Зачем нам Наблюдаемость IT-продуктов и как ее достичь

Бочаров Филипп

О себе

Бочаров Филипп

Руководитель проектов по разработке в МТС

Занимаюсь разработкой платформы Наблюдаемости. Помогаю продуктовым командам сделать работу сложных распределенных систем понятной и прозрачной.

Цели доклада

1. Какую проблему решает наблюдаемость?
2. Как ее достичь (в теории и на практике)?
3. Как наблюдаемость поможет лично Вам?

Наблюдаемость - возможность задавать вопросы о работе системы

Проблемы экосистем на примере МТС

Экосистема МТС

- **400+ цифровых продуктов**: телемедицина, Smart Farming, Big Data, умный дом, внутренние платформы …
- **Гетерогенность IT-ландшафта**: продукты используют разные языки, платформы и фреймворки
- **Экосистема продуктов:** продукты взаимодействуют между собой, увеличивая ценность предложения

Проблемы роста экосистемы

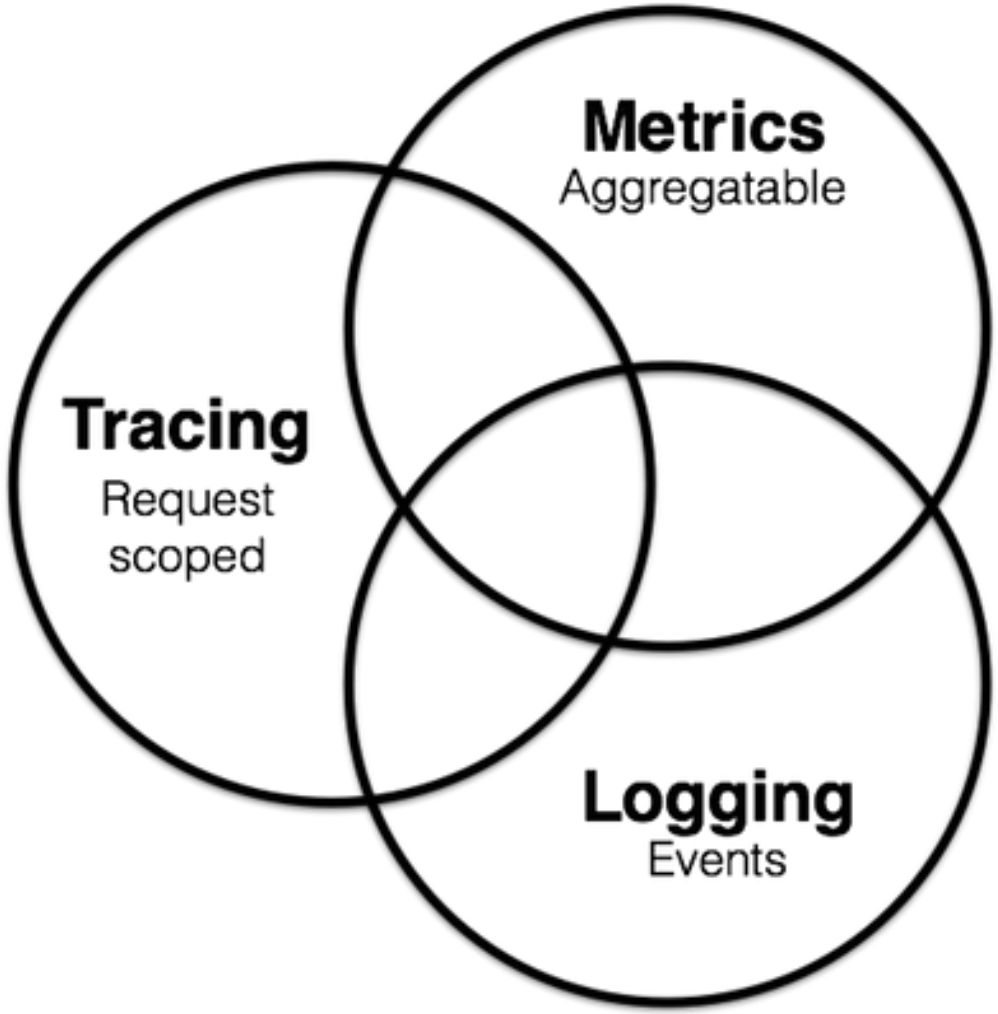
С ростом количества продуктов в экосистеме:

- Усложняется диагностика интеграционных проблем;
- Растет стоимость контроля качества;
- Растет время вывода продукта на рынок.

Как достичь наблюдаемости и снизить сложность?

Телеметрия

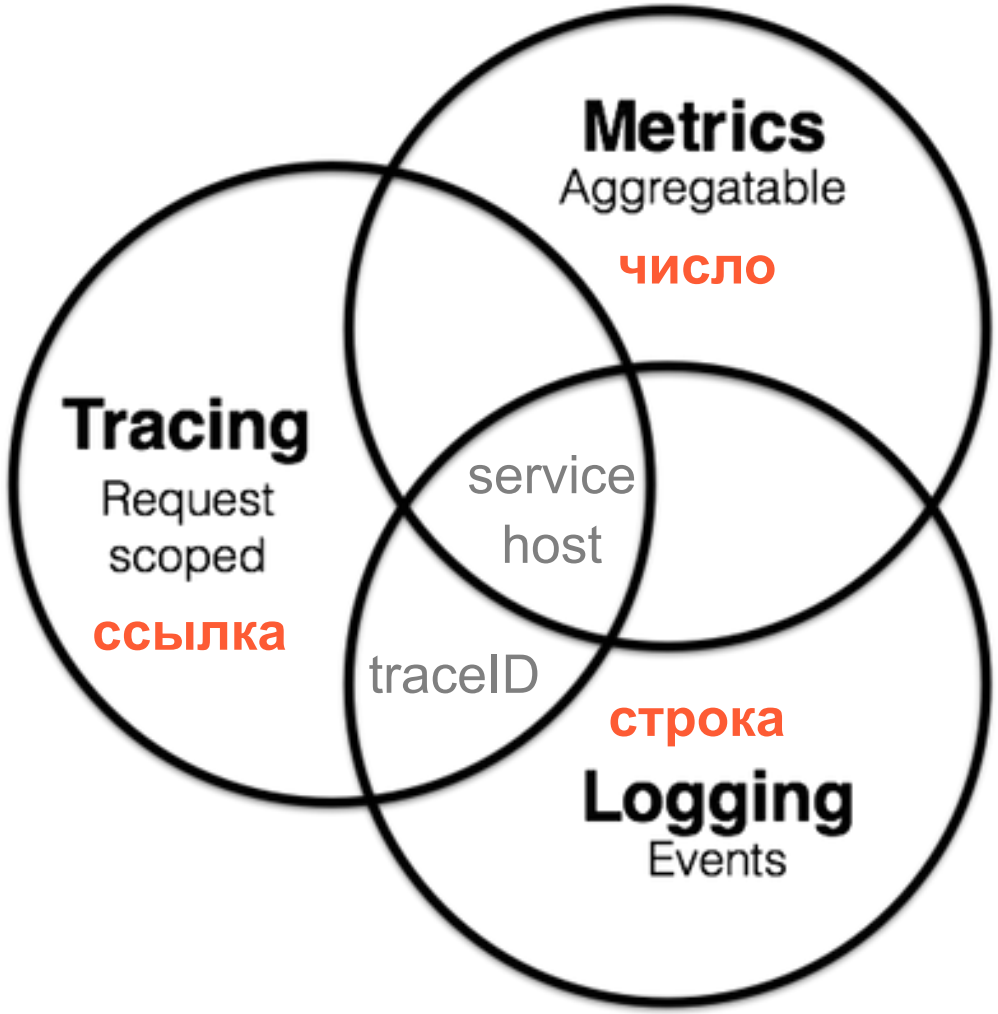
Единый связный массив данных,
позволяющий проводить анализ



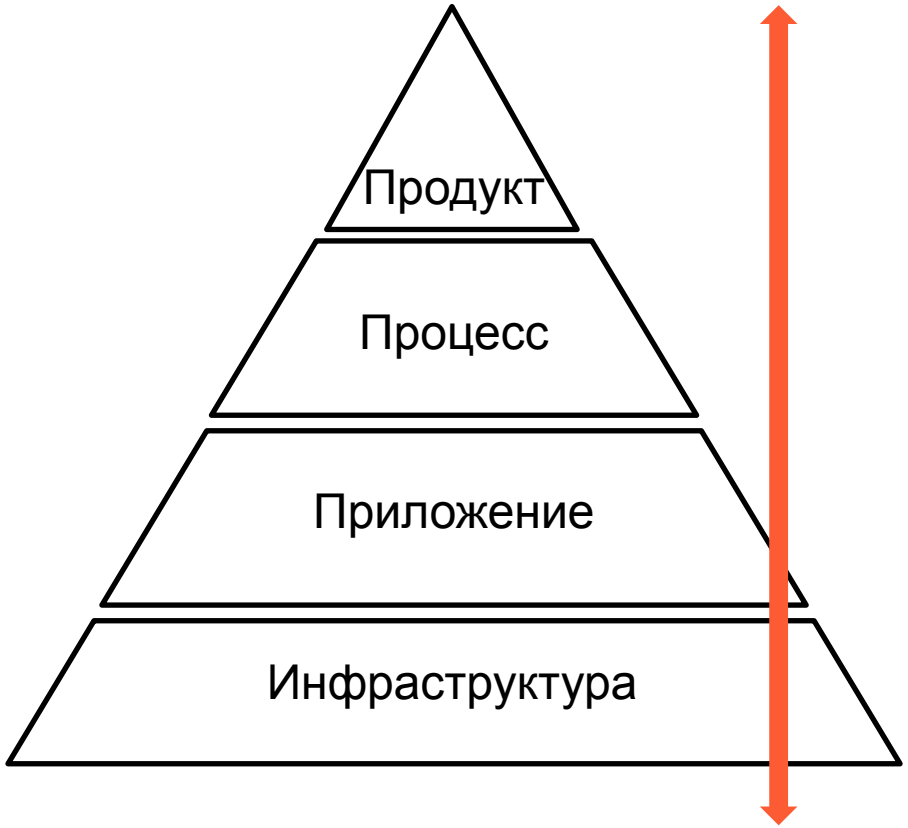
Телеметрия

Единый связный массив данных,
позволяющий проводить анализ

Time series, structured data



Уровни ландшафта



Количество продаж снизилось,

потому, что **процесс продажи** завершался с ошибкой,

из-за сбоя в **сервисе оплаты,**

так как закончилось **место на диске**

Способы обеспечения наблюдаемости

Инвазивные

выше точность

выше влияние

- Observability as a code (Jaeger)
- APM агент

VS

Неинвазивные

ниже точность

ниже влияние

- Сбор и анализ сетевого трафика
- Аналитика по базе данных
- Машинный анализ логов

Коробочное решение или набор инструментов

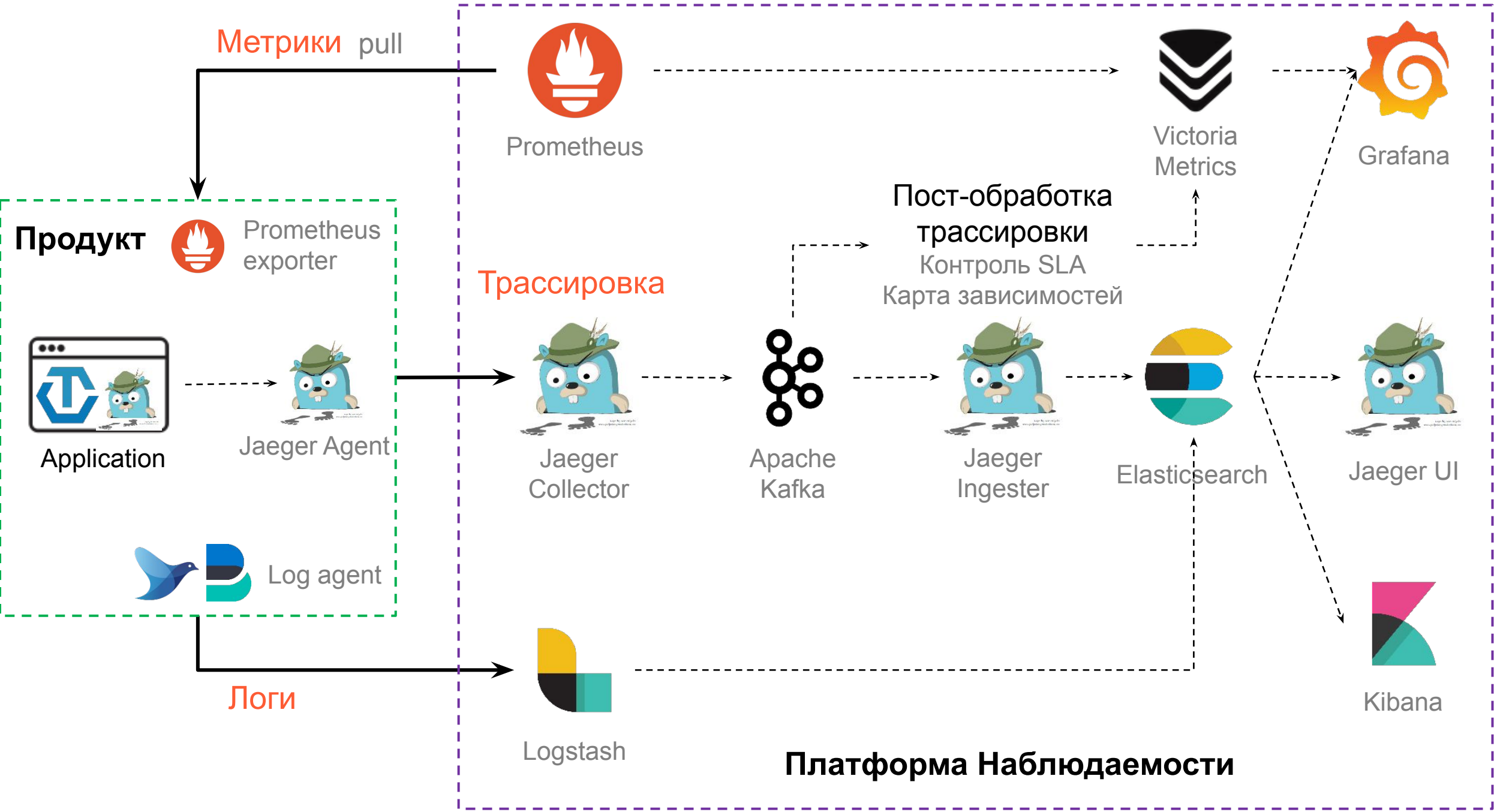
Критерии:

- Стоимость
- Кастомизация
- Vendor lock-in
- Готовность развивать экспертизу
- On-premise vs cloud

Платформа Наблюдаемости

Набор инструментов для решения задач наблюдаемости в экосистеме МТС

- Цель: наблюдаемость всей экосистемы
- Platform as a Service
- Единое хранилище данных
- Операционные задачи: диагностика
- Аналитические задачи: инсайты


Метрики pull
Prometheus
Victoria Metrics
Grafana
Продукт
Prometheus exporter
Пост-обработка трассировки
Контроль SLA
Карта зависимостей
Трассировка
Application
Jaeger Agent
Jaeger Collector
Apache Kafka
Jaeger Ingester
Elasticsearch
Jaeger UI
Log agent
Логи
Logstash
Kibana
Платформа Наблюдаемости

Наблюдаемость нужна вам… кем бы вы ни были

- Бизнес / владелец продукта
- DevOps / SRE / сотрудник техподдержки
- Разработчик
- Тестировщик / QA / специалист по качеству
- Архитектор

Разработчикам

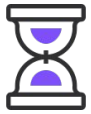
Схема работы

Client

Жалоба

Business

Требования



Время

Developer

Продукт

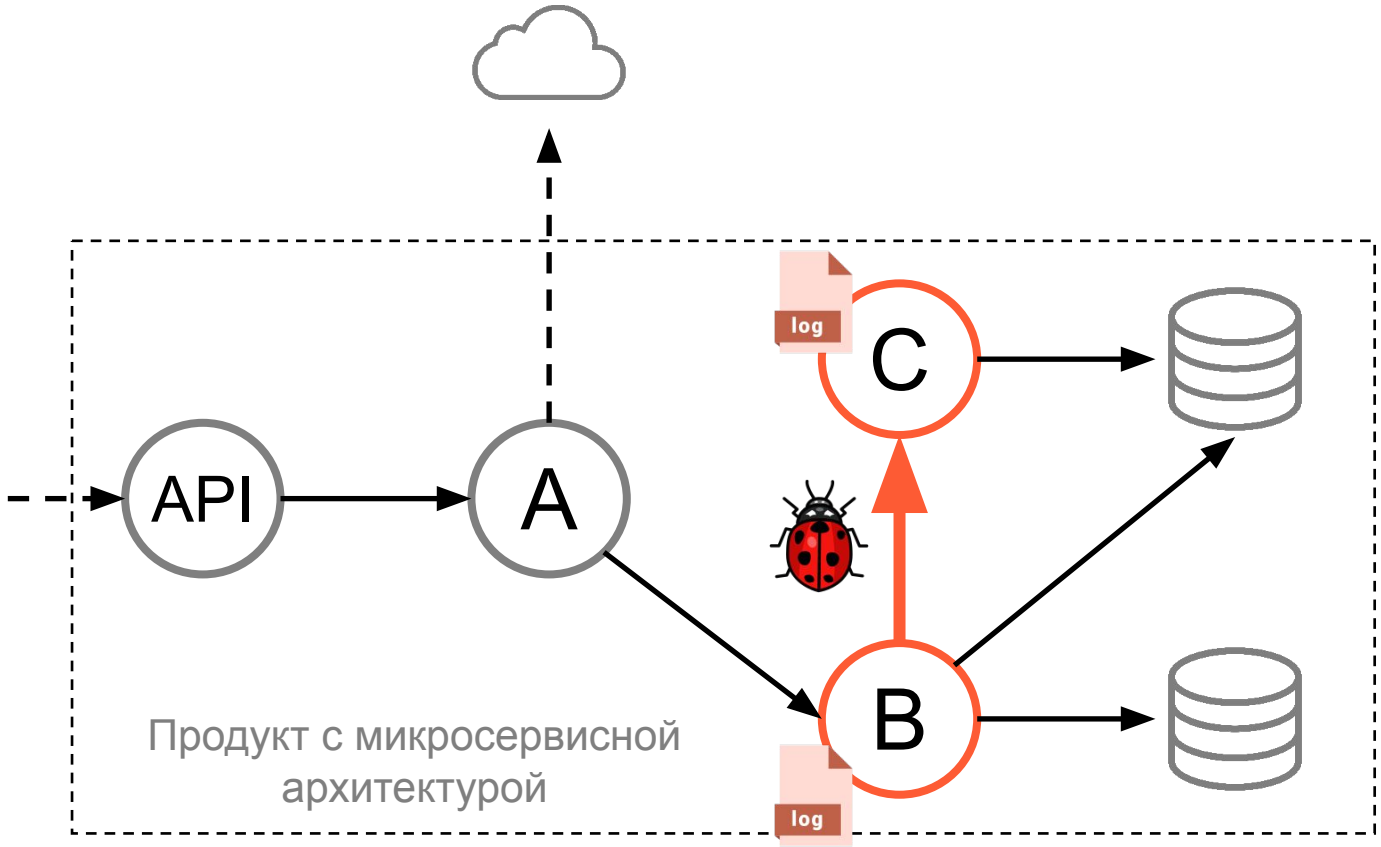
SRE

QA

Баг

- Вся телеметрия по багу
- Общие инструменты с SRE и QA
- Быстрый цикл отладки

Отладка распределенной системы



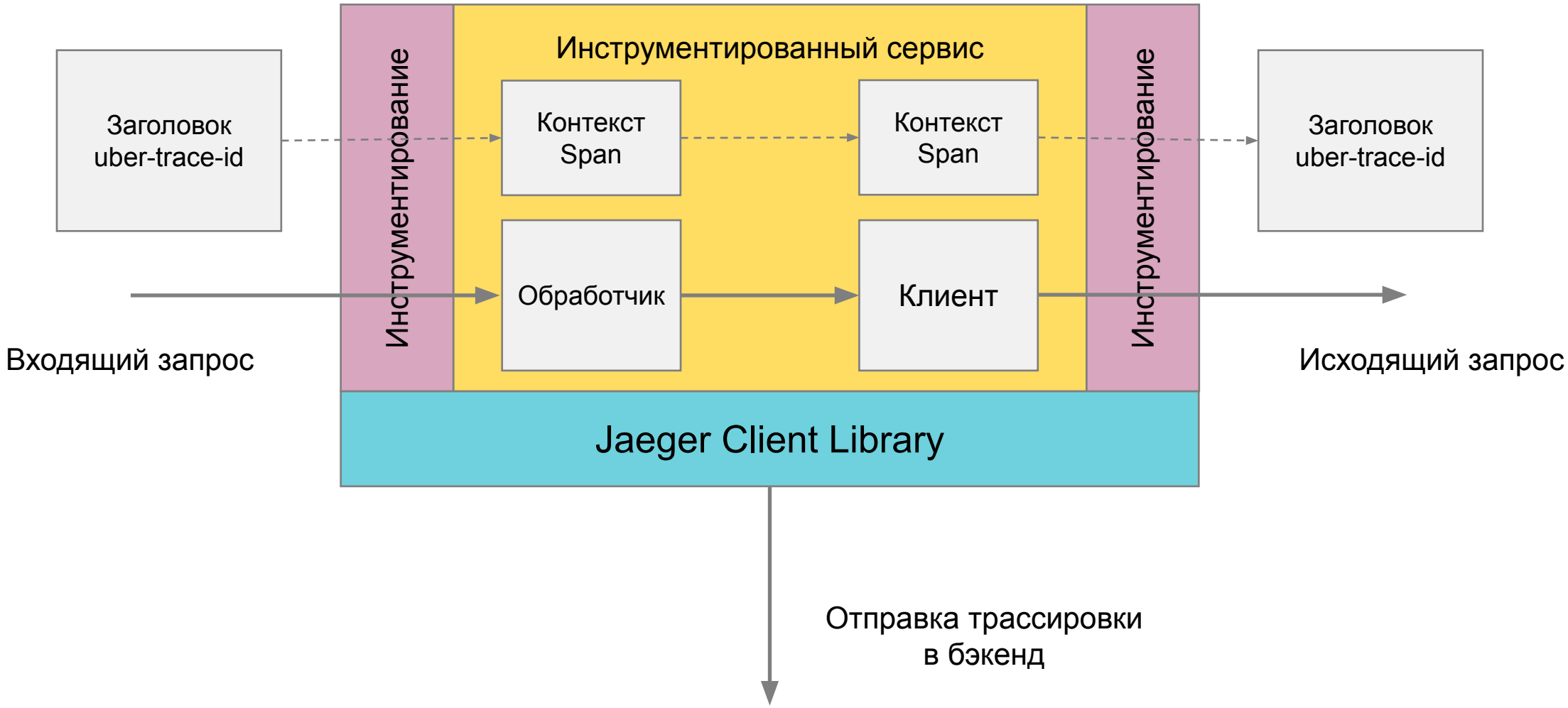
- На каком этапе произошла ошибка?
- В чем корневая причина ошибки?
- Не связана ли она с железом?

Распределенная трассировка

Причинно-следственная связь событий (дерево процесса)

Trace:

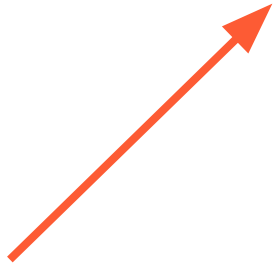
Схема работы



Переход от трейса к логам

Обогащение лога контекстом трассировки позволяет переходить от локализации к диагностике – от трассировки, к логам.

Time	Message	SpanID	TraceID	LogLevel
August 28th 2019, 14:10:41.525	2019-08-28 14:10:41,525[64] ERROR ArchiveSendingProcessor - ArchiveQueueId 3220: end with error.	eb798b4a79a3ce	10846ed40516dd5	ERROR

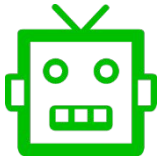


Логи привязаны к контексту
(Request scoped logs)

Структурное логирование и диагностика

Логи должны быть понятны и человеку и машине.

- Снижает требования к системе хранения
- Позволяет группировать события по типу
- Позволяет проводить аналитику и строить визуализации

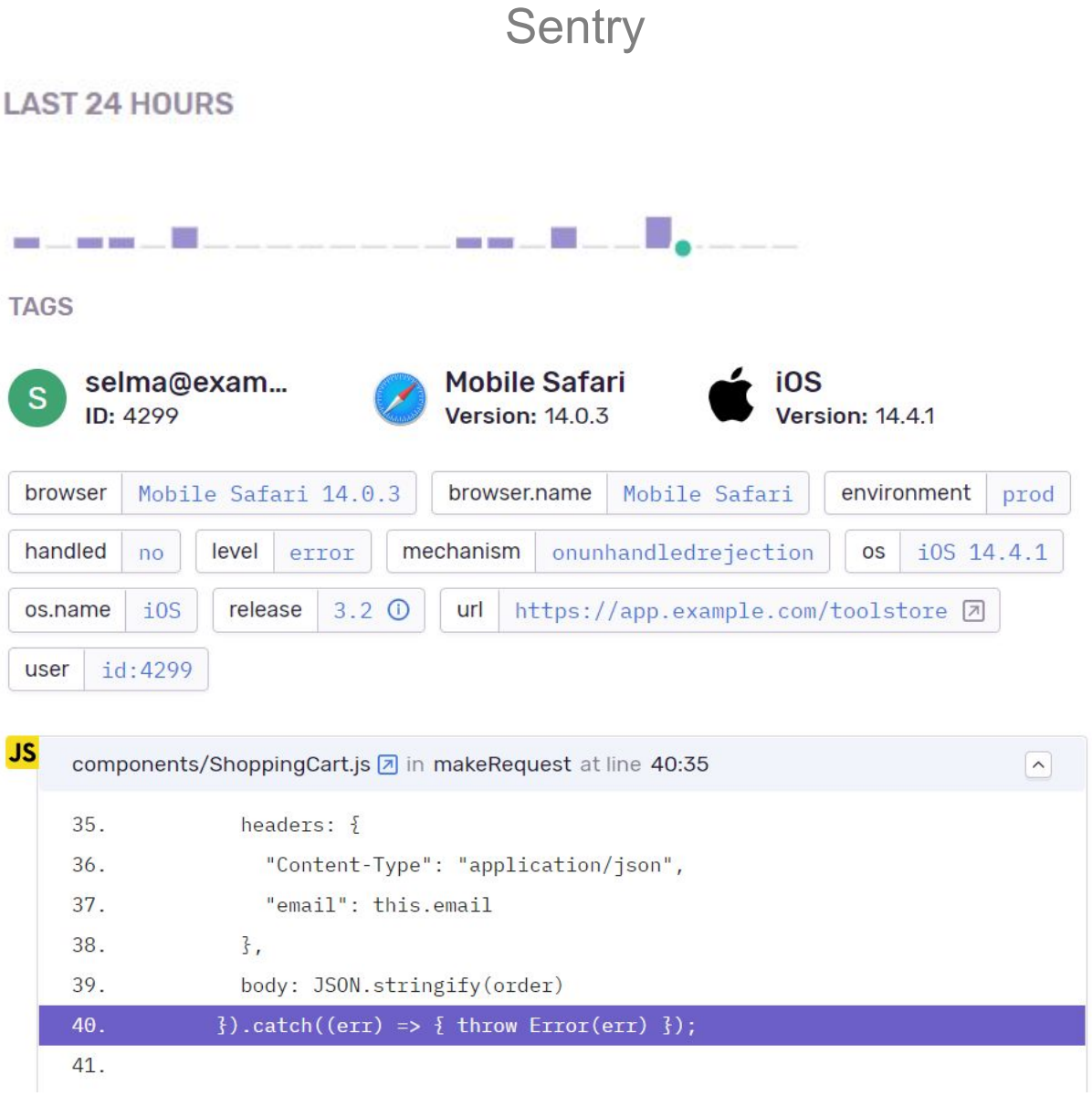


```
log("User {username} logged in from {ip_address}", username, ipAddress)
{
  "time": "2016-05-27T13:02:11.888",
  "template": "User {username} logged in from {ip_address}",
  "username": "alice",
  "ip_address": "123.45.67.89"
}
```

Как не пропустить важное событие

Error tracking - группировка ошибок одного типа и обогащение контекстом.

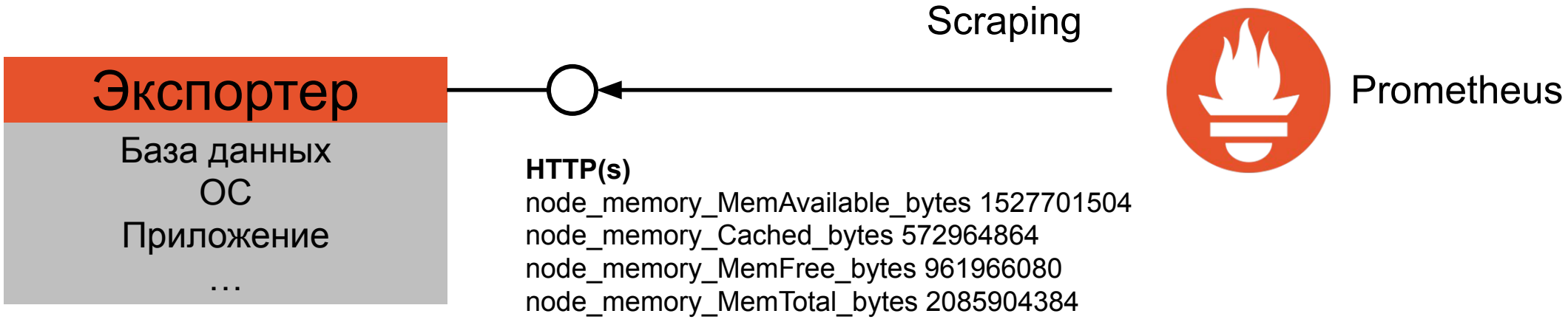
Это не замена, но отличное дополнение телеметрии!



Связано ли с железом?

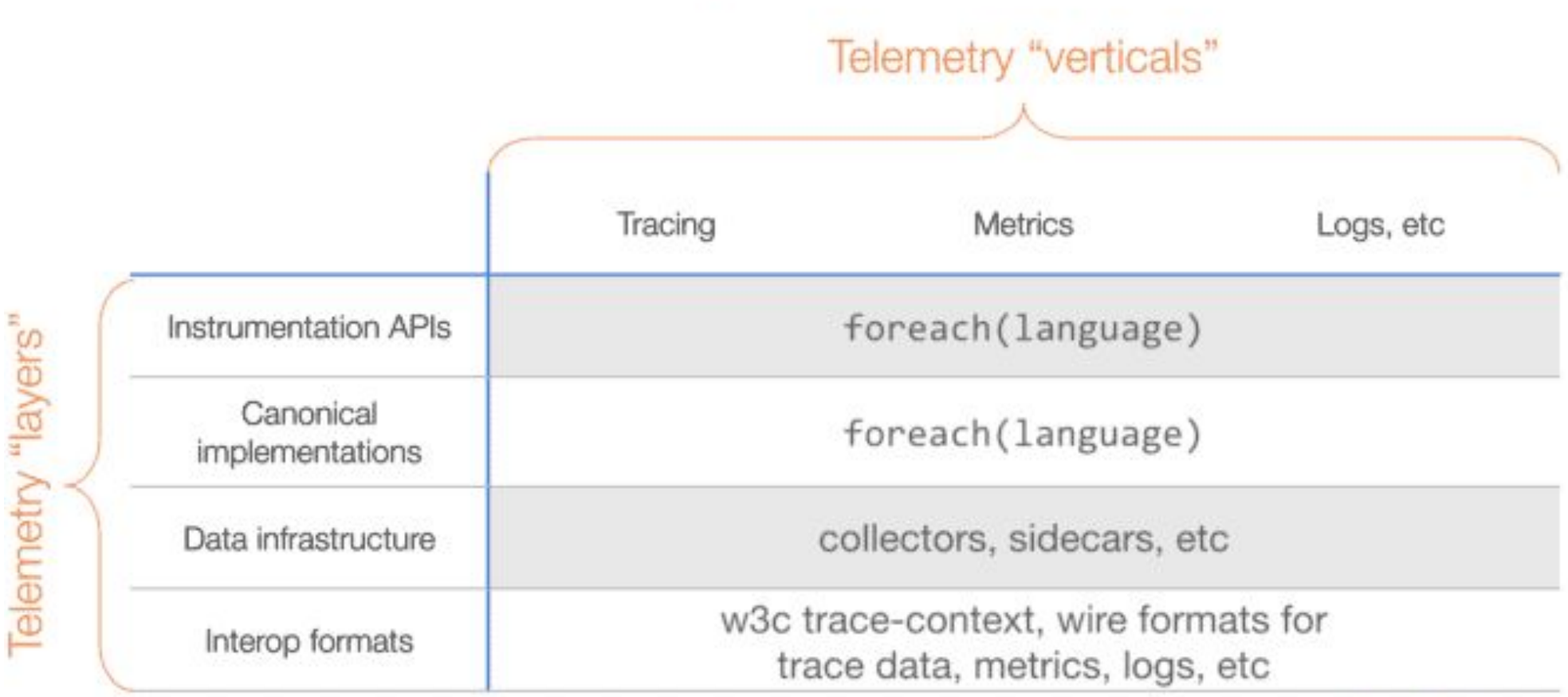
Что хотим собирать:

- Метрики железа (CPU, RAM, IO)
- Метрики операционных систем (windows / unix)
- Метрики сторонних компонент (базы данных, очереди, веб-сервера ..)
- Метрики приложения

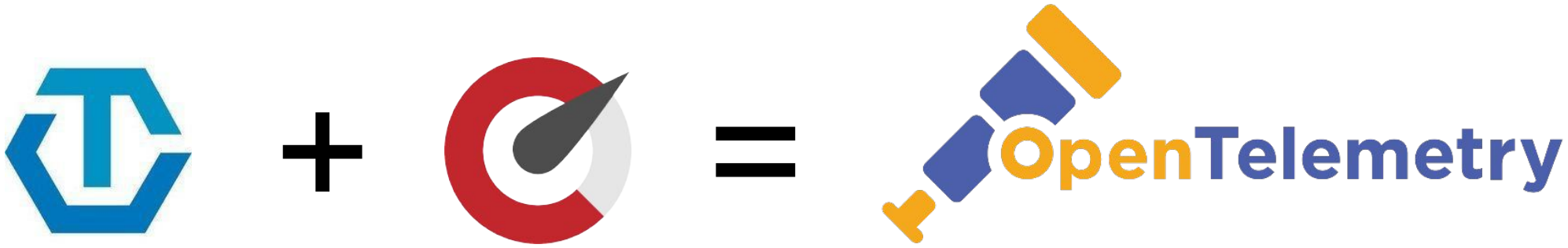


Фреймворк для настройки наблюдаемости

Observability as a code - возможность для разработчика встроить наблюдаемость в код и обеспечить себя и коллег телеметрией!



OpenTelemetry



OpenTracing + OpenCensus = OpenTelemetry

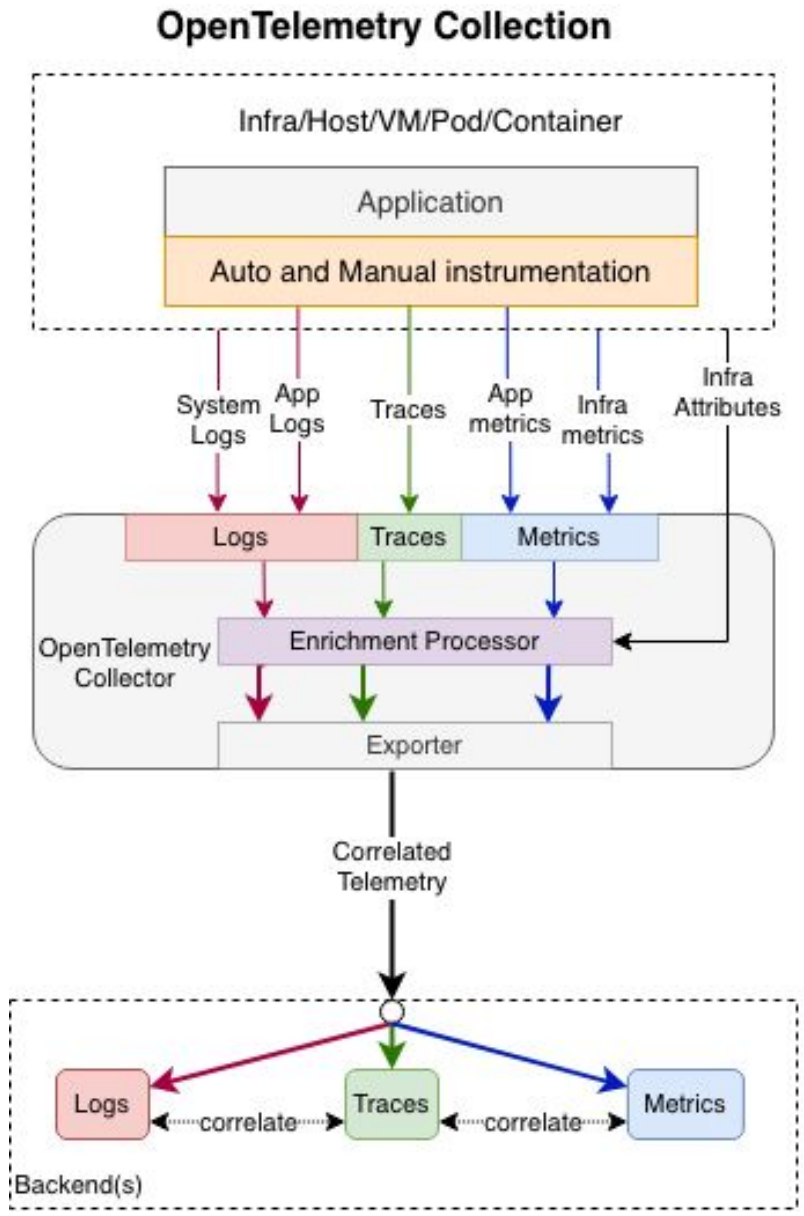
Трассировка | Трассировка и метрики | Трассировка, метрики, логи

OpenTelemetry

Швейцарский нож в области наблюдаемости.
Набор API для работы с трейсами, метриками и логами.

Преимущества:

- Поддержка множества языков и платформ
- Поддержка разных систем хранения
- Широко принят вендорами (Elasticsearch APM, NewRelic, …)
- Совместимость с OpenTracing и OpenCencus
- Одна инфраструктура для всех типов данных



Для бизнеса

Взаимодействие с клиентом

Деньги

Фидбек

Business

Client

Сервис

SLA / качество

- Ожидает высокого качества
- Жаждет новых возможностей
- Требует низкой цены

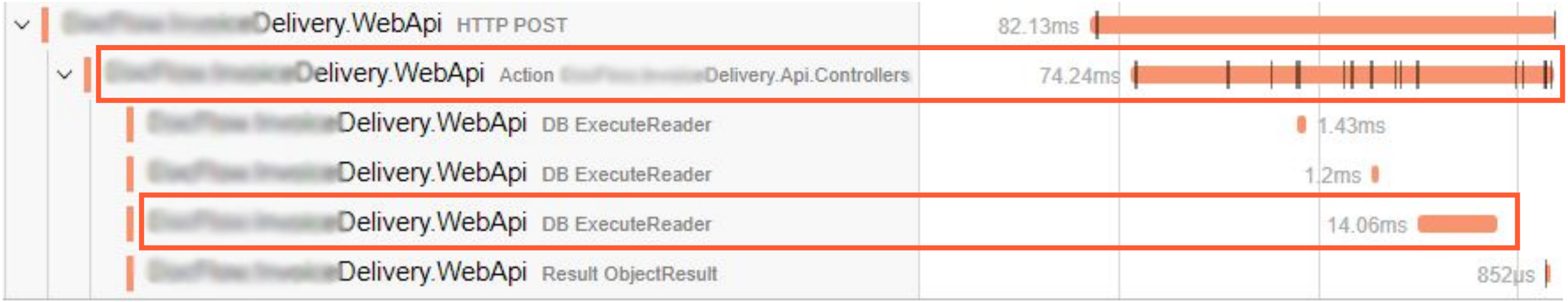
Модель качества продукта

Качество системы
ISO/IEC 25010:2011
ГОСТ Р ИСО/МЭК 25010-2015

Функциональная пригодность	Уровень производительности	Удобство использования	...	Защищенность
• Функциональная корректность • Полнота • Целесообразность	• Временны́е характеристики • Использование ресурсов • Уровень производительности			

Производные APM метрики

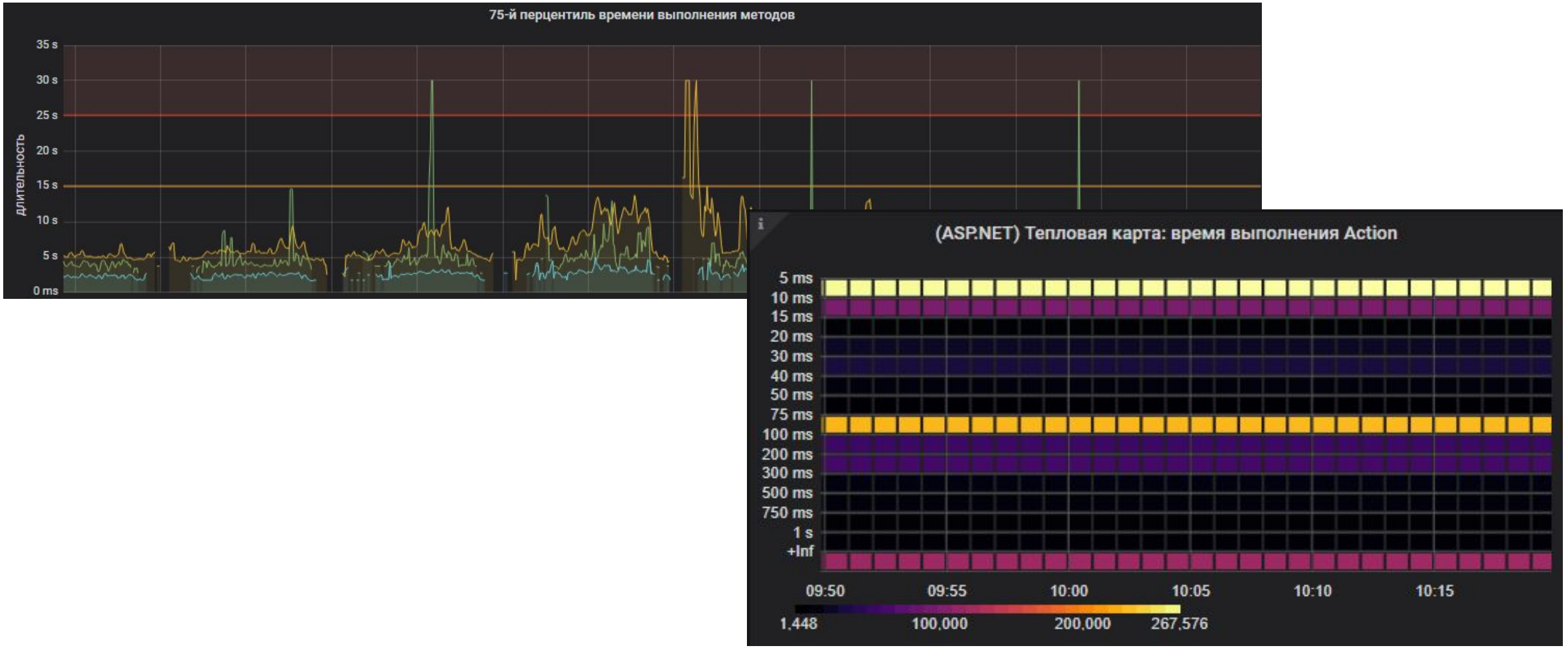
Длительность и ошибки каждого Action фиксируется в трассировке



Агрегация трассировки дает новые типы метрик

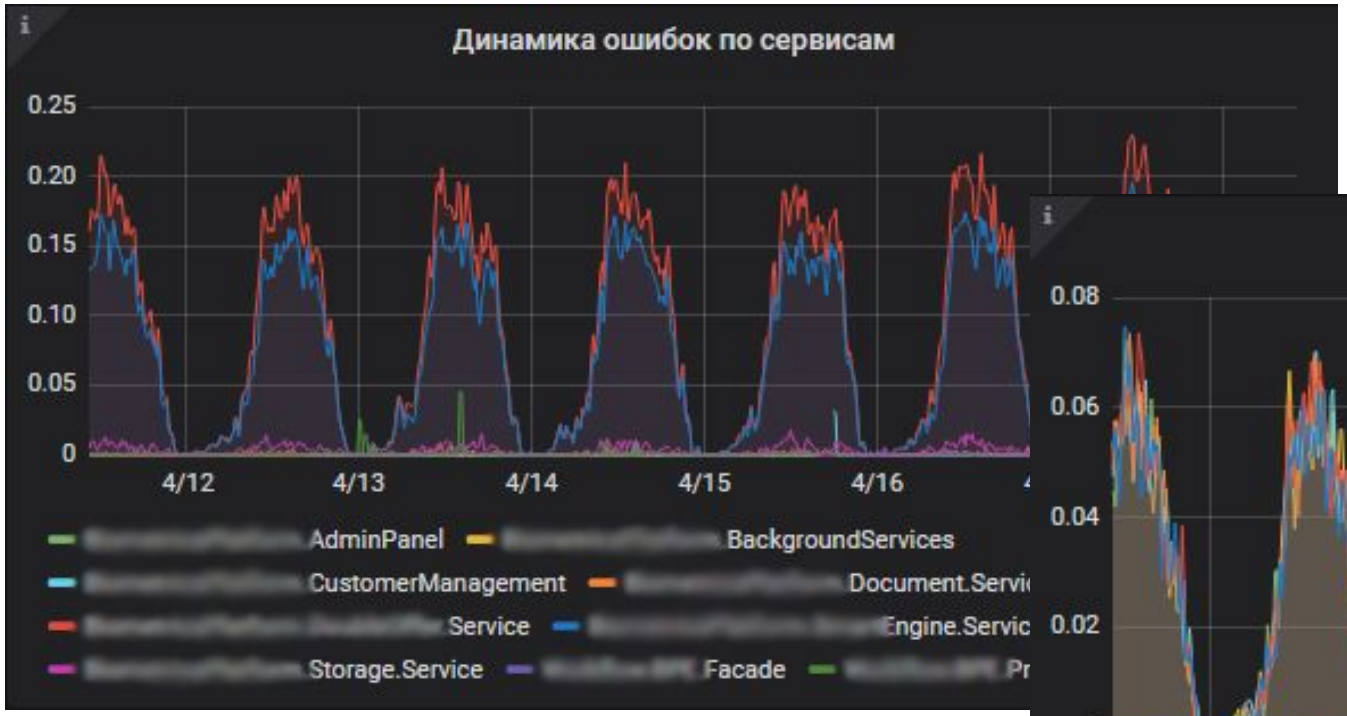
Дашборд “временные характеристики”

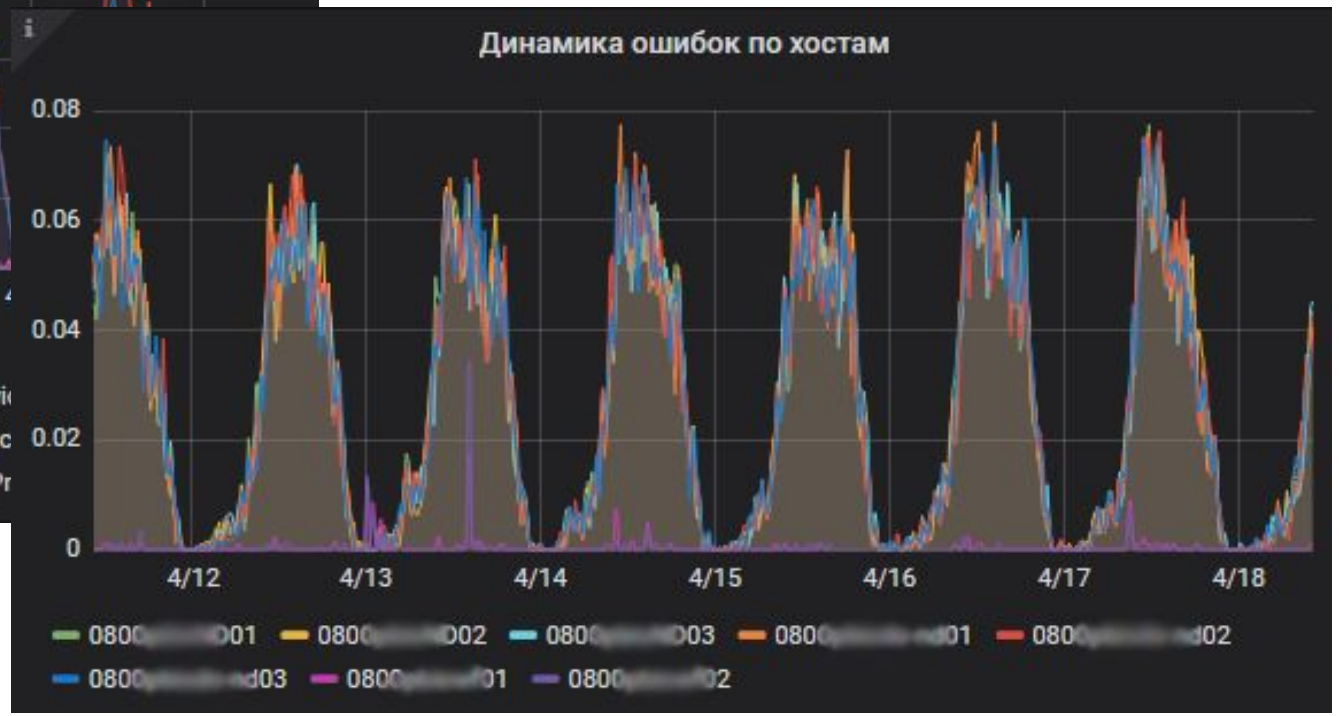
Тепловая карта и перцентиль длительности выполнения методов



Дашборд “функциональная корректность”

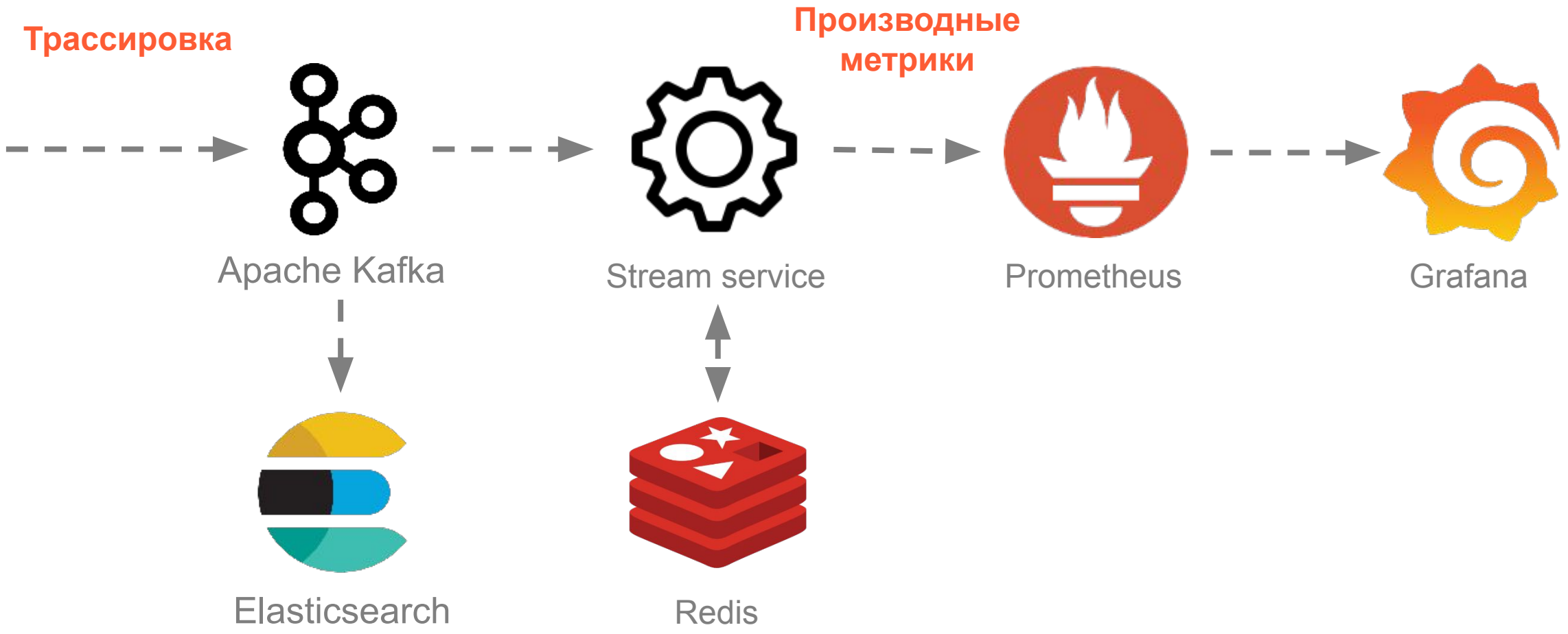
Количество ошибок с разрезе сервиса и хоста.





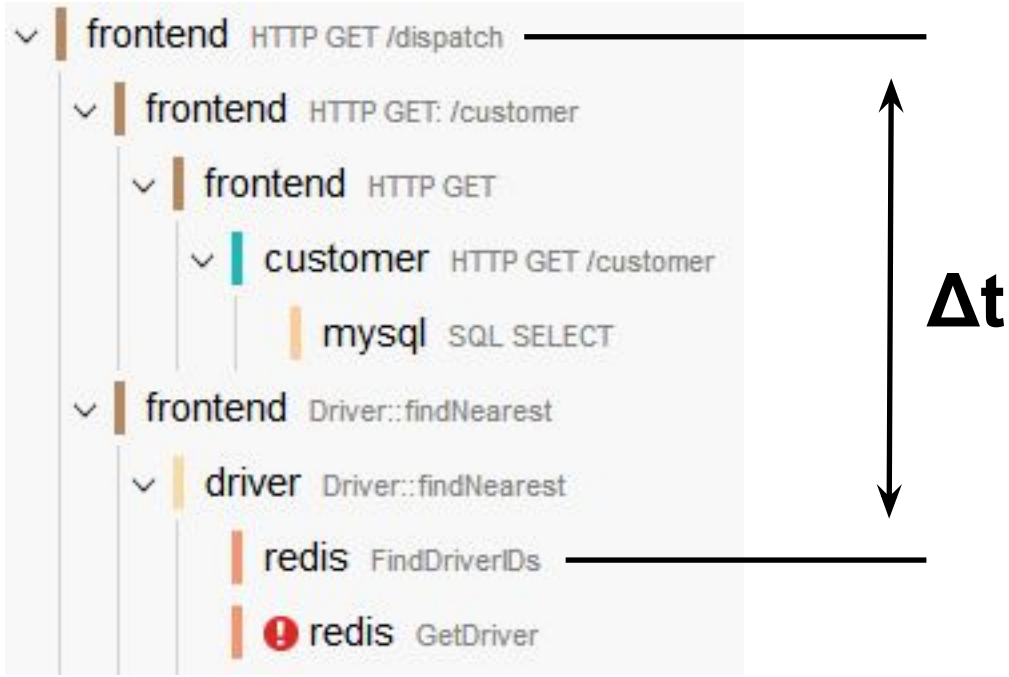
Пост обработка спанов

Постобработка позволяет считать производные метрики из трассировки



Контроль SLA процессов

Измеряем время от нажатия кнопки “Купить”, до отправки чека на почту.



Контроль времени между произвольными точками:

- Требует промежуточного хранилища для сессии
- Порядок событий не гарантирован
- Требует решения проблемы синхронизации времени на хостах

Real user monitoring

Довольны ли конечные пользователи?

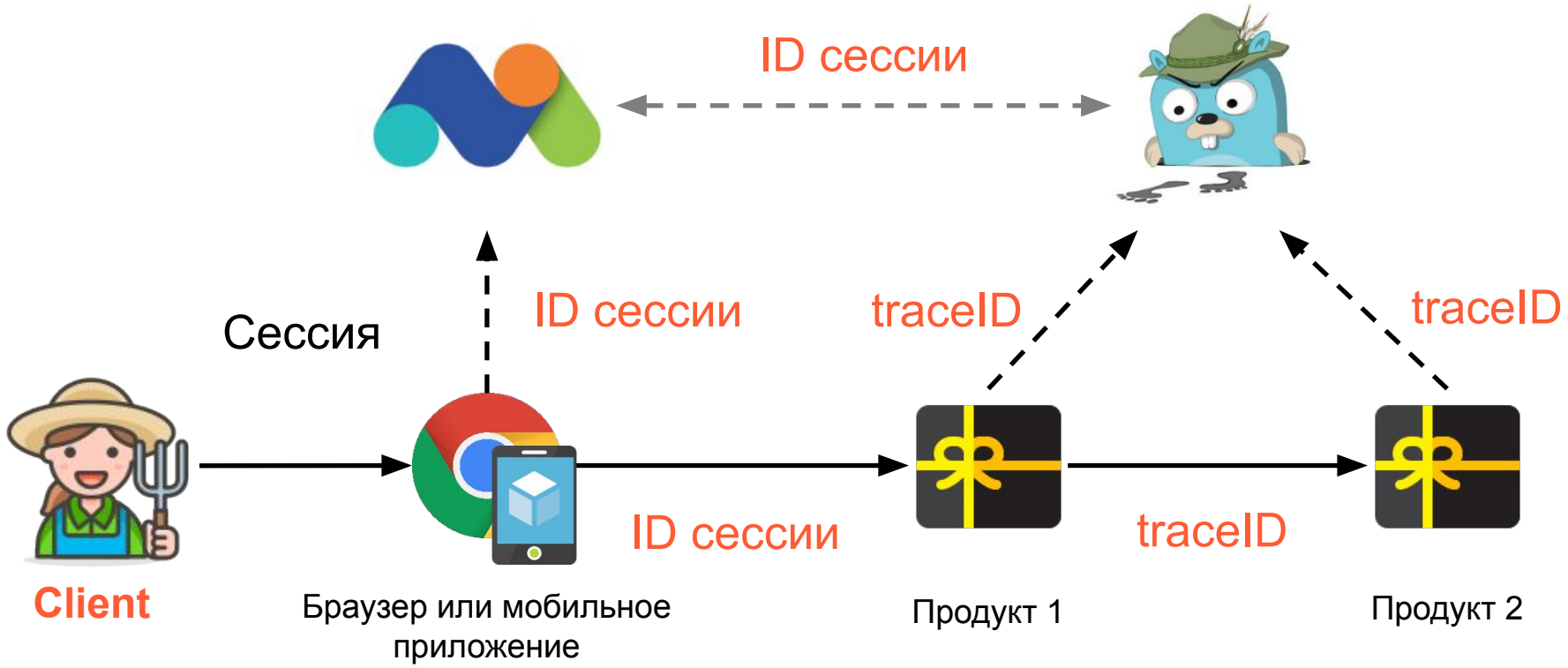
- Реальные пользователи и их транзакции
- Телеметрия с клиента
- Целые пользовательские сценарии

- Роботы и синтетический мониторинг
- Серверные метрики
- Отдельные методы и запросы

Связь Matomo и Jaeger

Связываем браузерную сессию Matomo с серверными трейсами Jaeger

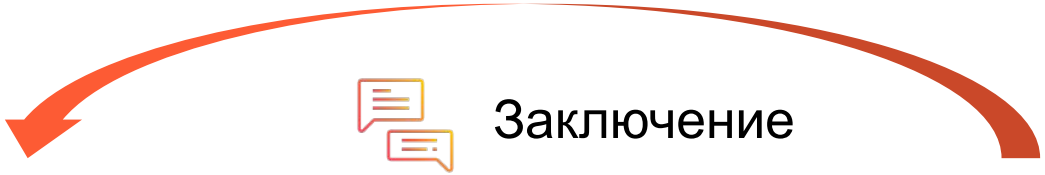
- Как ошибки сервера влияют на клиента?
- Сколько пользователей столкнулось с проблемой?
- Из чего состоит время загрузки страницы?



Тестировщикам и QA инженерам

Взаимодействие с QA

Баг



Заключение

Developer

QA

- Инструменты нагрузочного тестирования
- Автоматизация тестов
- Отчет о тестировании

Продукт

Модель наблюдаемости

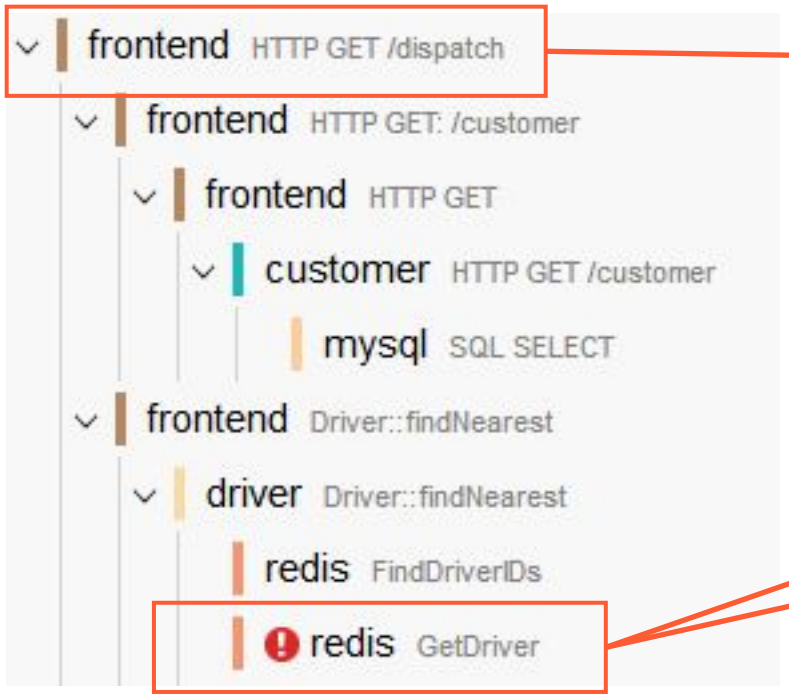
Профиль нагрузки

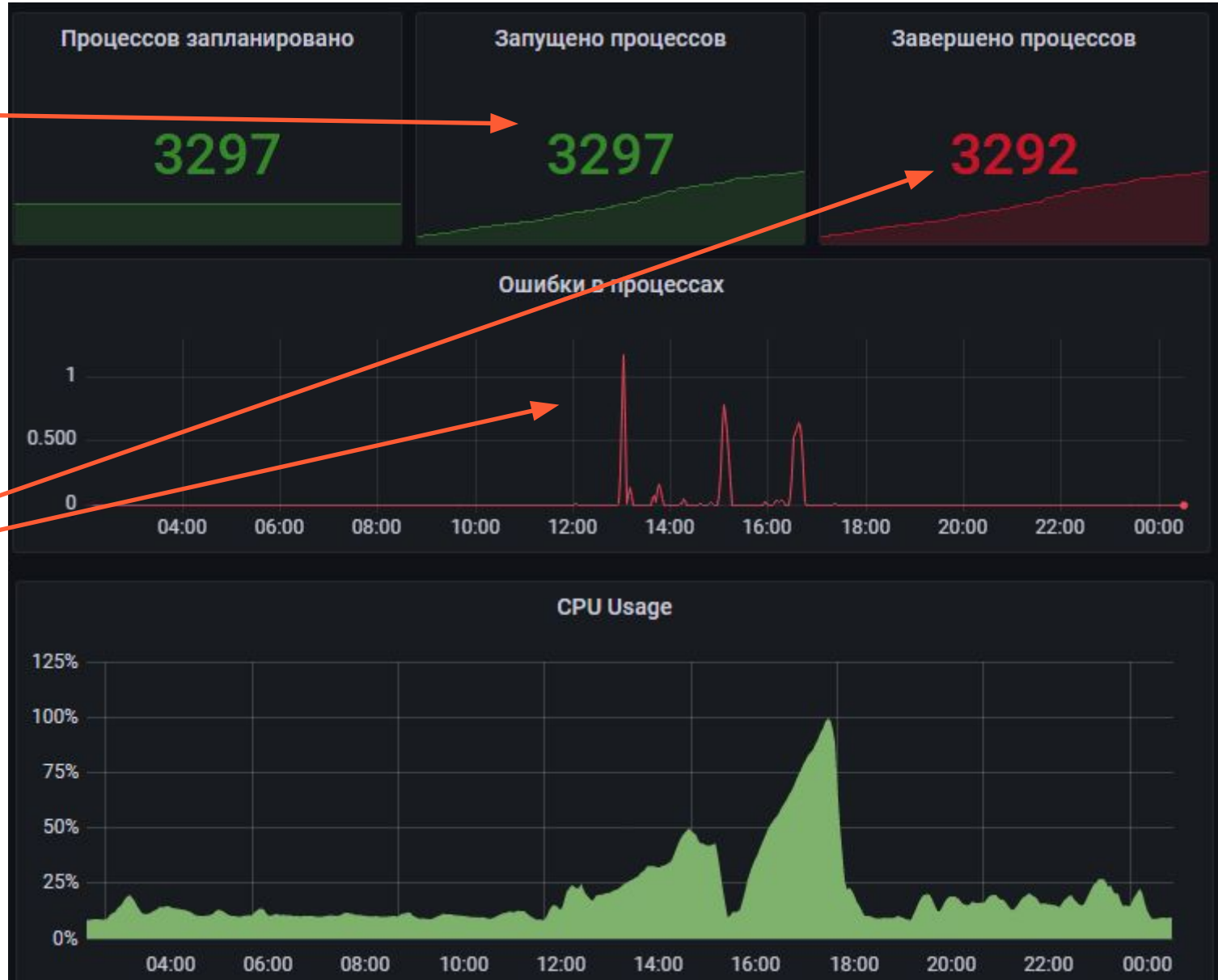
Автоматическое формирование профиля нагрузки любого стенда:

- Воспроизведение продуктивной нагрузки на тесте
- Контроль регресса производительности между релизами

Сервис	Метод	Кол-во вызовов	95й перцентиль, мс
WebApi	**ProductController.GetProducts**	18 435	12
Inventory	**ProductController.v2.GetProducts**	18 364	7
ResultStorage	**OperationResultController.FilerResult**	4 322	51

Формирование отчета о тестировании





- Корреляция проблем инфраструктуры и бизнес логики
- Контроль правильности выполнения процессов

Интеграционные тесты на базе трассировки

Malabi – фреймворк тестов на базе распределенной трассировки

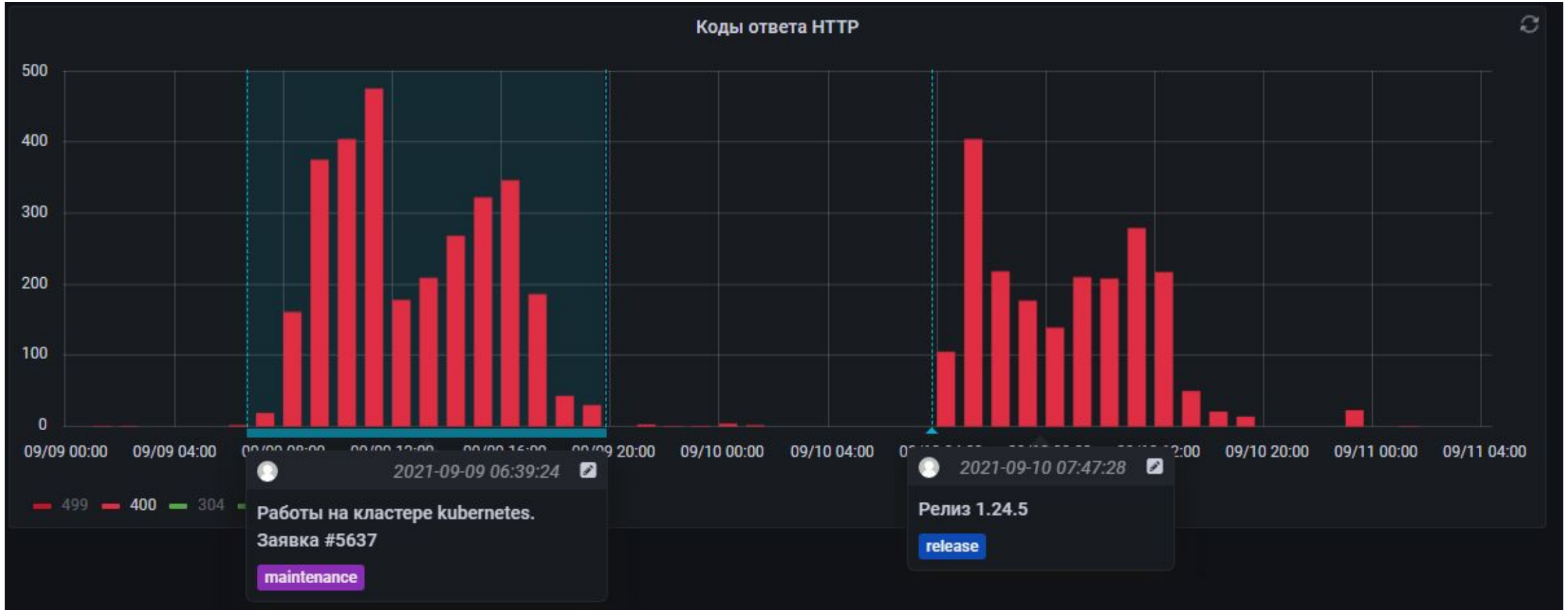
```
it('Запрос несуществующего пользователя - /users/Rick111', async () => {
    // Вызываем метод API
    const res = await axios.get("http://localhost/users/Rick111");

    // Получаем трассировку вызова
    const telemetryRepo = await getTelemetryRepository();

    // Проверяем соответствие трассировки
    const sequelizeActivities =  telemetryRepo.spans.sequelize();
    expect(sequelizeActivities.length).toBe(1);
    expect(sequelizeActivities.first.dbOperation).toBe("SELECT");

    const dbResponse = JSON.parse(sequelizeActivities.first.dbResponse);
    expect(Array.isArray(dbResponse)).toBe(true);
    expect(dbResponse.length).toBe(0);

    expect(telemetryRepo.spans.httpGet().first.statusCode).toBe(200);
});
```

- Можно проверить внутреннее состояние компонента
- Тесту не нужно множество клиентов к различным API компонент
- White box testing: тест “знает” о внутренней реализации

Для DevOps

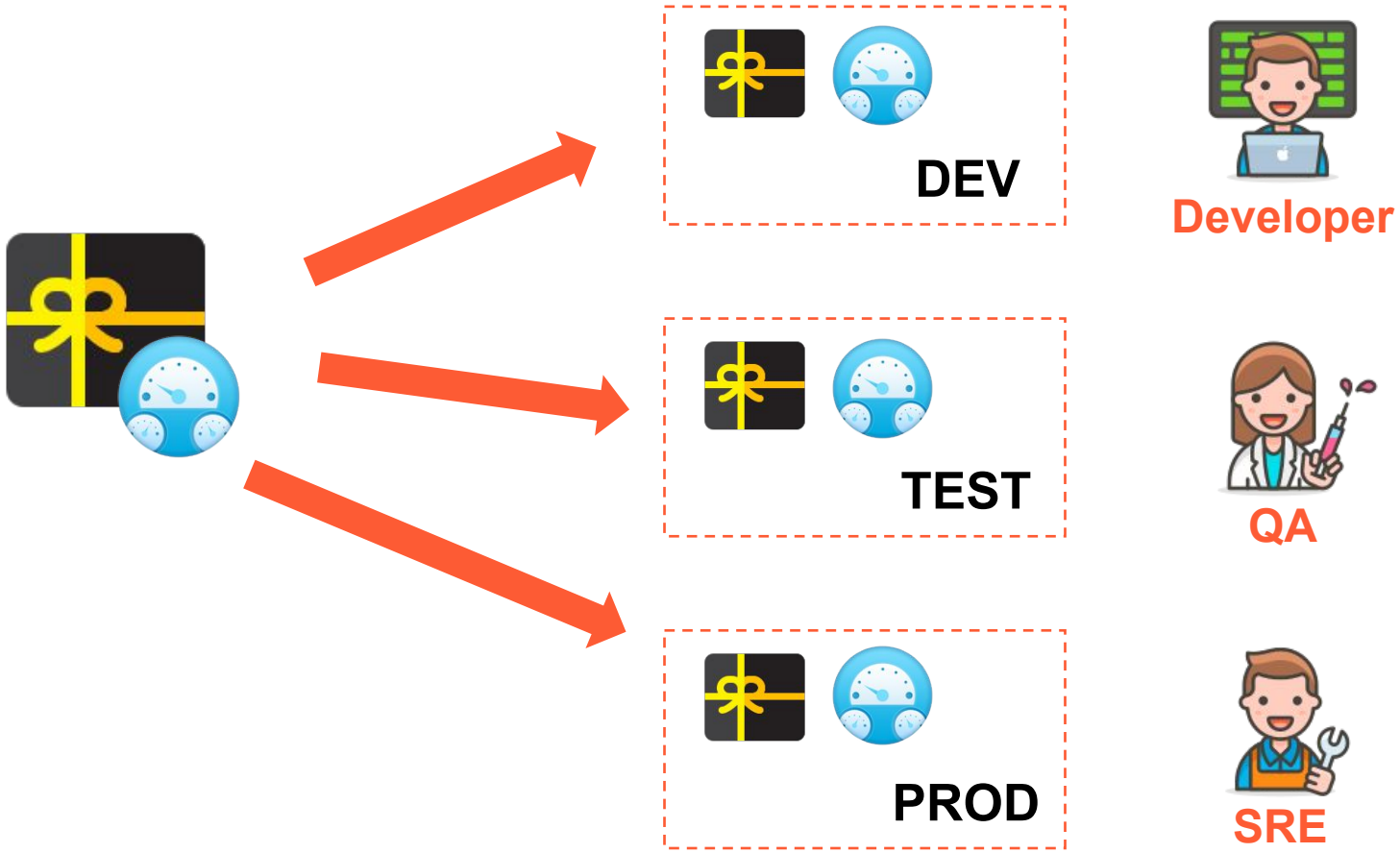
Связь метрик с событиями конвейера

- У нас проблемы или это влияние плановых работ?
- Проблемы связаны с новым релизом?



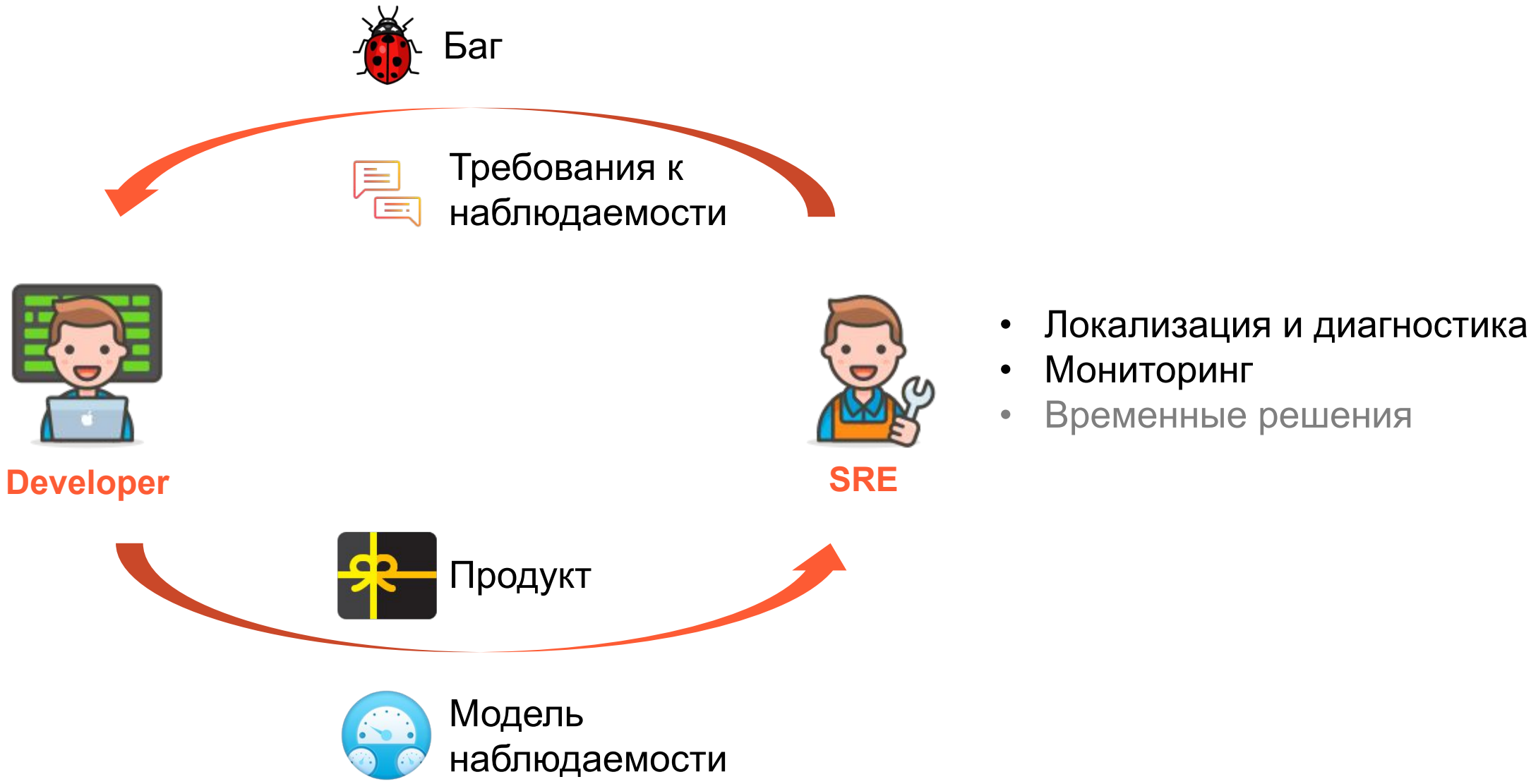
Observability as a code

Неважно сколько у вас сред – модель наблюдаемости поставляется вместе с кодом.
Не требуется настройка каждого нового стенда.



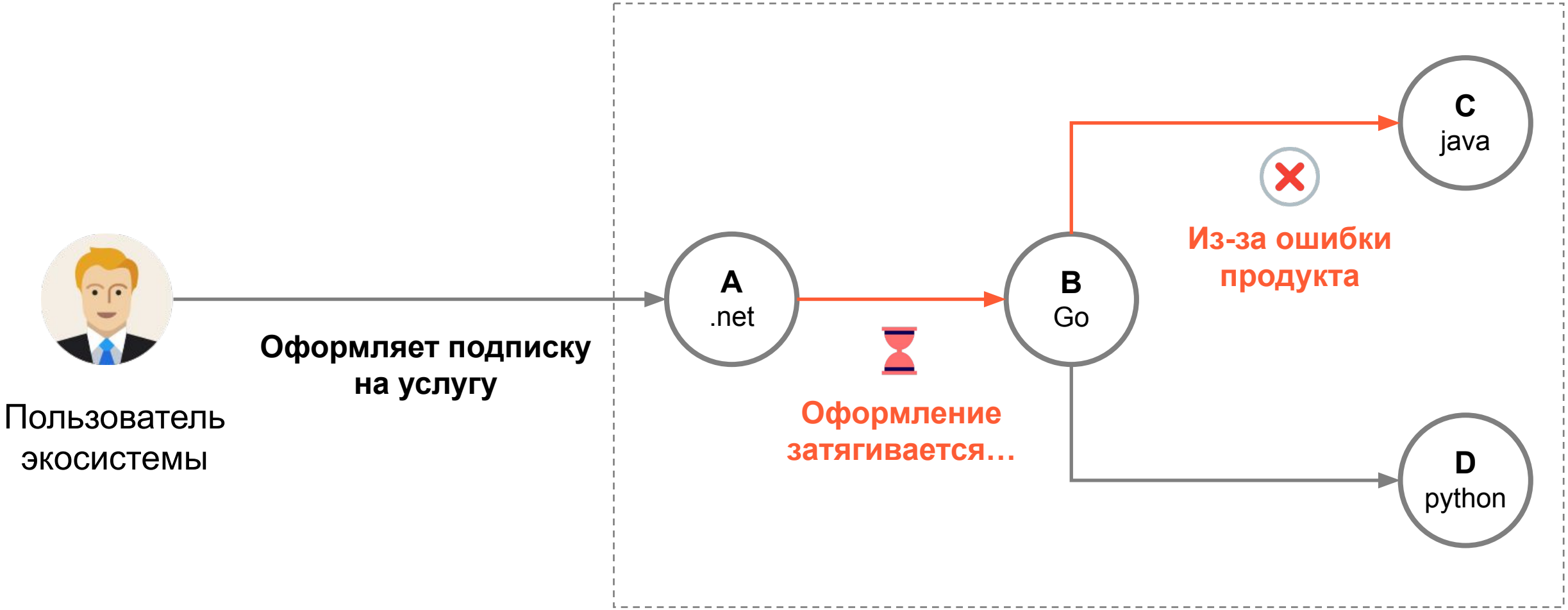
Для SRE и тех.поддержки

Идеальное взаимодействие с SRE



- Локализация и диагностика
- Мониторинг
- Временные решения

Локализация и диагностика в экосистеме



Хотим быстро понять, что во всем виноват продукт C

4 золотых сигнала мониторинга

Задержка

Время обработки запроса

99й перцентиль длительности спана обработки запроса в **распределенной трассировке**

Насыщение

Достижение предела производительности

Достижение порогового значения **метрик** потребления CPU / RAM / …

Трафик

Нагрузка на систему

Количество спанов обработки запроса в **распределенной трассировке**

Ошибки

Ошибки обработки запросов

Количество записей **логов** с уровнем ERROR или FATAL

Автоинструментирование

Сбор трассировки без изменения кода в управляемых средах.

```
java -javaagent path/to/opentracing-agent.jar <yourapp>
```

- ✓ не нужно привлекать разработчика
- ✗ невозможно управлять деревом трассировки
- ✗ работает не для всех платформ

Архитекторам

Карта IT ландшафта

Анализ связей между спанами в распределенной трассировке.

Сущности:

- ИТ-продукты
- Сервисы
- Сервера

Метрики:

- RPS
- Ошибки / коды ответа
- Уникальные методы
- Длительность

Стоимость решения и проблемы реализации

Выбор хранилища логов

Elasticsearch

Полнотекстовый поиск,
индексация ключей и сообщения

- Высокое потребление ресурсов
- Быстрый поиск по ключам
- Быстрый поиск по телу сообщения

VS

Grafana Loki

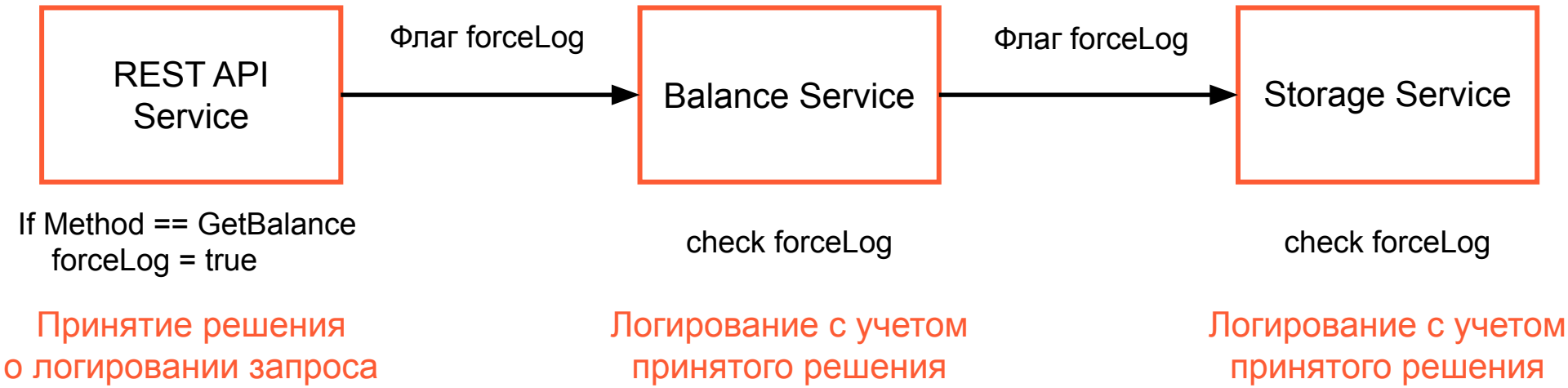
Индексация ключей
(меток)

- Низкое потребление ресурсов
- Быстрый поиск по меткам
- Медленный поиск по телу сообщения

Продуманное структурное логирование
может снизить стоимость хранения логов

Выборочный сбор логов

Нужен компромисс между влиянием на систему и подробностью логов.



Текущий подход:

- Простые условия сбора
- Привязка фильтра к определенному шагу процесса (однократное вычисление)
- Распространение “решения” фильтра через baggage

Системы распределенной трассировки

Jaeger

- Поддержка нескольких бэкендов
- Базируется на OpenTracing
- Совместимость с OpenTelemetry
- Совместимость с Open Zipkin
- Поддержка Remote sampling
- Apache 2.0 license

VS

Elasticsearch APM

- Трассировка и некоторые метрики
- Совместимость с OpenTracing
- Совместимость с OpenTelemetry
- Корреляция логов
- Поддержка Remote sampling
- Apache 2.0 license для клиентов
- SSPL / Elasticsearch license на сервер

Вывод:

1. Elasticsearch предлагает более комплексное решение
2. Мы выбрали Jaeger из-за простоты модификации и возможности “сменить поставщика”

Высоконагруженные системы

Наблюдаемость – не бесплатное удовольствие, особенно для высоконагруженных систем.

Стратегия сэмплирования	Как работает	Поддержка
Constant	Все или ничего Пример: полный сбор данных	Jaeger, OpenZipkin, Elastic APM
Probabilistic	Вероятностный сбор Пример: сбор 10% трейсов	Jaeger, Elastic APM
Rate Limiting	Ограничение потока трейсов Пример: не более 2 трейсов в секунду	Jaeger, OpenZipkin
Remote (Adaptive)	Централизованное управление динамическая настройка Пример: 10% для модуля A и 2 трейса/секунду для B	Jaeger, Elastic APM

Влияние на систему

Показатель	Влияние
RAM	Незначительный рост +100 KB
CPU	На уровне шума, <1%
Время отклика	Без изменений
Network	Незначительное увеличение

Реальный опыт:

- Продуктивный контур
- .net 4.7.2
- ~600 TPS
- Windows
- 100% сбор трассировки

Толерантность к ошибкам - ошибки Jaeger не влияют на приложение, но данные теряются

“Гигантские” трейсы

Проблемы больших трейсов (> 3000 спанов):

- Фризы интерфейса при отображении
- Большая вложенность, усложняющая поиск
- Нагрузка на бэкенд

Тысячи спанов это много… а как насчет миллиона?

- Длительные процессы с логикой переповторов и “зомби” процессы
- Обработка батчей данных
- Ошибки инструментирования: “склеивание” трейсов

Сокращение размера трейсов

Трейс – средство локализации интеграционных проблем, а не их диагностики!
Нужен баланс между подробностью трейса и удобством анализа.

Выбрали самые большие, сгруппировали, отсортировали:

POST /order/process	1446 шт.	33%	Важно
Action OrderController.ProcessOrder	1446 шт.	33%	Важно
SQL i_order.SaveData	907 шт.	12%	Не важно
SQL i_order.UpdateData	867 шт.	11%	Не важно

Объединяем

В логи

На практике: удавалось снизить количество спанов на 40%

Есть предел – далее только доработка UI

Выводы

Business

- Снижение time to market
- Снижение операционных расходов
- Контроль качества и пользовательского опыта

SRE

- Продукт, готовый к настройке мониторинга
- Простая локализация проблем

Developer

- Средства отладки распределенных систем
- Быстрый цикл отладки
- Простая коммуникация с SRE и QA

QA

- Средства автоматизации рутинных задач

Наблюдаемость системы положительно влияет на всех членов продуктовой команды

Стоимость обеспечения наблюдаемости

Дорого и сложно

- Потенциальные потери из-за низкого качества продукта
- Часы разработчиков, тестировщиков и инженеров для ручной диагностики и тестирования

Сложно и дорого

- Лицензии
- Железо для обработки и хранения
- Часы разработчиков и инженеров для настройки

Список литературы

Список литературы

1. **Gartner Magic Quadrant 2021 APM** - https://www.appdynamics.com/resources/reports/gartner-magic-quadrant-apm
2. **ГОСТ Р ИСО/МЭК 25010-2015 -** https://docs.cntd.ru/document/1200121069
3. **Google SRE Book -** https://sre.google/sre-book/monitoring-distributed-systems/#xref_monitoring_golden-signals
4. **Mastering Distributed Tracing -** https://www.amazon.com/Mastering-Distributed-Tracing-performance-microservices-ebook/dp/B07MBNGF7Q
5. **Cloud Native Computing Foundation (CNCF) -** https://www.cncf.io/
6. **OpenTelemetry Specification -** https://github.com/open-telemetry/opentelemetry-specification
7. **Trace-based testing with Malabi** - https://www.cncf.io/blog/2021/08/11/trace-based-testing-with-opentelemetry-meet-open-source-malabi/
8. **Error tracking vs logging** - https://sentry.io/vs/logging/