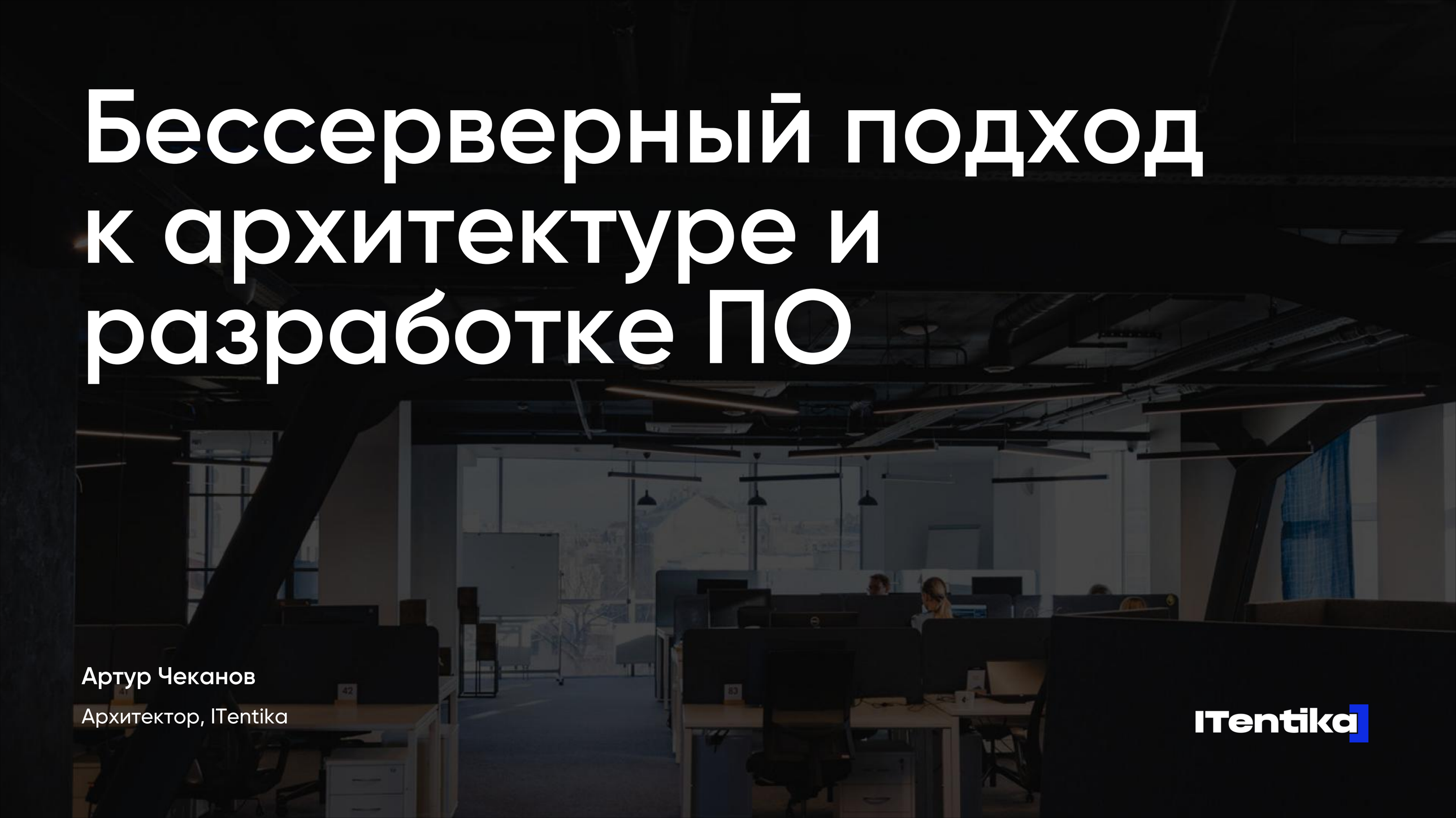
Бессерверный подход к архитектуре и разработке ПО



Артур Чеканов

Архитектор, ITentika

ITentika

Пара слов обо мне

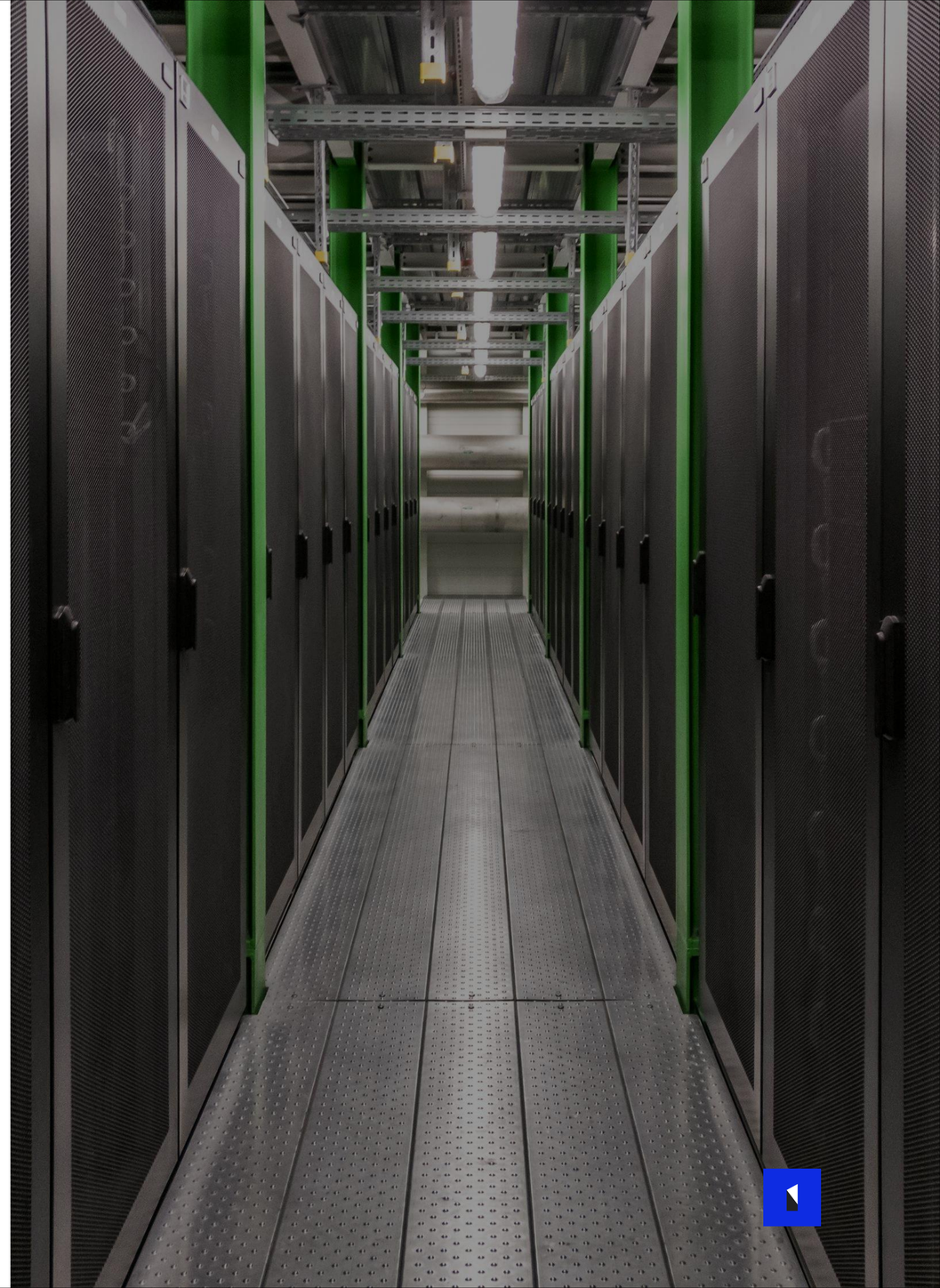
- 14+ лет в IT
- Все еще не надоело
- Python в сердечке
- Сфера интересов: Solution Architecture, High Load, Data
- Главный архитектор ITentika

О чем поговорим

- Общая вводная про бессерверный подход
- Мой опыт
- Пример написания приложения в Yandex Cloud
- Выводы

Что такое Serverless

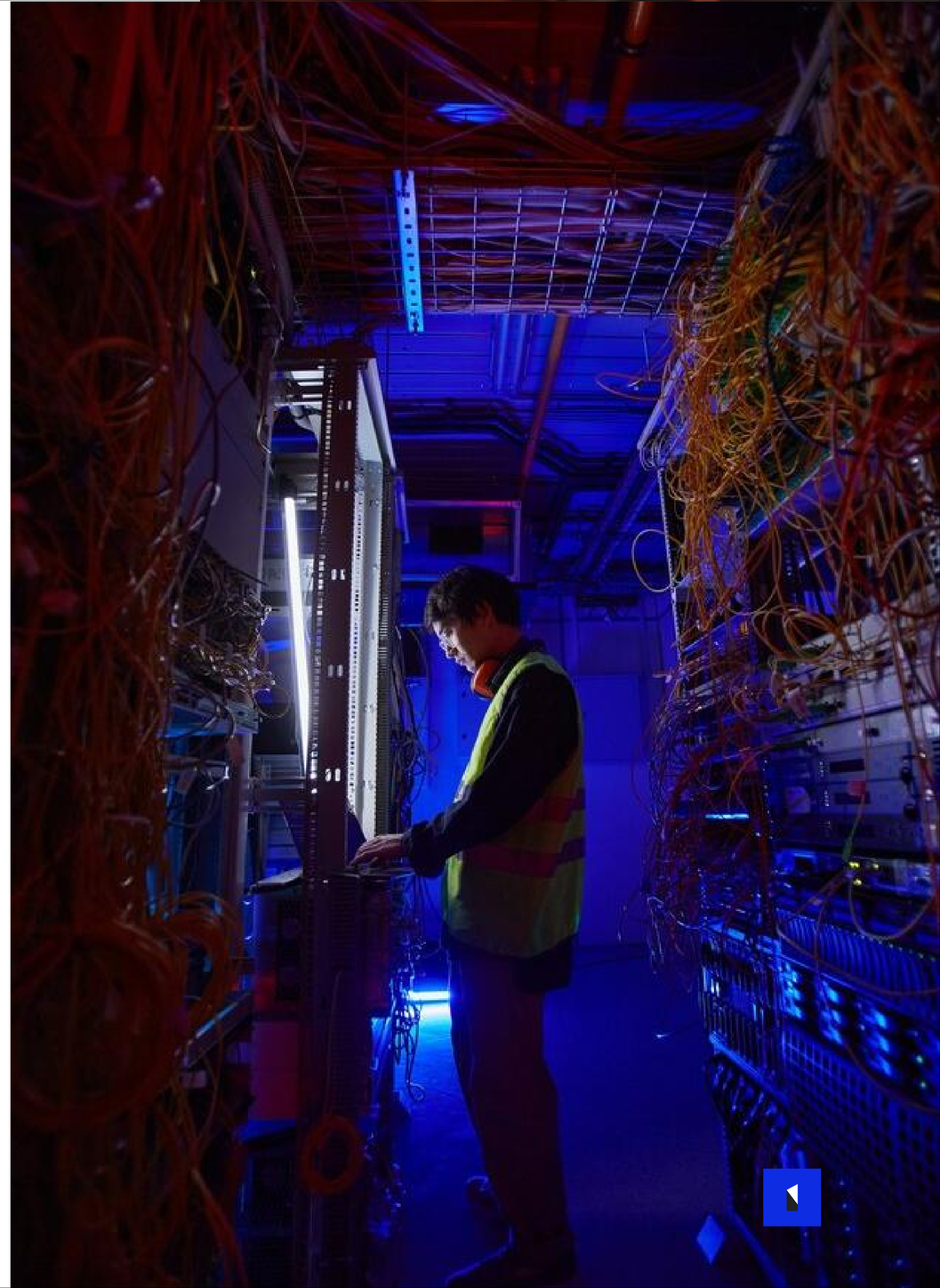
- Server-less, нет серверов нет проблем
- Автоматическое управление ресурсами
- Автоматический maintenance
- Динамический скейлинг, в пределах квот
- FaaS
- Микросервисная архитектура

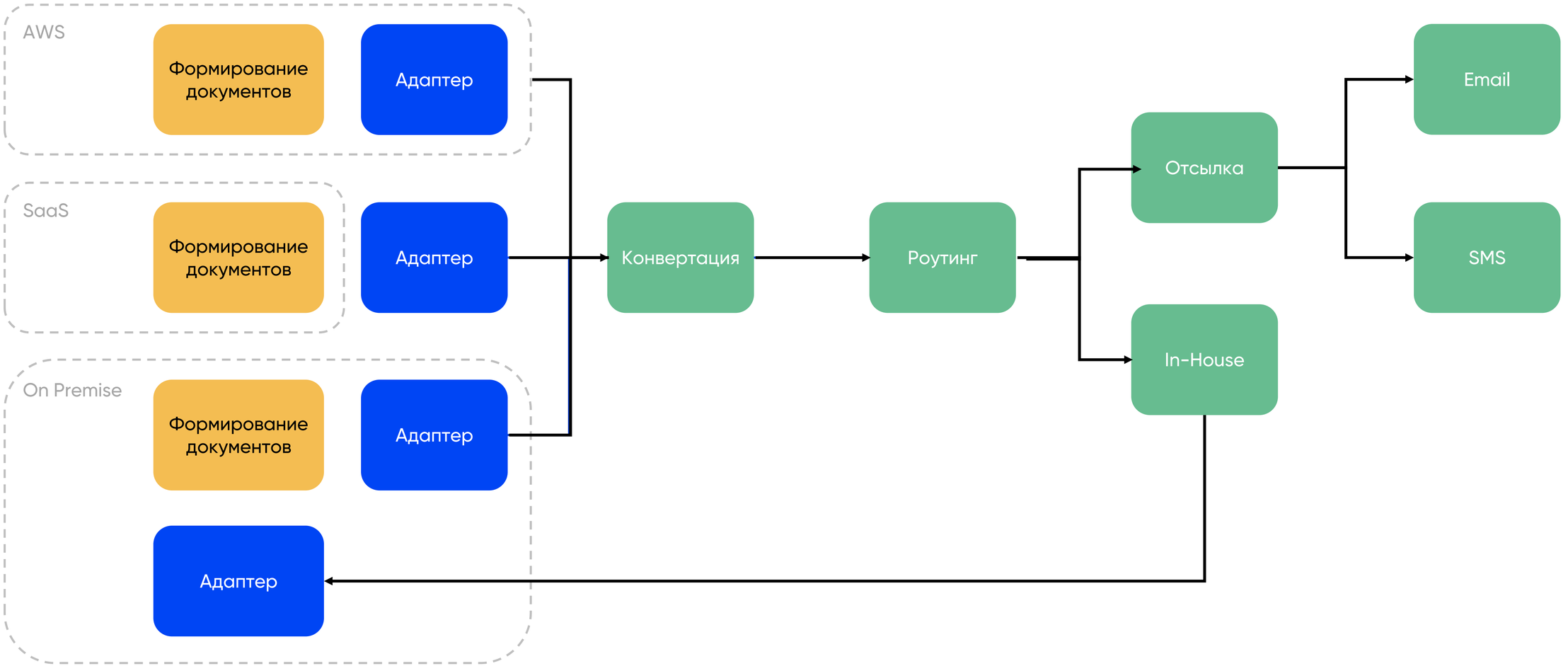


Как мы однажды до serveless докатились

История одного продукта

- Централизация отсылки, шаринга и обмен документов
- Разработка с нуля
- Инфраструктура AWS
- от PoC до продакшена
- ~50 сервисов
- Тех стек:
  - Python
  - C++
  - JS

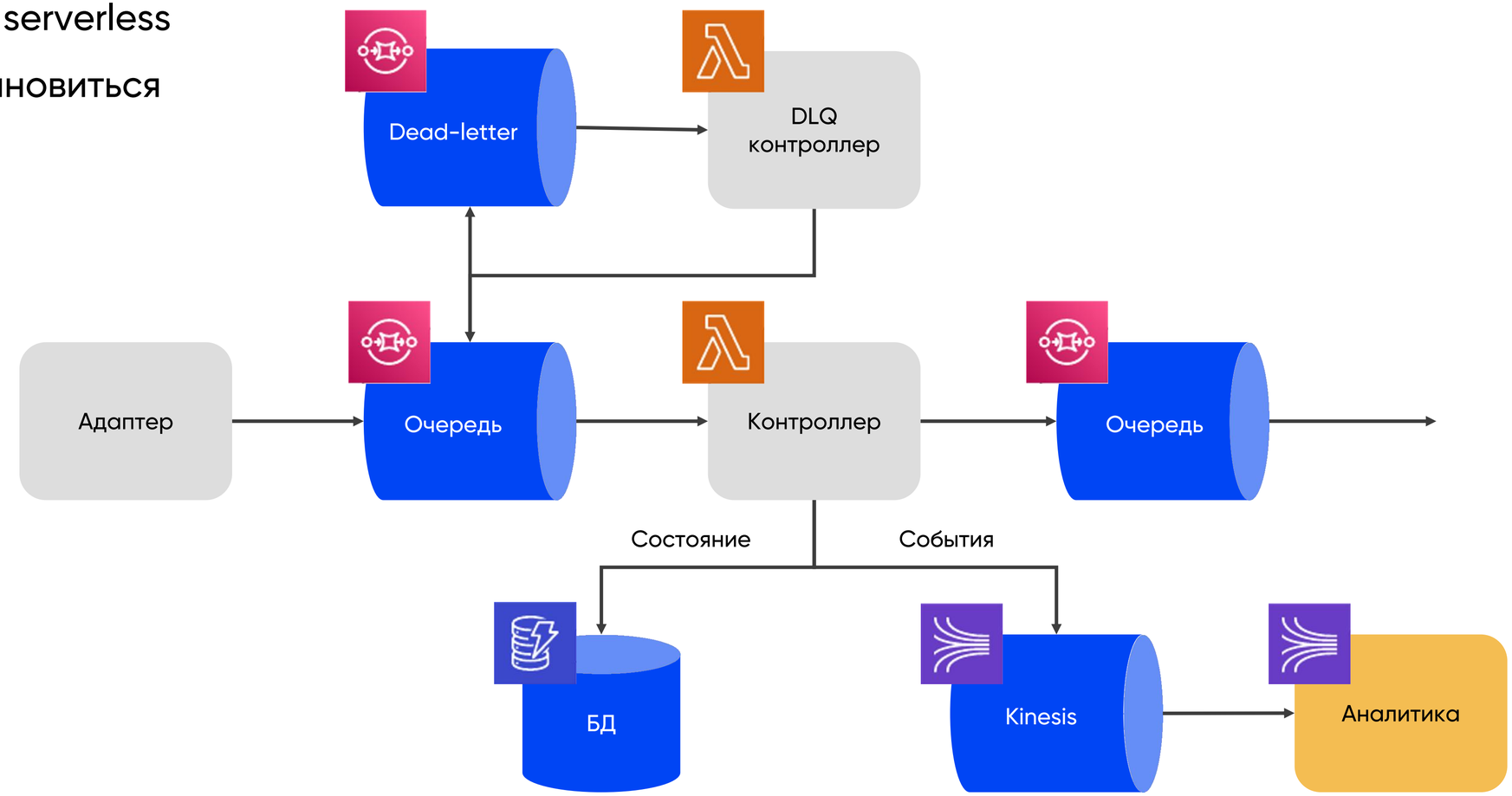




Helicopter view

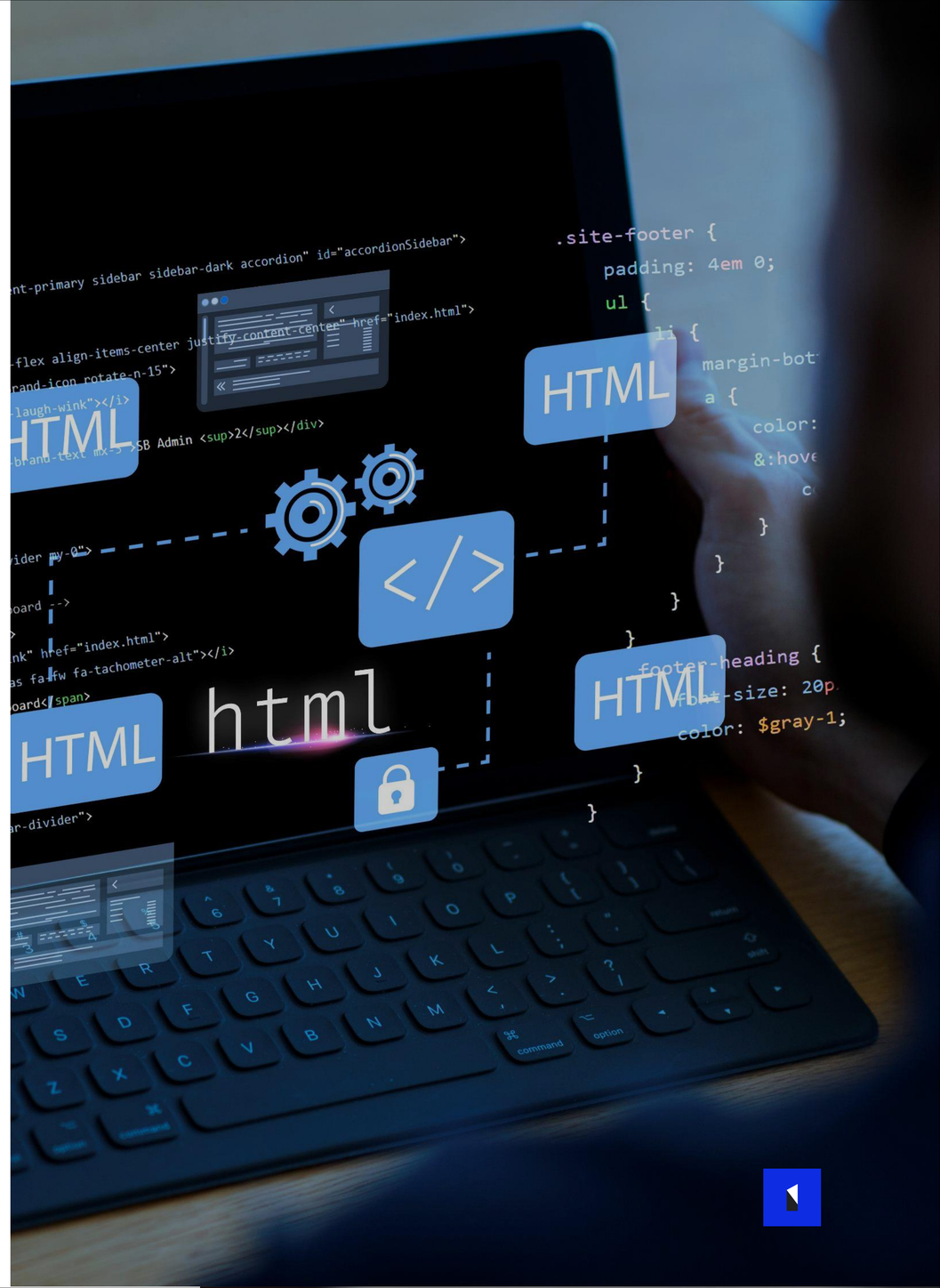
Типичный компонент

- Примерно это и считается serverless
- На этом планировали остановиться
- Но не остановились



REST API на serverless

- В процессе развития продукта появилась необходимость иметь REST API
- Отказались от EC2
- Отказались от EKS на AWS Fargate
- Взяли опять лямбды

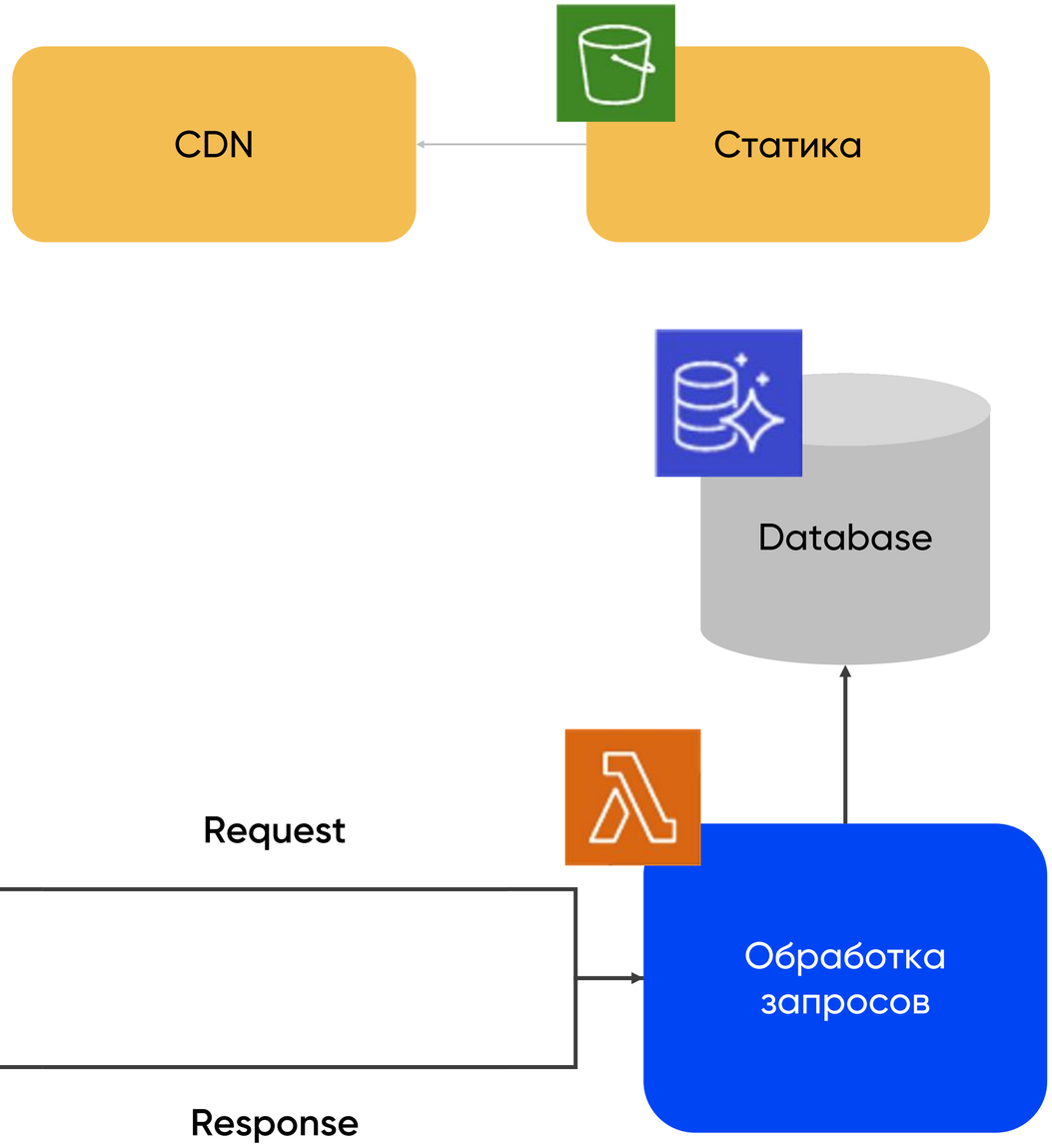


REST API на serverless

- API Gateway принимает запросы
- Лямбда их обрадатывает
- Принимает dict, отдает dict
- Одна лямбда может обрабатывать несколько путей

```python
def handler(event, context):
    return {
        'statusCode': 200,
        'body': 'Hello World!',
    }

```



CI/CD

- **Serverless – фреймворк для деплоя**
  - Слишком мало контроля над происходящим
  - CloudFormation в целом неудобен
  - Инфрастуктура и деплой вроде как смешаны вместе
  - Брали как временное решение
  - Дошел до продакшена

CI/CD

- **Serverless – фреймворк для деплоя**
  - Слишком мало контроля над происходящим
  - CloudFormation в целом неудобен
  - Инфрастуктура и деплой вроде как смешаны вместе
  - Брали как временное решение
  - Дошел до продакшена
- **AWS Chalice**
  - Хороший веб фреймворк для лямбд
  - Может делать деплоймент, но тут те же нюансы что и у serverless

CI/CD

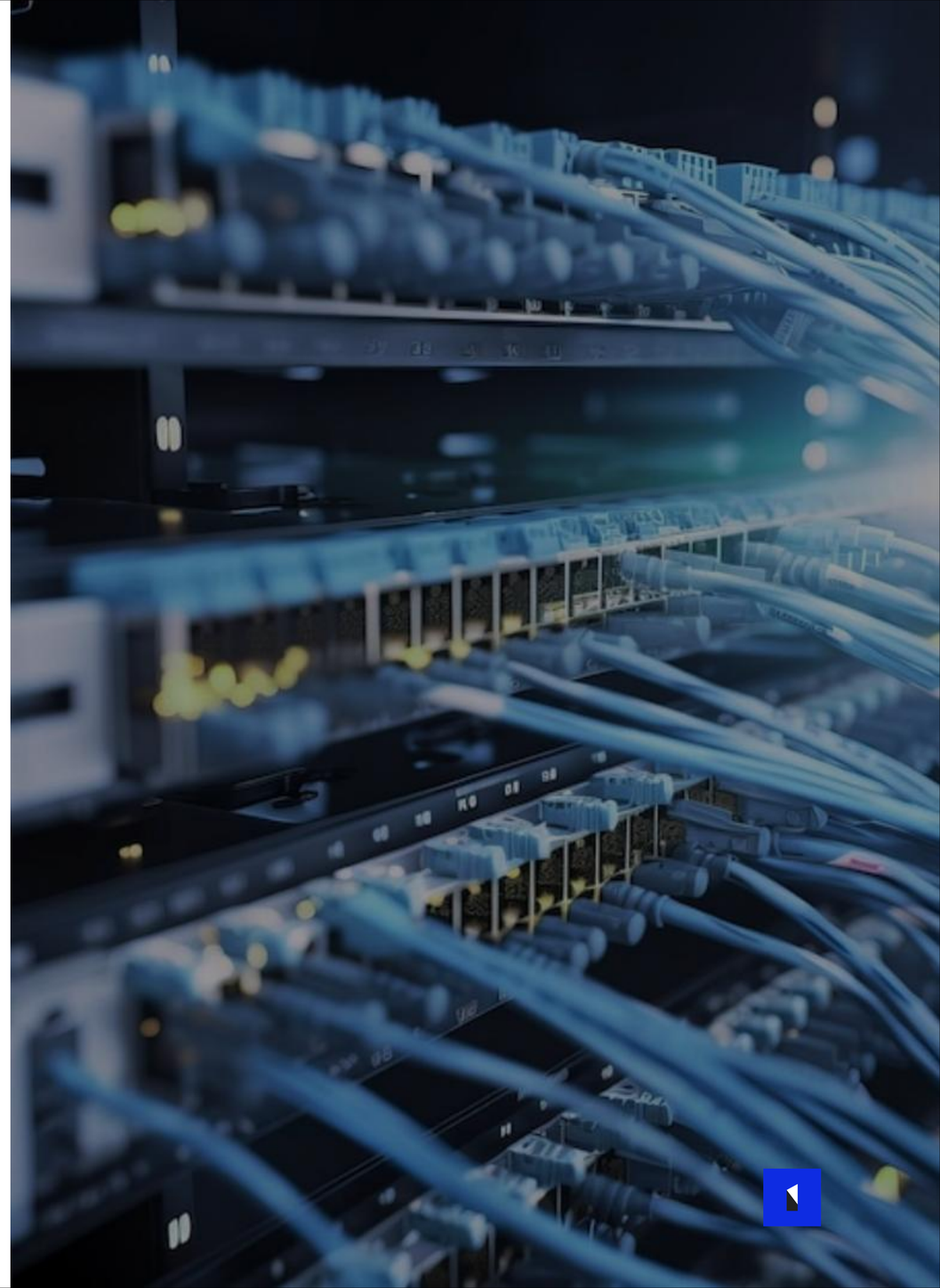
- **Serverless – фреймворк для деплоя**
  - Слишком мало контроля над происходящим
  - CloudFormation в целом неудобен
  - Инфрастуктура и деплой вроде как смешаны вместе
  - Брали как временное решение
  - Дошел до продакшена
- **AWS Chalice**
  - Хороший веб фреймворк для лямбд
  - Может делать деплоймент, но тут те же нюансы что и у serverless
- **AWS Стек: AWS CDK + AWS Code Build/Pipeline**
  - CDK в целом понравился, но его надо учить
  - Code Build в целом хорош для мелких проектов, или в ситуации когда DevOps отдел не хочет стандартизированный подход к пайплайнам

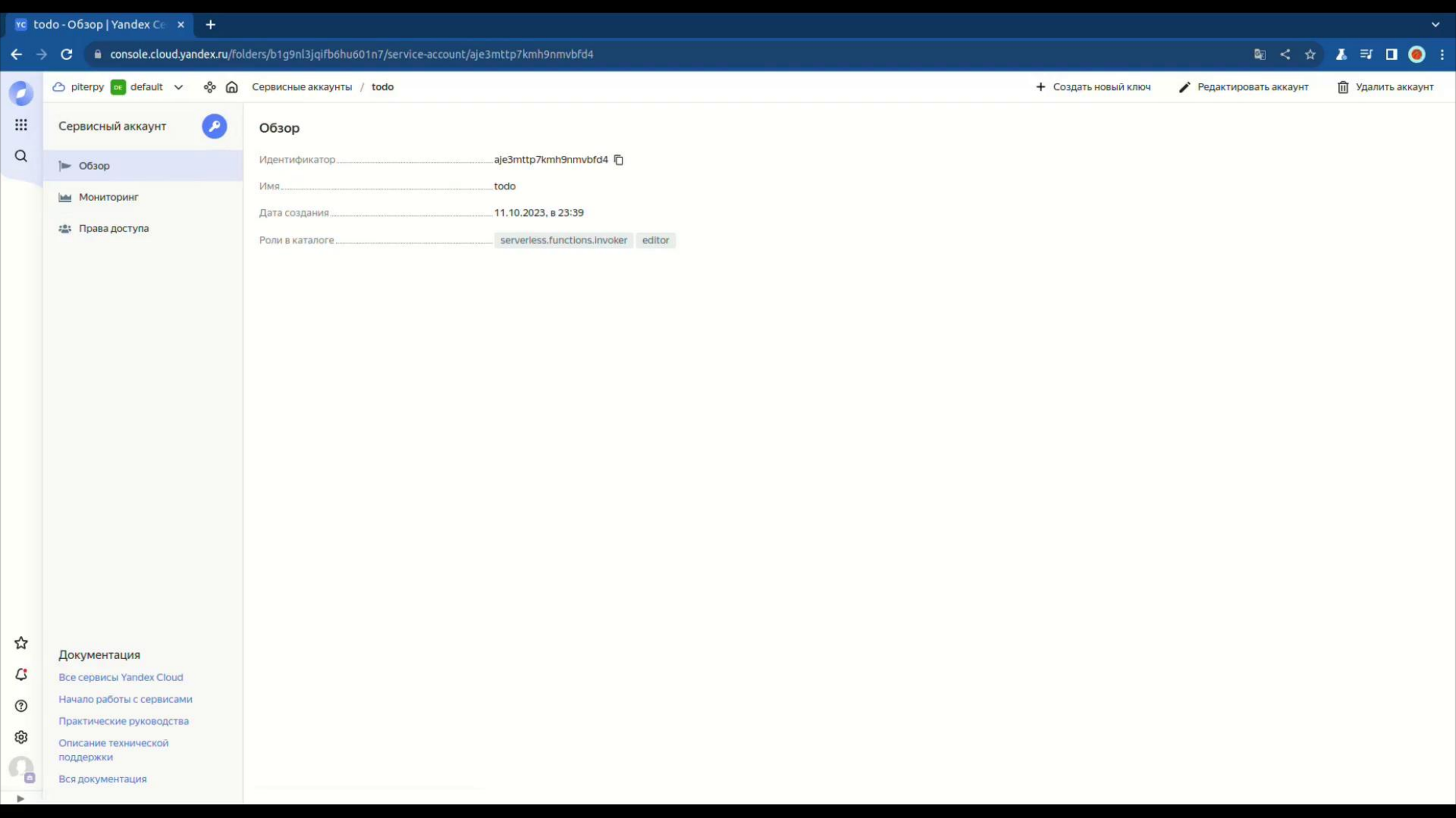
CI/CD

- **Serverless – фреймворк для деплоя**
  - Слишком мало контроля над происходящим
  - CloudFormation в целом неудобен
  - Инфрастуктура и деплой вроде как смешаны вместе
  - Брали как временное решение
  - Дошел до продакшена
- **AWS Chalice**
  - Хороший веб фреймворк для лямбд
  - Может делать деплоймент, но тут те же нюансы что и у serverless
- **AWS Стек: AWS CDK + AWS Code Build/Pipeline**
  - CDK в целом понравился, но его надо учить
  - Code Build в целом хорош для мелких проектов, или в ситуации когда DevOps отдел не хочет стандартизированный подход к пайплайнам
- **Кастом**
  - Terraform для всего инфрастуктурного, самописные модули для стандартизации и DRY
  - Jenkins, Groovy опять же помогает в DRY

Что предоставляют провайдеры в РФ

- Yandex Cloud
  - API Gateway
  - Cloud Functions
  - Message Queue
  - Object Storage
  - YDB
  - Serverless Containers




todo - Обзор | Yandex Ce
console.cloud.yandex.ru/folders/b1g9nl3jqifb6hu601n7/service-account/aje3mttp7kmh9nmvbfd4
piterpy
default
Сервисные аккаунты / todo
Создать новый ключ
Редактировать аккаунт
Удалить аккаунт
Сервисный аккаунт
Обзор
Мониторинг
Права доступа
Обзор
Идентификатор aje3mttp7kmh9nmvbfd4
Имя todo
Дата создания 11.10.2023, в 23:39
Роли в каталоге serverless.functions.invoker editor
Документация
Все сервисы Yandex Cloud
Начало работы с сервисами
Практические руководства
Описание технической поддержки
Вся документация

Что мы получили (и нам понравилось)

- Упростилась поддержка инфраструктуры
- Логирование осуществлялось самим AWS
- Все плюшки микросервисной архитектуры
  - Независимый релизный цикл компонент
  - Быстрый rollback
  - Разный тех стек
- Оптимизация расходов
  - Не нужно тушить env на выходные

Что мы получили (и нам не понравилось)

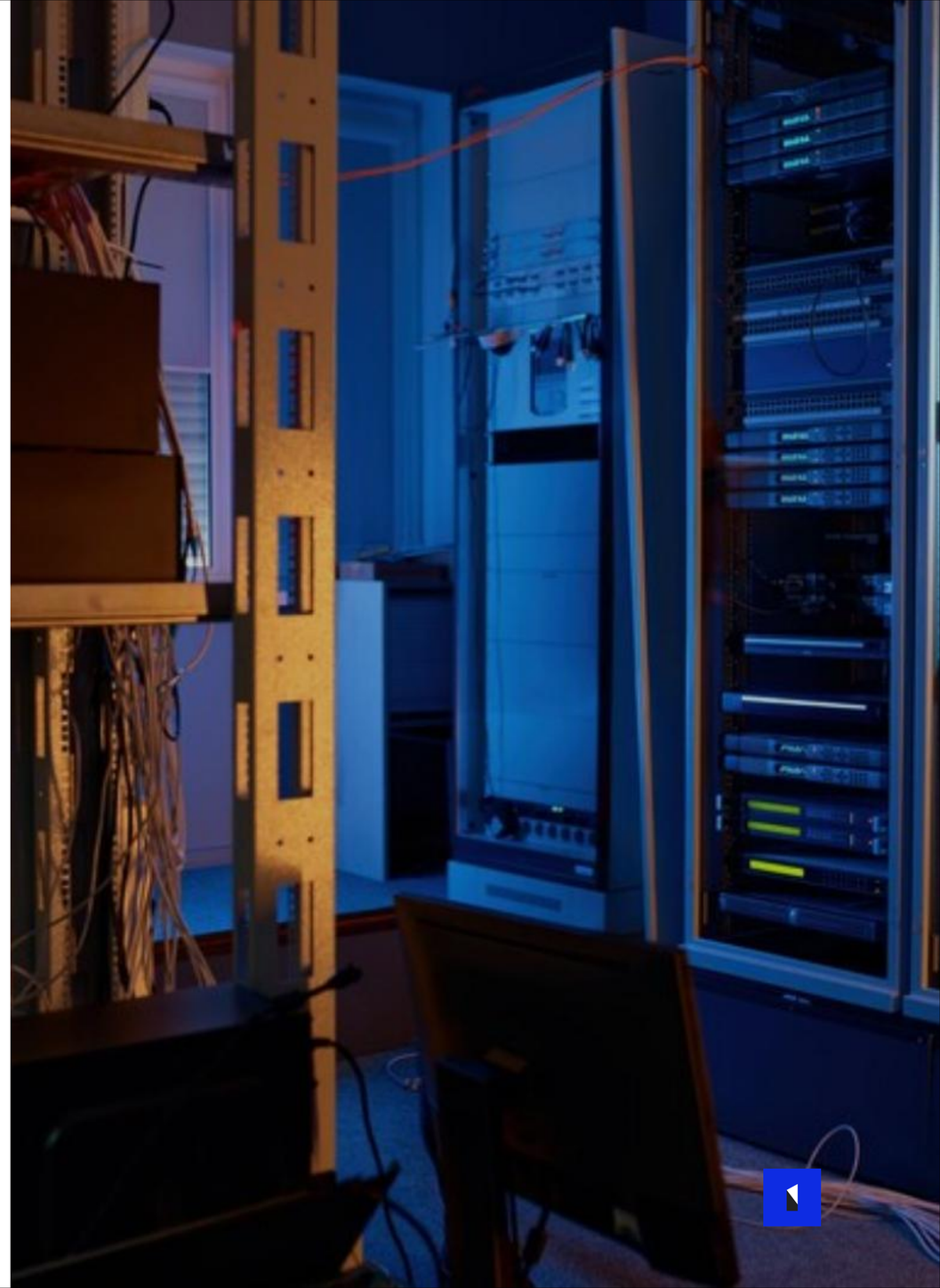
- Все плюшки микросервисной архитектуры
  - Релизный цикл все же имеет свои зависимости
  - Труднее для понимания в отличии от монолита
  - Разный тех стек
  - Проблемы взаимодействия между под командами
- Редактирование кода прямо в браузере

О чем я не рассказал

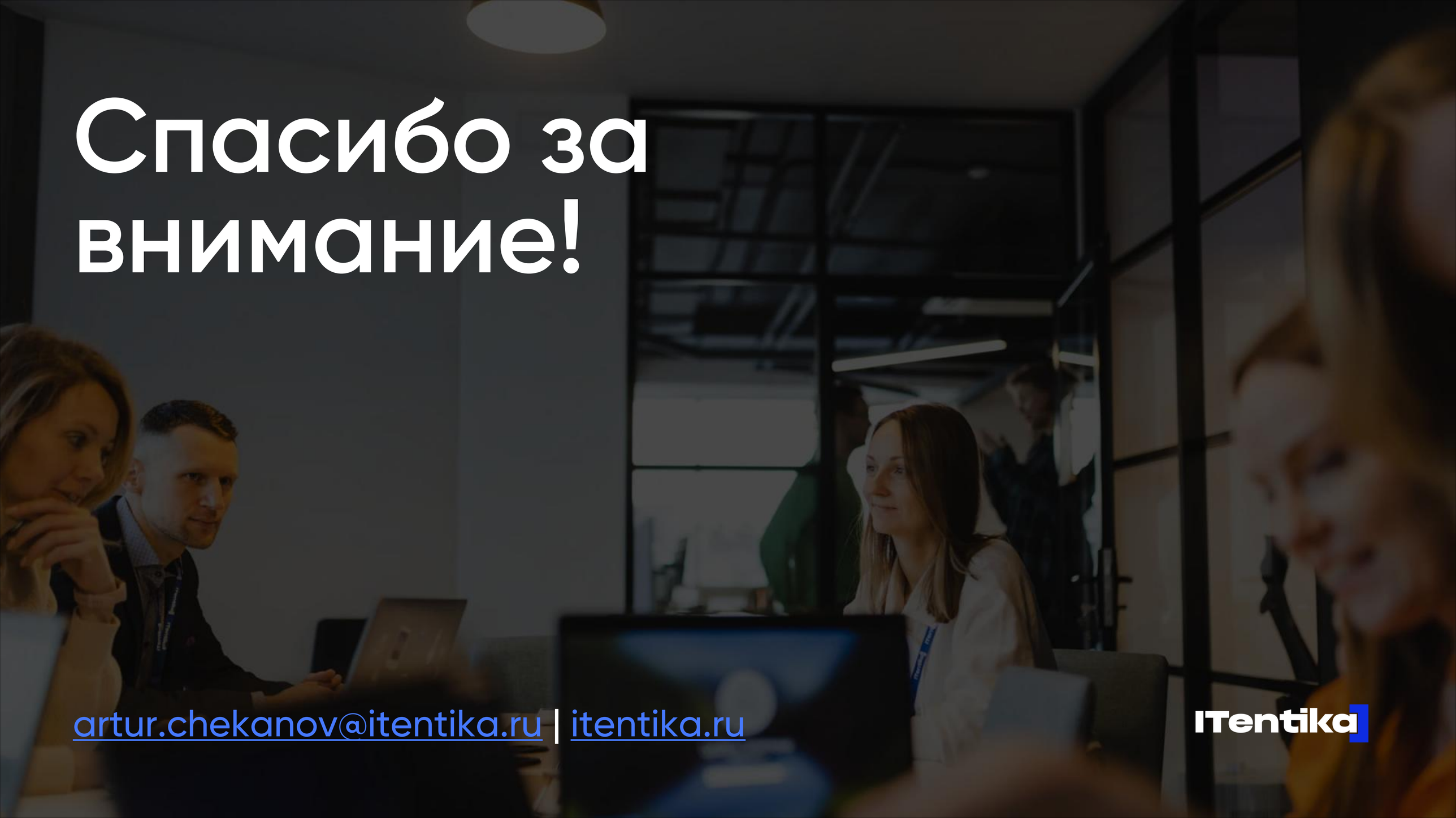
- Сложности с локальным окружением
  - Эмуляция AWS сервисов
  - Использование дев окружения
  - Окружение per-develper
- Квоты
- Как мы боролись с логированием
- Как мы однажды зациклили лямбду на бесконечный повтор DLQ
- Как мы устанавливали Ops процессы
- Как мы делали стартап на serveless (и на этот раз посчитали расходы)

Для кого serverless

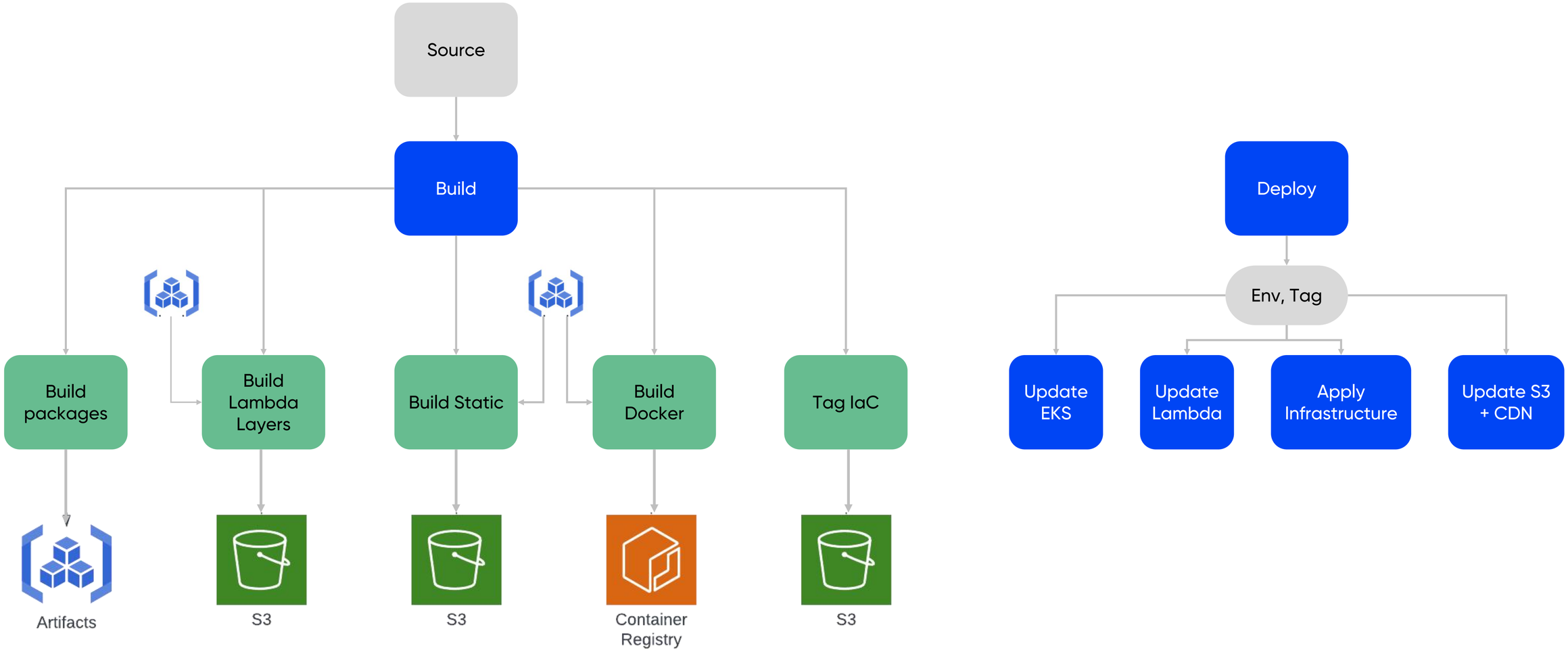
- Для сильно неоднородного характера нагрузки на сервис
  - Внутренние сервисы компаний
- Для расширения технического кругозора
  - Новые решения и технологии появляются постоянно
  - Возможность их вживую пощупать
- Для CV-driven разработки
  - Это модно = много баззвордов в резюме
- Для стартапов
  - Когда каждая копейка на счету



Спасибо за внимание!



artur.chekanov@itentika.ru | itentika.ru


Source
Build
Build packages
Build Lambda Layers
Build Static
Build Docker
Tag IaC
Artifacts
S3
S3
Container Registry
S3
Deploy
Env, Tag
Update EKS
Update Lambda
Apply Infrastructure
Update S3 + CDN

CI/CD