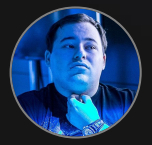
DEVOOPS 2026



СЕРГЕЙ БУХАРОВ
АРХИТЕКТОР ОБЛАЧНОЙ ИНФРАСТРУКТУРНОЙ ПЛАТФОРМЫ, DODO ENGINEERING

1,5 квартала | 8 инструментов | 1 форк

OpenTelemetry: через грабли к... чему?

Масштаб, на котором ломаться не должно

2 облака

4 кластера Kubernetes

100+ нод

100+ сервисов

Мониторингу всё равно, сколько у вас нод. Ему важен объём данных

~10 тыс.

SCRAPE ТАРГЕТОВ

~20 млн.

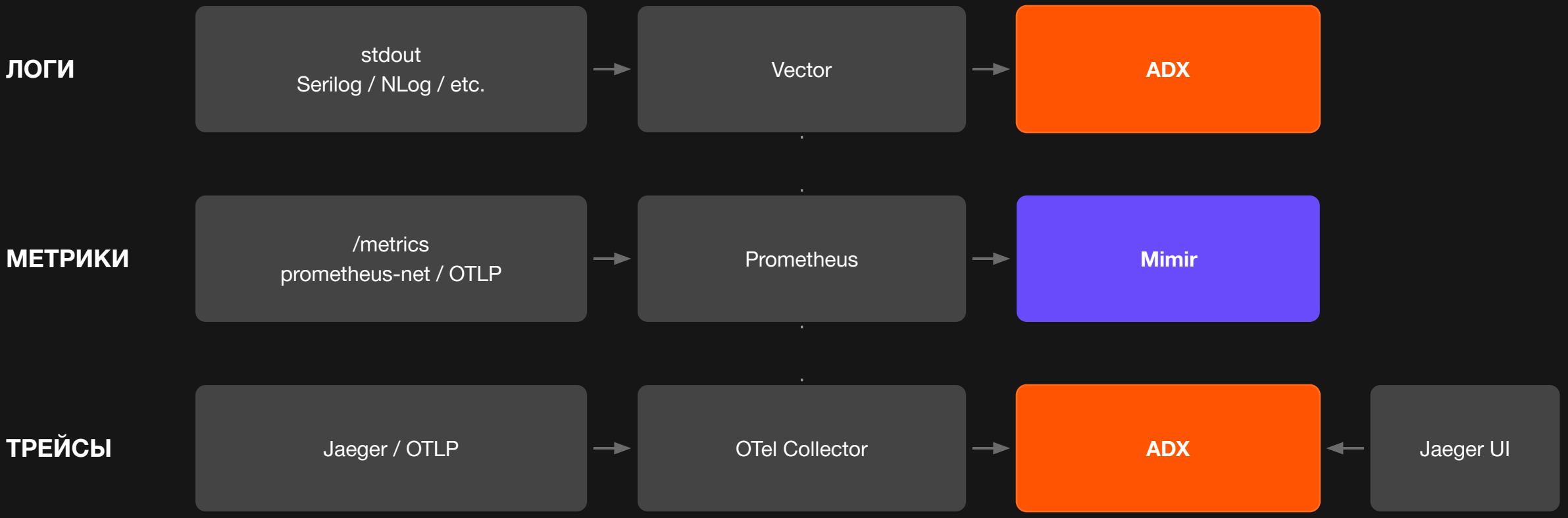
АКТИВНЫХ СЕРИЙ

~1 млн.

ЛОГОВ В МИНУТУ

Это не космический масштаб.

Три потока. Три технологии. Ни одной связи



ADX ≈ ваш ClickHouse / Splunk / Elasticsearch.

Пять причин

1 Vendor-lock

ADX не существует за пределами Azure.

Вся телеметрия — в одном облаке одного провайдера.

2 Нет связности

Из метрики не провалиться в трейс.

Из трейса — не провалиться в лог.

Три потока живут отдельными жизнями.

3 Нет стандарта

Каждый пишет как умеет. Любимый пример — код страны:

```
"country": "RU"
"country": "Russia"
"country": 643
```

Поле есть везде. Отфильтровать нельзя нигде.

4 Vector

Наследство с отдельным сервисом на Rust для healtcheck'ов.

Bus-фактор, на который не дышали.

5 «Архитектура попросила»

Подготовить инфраструктуру под OTel.

Что делаем

ЦЕЛЬ 1

Разорвать зависимость от ADX

Не выкинуть, а перестать стоять на нём двумя ногами.
Grafana Stack разворачивается где угодно — хоть на Raspberry.

ЦЕЛЬ 2

Получить связность трёх сигналов

Здесь в кадре появляется OpenTelemetry. Как обещание,
что связность приедет из коробки вместе со стандартом.

1 **Grafana Stack + Prometheus Operator**

Поднимаем рядом, заодно закрываем техдолг.

2 **Grafana Alloy в двух ролях**

Сборщик логов вместо Vector и OTel-коллектор в одном.
Сервисы переводим на OTLP.

3 **ADX остаётся с нами**

В роли долгосрочного хранилища.
Grafana Stack — оперативный контур.

На бумаге план скучный.

Статистика уже намекает

46%
ДОРАБАТЫВАЮТ КОЛЛЕКТОР ИЛИ ЕГО КОМПОНЕНТЫ

63%
МУЧАЮТСЯ С УПРАВЛЕНИЕМ КОНФИГУРАЦИЕЙ

52%
СТАБИЛЬНОСТЬ И ПРОИЗВОДИТЕЛЬНОСТЬ

43%
КОЛЛЕКТОРУ НЕ ХВАТАЕТ OBSERVABILITY

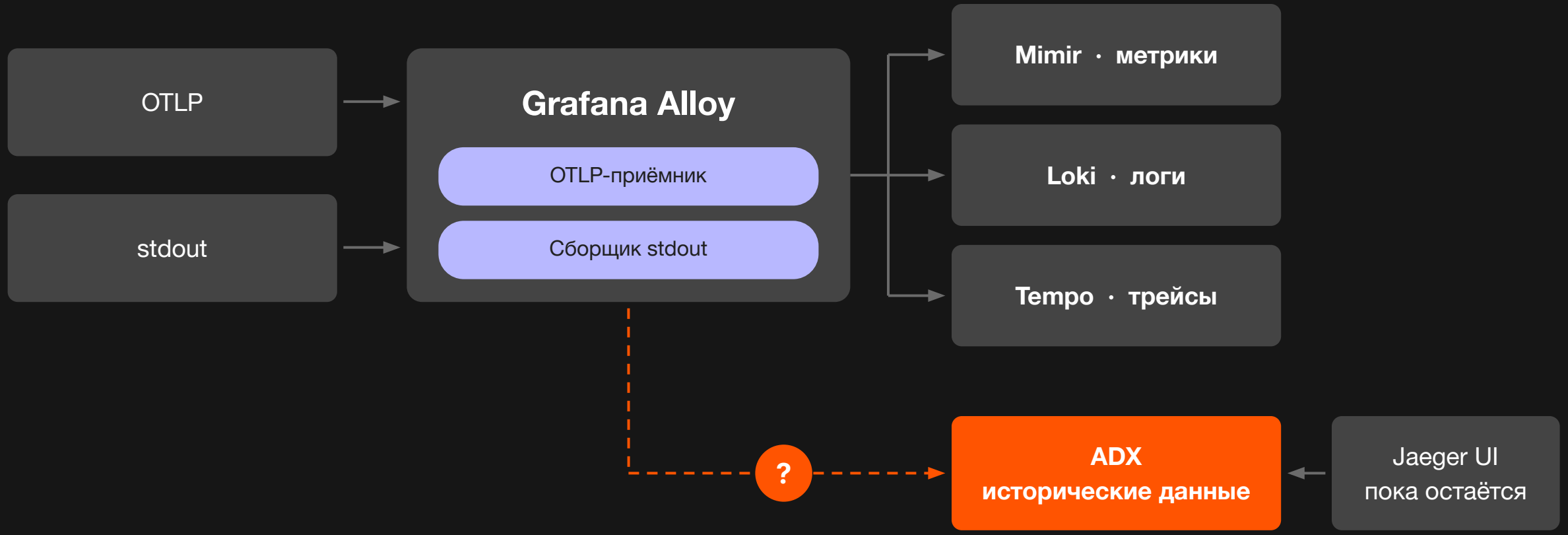
29%
ХОТЯТ БОЛЬШЕ ИНТЕГРАЦИЙ

Collector Builder считают сложным: **2 из 61** в 2024 → **21 из 55** в 2026

Читайте такие опросы до внедрения, а не после

OpenTelemetry Collector Survey, 2026 (follow-up) · opentelemetry.io

Один вход, три сигнала, один стек



Способ доставки данных в ADX под вопросом.

Лимиты — сразу, а не после первого инцидента

ADX · долгосрочное хранилище

- Приучил всю компанию: хранилище бездонное
- Можно спросить что угодно за любой период, не думая о цене выполнения
- «Бездонное» ≠ «бесплатное». Счёт приходил не автору запроса, а нам
- Один регулярный жирный запрос заметно поднимал стоимость

РОЛЬ: тяжёлая ретроспектива

Grafana Stack · оперативный контур

- Наши сервера, наша память, наш диск
- Tempo и Loki с первого дня зажаты таймаутами и лимитом окна запроса
- Принцип: стек живёт в тех ресурсах, которые ему выдали
- И именно здесь появляется связность между сигналами

РОЛЬ: «что происходит прямо сейчас»

Разработчикам объясняли прямо: это не замена ADX, это другой инструмент

Alloy — форк otelcollector. Но не близнец

OpenTelemetry Collector
тот же движок, те же пайплайны

Grafana Alloy

Стандартные приёмники и экспортеры — всё, что нужно для Grafana Stack

экспортера в ADX нет

OTel Collector

Широкий зоопарк вендорских компонентов

azuredataexplorerexporter есть



Собрать Alloy вместе с компонентом для ADX за разумное время не получилось

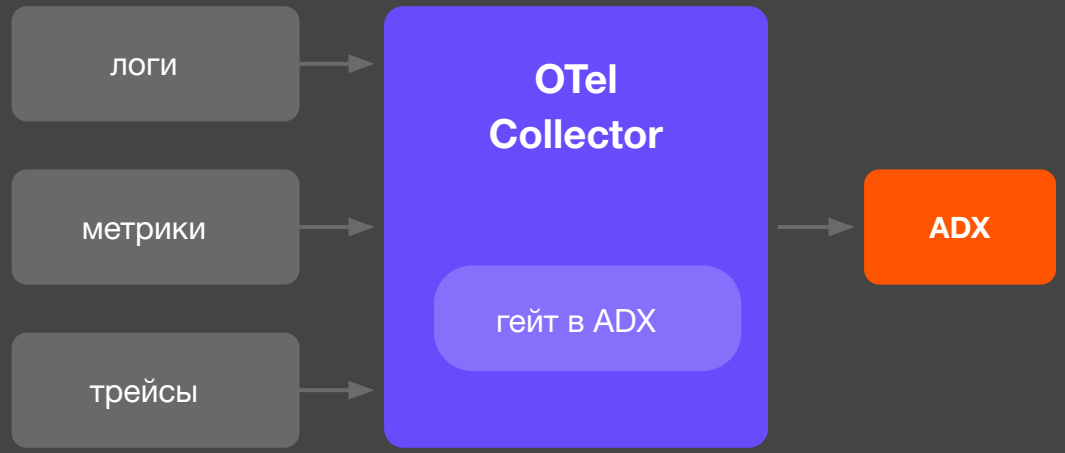
Не патчить Alloy — расширить то, что уже есть

БЫЛО



Стоял давно, тихо и незаметно.

СТАНЕТ



Полноценный гейт в ADX сразу для всех трёх сигналов.

«Есть работающий инструмент — доведём его до ума»

OOM. Под абсолютно штатной нагрузкой

Сам Grafana Stack завёлся без приключений

При первом же нагрузочном тесте OTel Collector лёг по OOM.

Коллектор теряет телеметрию всё, что держал в памяти

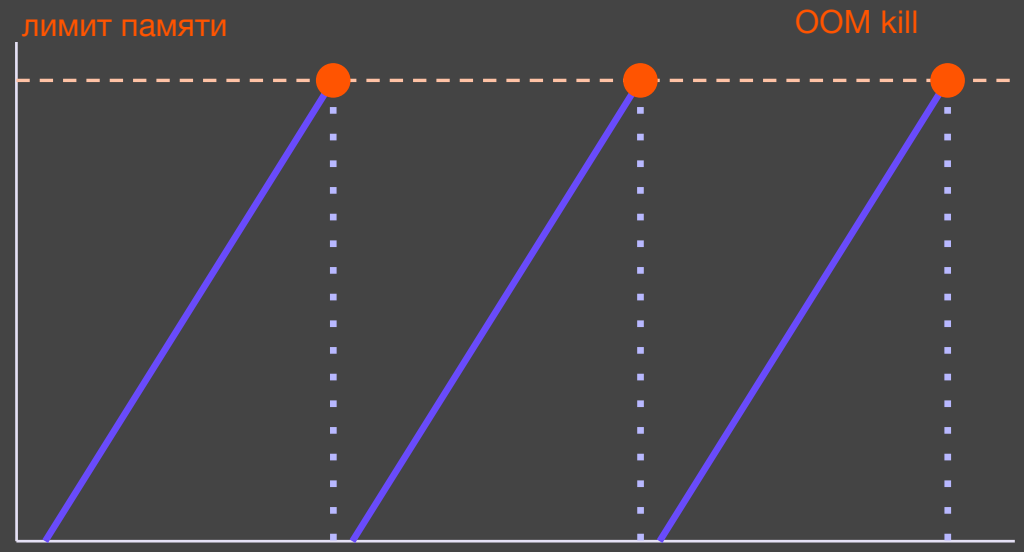
↓

Backpressure на Grafana Alloy приём упирается в вставший гейт

↓

Alloy складывается следом встаёт весь путь телеметрии

ПАМЯТЬ КОЛЛЕКТОРА

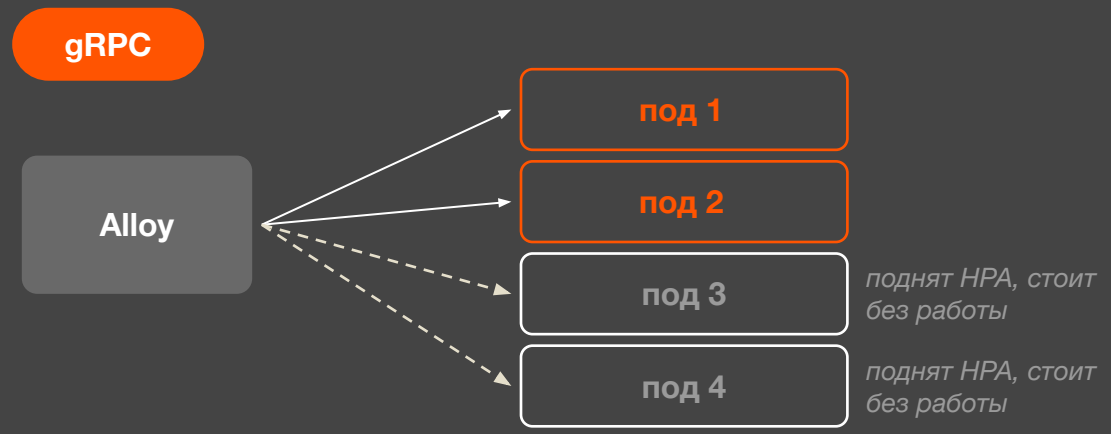


Схема, а не исторический график: метрик того периода уже нет

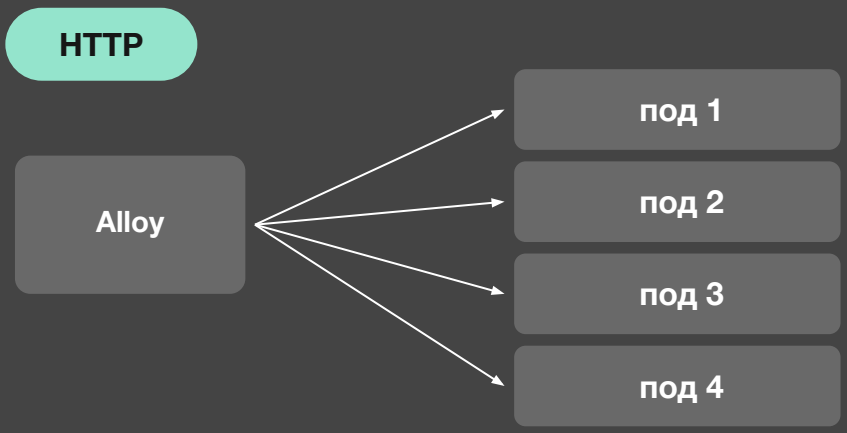
Хорошая новость: это был стенд. Плохая — то, что он нам показал

Ложный след: gRPC-балансировка

Симптом: часть подов явно горячее остальных, часть почти простаивает.



Балансировка на уровне соединений, а не запросов. Долгоживущие соединения уже приземлились на старые поды и не переезжают.



Перевели транспорт между Alloy и коллектором на HTTP. Перекос исчез, нагрузка легла по подам ровно.

А OOM остался. Ровно такой же, как был

У трубы две стороны. Крутить можно с обеих



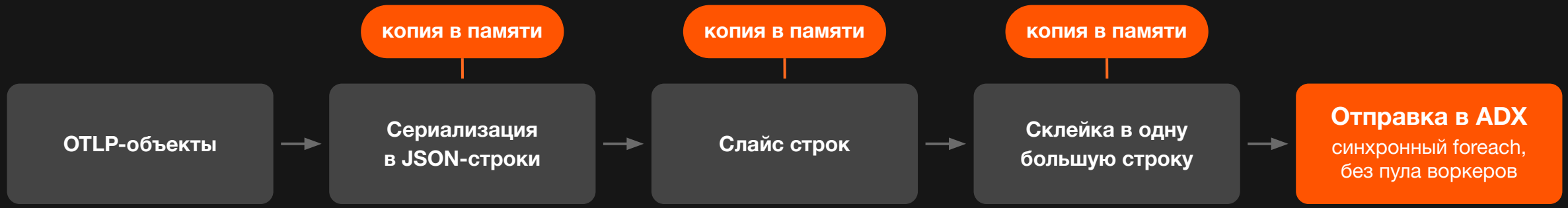
8000

КОНСЬЮМЕРОВ В МОМЕНТЕ

Это уже не тюнинг. Это акт отчаяния, оформленный как конфиг

Нулевой результат с обеих сторон трубы.

azuredataexplorerexporter: дело не в настройках



ДВА СИНХРОННЫХ ПОХОДА НА КАЖДУЮ ЗАПИСЬ

1. Загрузить данные во временное хранилище
2. Оповестить ADX, что данные загружены

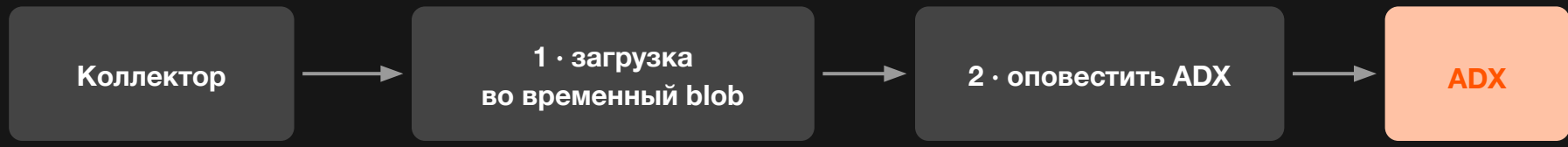
ЧТО ИЗ ЭТОГО СЛЕДУЕТ

сетевой сбой или тяжёлый батч → подпирает всю цепочку → батчи копятся бесконтрольно → OOM

Менять надо не настройки доставки — а способ доставки.

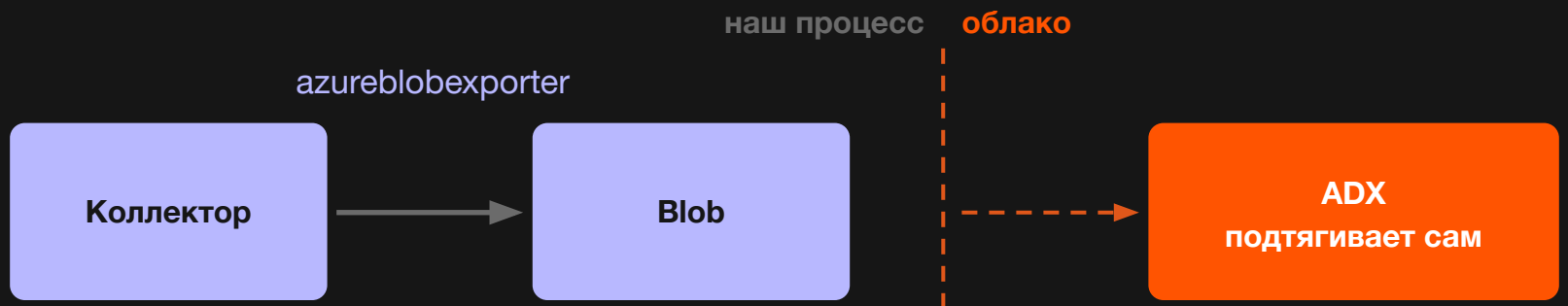
Один поход вместо двух

БЫЛО



Два синхронных похода, конвертация и дублирование в памяти — всё внутри нашего процесса

СТАЛО



Ровно так же работал Vector —
решение было под носом с самого начала

Наша память и наше управление
заканчиваются здесь

ОТБРОСИЛИ

Event Hub
(managed Kafka)

Упёрлись в максимальный размер сообщения

azuredataexplorerexporter выкинут целиком.

Данные доехали. Но не доехали

AZUREBLOBEXPORTER КЛАДЁТ ТАК · ОДИН JSON-МАССИВ НА ВЕСЬ БАТЧ

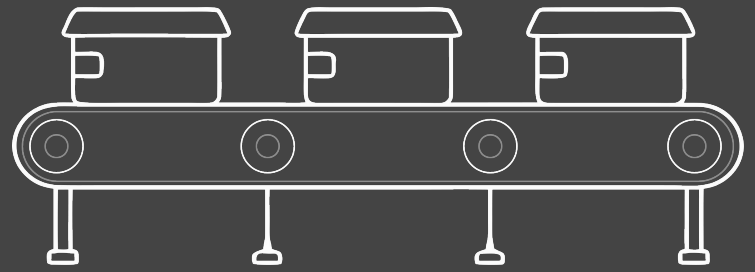
```
[{"ts":…,"msg":…},{"ts":…,"msg":…},{"ts":…,"msg":…}]
```

ADX ЖДЁТ ТАК · JSONL, ПО ОБЪЕКТУ НА СТРОКУ

```
{"ts":…,"msg":…}
{"ts":…,"msg":…}
{"ts":…,"msg":…}
```

ADX видит один JSON → **кладёт весь файл в одну строку** → **строка превышает лимит** → **тихо отбрасывается**

В таблице-приёмнике остаётся timestamp и пустота вместо содержимого



Форкаем коллектор.

«А чего так дорого?»

Вопрос, который инженеру всегда приятно услышать от финопса

БЫЛО — **Vector жал данные**

Компрессия всегда работала и про нее забыли.

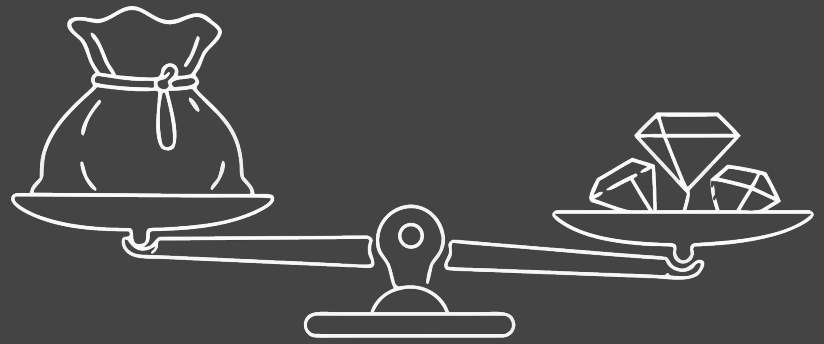
СТАЛО — **azureblobexporter не жмёт вообще**

Складывает в блоб сырой несжатый JSON.
В горячке внедрения про это никто не подумал

ИТОГ — **Потеряли то, что уже работало**

Компрессию внедрили сами — ещё одна доработка в форк

Поймать это на стенде было нельзя в принципе.
На FinOps-грабли наступают только в проде



Стандарт не описывает дисциплину логирования

1
Что с логами, написанными **до инициализации** OTel SDK?

2
Что с логами, если сервис **упал до инициализации** SDK?

3
Что, если **упал OTLP-приёмник**?

4
Отключать ли stdout? Как **избежать дублирования**, но не потерять важное?

OTel описывает протокол, формат и семантику полей. **Дисциплину логирования он не описывает**

ДОГОВОРИЛИСЬ ДЛЯ СЕБЯ ТАК

Собираем весь stdout
ничего не отключаем

Fatal — в оба канала
и в stdout, и в OTLP

Дублирование
минимальное и осознанное

Если сервис умирает, я хочу видеть причину даже тогда, когда вся система мониторинга недоступна. Это решение живёт у нас до сих пор.

Кто в лес, кто по дрова

ТРЕЙСЫ

- Трейсы длиной в аптайм сервиса
- Трейсы без «головы»
- Протокол Jaeger

МЕТРИКИ

- Большая часть сервисов пушит метрики по OTLP
- Exemplars пишут единицы

ЛОГИ

- По OTLP пушат буквально несколько сервисов
- Большинство пишет только в stdout, trace id отсутствует или пишется как попало
- Все старые проблемы остались — теперь в рамках стандарта

снова здравствуйте

```
"country": "RU"
"country": "Russia"
"country": 643
```

Пошли через архитектуру и НФТ. Требования обновили — а воз и ныне там

Половина, за которую отвечали мы, — сделана

ЦЕЛЬ 1 ВЫПОЛНЕНА

Уйти от зависимости от ADX

Стек стоит, он независимый,
его можно развернуть где угодно

ЦЕЛЬ 2 ВЫПОЛНЕНА НАПОЛОВИНУ

Связность трёх сигналов

метрики — ок | трейсы — 50/50 | логи — беда

Половина, за которую отвечали мы, — сделана. Половина, за которую отвечали не мы, — под вопросом.

Результат, который окупился

Унификация метрик

Сервисы отдают метрики в едином формате, с предсказуемыми именами и лейблами.

Prometheus Operator

Заехал под шумок: «раз уж мы тут всё равно всё перебираем».

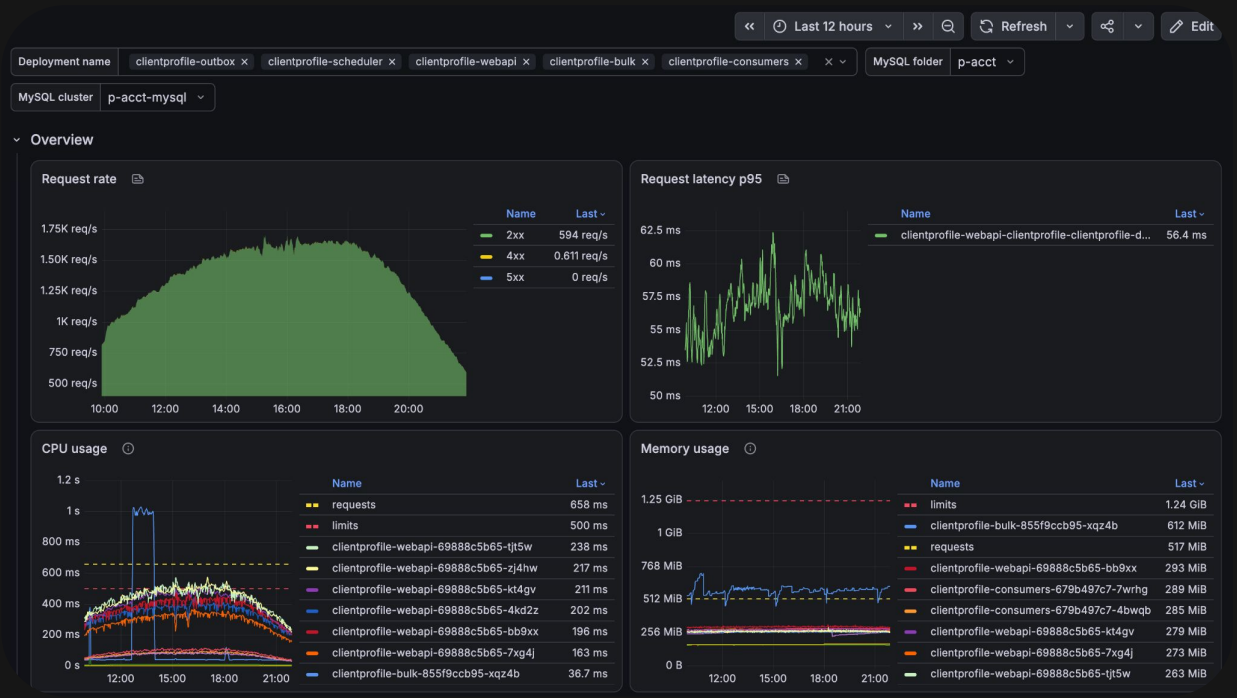
Мониторинг «как сервис»

1. Декларативное описание алертов
2. Базовые дашборды “из коробки”
3. Упрощенный UX для разработчиков

Как это выглядит — на следующем слайде

Один CR — и мониторинг есть

```yaml
apiVersion: k8s.paas.dodois.io/v1
kind: DodoAppServiceMonitor
metadata:
  name: clientprofile
spec:
  serviceRef:
    name: clientprofile
    kind: DodoAppService
  alerting:
    kafka:
      enabled: true
      max_queue: 5000
    local_command_queue:
      enabled: true
```



ИЗ КОРОБКИ

- **состояние подов**
- **состояние runtime**
- **error rate**
- **mysql**

Никто не рисовал дашборд под конкретный сервис — **он основан на конвенции именования метрик.**

Связность всё-таки пришла



Те же хранилища, что и у бэкенда

Grafana Faro — observability SDK для frontend и мобильных приложений

СКВОЗНАЯ ТРАССИРОВКА нажатие кнопки в браузере или мобильном приложении → поход в базу

Еще не носит массовый характер, но уже работает

Та самая связность, ради которой всё затевалось, только с другой стороны.

Форк, который не просил есть

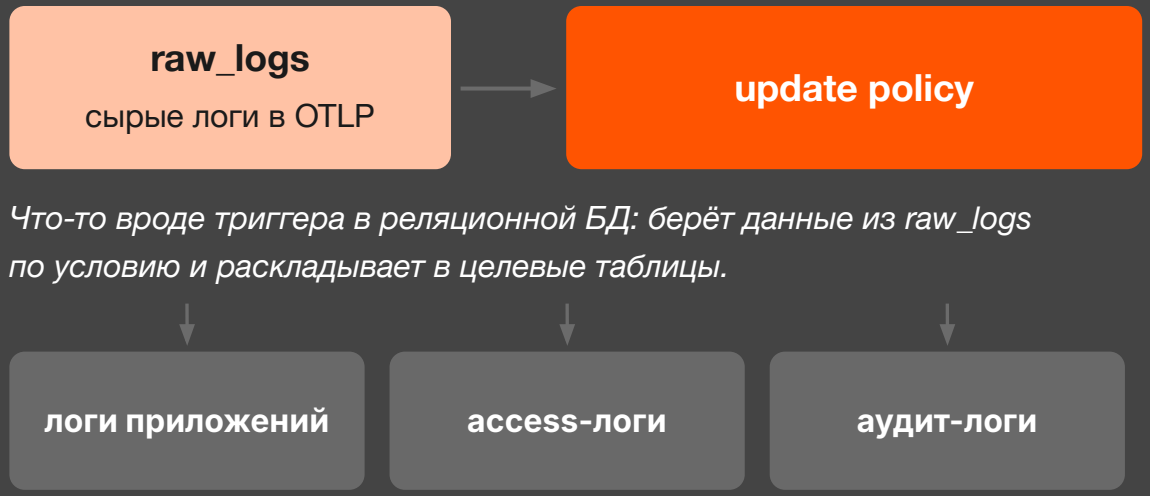
2 ДОРАБОТКИ, ГОД ЖИЗНИ

Обычно форк — это долг
ребейзы на апстрим, разбор конфликтов, догонялки за чужими релизами

Мы не страдали
зафиксировали версию и жили на ней. *Ни инцидентов, ни человека, который его пасёт*

Цена решения
почти год без фиксов апстрима и новых компонентов.

ЧТО ВНУТРИ НАШЕГО ADX



Работал ровно до того момента, **пока мы не упёрлись в update policy**

Три пары «ключ — значение». Слева и справа

ПРИЕЗЖАЕТ ОТ OTEL COLLECTOR

```
"attributes": [
 {
  "key": "service.name",
  "value": { "stringValue": "orders" }
 },
 {
  "key": "service.version",
  "value": { "stringValue": "1.4.2" }
 },
 {
  "key": "deployment.environment",
  "value": { "stringValue": "prod" }
 }
]
```

ХОЧЕТ ВИДЕТЬ ADX И ЛЮБОЙ ЧЕЛОВЕК

```
"service.name": "orders"
"service.version": "1.4.2"
"deployment.environment": "prod"
```

три строки плоского JSON

update policy разворачивает левую структуру в правую — на каждой вставке, проходя по массиву объектов с динамической типизацией

Протокол такой, какой есть. С ним можно либо работать, либо не работать.

Десять минут задержки

10

МИНУТ ЗАДЕРЖКИ ЛОГОВ

МЕХАНИКА

1. Update policy выполняется для каждой вставки
2. Чем больше данных и чем тяжелее policy — тем больше процессорного времени уходит на обработку
3. В какой-то момент данные начинают отставать

Во время инцидента человек смотрит не на то, что происходит, а на то, что происходило десять минут назад.

Это не «немного медленнее». Это неисправный инструмент

Оптимизировать попробовали. Хватило ненадолго — операция дорогая по своей природе, а данных меньше не становится.

Отказались не потому, что сломался

Потолок находился не в OTel Collector, а в его стыке ADX.

Инжест через блоб держал нагрузку, компрессия экономила деньги, JSONL доезжал как задумано — чинить в форке было нечего.



Промежуточного блоба больше нет

gRPC вернулся — в своём коде его можно приготовить правильно

Уплощение масштабируется горизонтально, а не деньгами в ADX

Форк можно чинить бесконечно.

Иногда дешевле написать заново маленький инструмент, который делает одну вещь, но хорошо

Всего, что обещает OpenTelemetry, можно добиться и без OpenTelemetry

1. Приведите телеметрию в порядок: договоритесь о нейминге метрик, лейблов и полей в логах и трейсах
2. Включите exemplars и настройте связность между датасорсами в Grafana
3. **Стандарт не покроет вашу специфику.** Подумайте о собственной библиотеке. Правильное сэмплирование, стандарты нейминга метрик и единые поля из стандарта к вам не придут

Здесь OTel действительно помогает: даёт готовые соглашения, общий протокол и общую терминологию.
Он делает задачу проще

Но он не делает её за вас. Наведение порядка — это работа с людьми и с кодом ваших сервисов

Стандарт и инструменты — не одно и то же

ИНСТРУМЕНТЫ

Про них вы слушали сорок минут

- Вендорские компоненты с плохой производительностью
- Вендорские компоненты с недостаточным функционалом
- Сложная система сборки
- Кривая gRPC балансировка
- Баги в клиентских SDK

46% дорабатывают коллектор напильником.

СТАНДАРТ

И здесь всё ровно наоборот

- Наличие стандарта лучше, чем его отсутствие
- Его поддерживает всё больше платформ и инструментов — и эта поддержка приходит к вам бесплатно
- Любой ИИ-агент разберёт телеметрию по OTLP из коробки: ему не нужно объяснять ваш формат

Ваш внутренний формат этого не умеет.

Инструментарий — с открытыми глазами и с заложенным на него временем

Стандарт — берите

Посчитаем честно

Дней потрачено	140
Людей задействовано	4
Инструментов изучено	7
Форков создано	1
Форков выброшено	1
Инструментов создано	1
Нервных уничтожено	**300 000 000**

ПОЧЕМУ НЕ ОКУПИЛОСЬ

Нас не просили идти за хайпом

Нас просили exemplars, фронтенд-обсервабилити и больше трейсов.
Точечные запросы заинтересованных людей, а не массовая боль

Новые инструменты проиграли привычным

Мы продавали связность сигналов и мониторинг в одном окне. Окно неудобное, Kibana и Jaeger привычнее, а ADX сложно переплюнуть

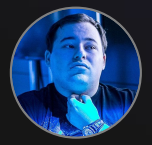
Качество телеметрии стандарт не улучшает

Плохой она не была. Но значимо лучше от стандарта не стала

Обновить NFR — не значит внедрить

Понятных доработок в сервисах и сроков под них не появилось. Такое проталкивается с двух сторон

У хайпа вокруг OpenTelemetry есть цена. Мы её заплатили, оно того не стоило.



СЕРГЕЙ БУХАРОВ
АРХИТЕКТОР ОБЛАЧНОЙ ИНФРАСТРУКТУРНОЙ ПЛАТФОРМЫ, DODO ENGINEERING

@sbukhar0v

Спасибо!
Давайте обсудим

О ЧЁМ МОЖНО СПРОСИТЬ

- форк и свой сервис
- Grafana Stack
- адопшен и NFR