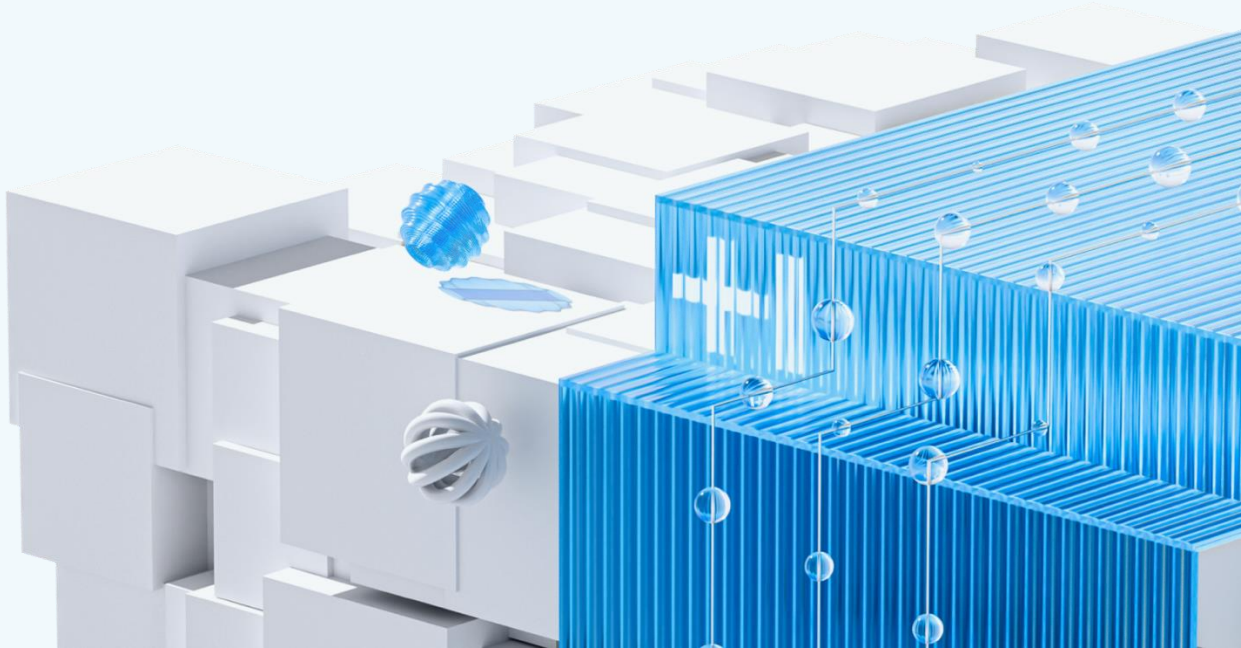
Наблюдаем за микросервисами правильно

Руслан Гайнанов
главный инженер DevOps

TI



Т1 сегодня — лидер российского ИТ-рынка

Многопрофильный ИТ-холдинг, предоставляющий полный спектр решений для реализации высокотехнологичных проектов с учётом актуальных потребностей и отраслевой специфики заказчиков

249,6 млрд руб.
+29% к 2023 г.
выручка за 2024 г. по МСФО

27 000+
сотрудников в ИТ-холдинге Т1

Рынок

850+
корпоративных и государственных клиентов

№1 в России

c-news
Крупнейшие ИТ-компании

IRX
Крупнейшие группы и компании в области ИКТ

Решения

97
единиц ПО внесено в ЕРРП

40+
центров компетенций

Партнёры

138
компаний в партнёрской сети НОТА

369
вендоров интеграции

485
сервисных партнёров

Обо мне

Руслан Гайнанов

главный инженер DevOps

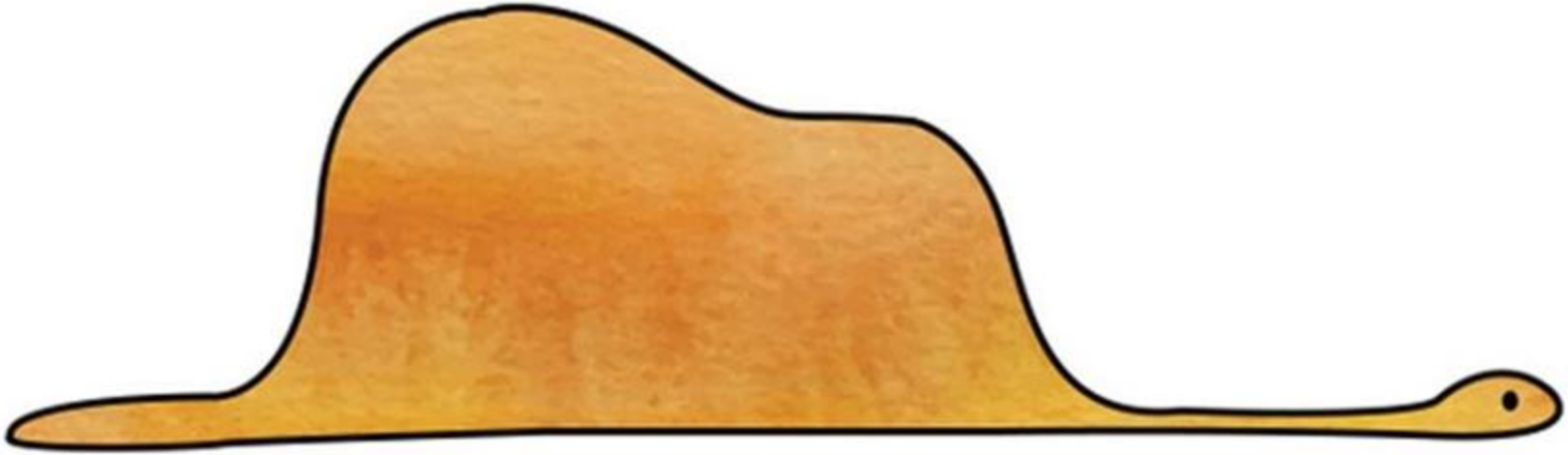
+ Пишу код
+ Управляю серверами
+ Оптимизирую процессы разработки

TI Иннотех

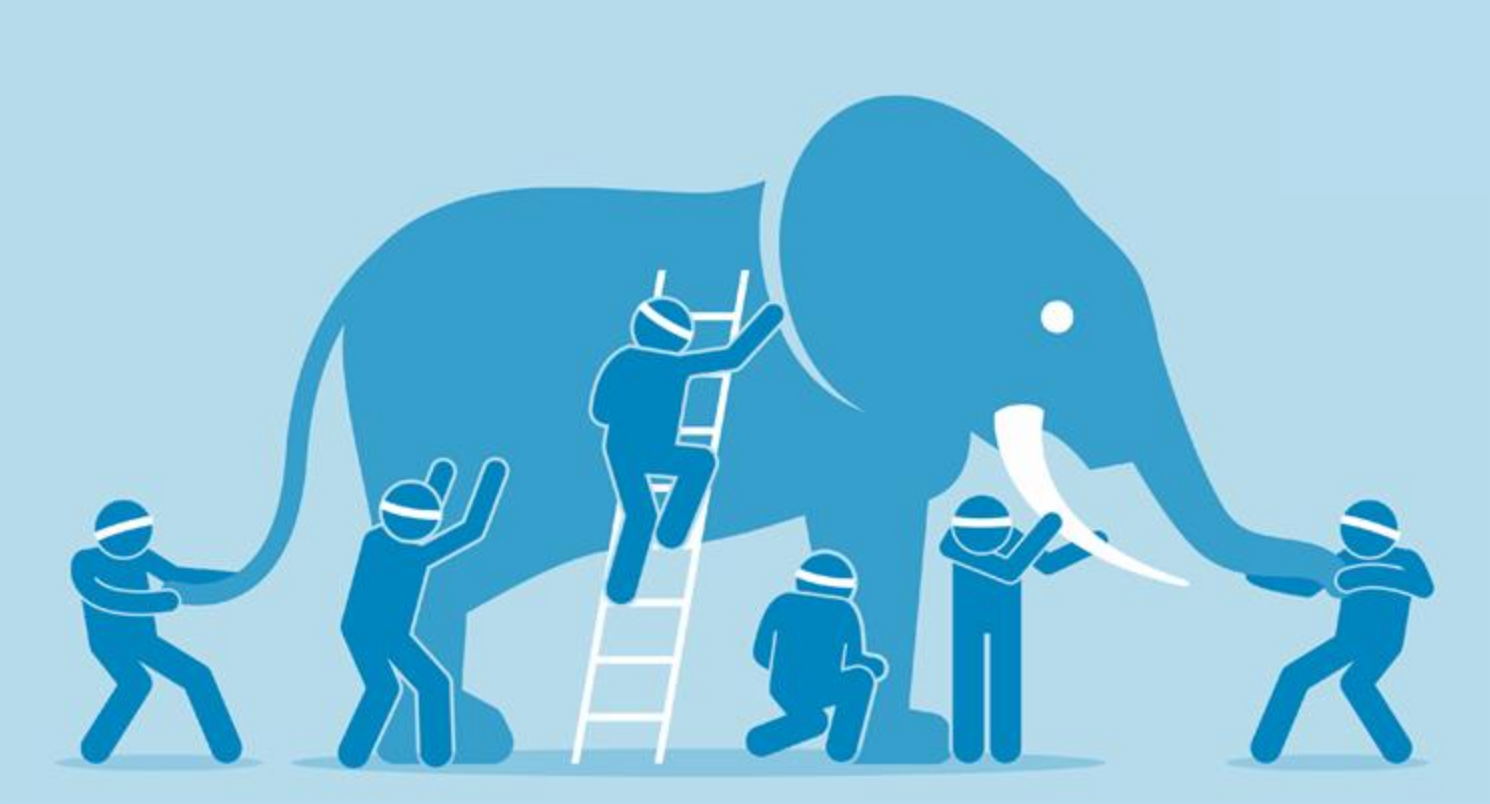
Каждый видит то, что хочет увидеть…

Одна мудрая притча

Рисунок №1



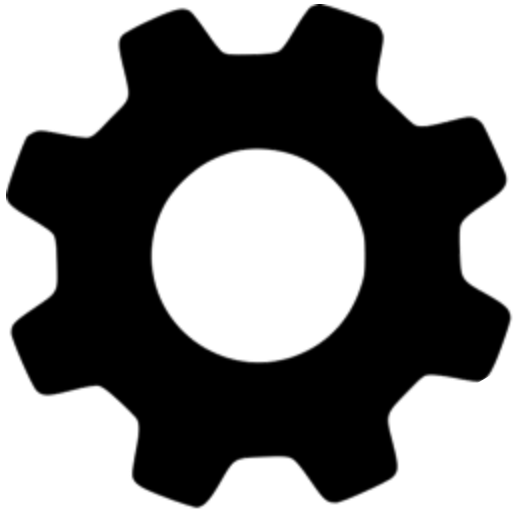
«Шесть мудрецов смотреть слона...»



ИМПУЛЬС ТІ

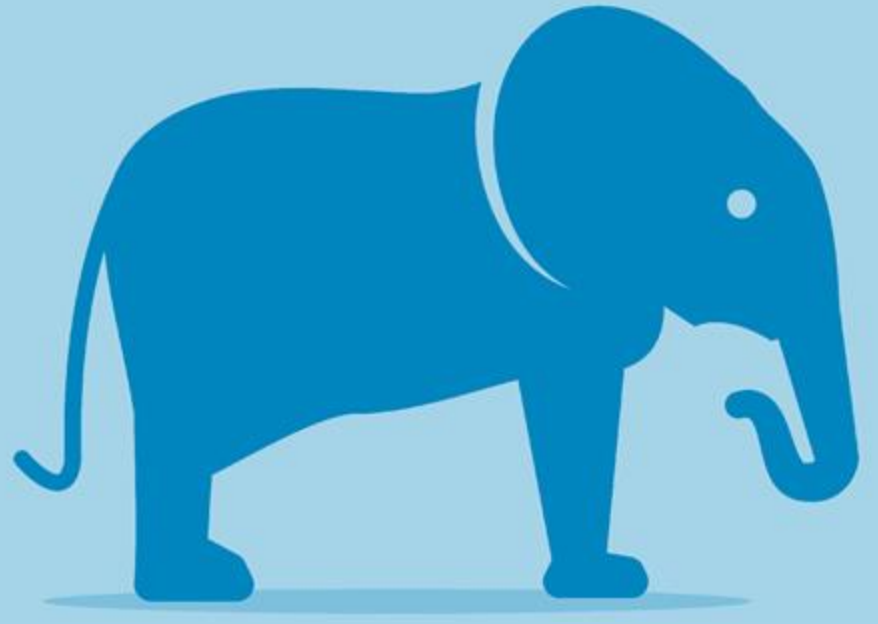
https://dzen.ru/a/YTETjO9zYlgUS5QU

Monolith

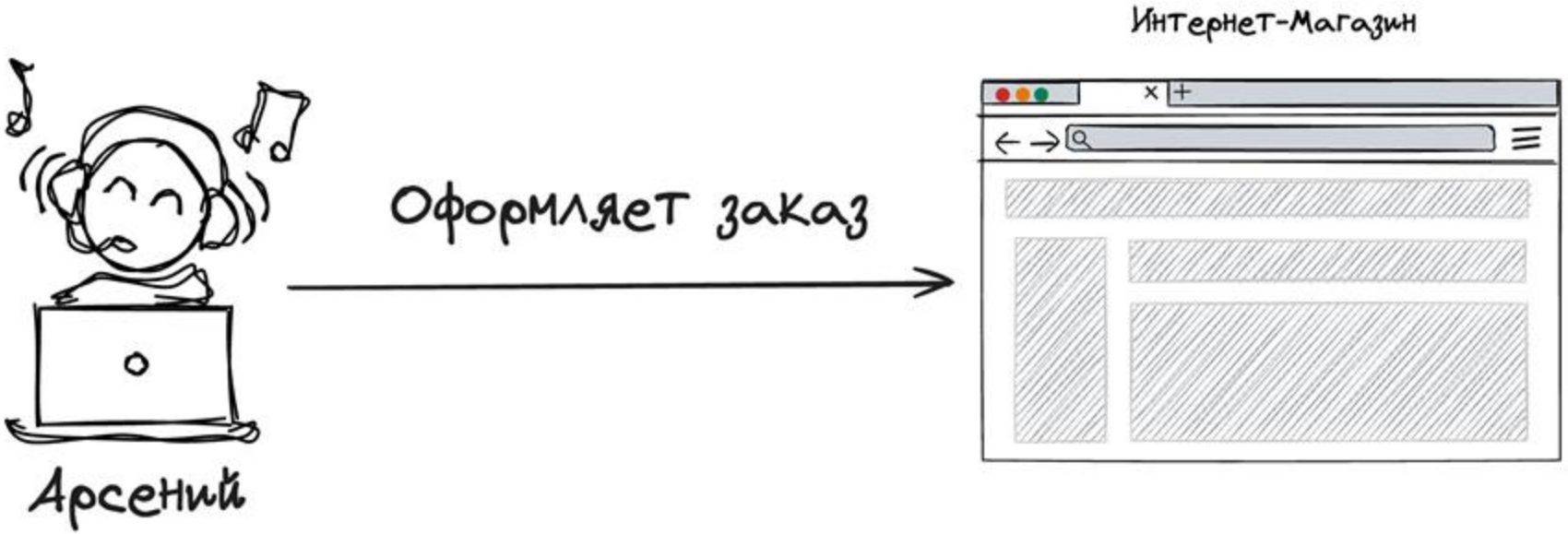


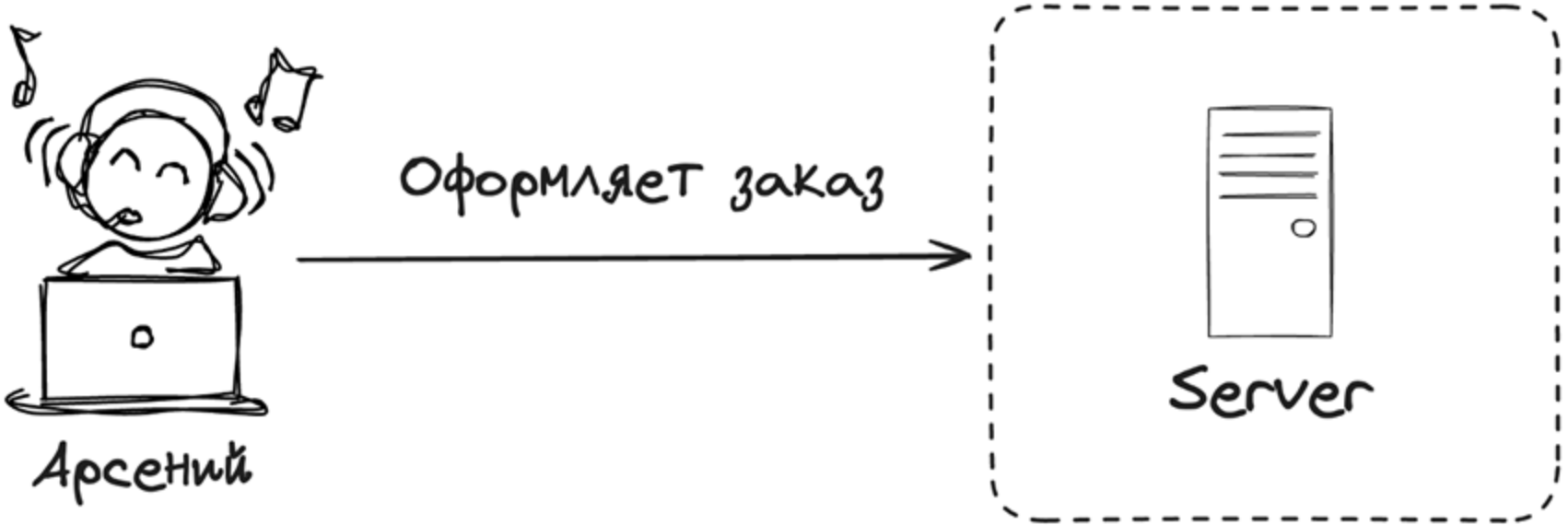
Microservices

Микросервис — что же ты такое?

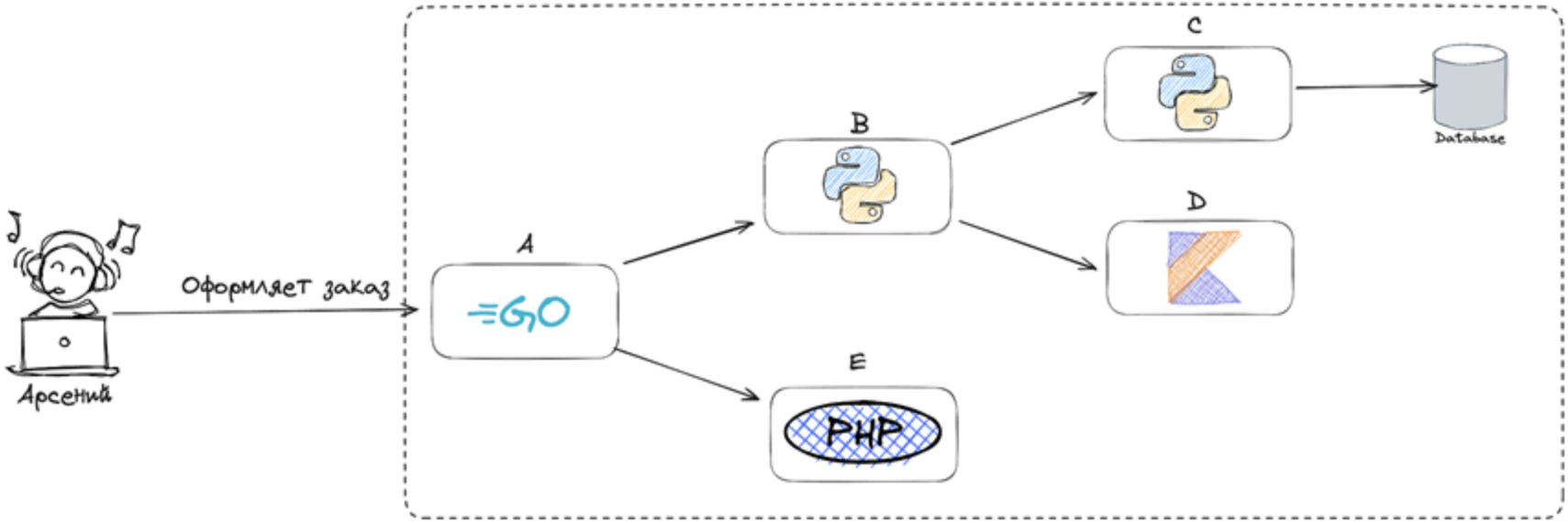


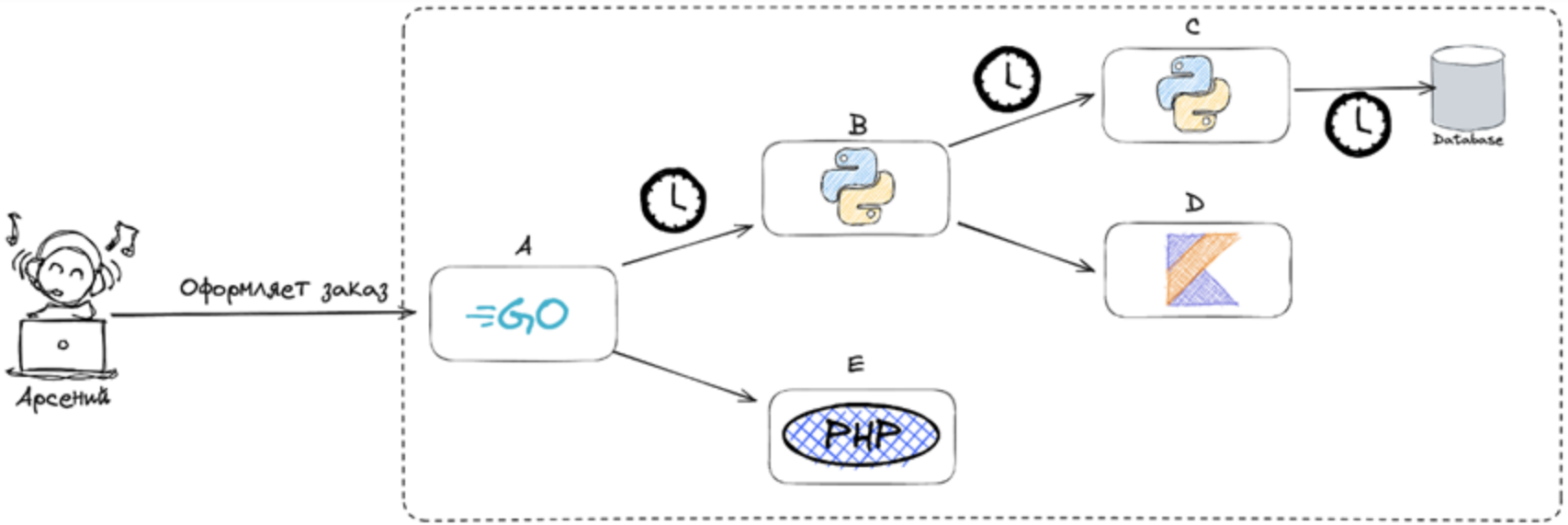
Ситуация

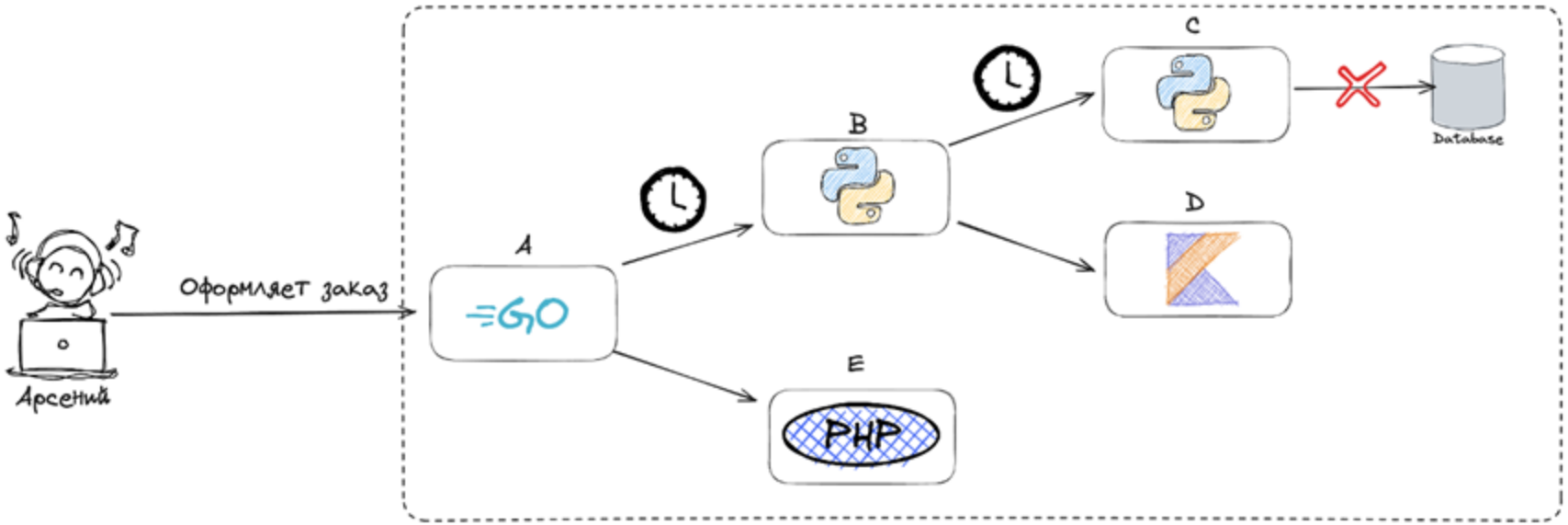

Интернет-Магазин
Оформляет заказ
Арсений

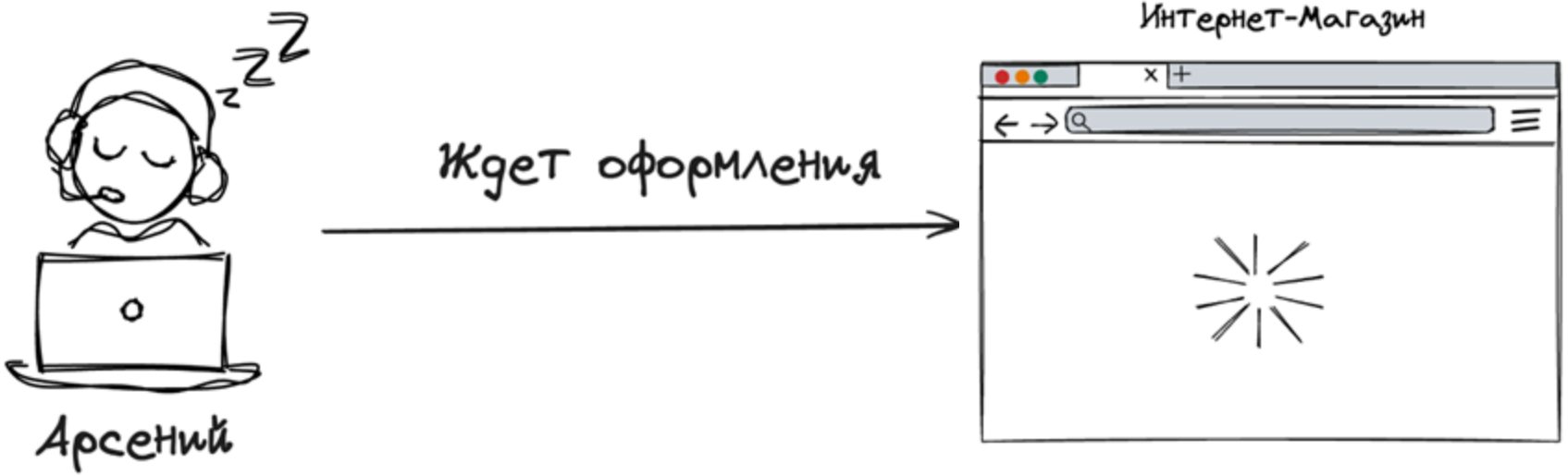

Оформляет заказ
Арсений
Server

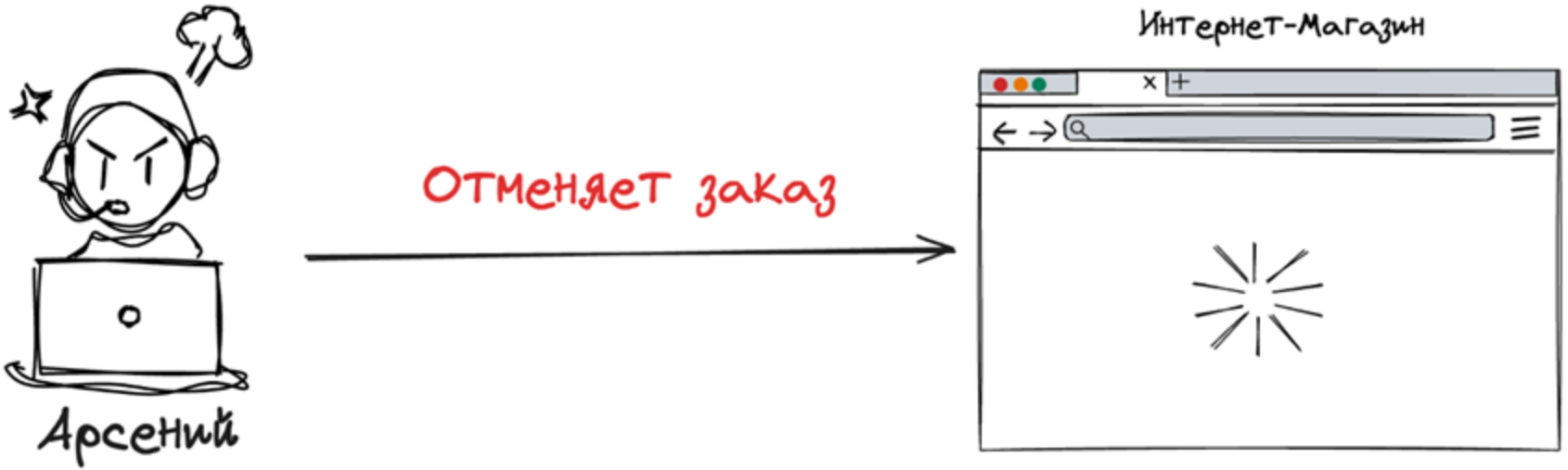
Оформляет заказ


Арсений
Оформляет заказ
A
B
C
D
E
Database
PHP


Арсений
Оформляет заказ
A
B
C
D
E
Database
PHP


Оформляет заказ
Арсений
A
B
C
D
E
Database
PHP


Интернет-Магазин
Ждет оформления
Арсений


Интернет-Магазин
Отменяет заказ
Арсений

Как этого избежать?

Организовать наблюдение за системой

Процесс мониторинга

Сбор данных телеметрии

Анализ

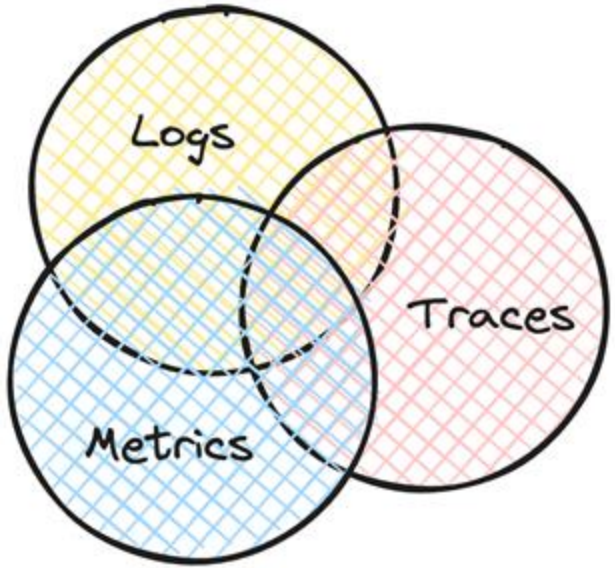
Визуализация

Контроль и оптимизация

Какие данные собирать?

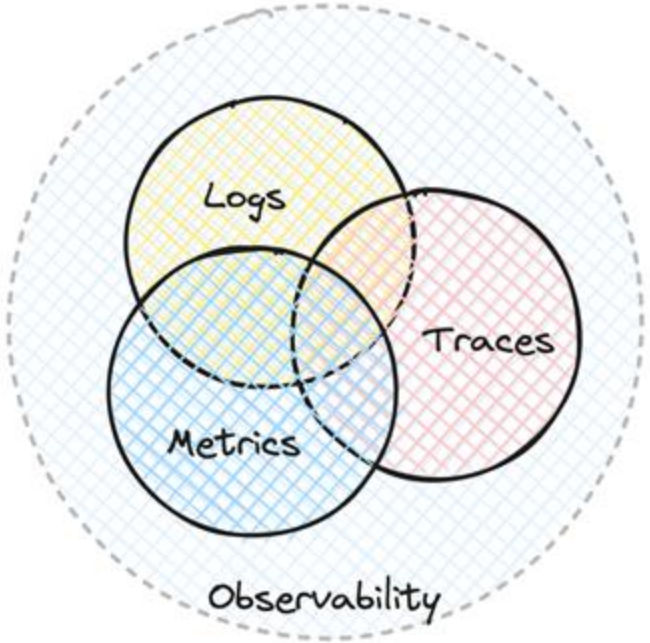
Три сигнала телеметрии

1. **Метрики** — количественные данные некоторых параметров системы
2. **Логи** — структурированные данные о поведении и события в системе
3. **Трейсы** — данные о выполняемых операциях внутри системы



Три сигнала телеметрии = Observability

1. **Метрики** — количественные данные некоторых параметров системы
2. **Логи** — структурированные данные о поведении и события в системе
3. **Трейсы** — данные о выполняемых операциях внутри системы

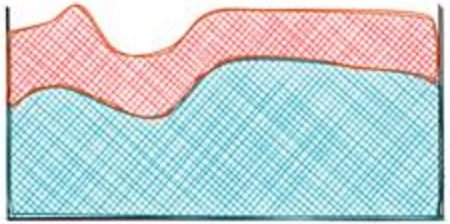


Observability — сила в единстве

Detect

Seconds

SLI/SLO/KPIs

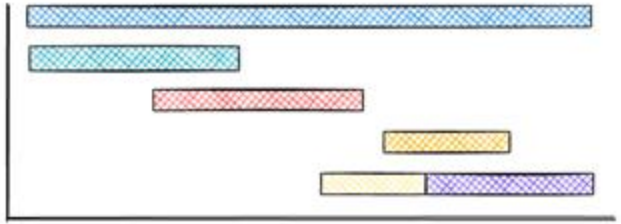


Возникла проблема?

Troubleshoot

Seconds..Minutes

Связи, задержки, ошибки



Где именно возникла проблема?

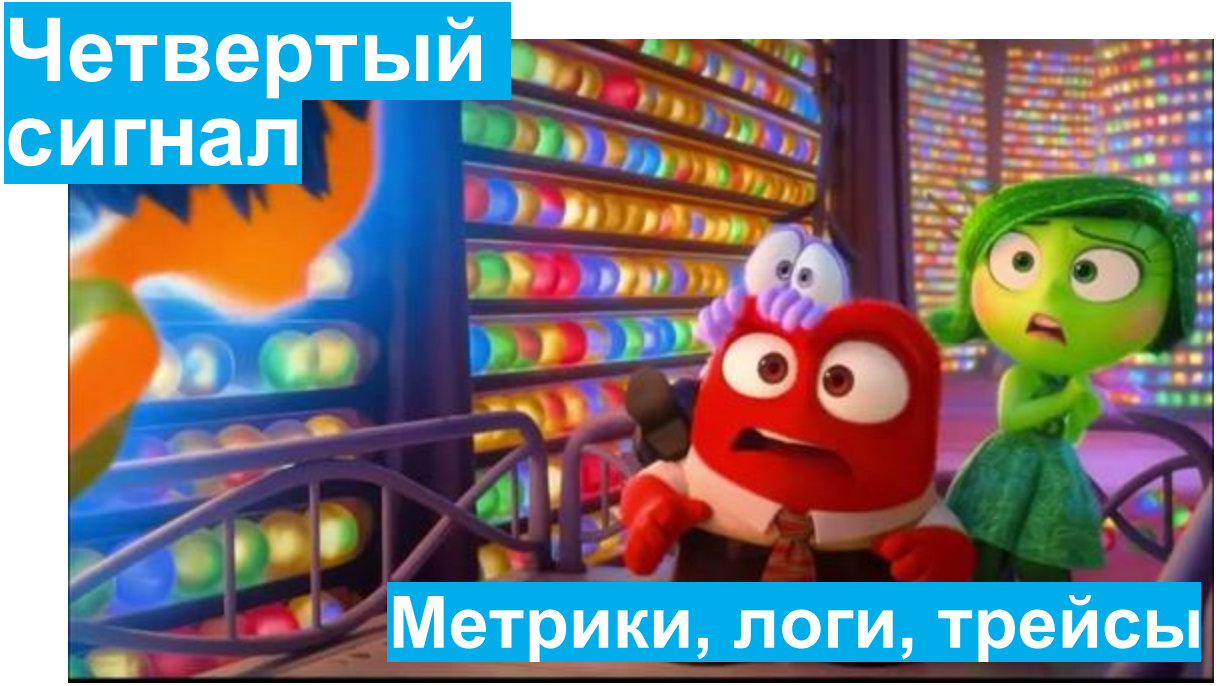
Root Cause

Minutes..Hours

Полная информация

Почему возникла?

Четвертый сигнал?

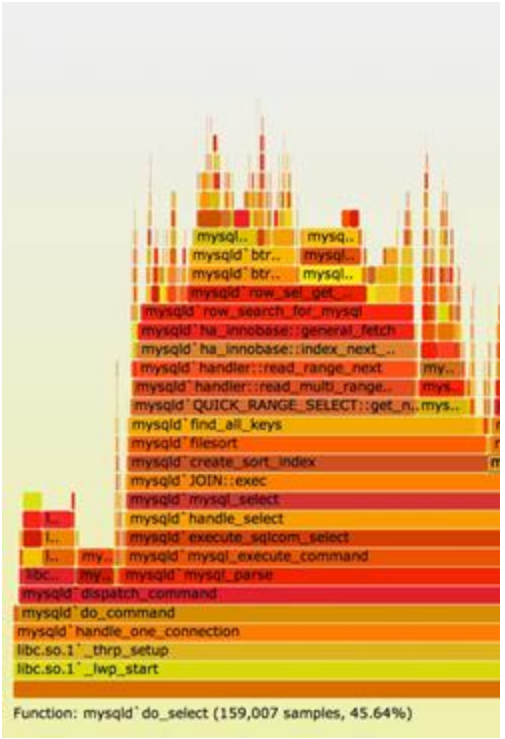


Профили — куда уходит …время

Профили (**Profiles**)

- трассировки стека вызовов
- подробное представление на что в коде тратятся ресурсы:
  - циклы процессора, память и т.д.
- поиск узких мест

Непрерывное профилирование (**Continuous profiling**) — инструмент для оптимизации производительности приложений и эффективного использования ресурсов



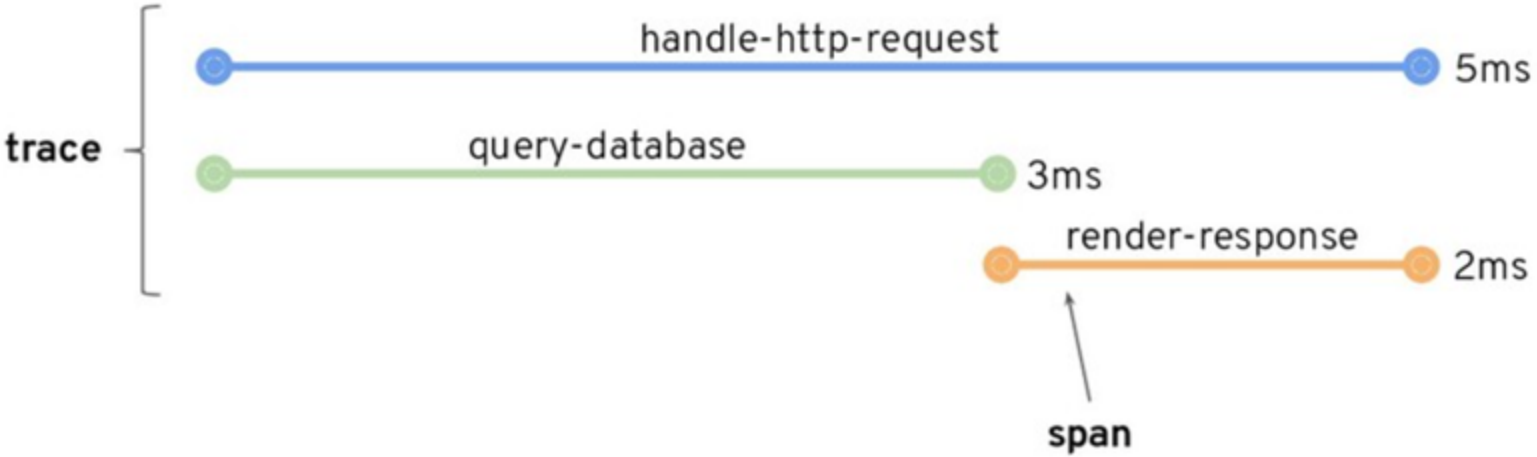
Вопрос

Кто использует **все три сигнала** в своих проектах?

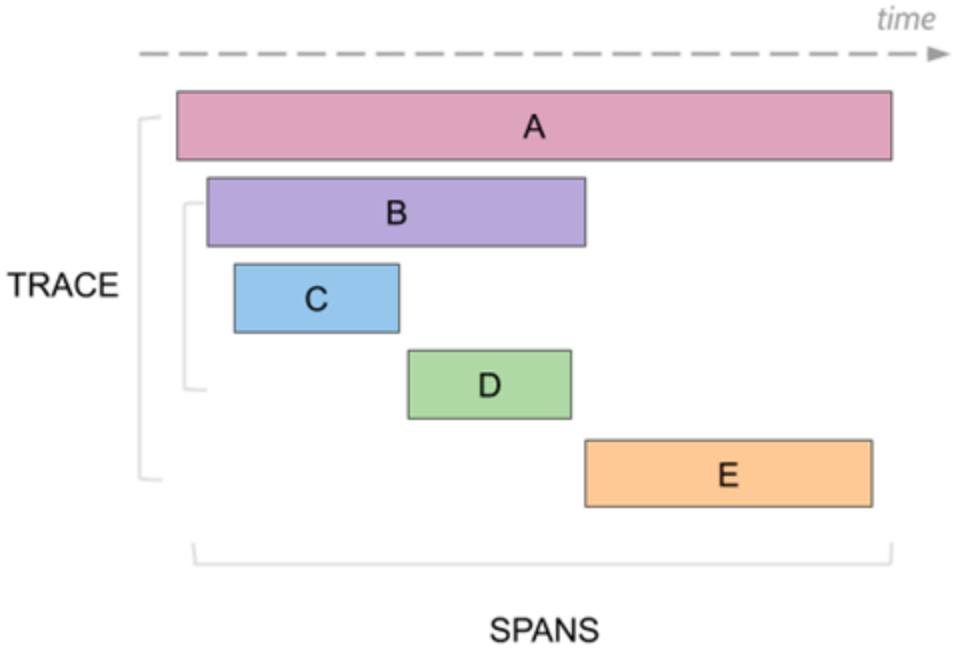
Из чего состоит трассировка?

Спан (span) — единица какой-либо работы/операции

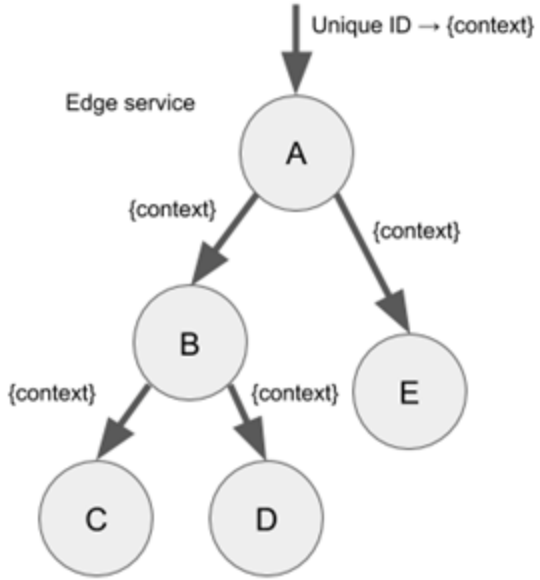
Трейс (trace) — объединение спанов по одному запросу

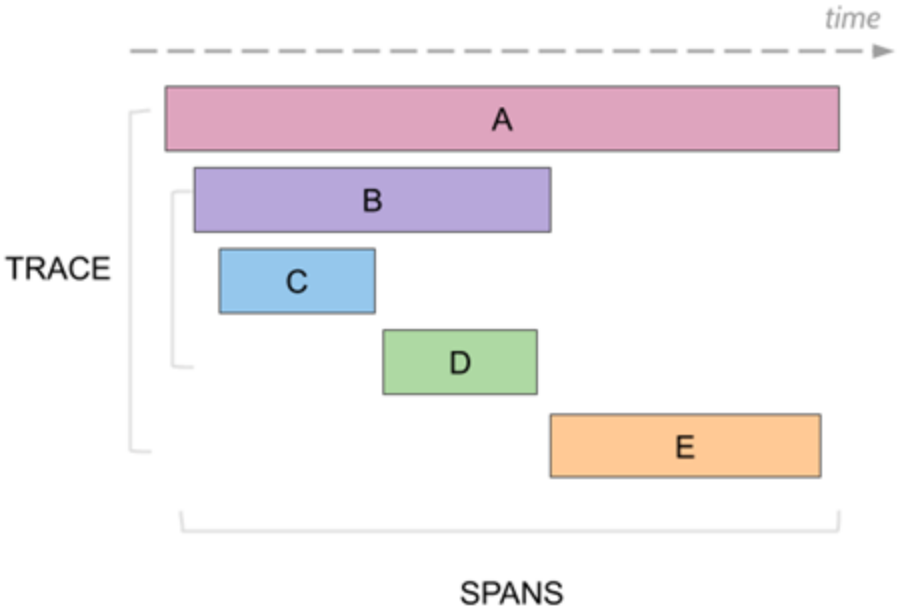


Трассировка в микросервисах

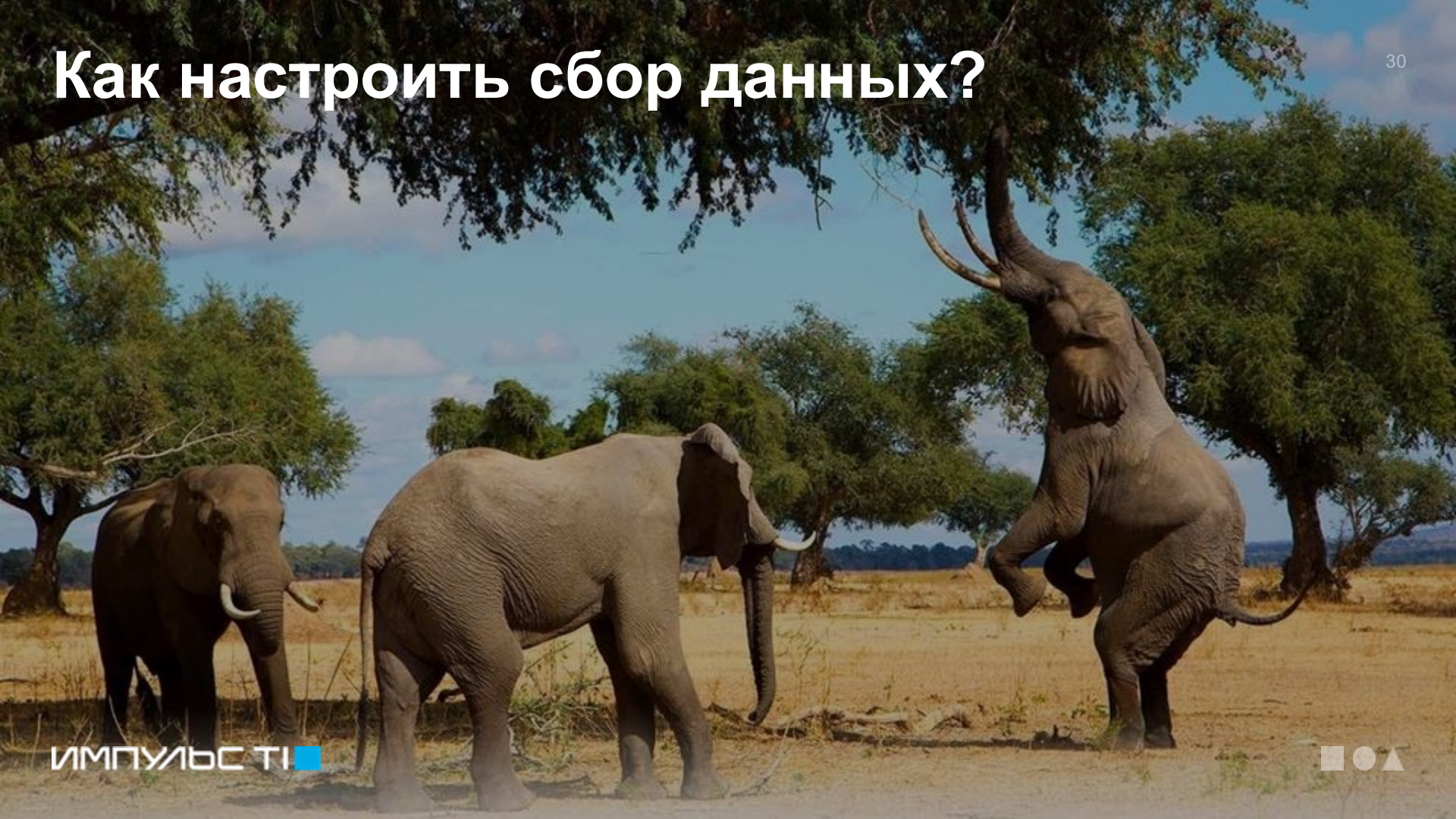


Трассировка в микросервисах





Как настроить сбор данных?

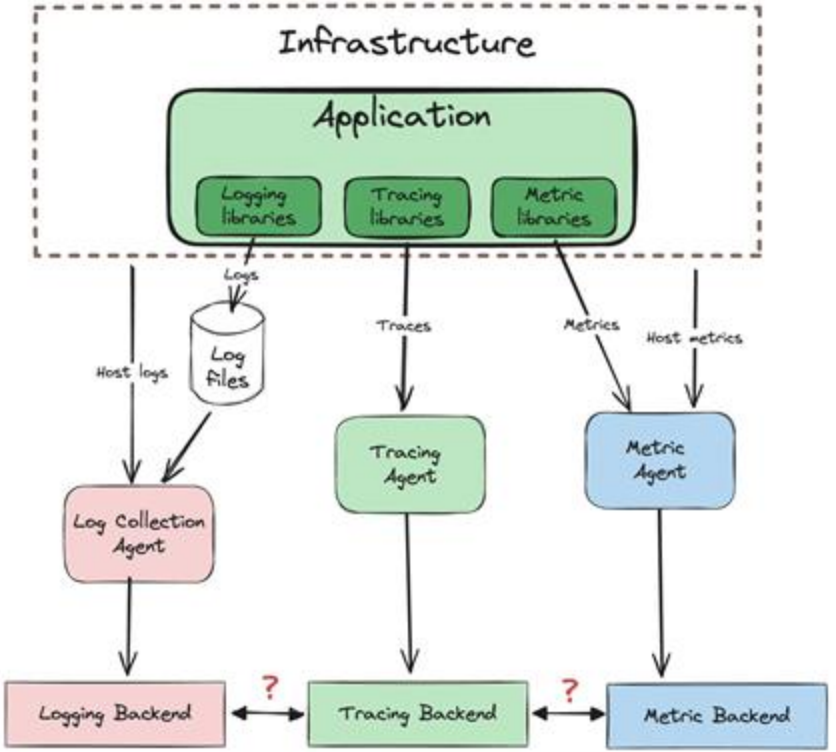


Классический подход

Телеметрия собирается
с помощью различных библиотек

Данные обрабатываются
и хранятся раздельно

Нет корреляции
между событиями



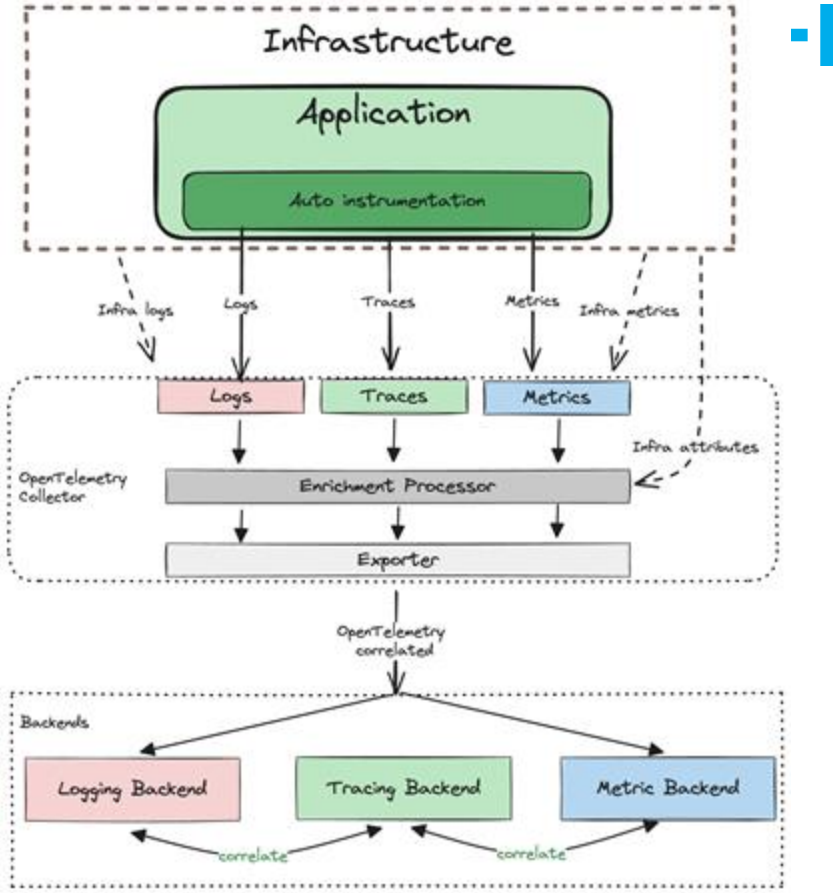
Современный подход

С помощью OpenTelementry

Сбор телеметрии через инструментацию

Единый обработчик
с обогащением данными

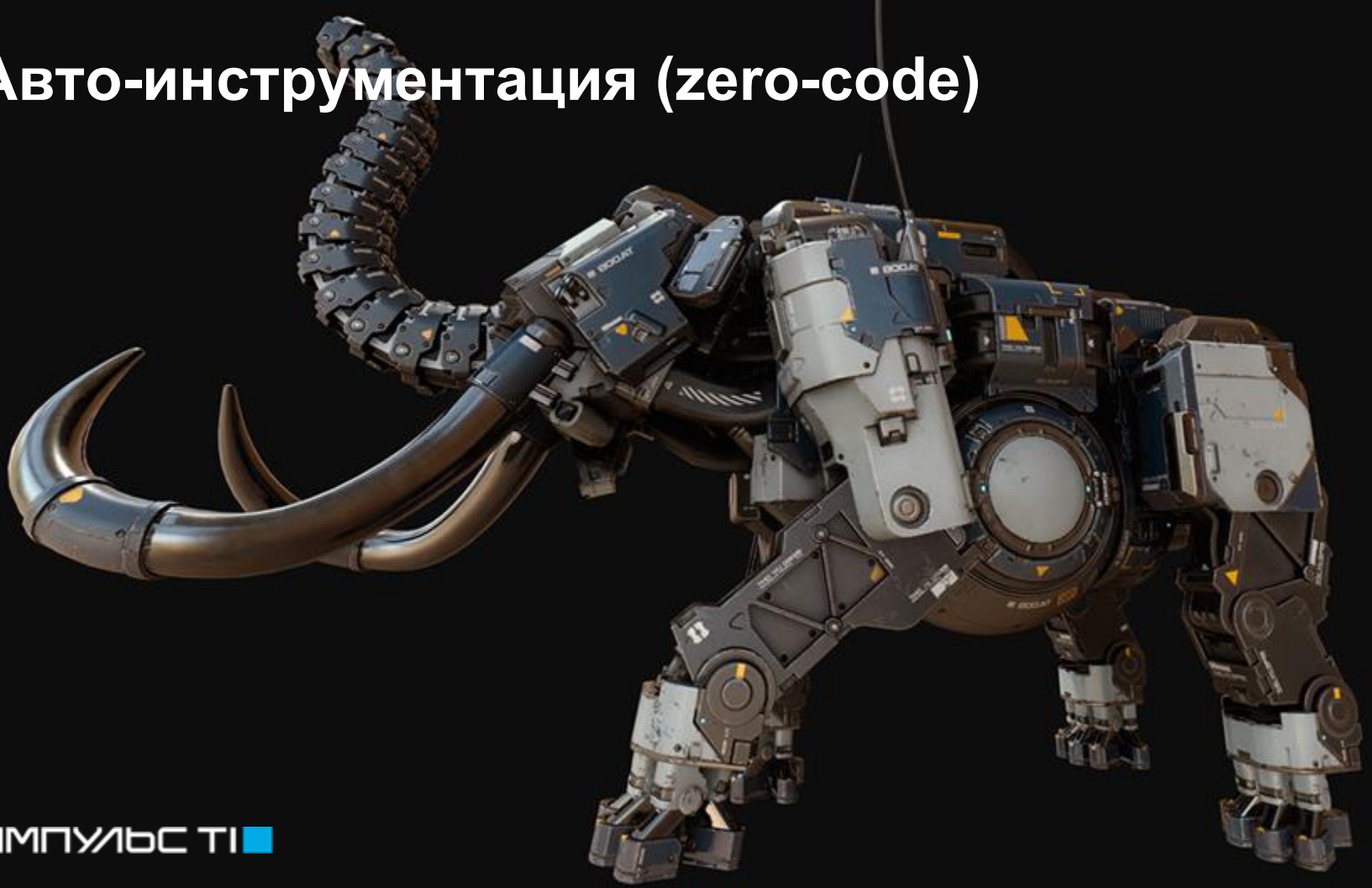
Связанные между собой события



Стандарт в телеметрии

OpenTelemetry

Авто-инструментация (zero-code)



Простой веб-сервер на Python-Flask

Файл app.py:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "hello-world"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8001)
```

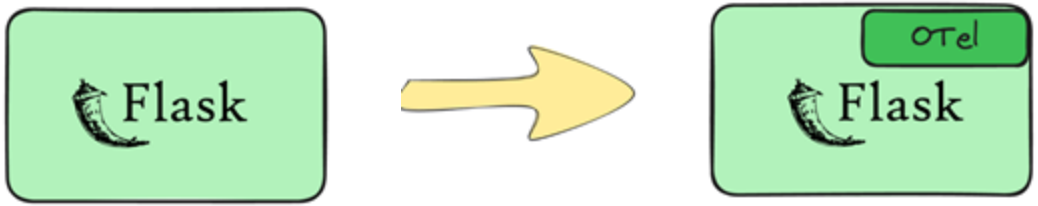
Запуск веб-сервера

```
$ python app.py
```


```
❯ curl localhost:8001
hello-world
```

Авто-инструментация

“Collection methods that do not require the end-user to modify application’s source code. Methods vary by programming language, and examples include bytecode injection or monkey patching.” (c) OpenTelemetry glossary




```
$ opentelemetry-instrument python app.py
```

node.js
Java
Microsoft .NET
GO
python

Monkey patching: runtime

```python
import time
from functools import wraps
...

def instrument_function(original_function):
    @wraps(original_function)
    def instrumented_function(*args, **kwargs):
        start_time = time.perf_counter_ns()
        result = original_function(*args, **kwargs)
        duration = time.perf_counter_ns() - start_time
        print(f"{original_function.__name__} took {duration} nanoseconds")
        return result
    return instrumented_function
```

Bytecode Instrumentation: VM (URL)

Compile-Time Instrumentation: Binary (URL)

instrumentation/opentelemetry-instrumentation-flask/src/opentelemetry/instrumentation/flask#L337

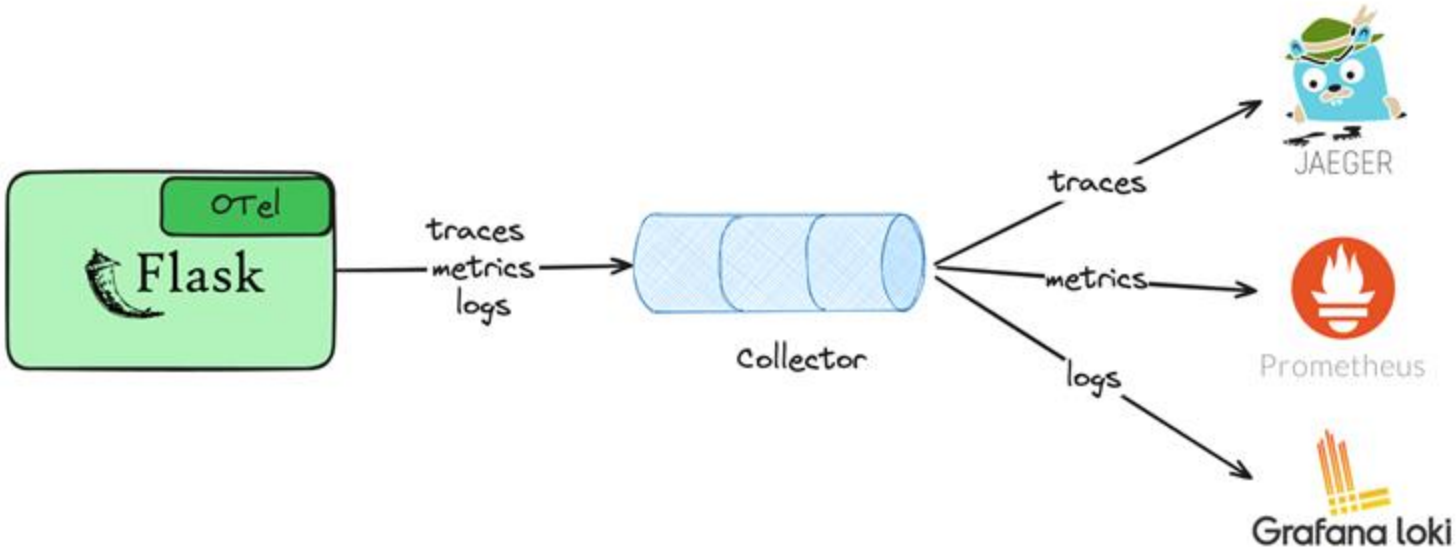
Пример запроса

```
❯ curl localhost:8001
hello-world
```

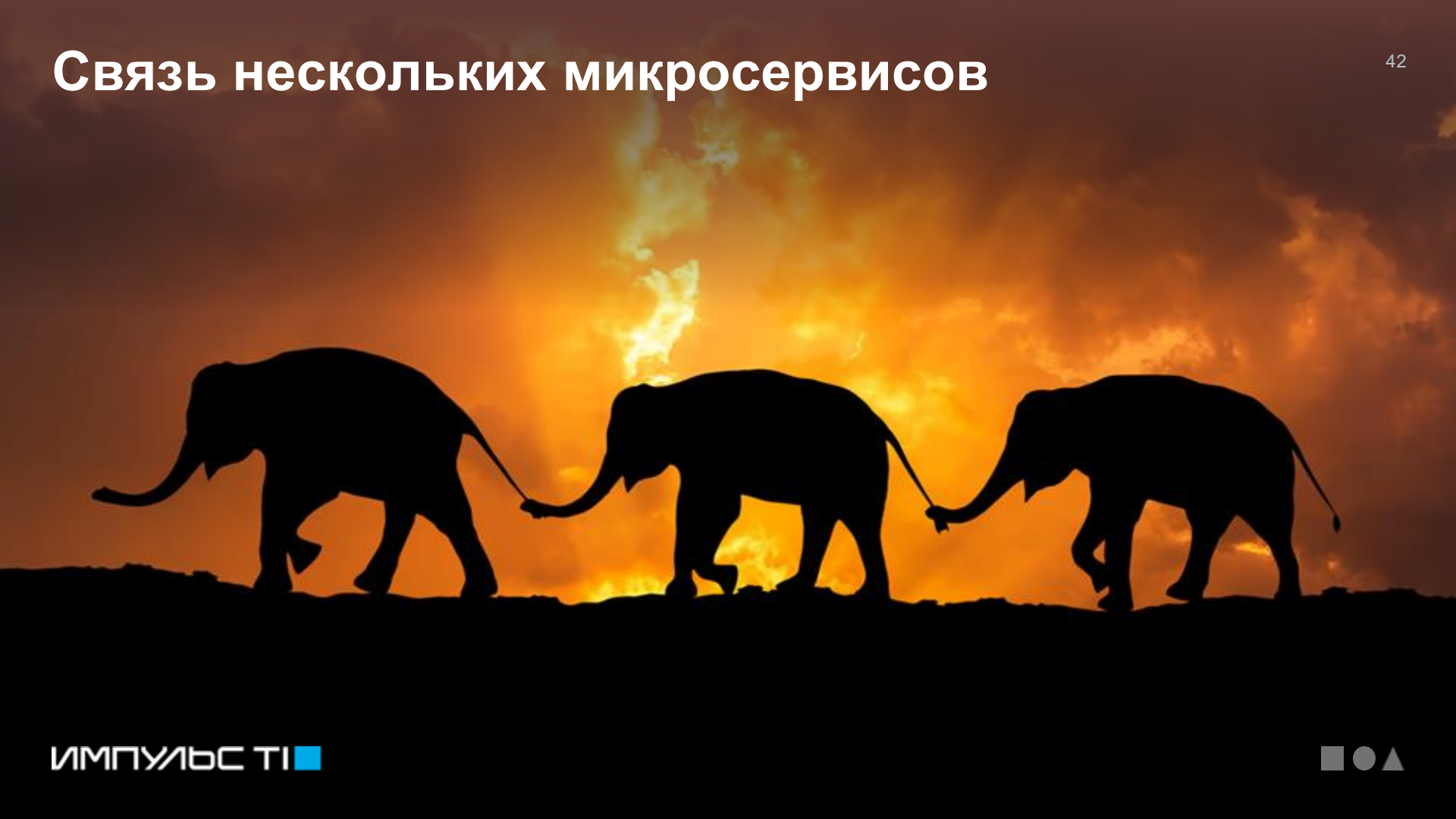
Пример спана в консоли


```
{
    "body": "hello-world",
    "severity_text": "INFO",
    "attributes": {
        "otelServiceName": "flask-app",
        "code.filepath": "./docker-tracing-demo/flask_app/app.py",
        "code.function": "index",
        "code.lineno": 11
    },
    "timestamp": "2024-08-24T18:52:11.966384Z",
    "trace_id": "0x7ea0f1afc744150ffc7bb451d1906cf0",
    "span_id": "0xf296a5fd627ea074",
}
```

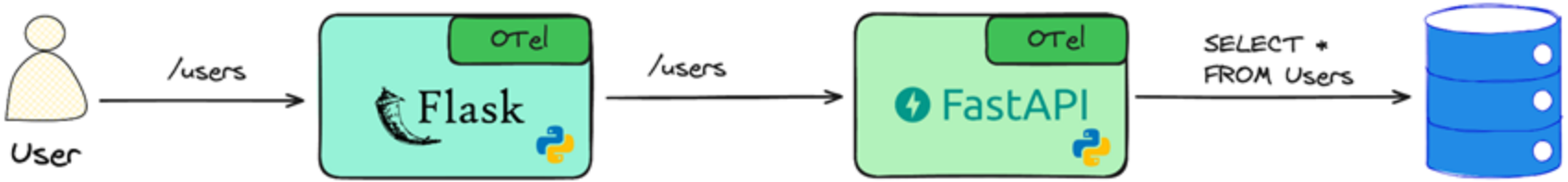
Отправка спанов в Collector



Связь нескольких микросервисов

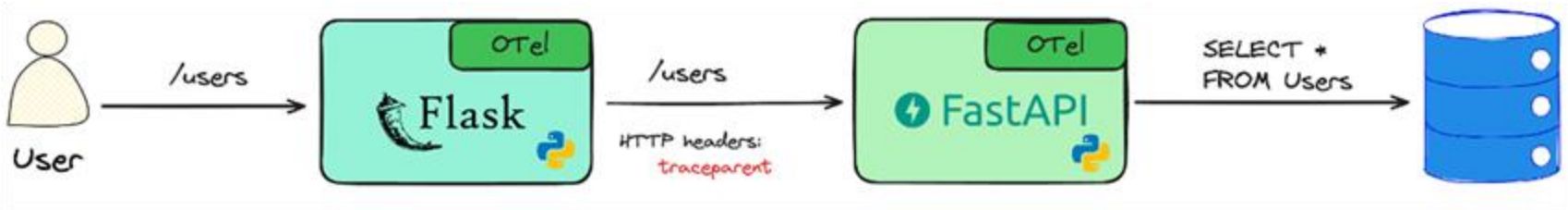


Как связать одним трейсом два сервиса?



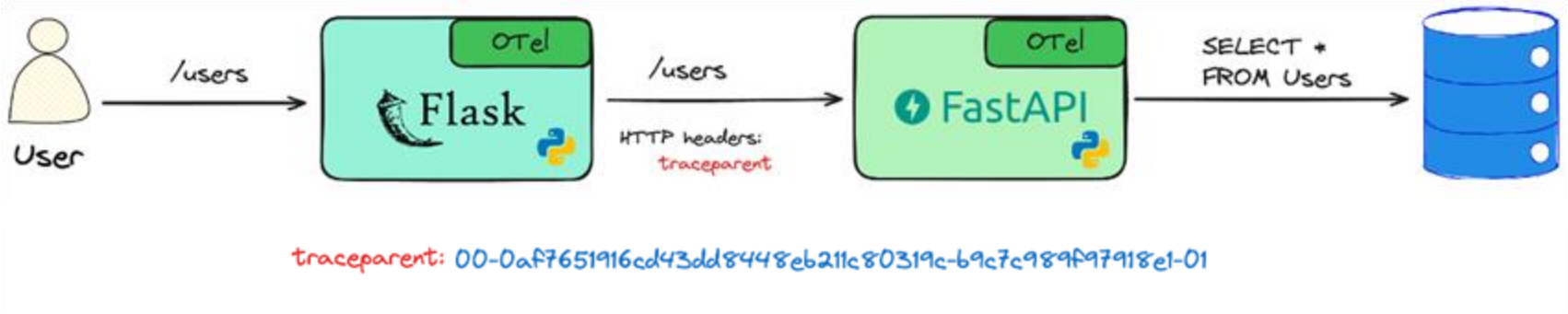
Как связать одним трейсом два сервиса?

Trace Context — передача контекста в HTTP заголовках



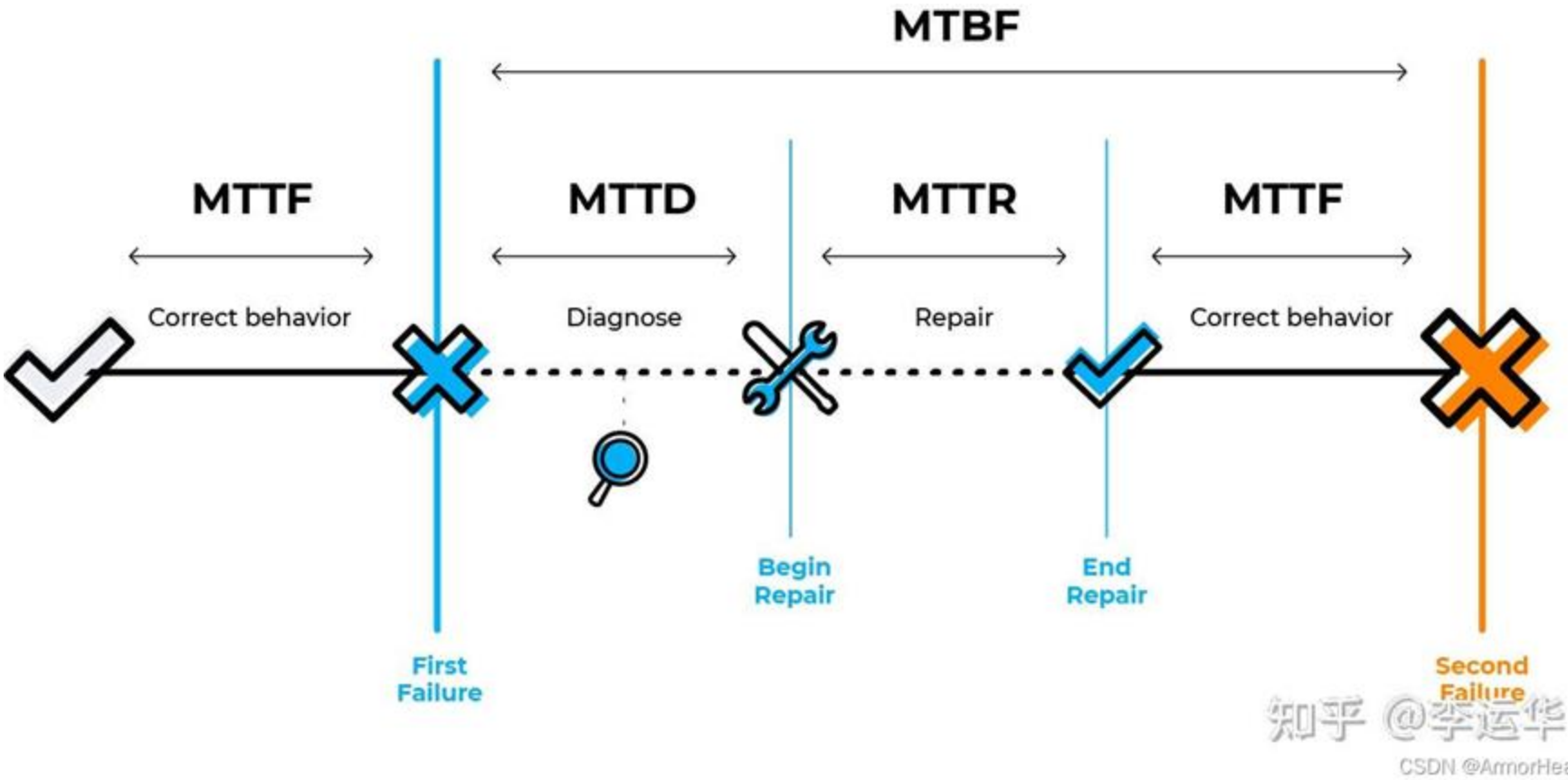
Как связать одним трейсом два сервиса?

Trace Context — передача контекста в HTTP заголовках

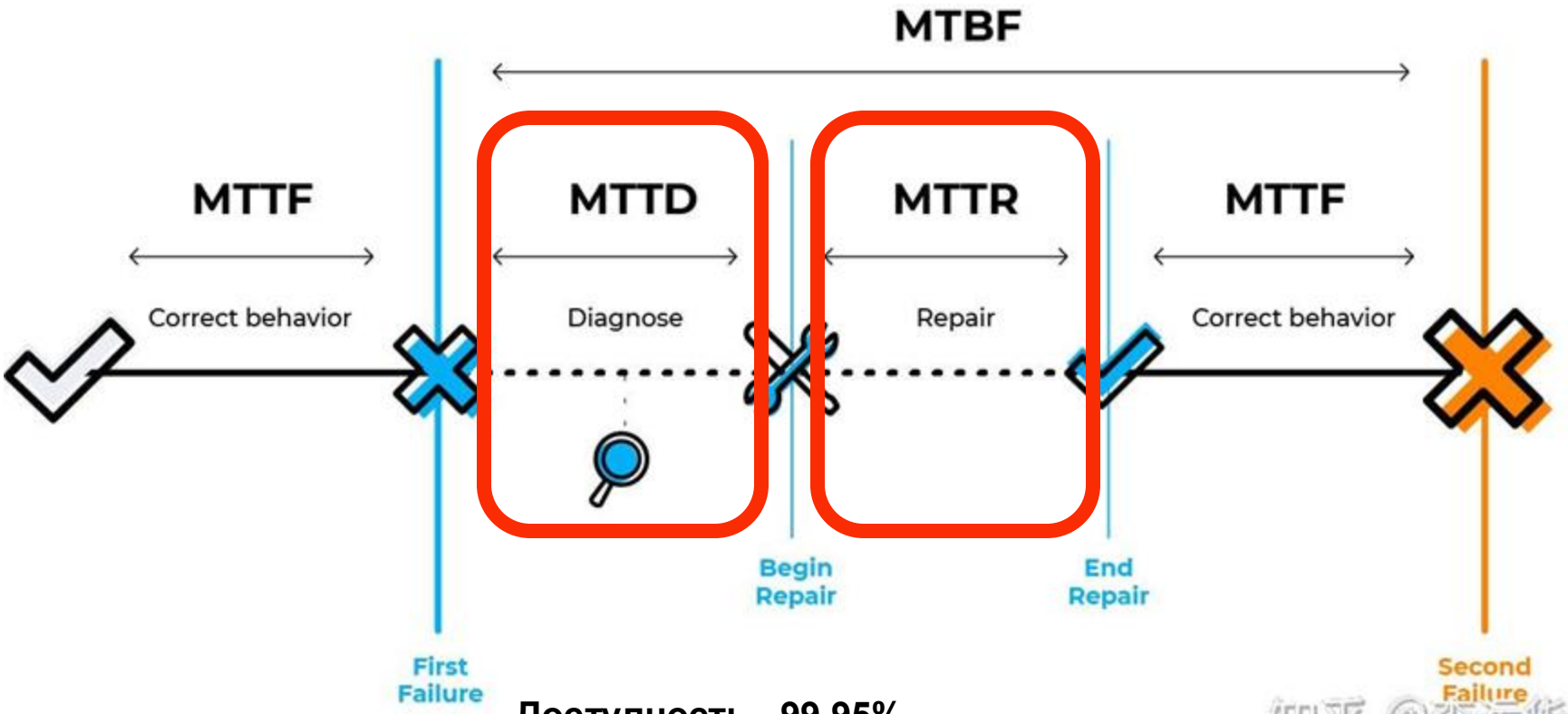


Подключил алерты ко всем сервисам +1

MTTD и MTTR



Найти и обезвредить

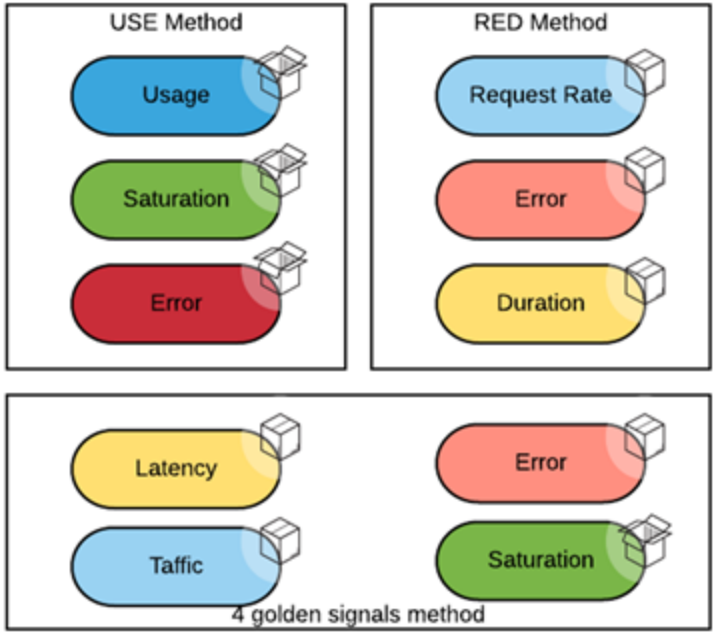


Доступность - 99,95%
Время простоя - 4ч 22мин

Что искать и как?

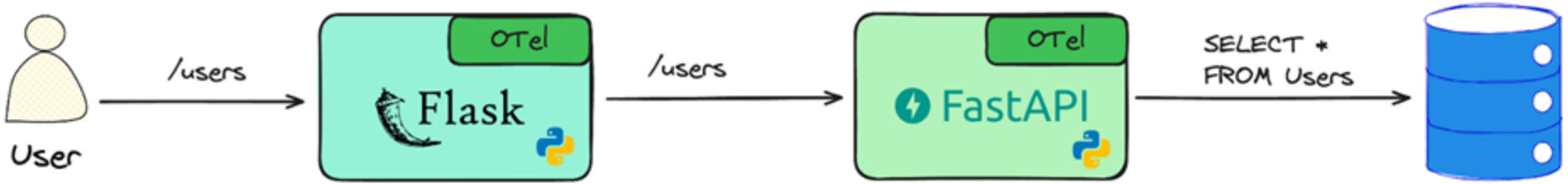
Техники SRE: LETS, STELA, RED и USE

- **Ошибки обработки (500-ки)**
- **Долгие запросы (задержка>1с)**



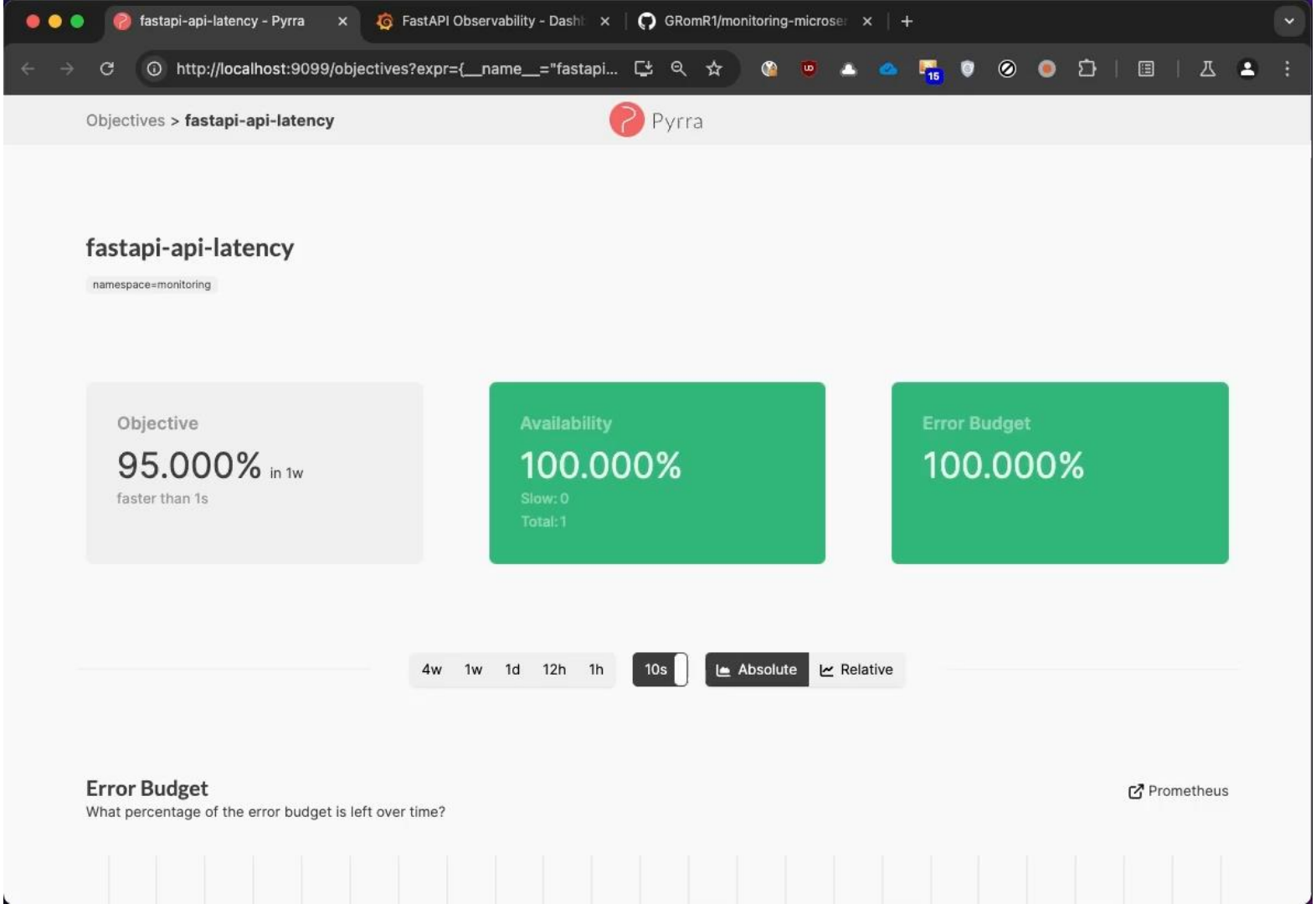
Пора искать ошибки

пример



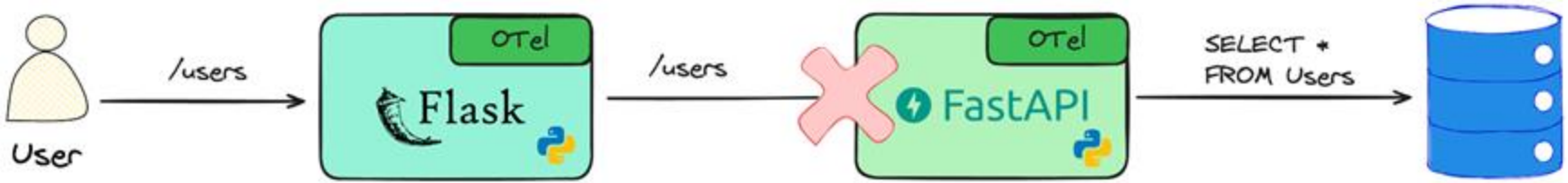
Пользователи жалуются, что иногда возникают ошибки
при запросе информации о других пользователях

Найти и обезвредить

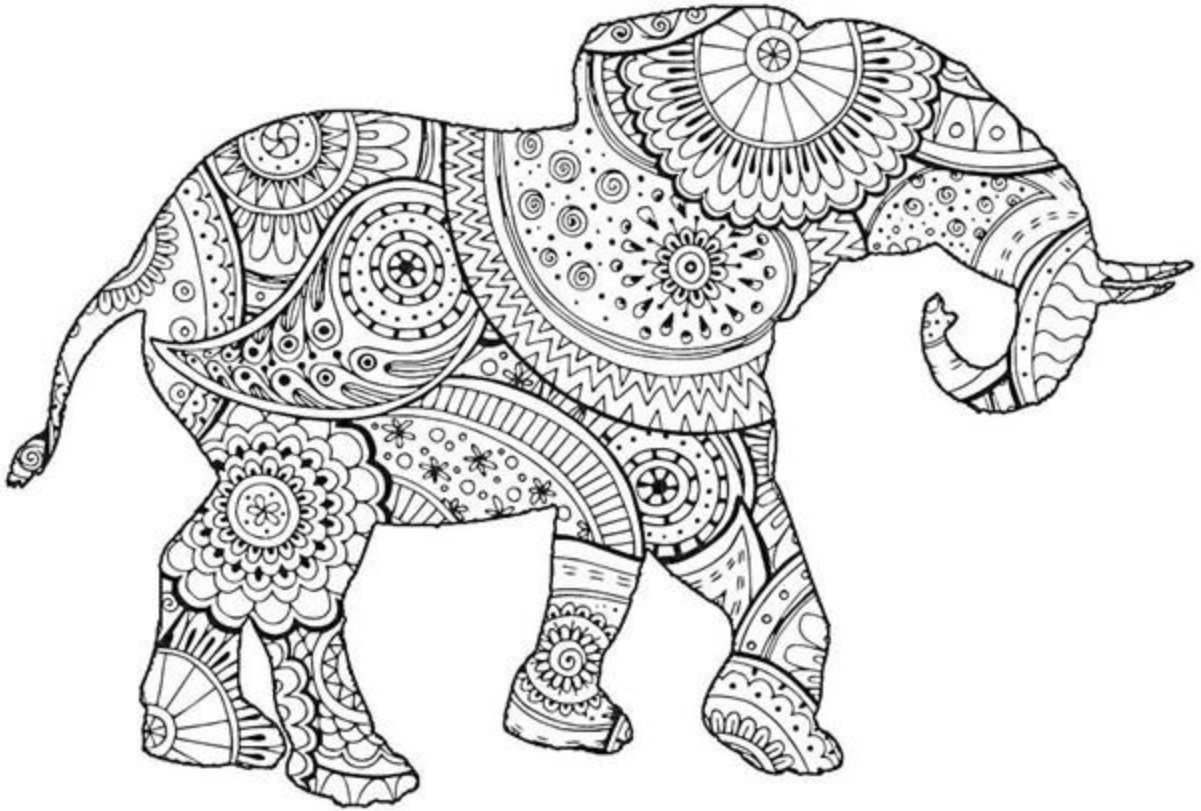

fastapi-api-latency - Pyrra
FastAPI Observability - Dash
GRomR1/monitoring-microse
http://localhost:9099/objectives?expr={__name__="fastapi...
Objectives > fastapi-api-latency
Pyrra
fastapi-api-latency
namespace=monitoring
Objective
95.000% in 1w
faster than 1s
Availability
100.000%
Slow: 0
Total: 1
Error Budget
100.000%
4w
1w
1d
12h
1h
10s
Absolute
Relative
Error Budget
What percentage of the error budget is left over time?
Prometheus

URL

Проблема локализована и устранена



Как попробовать самому?

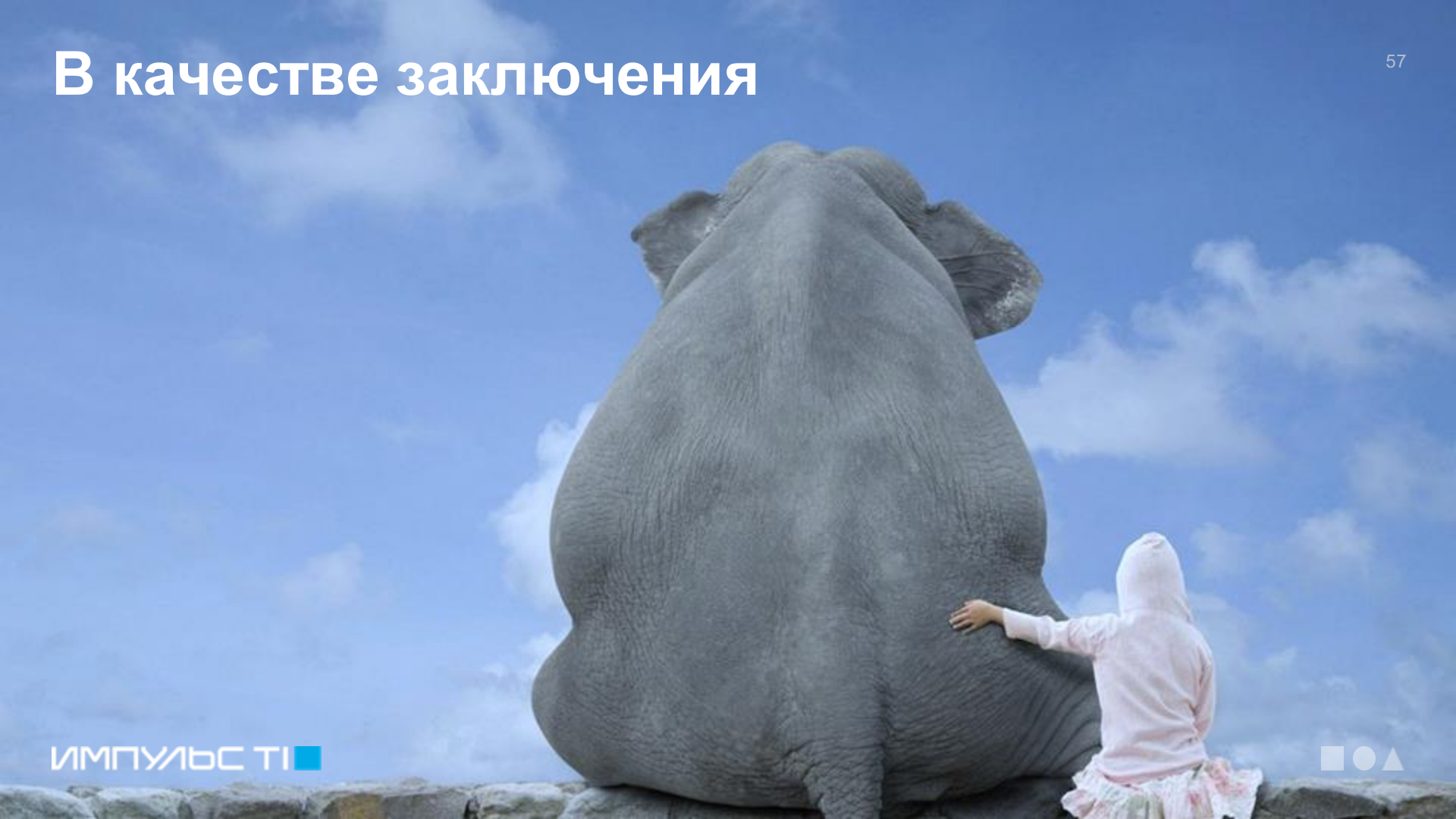


Посмотреть и попробовать

- Useful Links
- Logs-to-Trace & Trace-to-Logs
- **Tempo** for Traces
- **Prometheus** for Metrics
- **Loki** for Logs
- **OpenTelemetry** Collector for Processing
- **Grafana** & **Jaeger** for UI
- **Flask** & **FastAPI** applications
- **Auto**- & **Manual**-Instrumentation
- **Pyrra** for calculating SLOs, Error Budget
- **AlertManager** & **Karma** for alerts
- **Beyla** for eBPF instrumentation
- **Pyroscope** as UI for Profiling

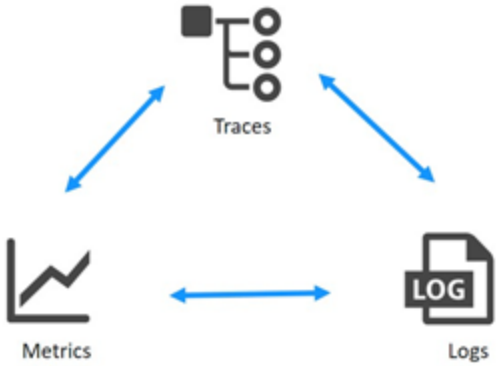
@GRomR1/monitoring-microservices-demo

В качестве заключения



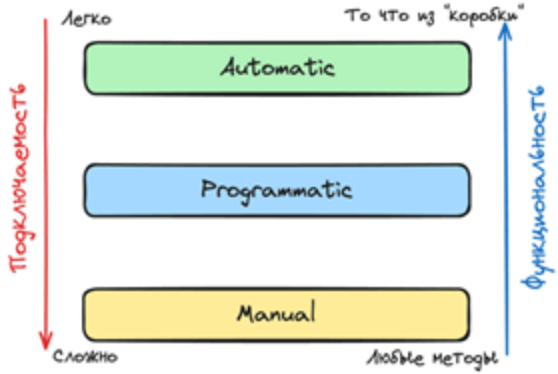
Выводы

Три сигнала



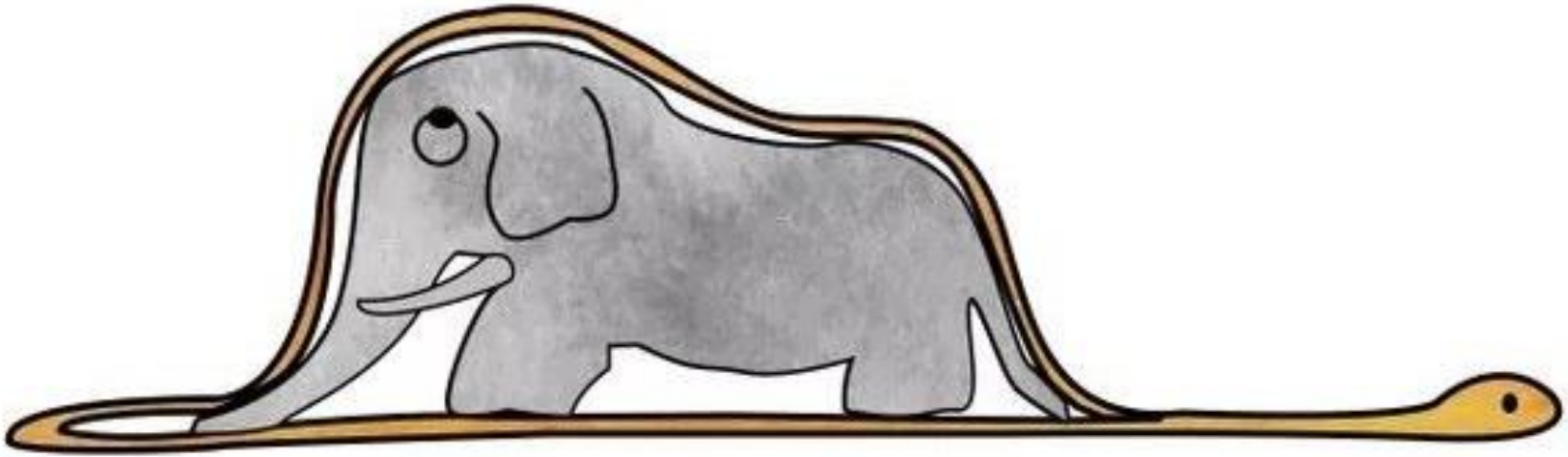
Траблшутинг

Инструментация



«Тогда я нарисовал удава изнутри…» +I

«Маленький принц». А. де Сент-Экзюпери.



Спасибо

Руслан Гайнанов
главный инженер DevOps,
T1 Иннотех
@gainanovrus