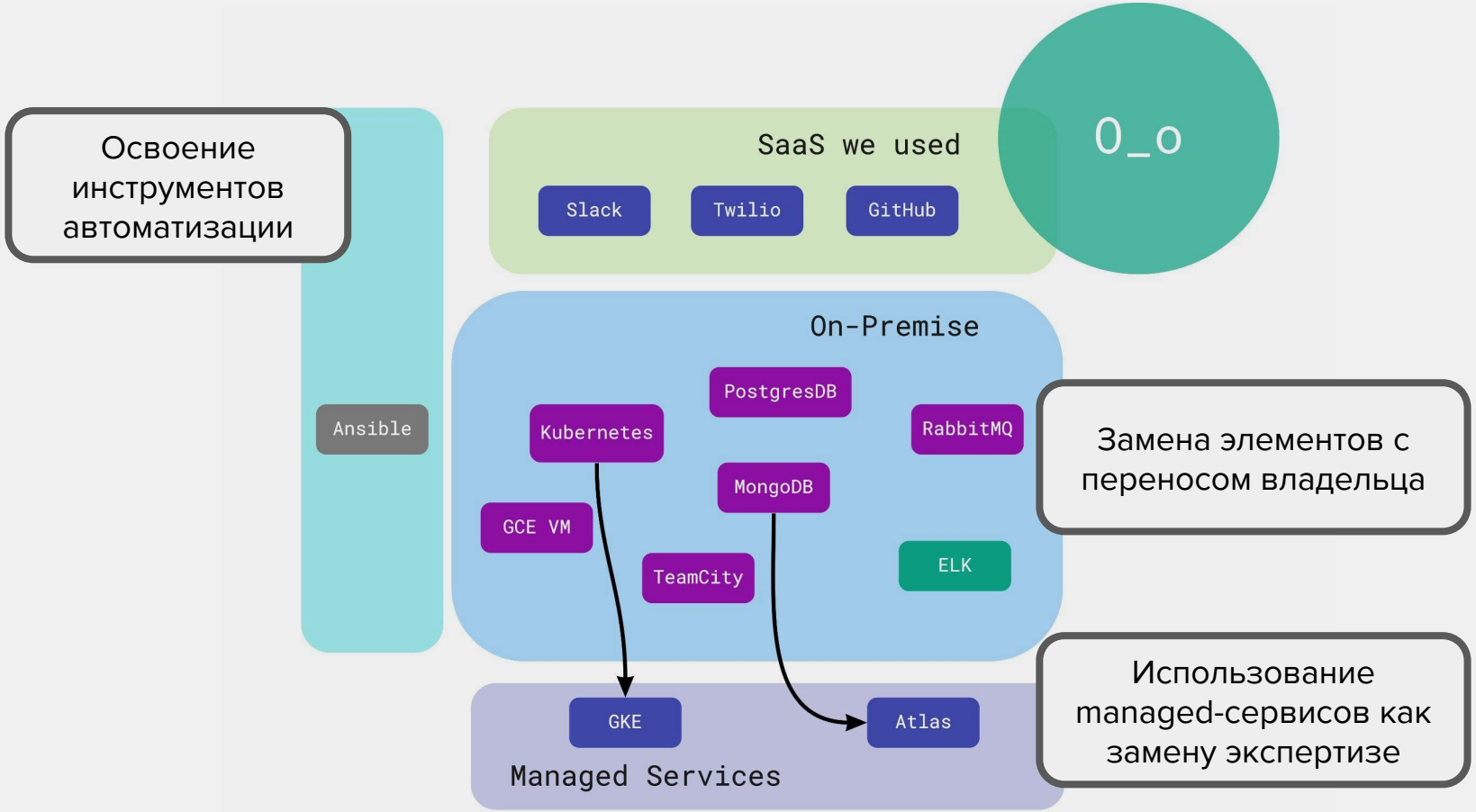
Стартовый состав команды

- CTO
- Director of Engineering
- Third-Party Engineers

Стартовый состав команды

- CTO
- Director of Engineering
- Third-Party Engineers
- + SRE Engineer

Едим слона по частям



Состав команды

- CTO
- Director of Engineering
- SRE Engineer
- Lead Developer x2
- Head of Mobile
- No Third-Party Engineers

Скиллы команды SRE позволяют целеполагать и реализовывать любую задачу

Скиллы команды SRE позволяют **целеполагать** и реализовывать любую задачу

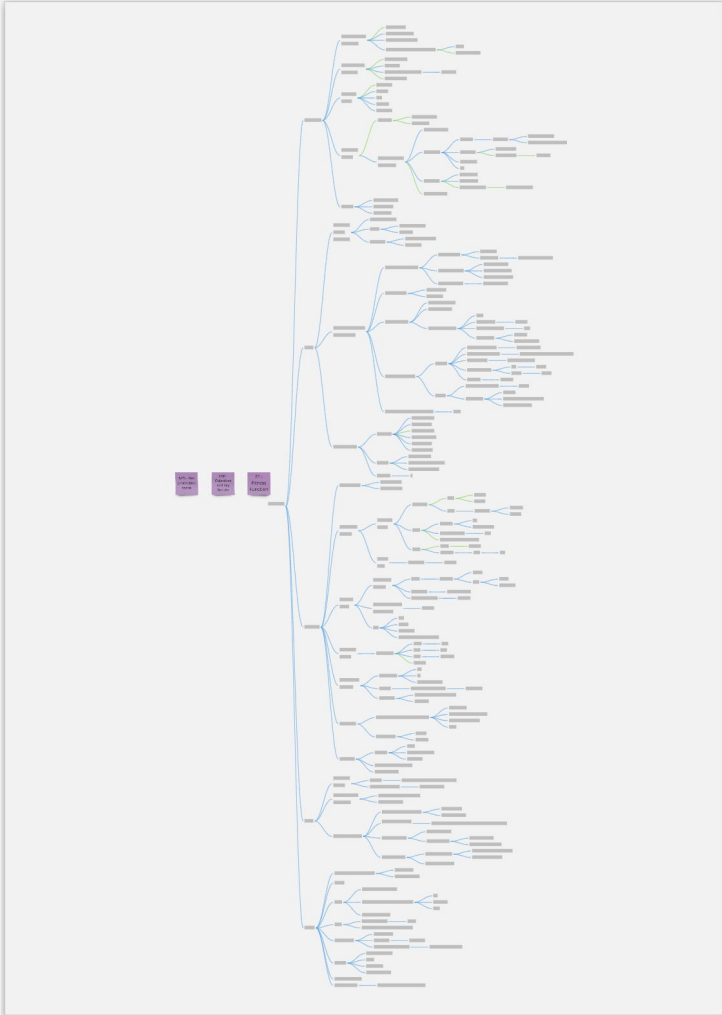
Engineering Vision

Зачем?

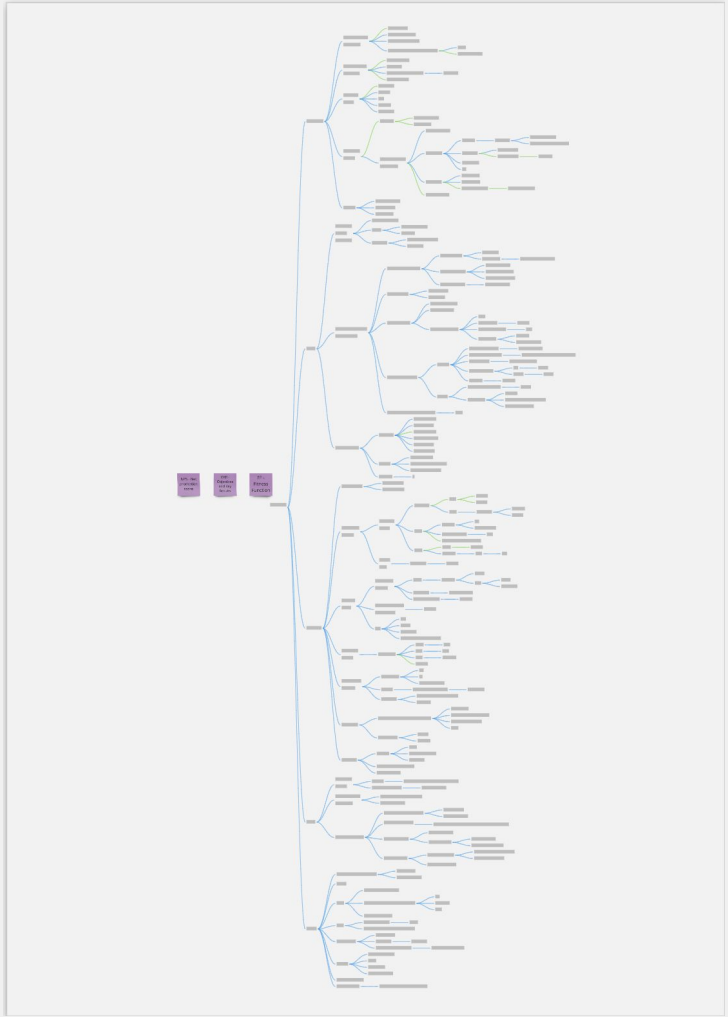
- **Приспособить к требованиям бизнеса**
  - **количество пользователей увеличится в N раз**
  - **ожидается интенсивный найм**

Зачем?

- Приспособить к требованиям бизнеса
  - количество пользователей увеличится в N раз
  - ожидается интенсивный найм
- Комфорт команд разработки
  - коммуникация что ожидать
  - улучшение качества инструментария



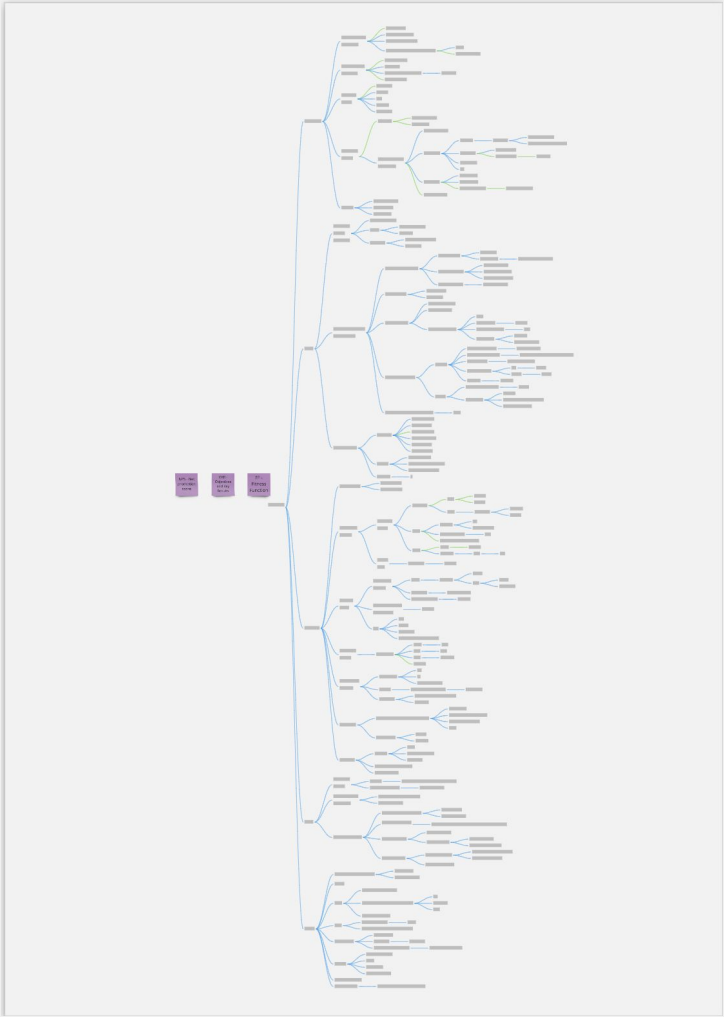
Набор идей для реализации в виде mind-map



Набор идей для реализации в виде mind-map

Каждая идея — эпик или некоторая цель:

- try Service Mesh
- GH Actions for 30 services



Набор идей для реализации в виде mind-map

Каждая идея — эпик или некоторая цель:

- try Service Mesh
- GH Actions for 30 services

План — не KPI и не OKR

Как мы понимаем, что эволюционируем в нужном направлении?

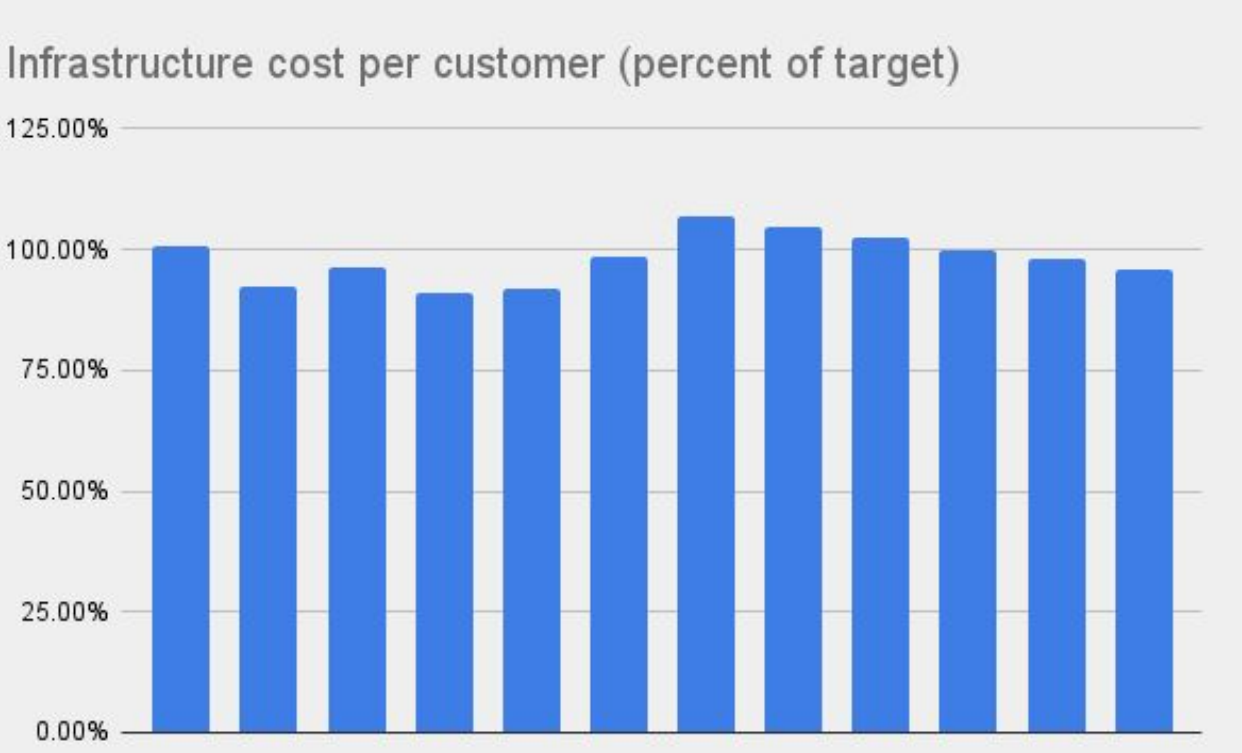
Fitness Functions

Чтобы наш ANNAzaur был в форме

Примеры Fitness-функций

- F_1 = OPEX < x$ per customer
- F_2 = Availability > 99,999%
- F_3 = Developers are Happy

Затраты на клиента
(в % от целевого значения)



Скиллы команды SRE позволяют целеполагать и реализовывать любую задачу

“Простейшие” операции

- Добавление элемента
  - не было X -> стало X
- Замена элемента
  - было X -> стало Y
- **Модификация элемента**
  - **было X -> стало X***
- Удаление элемента
  - было X -> нет X

2. Постоянные инкрементные улучшения

Проблема

нехватка ресурсов команды платформы

Managed Services

Примеры

- Google Kubernetes Engine
- Atlas (MongoDB)
- Google CloudSQL (PostgreSQL)
- CloudAMQP (Cloud RabbitMQ)

Бенефиты

- Постоянные апдейты
- Легкость миграций
- Тулинг из коробки
- Минимум времени на саппорт

Trade-Offs

- **Vendor Lock**
- **Ограничения сервиса**
- **Дороже** (но это не точно)

Vendor Lock не так уж страшен

- **Cloud - это уже Vendor Lock**
  - **наша инфраструктура полностью облачная**

Vendor Lock не так уж страшен

- Cloud - это уже Vendor Lock
  - наша инфраструктура полностью облачная
- Возможность обратной миграции
  - GKE -> собственный кластер K8s
  - Atlas -> MongoDB
  - CloudSQL -> PostgreSQL

Ограничения сервиса

- **Узнать заранее**
  - **GKE позволяет не больше 110 подов на ноду**

Ограничения сервиса

- Узнать заранее
  - GKE позволяет не больше 110 подов на ноду
- Обращение в саппорт
  - есть вероятность, что ограничения снимут

Ограничения сервиса

- Узнать заранее
  - GKE позволяет не больше 110 подов на ноду
- Обращение в саппорт
  - есть вероятность, что ограничения снимут
- Воркэарунд и хаки никто не отменял
  - https://aatarasoff.medium.com/how-we-hack-densely-packed-gke-cluster-ccbc4b2848e6

Own K8s (GCP) -> GKE

До: N$ в месяц
(VMs + внешняя поддержка)

После: 0.8N$ в месяц
Плюс плюшки: (security, alerts, updates)

Managed Service Fitness Function

F = (exists) && (fit) && (cost < (2 * x + ½ * s)),

где x — стоимость self-hosted,

а s — средняя ставка SRE-инженера

Скиллы команды SRE позволяют целеполагать и реализовывать любую задачу

Прям так совсем любую задачу?

3. Особенности мутирования

“Простейшие” операции

- **Добавление элемента (в меньшей степени)**
  - **не было X -> стало X**
- **Замена элемента (в бОльшей степени)**
  - **было X -> стало Y**
- **Модификация элемента**
  - **было X -> стало X***
- **Удаление элемента**
  - **было X -> нет X**

Виды мутаций по исполнителю

- Простой случай
- Планирование на неделю
- Стендапы 2-3 раза в неделю
- Канал в слаке для информирования

Виды мутаций по исполнителю

- Конфликт бизнес и технических задач
  - Пассивность команд
  - “Продажа” пользы “мутации”

Не работает

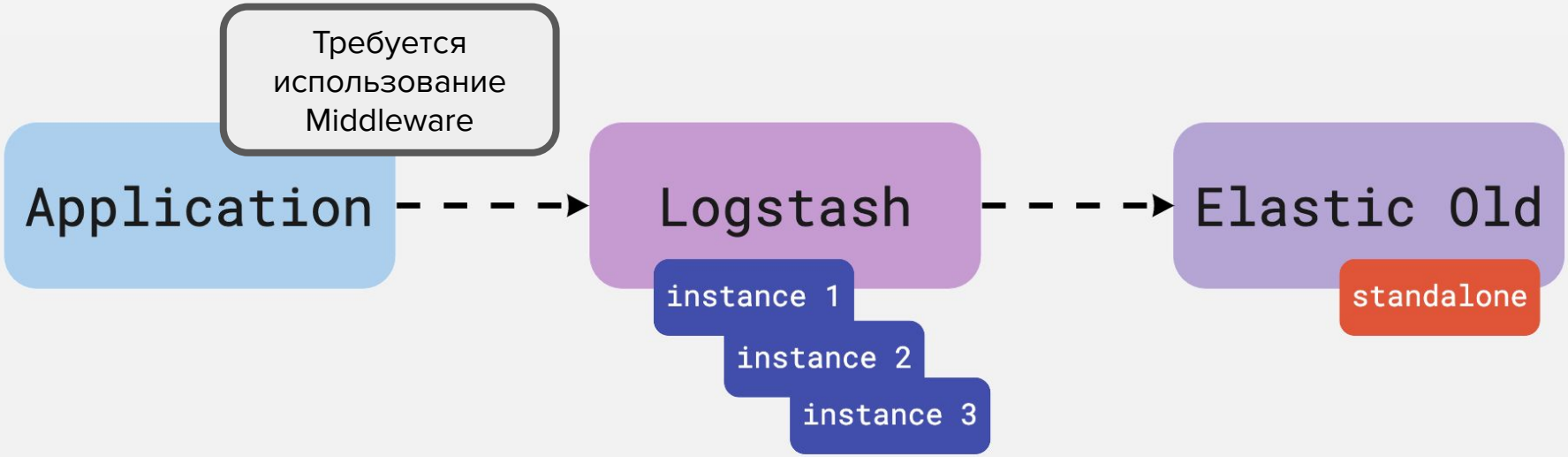
(во всяком случае у нас)

- **Авторитаризм**
  - **будет так с 3 сентября**
- **Демократия**
  - **давайте обсудим, а потом ещё раз обсудим**
  - **убеждение, “продажа”, просьба**

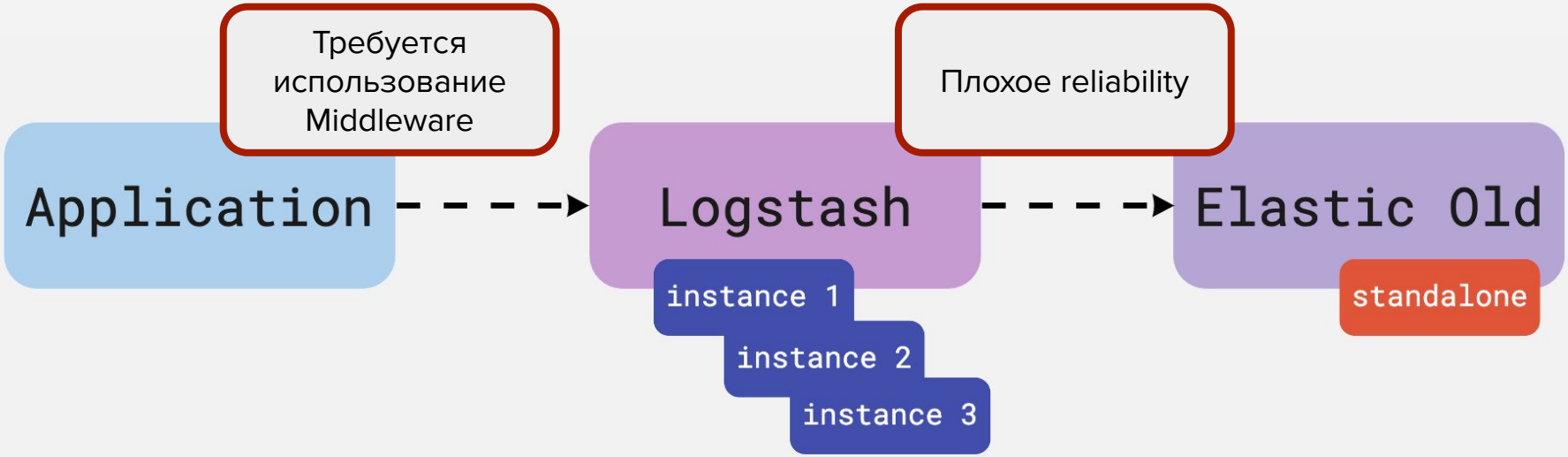
Soft-Hard Deadlines (SHD)

Кейс: переход на новую систему логов

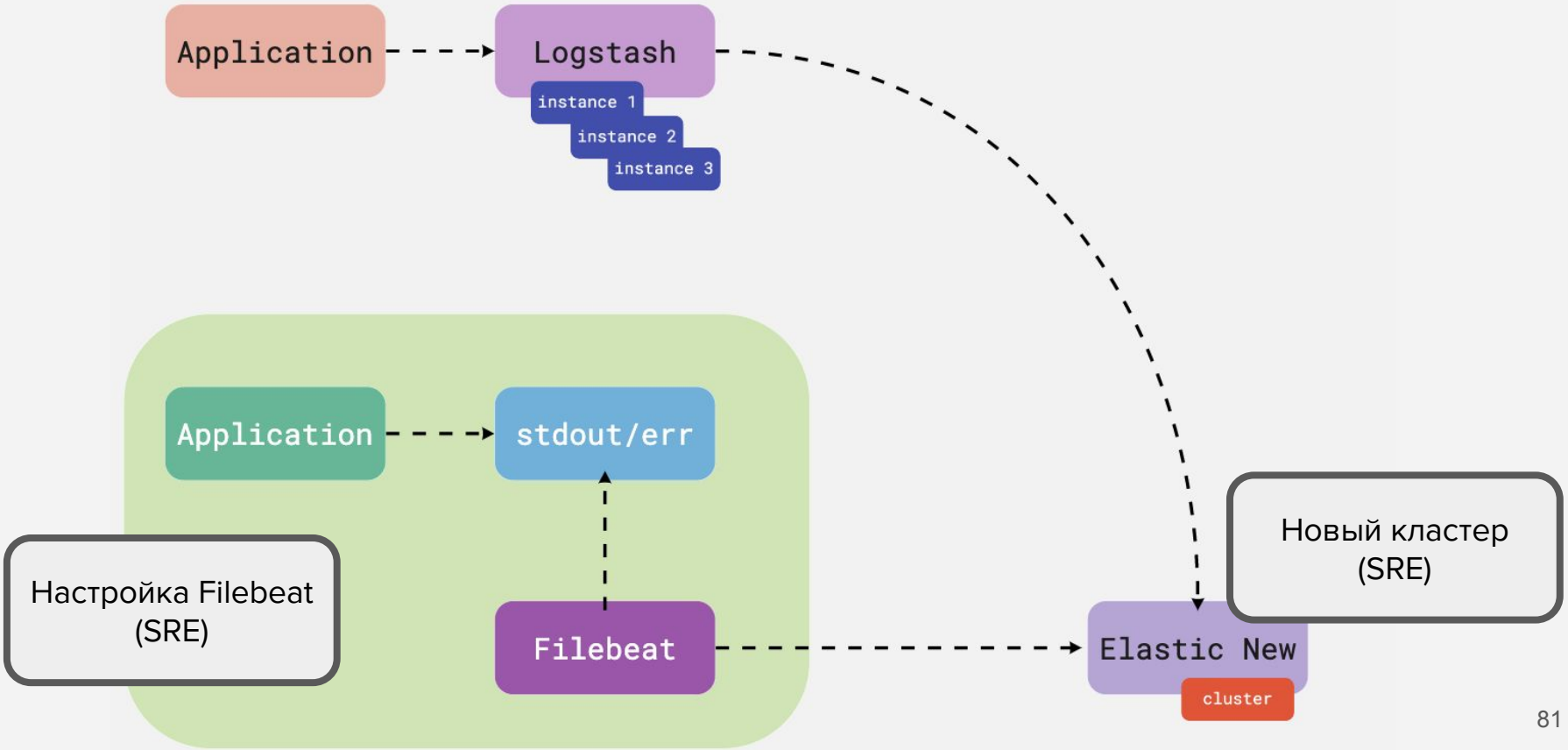
Старая схема логирования



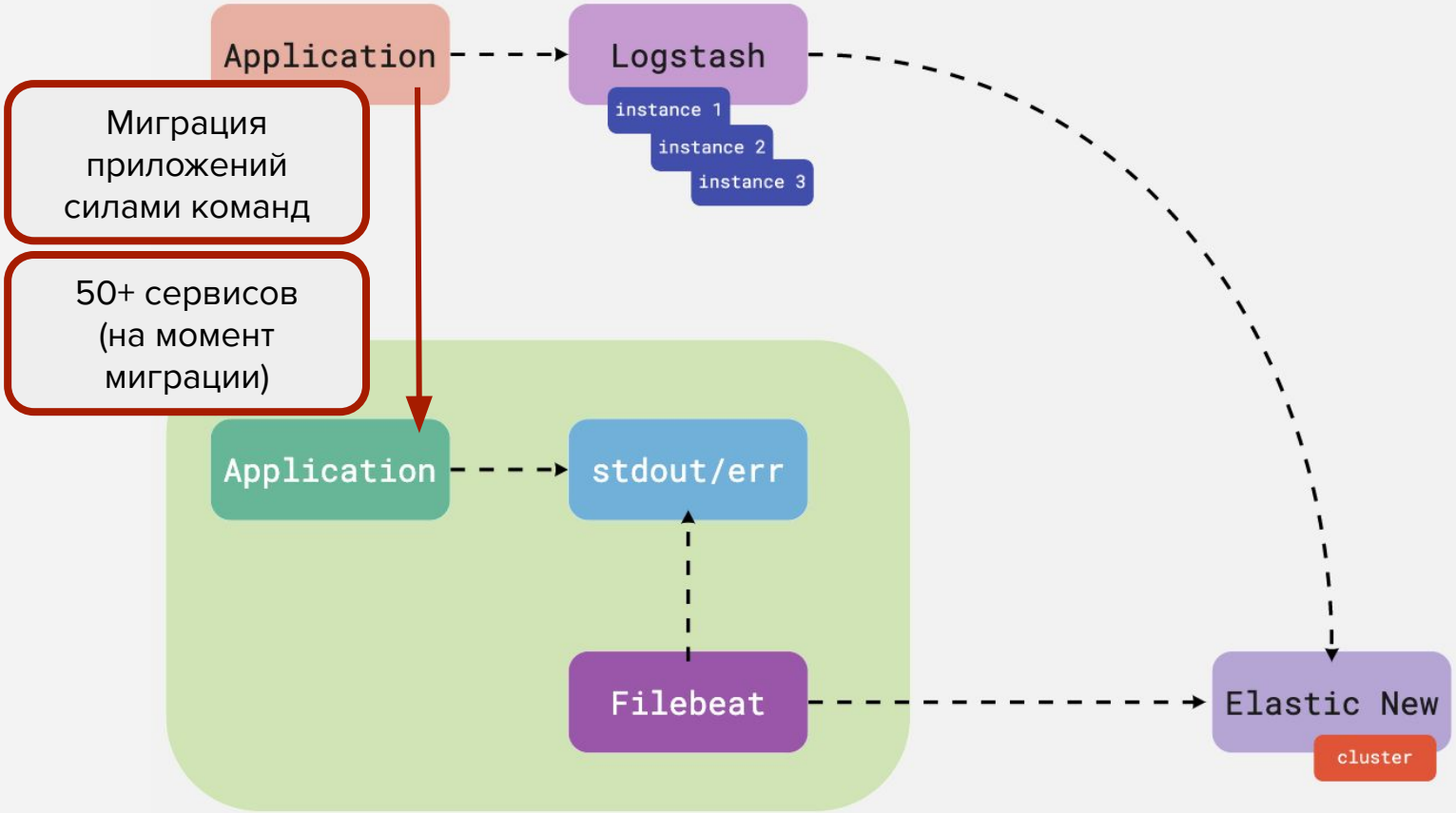
Очевидные проблемы



Новая схема логирования



“Узкое” место


Application
Logstash
instance 1
instance 2
instance 3
Миграция приложений силами команд
50+ сервисов (на момент миграции)
Application
stdout/err
Filebeat
Elastic New
cluster

Конфигурация сервисов

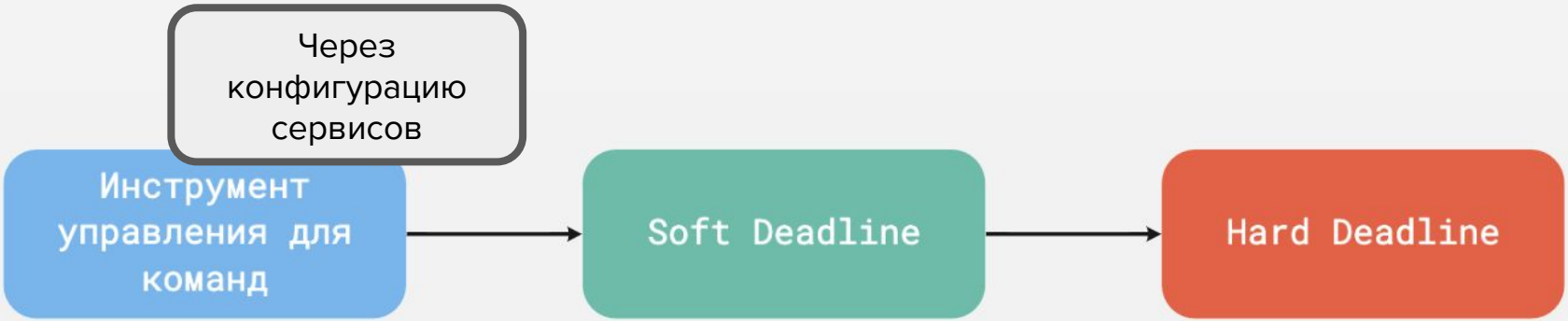
```
#feature-toggle to enable standardized logs
logs: True
```

Конфигурация сервисов

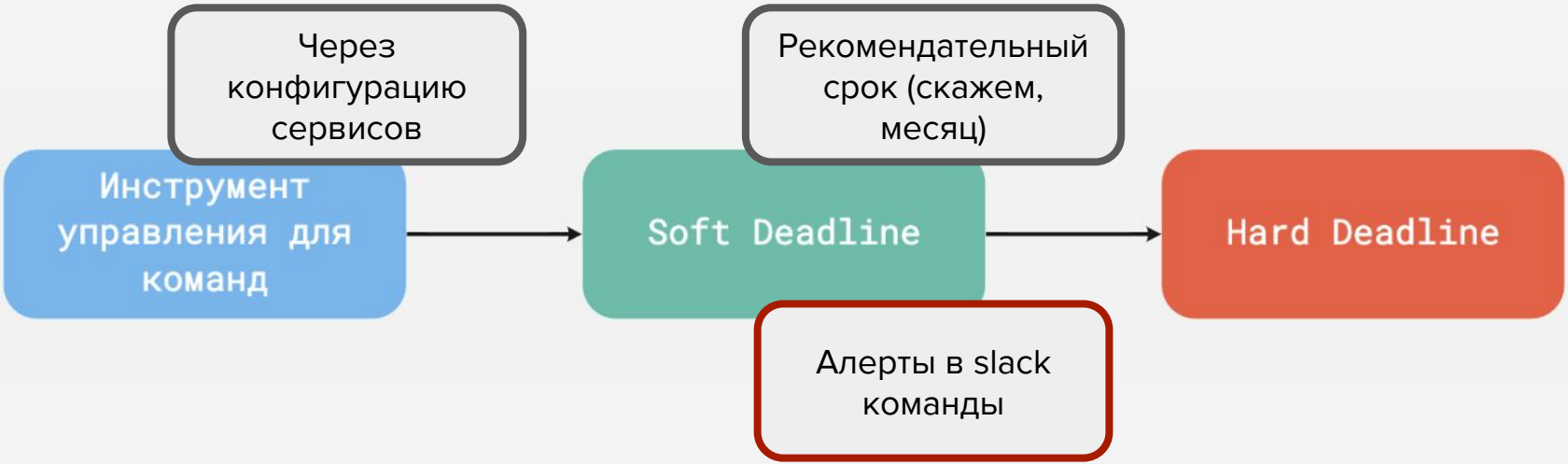
```
#feature-toggle to enable standardized logs
logs: True

#pod modification if True
env:
  ...
  LOGS_STD_ENABLED: "{{ logs }}"
  LOGS_LEVEL: "{{ logs_std_level }}"
  LOGS_FORMAT: "{{ logs_std_format }}"
labels:
  ...
  logs-std-enabled: "{{ logs }}"
```

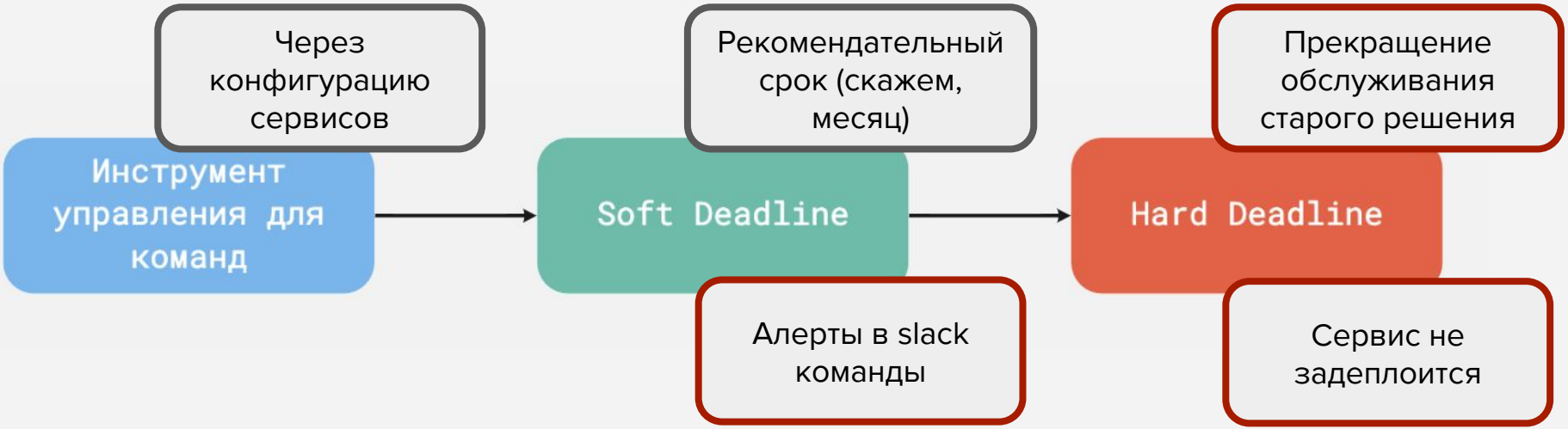
Soft-Hard Deadlines



Soft-Hard Deadlines



Soft-Hard Deadlines



Почему Soft-Hard Deadlines работает?

- Смесь авторитаризма и демократии
  - нам надо это сделать, но в разумные сроки
  - возможность планировать переход внутри каждой бизнесовой команды

Виды мутаций по исполнителю

- “Спонтанные” улучшения
  - внутренние проблемы команды
  - проблемы уровня компании
- Поддержка командой SRE
  - тиражирование через SHD

4. Как не делать больно?

“Простейшие” операции

- Добавление элемента
  - не было X -> стало X
- **Замена элемента (почти всегда)**
  - **было X -> стало Y**
- **Модификация элемента (практически никогда)**
  - **было X -> стало X***
- Удаление элемента
  - было X -> нет X

Кто наши клиенты?

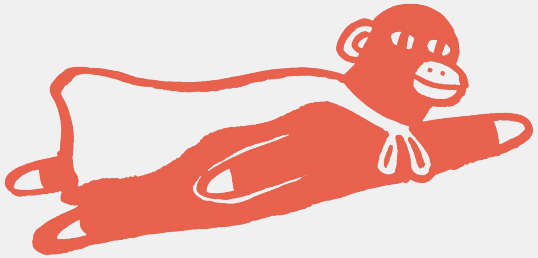
Типы клиентов

- **Пользователи, которые несут нам свои денежки**

Типы клиентов

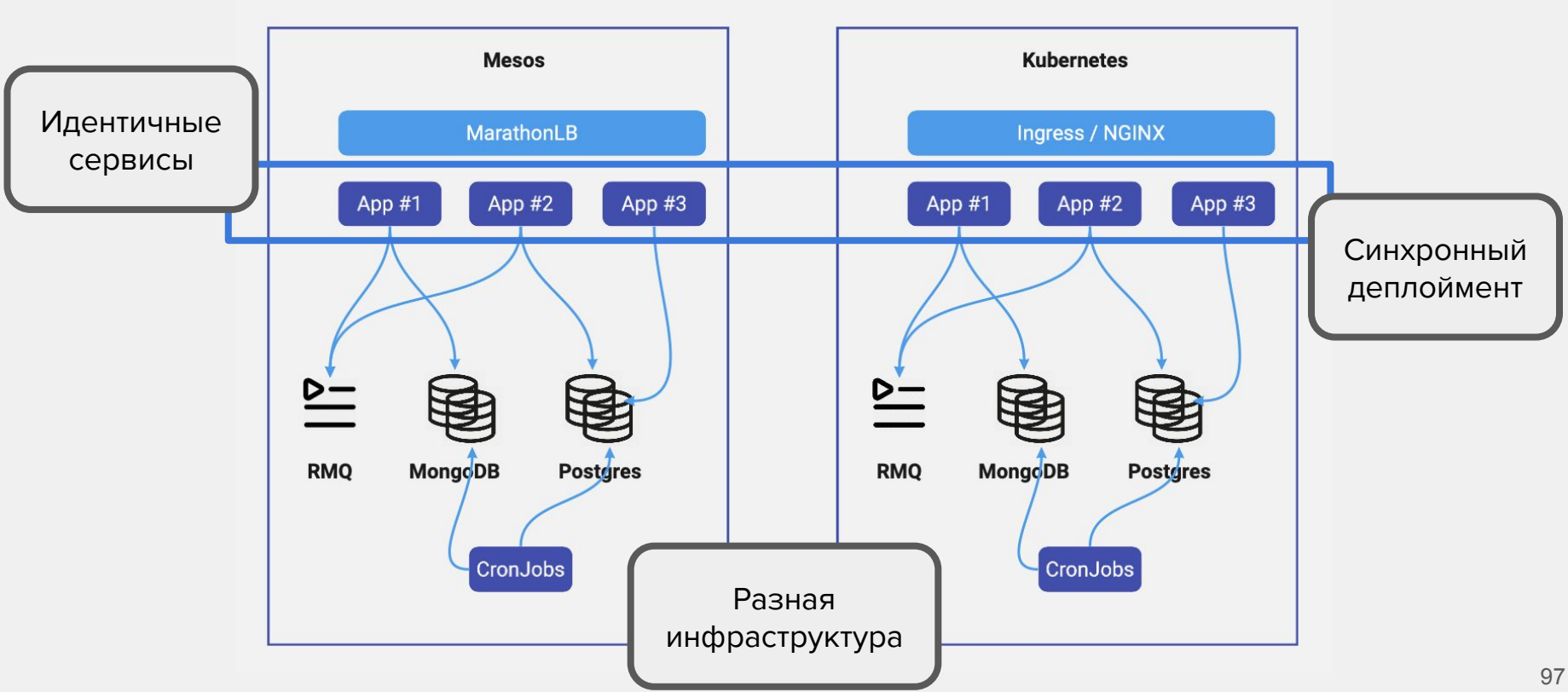
- Пользователи, которые несут нам свои денежки
- Команды разработки, которые делают так, чтобы пользователи несли больше денежек

Проблема #X
как не делать больно пользователям?

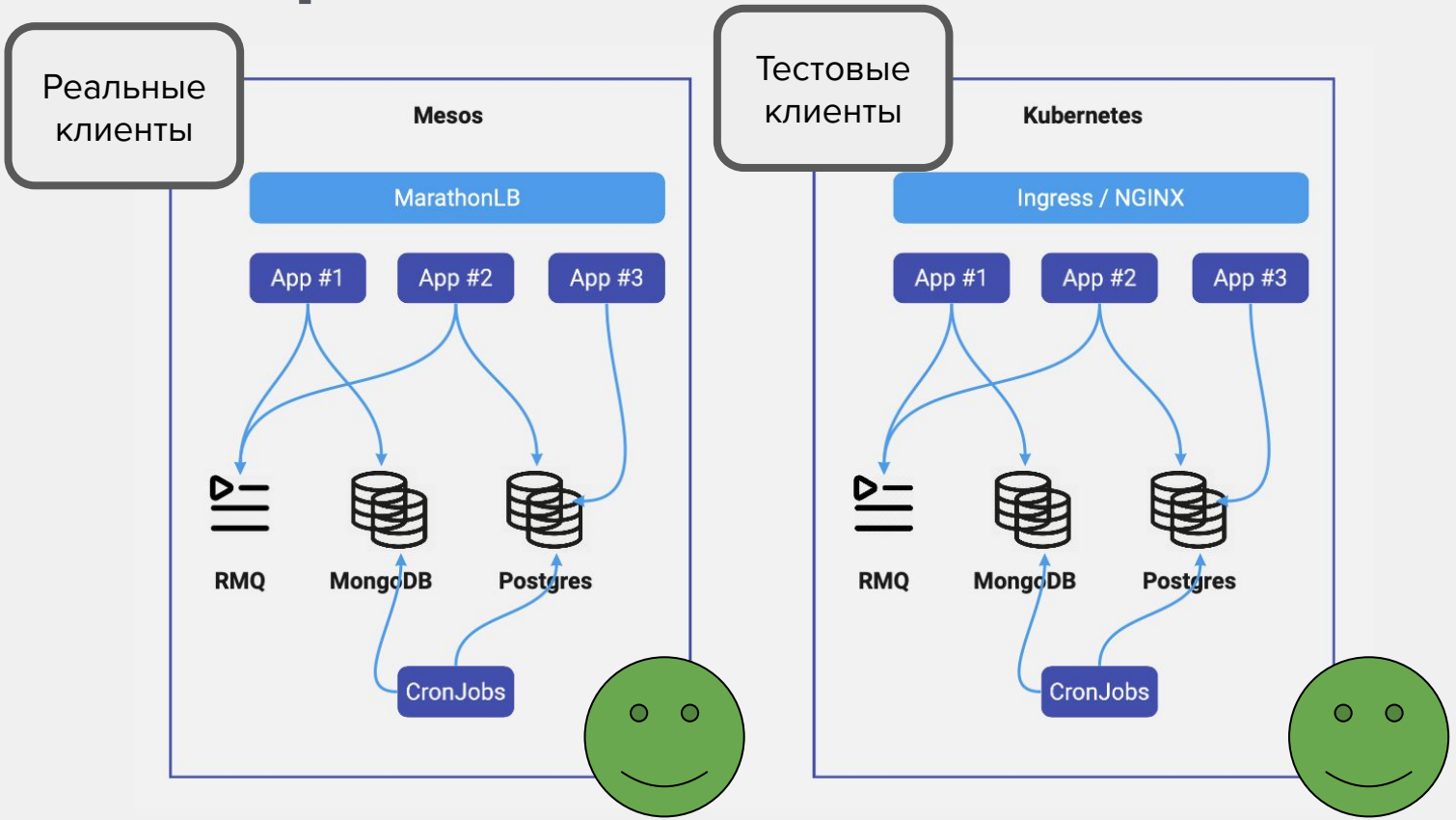


Blue-Green Infrastructure Testing

Схема миграции (mesos->K8s)

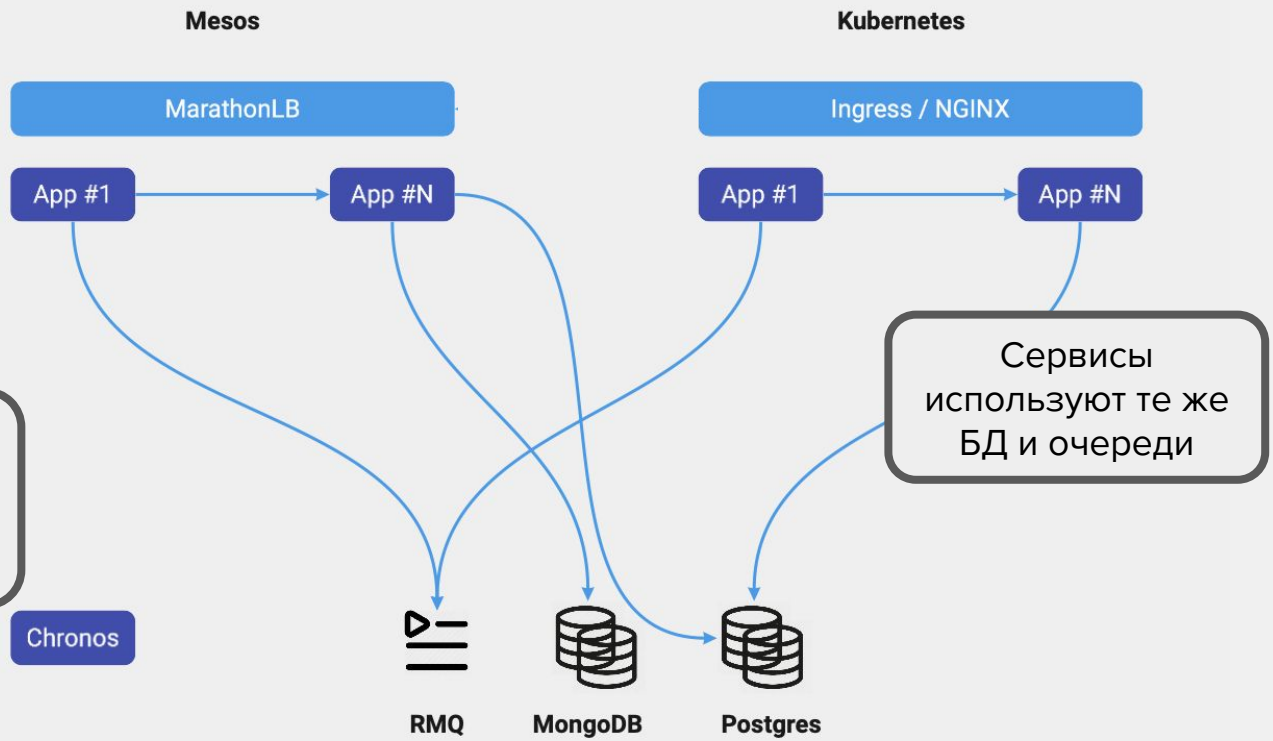


Тестирование основных кейсов

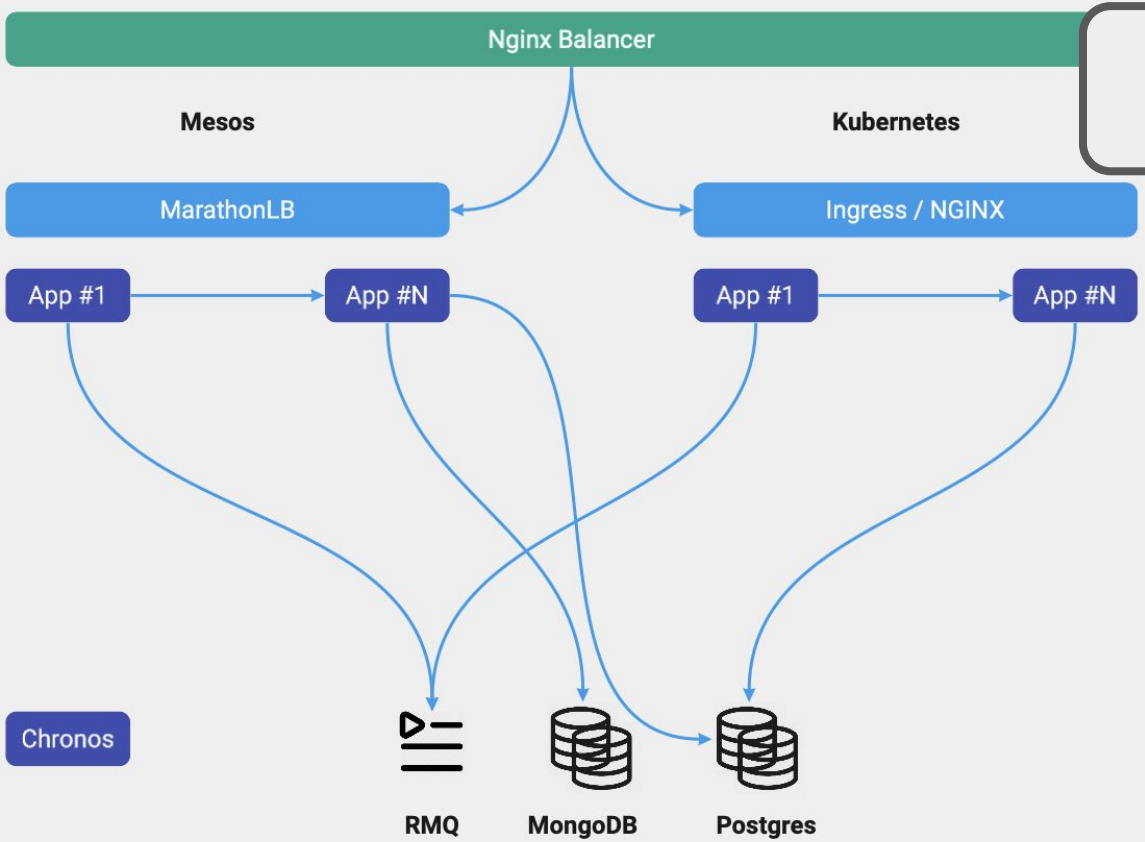


Но дальше так идти опасно

Рабочая схема миграции



A/B-тестирование



Постепенно переводим на новый кластер

Blue-Green Infrastructure Testing

- **Идентичный программный слой**
  - **тестируем инфраструктуру, но не софт**

Blue-Green Infrastructure Testing

- Идентичный программный слой
  - тестируем инфраструктуру, но не софт
- Сбор телеметрии для наблюдаемости
  - логи, метрики
  - алерты на ошибки

Blue-Green Infrastructure Testing

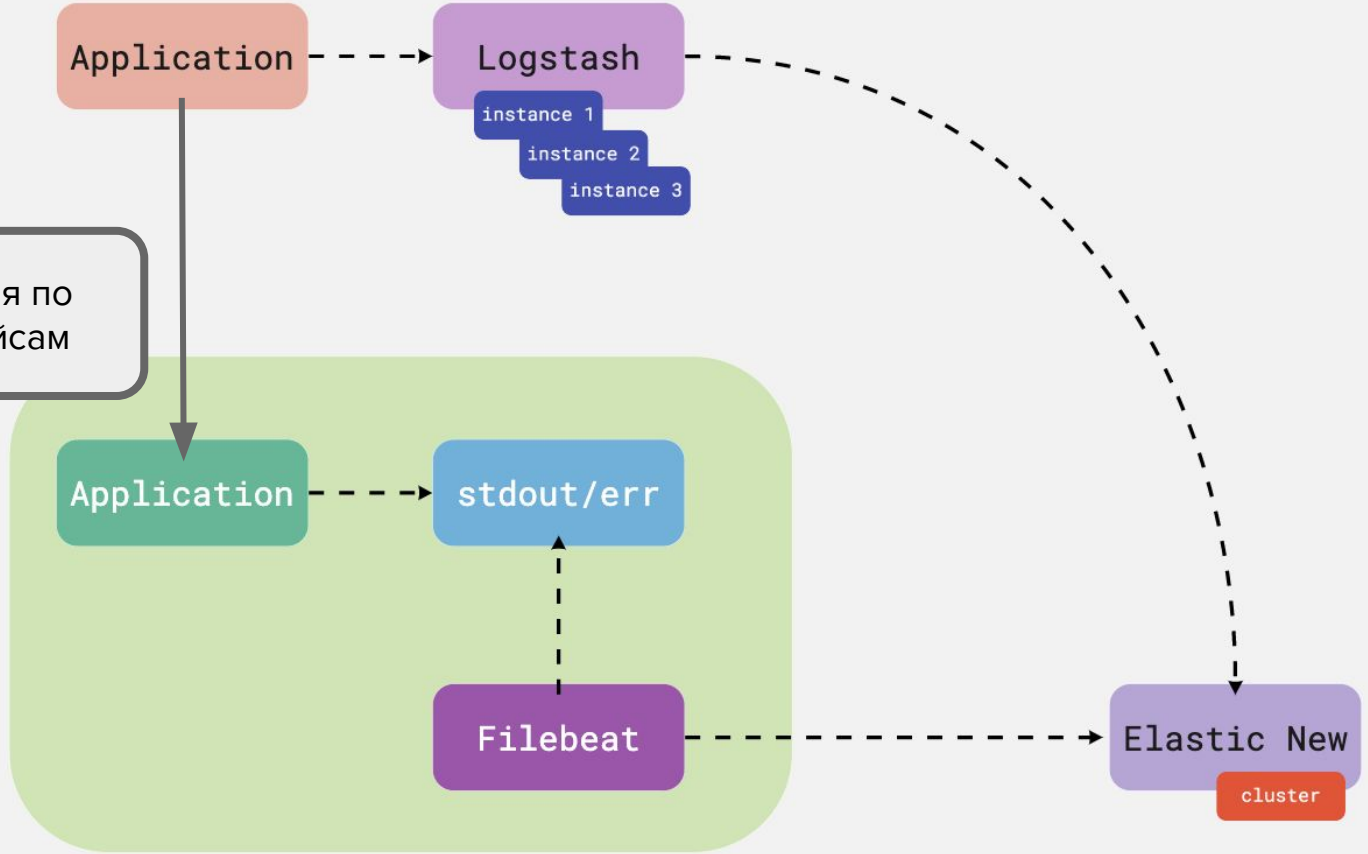
- Идентичный программный слой
  - тестируем инфраструктуру, но не софт
- Сбор телеметрии для наблюдаемости
  - логи, метрики
  - алерты на ошибки
- Требует индивидуального подхода
  - подходит для больших и опасных задач

Service-by-Service Infrastructure Testing

Вернемся к задаче о логах

Снижение рисков отказа в обслуживании

*Миграция по неймспейсам



Проблема
как не делать больно разработчикам?

Цель: минимизировать
негативное влияние на процесс разработки
внутри бизнесовых команд

Нам нужна абстракция
(в терминах DDD - anti-corruption layer)

Кейс: движок для деплоймента (AppYAML)

Принципы AppYAML

- Один сервис - один файл
- As small as possible (только необходимое)
- Устойчивый интерфейс для разработчиков

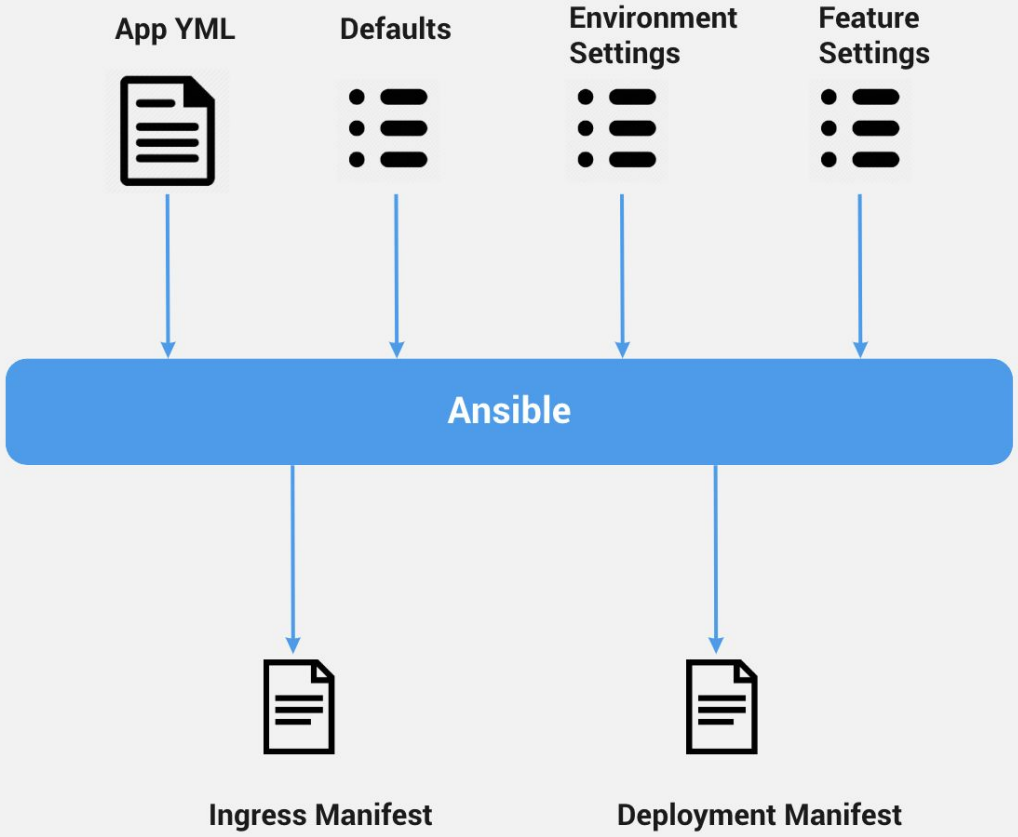
Минимальный манифест

```
application:
  app_name: hcn
  test:
    cpu: 0.5
    mem: 512.0
    instances: 2
  prod:
    cpu: 0.5
    mem: 512.0
    instances: 4
  rmq: True
  mongo: True
```

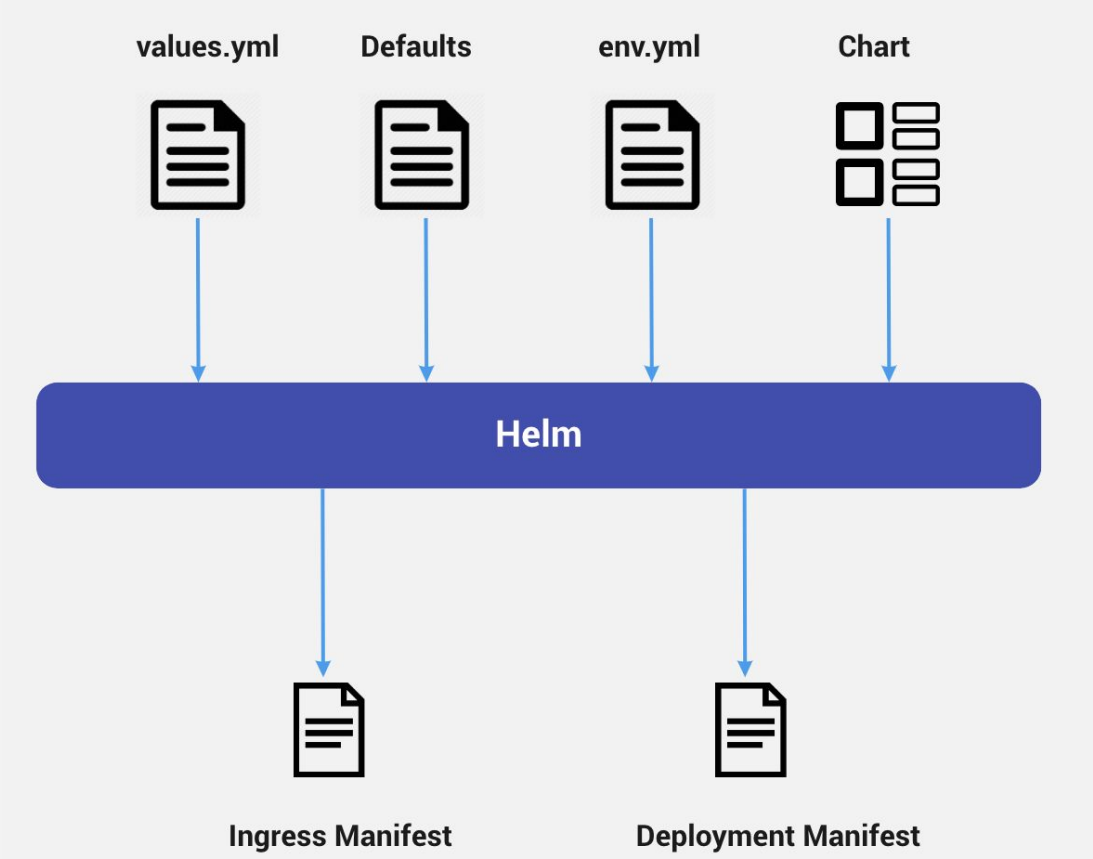
Минимальный манифест

```
application:
  app_name: hcn
  test:
    cpu: 0.5
    mem: 512.0
    instances: 2
  prod:
    cpu: 0.5
    mem: 512.0
    instances: 4
  rmq: True
  mongo: True
```

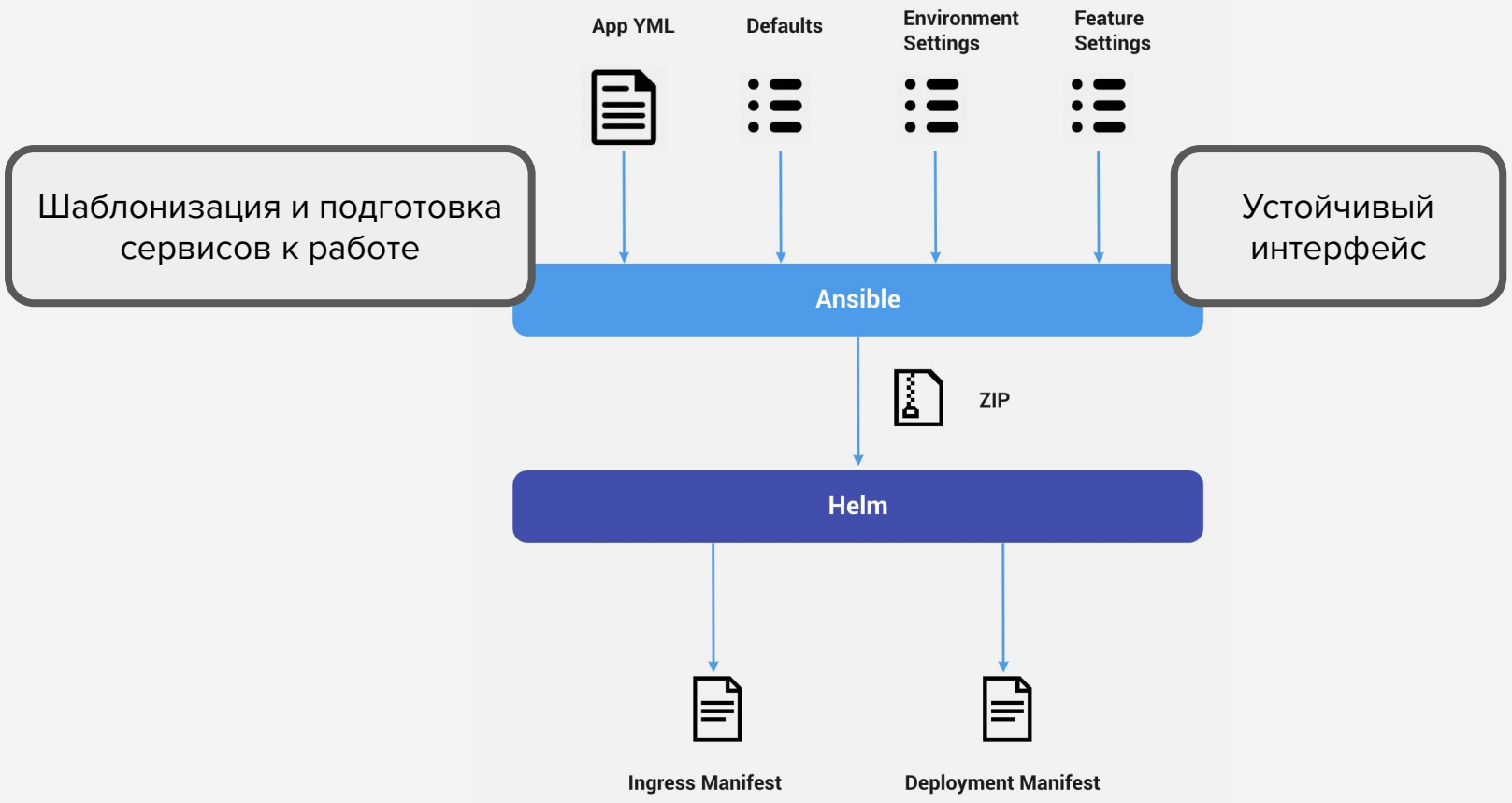
Генерация через AppYAML



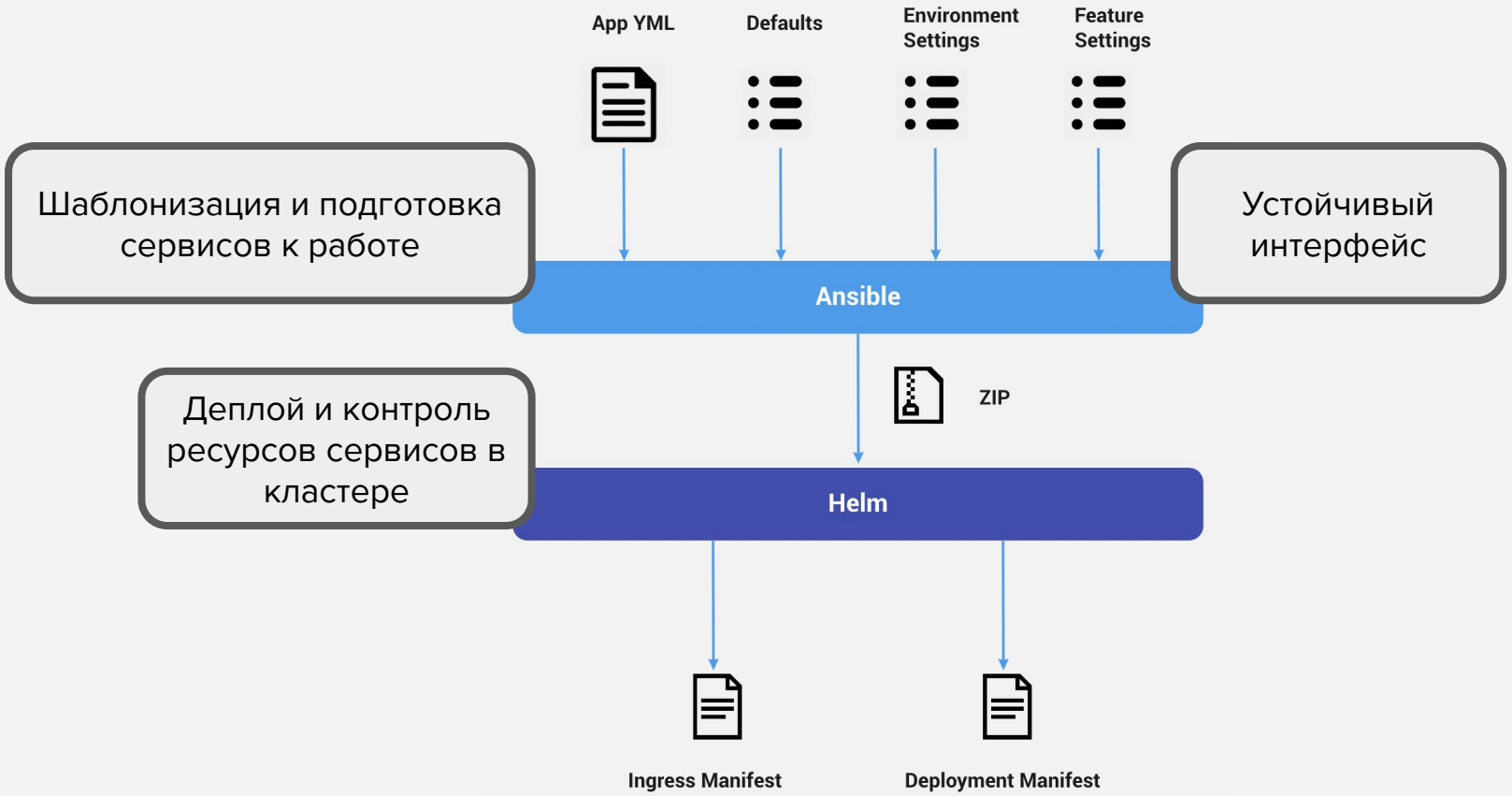
Тоже, но через Helm



Как мы встроили Helm



Как мы встроили Helm



Что помогает в смягчении боли?

- **AppYAML / Helm**
  - **описание сервисов**

Что помогает в смягчении боли?

- AppYAML / Helm
  - описание сервисов
- CLI / API / Админки
  - инструменты самообслуживания

Что помогает в смягчении боли?

- AppYAML / Helm
  - описание сервисов
- CLI / API / Админки
  - инструменты самообслуживания
- One-Page Documentation
  - базы / доступы / базовые команды

5. Тупиковая ветвь эволюции

“Простейшие” операции

- Добавление элемента
  - не было X -> стало X
- Замена элемента (почти всегда)
  - было X -> стало Y
- Модификация элемента (практически никогда)
  - было X -> стало X*
- **Удаление элемента**
  - **было X -> нет X**

Проблема

решением X ~~почти~~ никто не пользуется

Почему X становится ненужным

- Устарело / не отвечает потребностям
  - Logstash из задачи о логировании
  - Elastalert уступил место Alertmanager

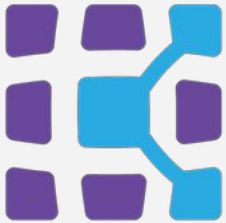
Почему X становится ненужным

- Устарело / не отвечает потребностям
  - Logstash из задачи о логировании
  - Elastalert уступил место Alertmanager
- Переоценка значимости
  - полная (X не важен совсем)
  - частичная (некоторая часть X не важна)

Кейс: e2e-тестирование

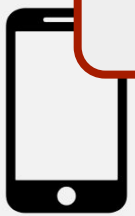
Попытка #1

Попытка #1

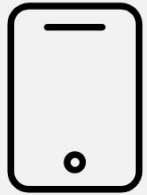


Kobiton

Flaky-девайсы



Flaky-тесты



appium

Сложность мокирования

Попытка #2

Облачный сервис с готовой инфраструктурой

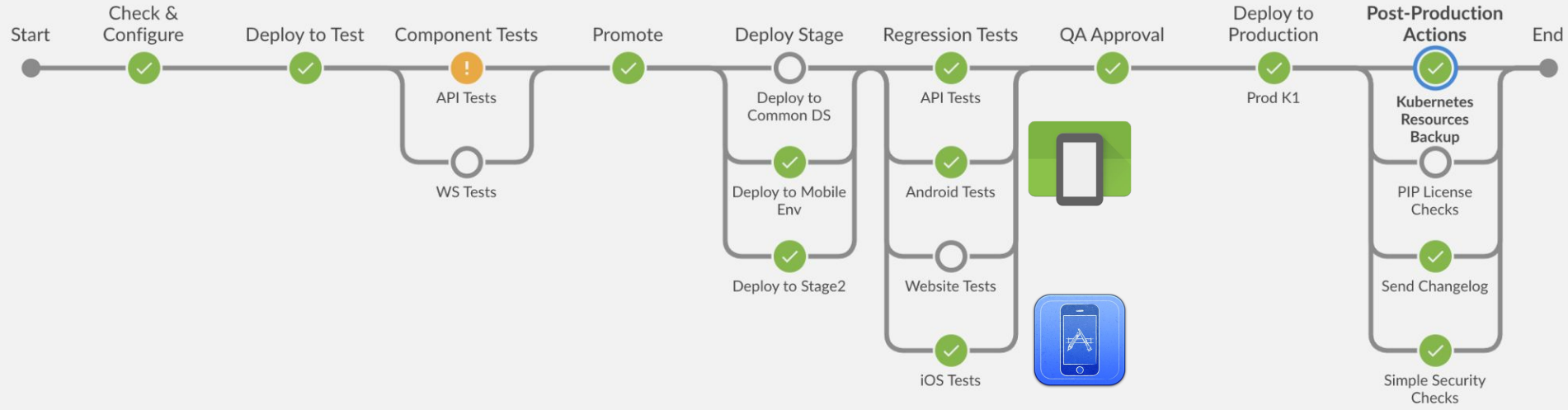
SAUCELABS

Эмулятор под каждую платформу

appium

Количество Flaky-тестов значительно снизилось

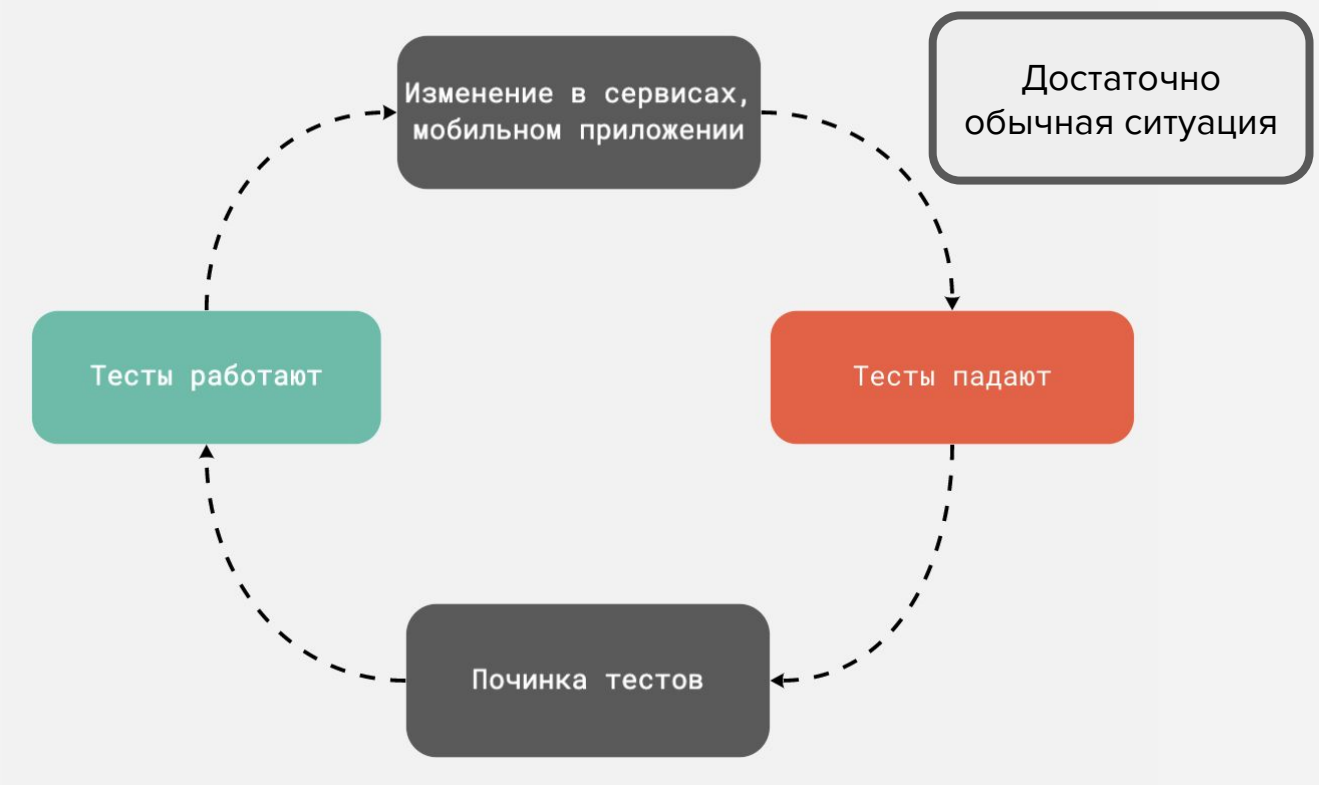
Прогон тестов - часть пайплайна



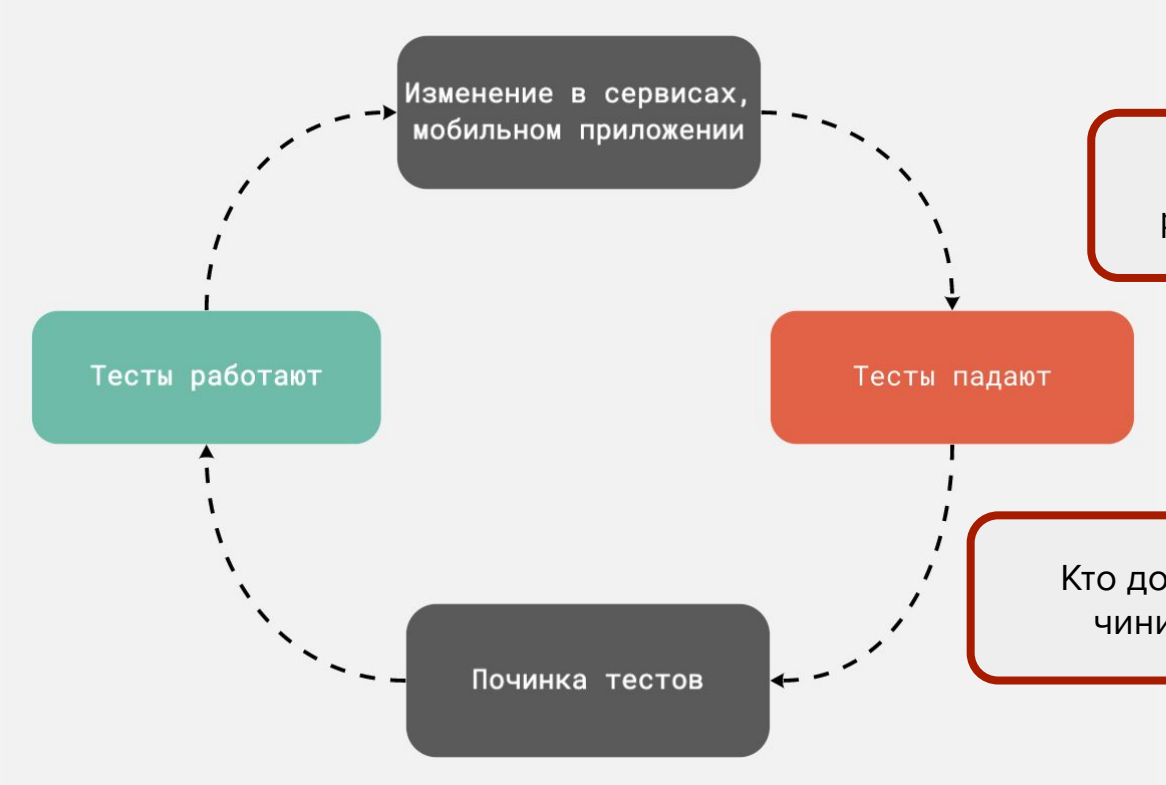
Похоже на success-story,
где тут тупик эволюции?

Проблема в том,
что у тестов не оказалось владельца

Красно-зеленый круговорот



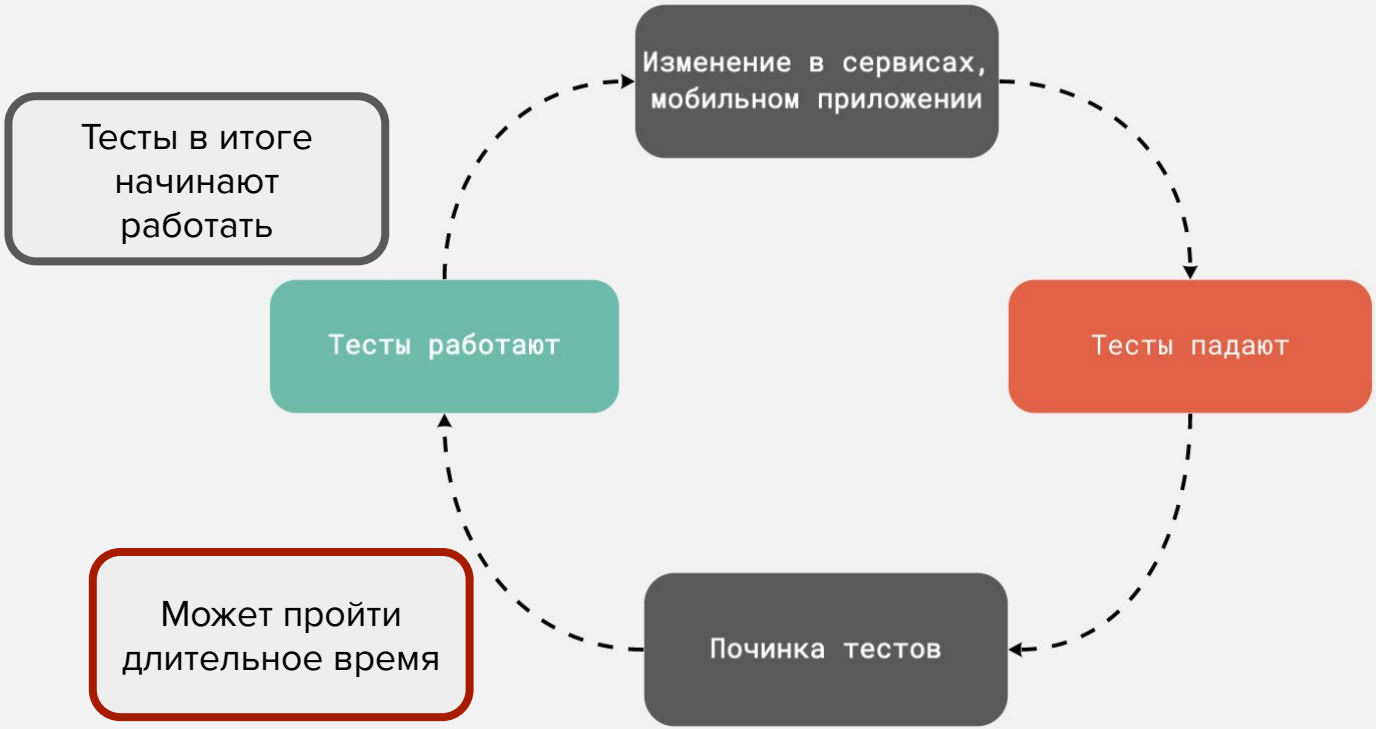
Красно-зеленый круговорот



“Нервные” разработчики

Кто должен чинить?

Красно-зеленый круговорот



Попытки спасти e2e-тесты

- **Запуск тестов без блокировки пайплайна**
  - **привел к тому, что нужность стала ещё меньше**
- **Обязательный запуск при релизе моб. приложения**
  - **не выдержал конкуренции с мокированными UI-тестами**

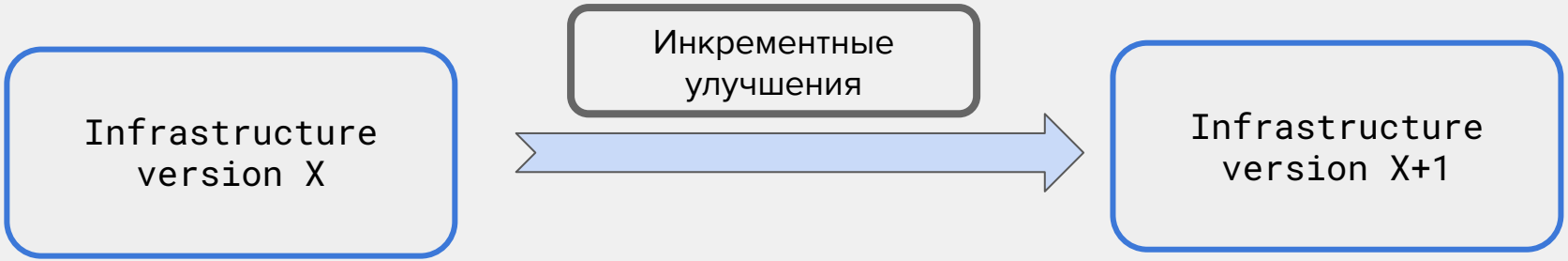
Цель: не тратить время и ресурсы на то, что не пользуется спросом

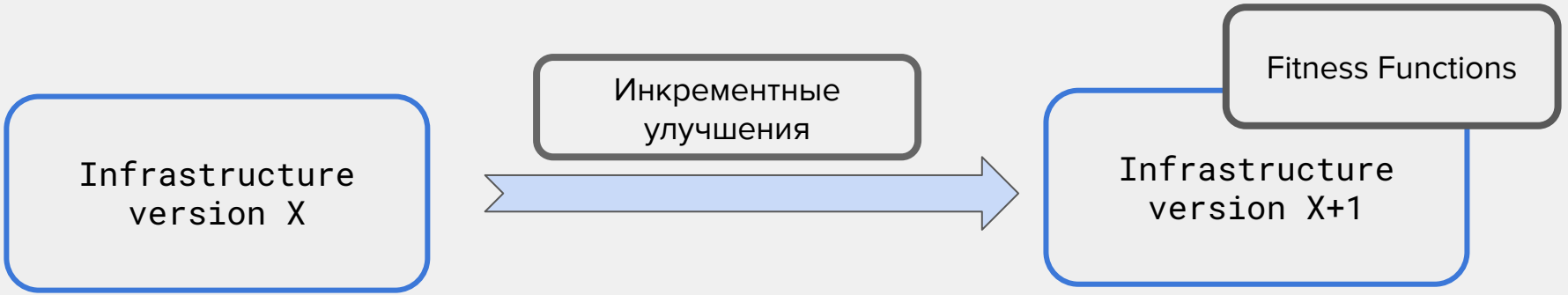
Положительные моменты

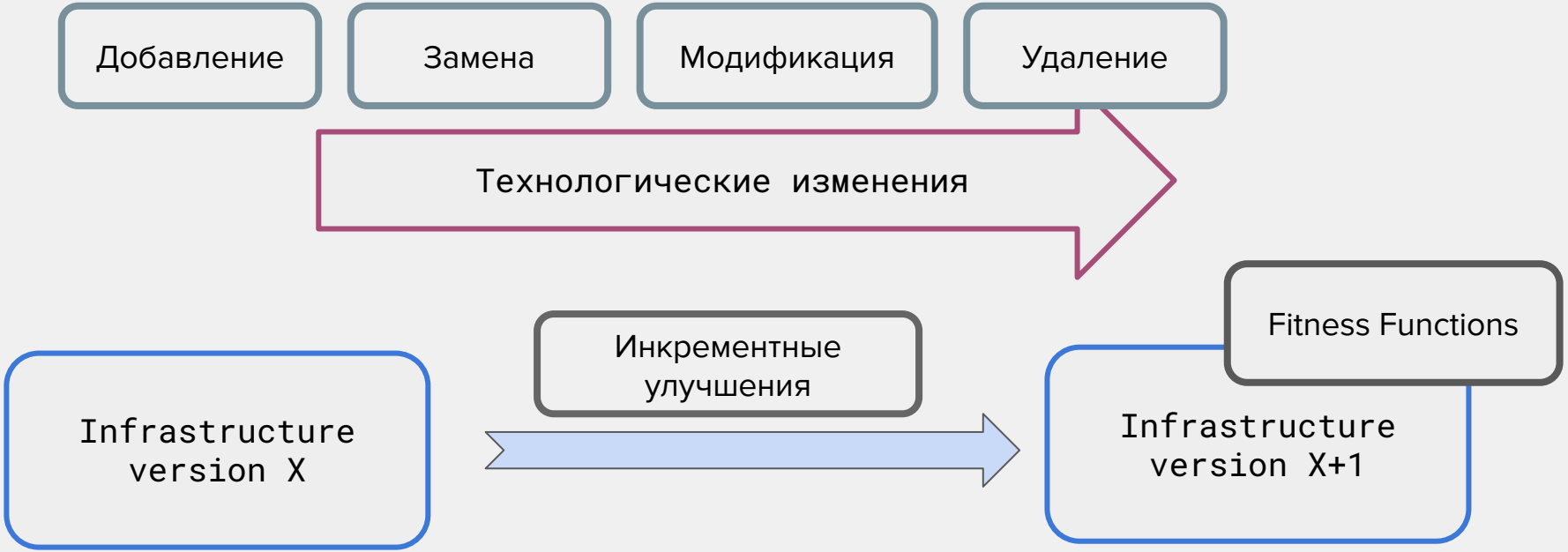
- Было найдено несколько критичных багов
- Улучшена работа механизма мокирования
  - переход на фейки
- “Три ~~х**ни~~ фигни”
  - планируем вернуться к задаче

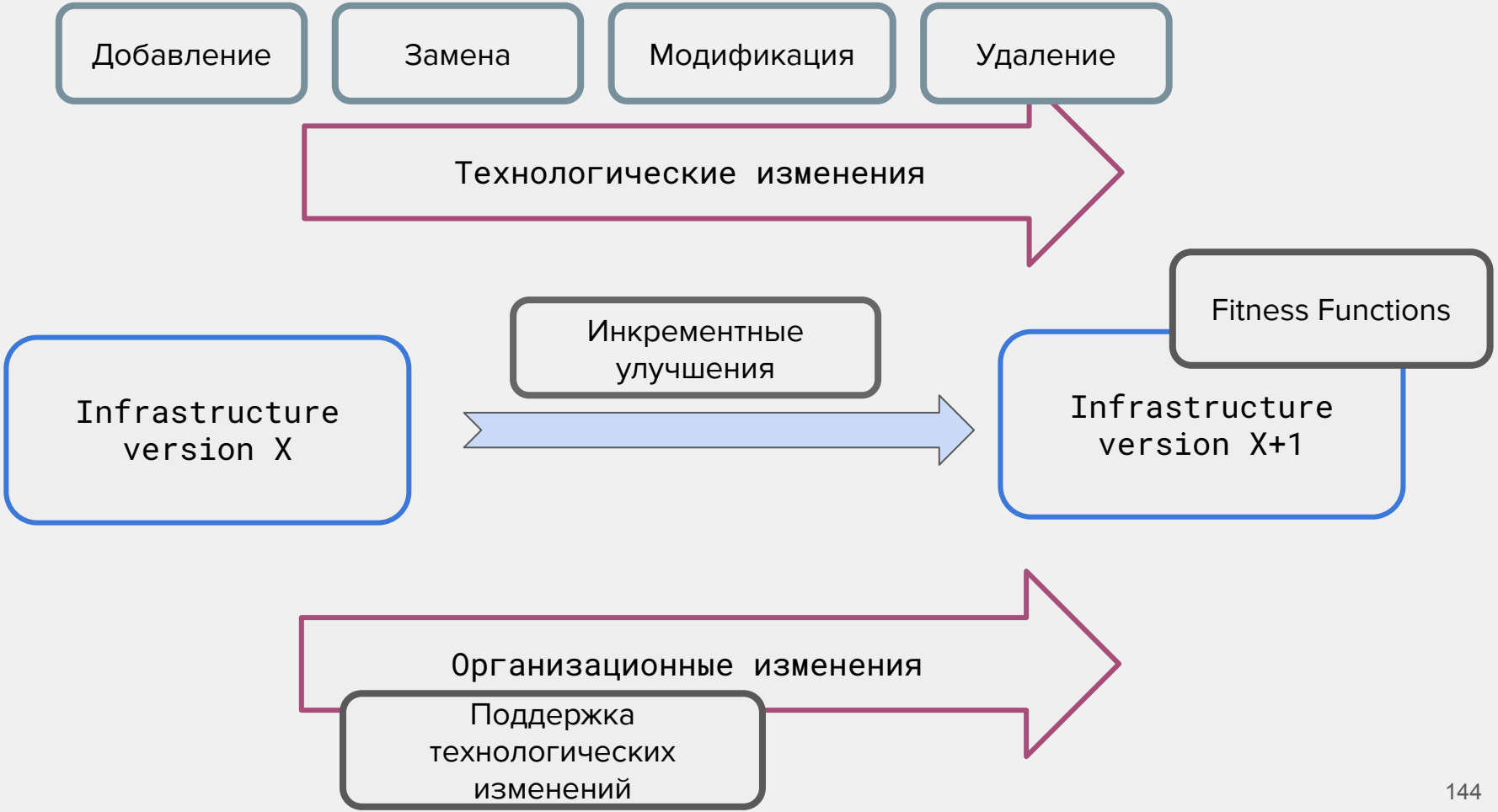
Эпилог: эволюционный фреймворк


Infrastructure
version X


Infrastructure
version X
Инкрементные
улучшения
Infrastructure
version X+1


Infrastructure
version X
Инкрементные
улучшения
Fitness Functions
Infrastructure
version X+1


Добавление
Замена
Модификация
Удаление
Технологические изменения
Fitness Functions
Инкрементные улучшения
Infrastructure version X
Infrastructure version X+1


Добавление
Замена
Модификация
Удаление
Технологические изменения
Fitness Functions
Инкрементные улучшения
Infrastructure version X
Infrastructure version X+1
Организационные изменения
Поддержка технологических изменений

Дьявол в деталях!

Ваши вопросы?

ANNA

@aatarasoff

@aatarasoff

@aatarasoff

@aatarasoff