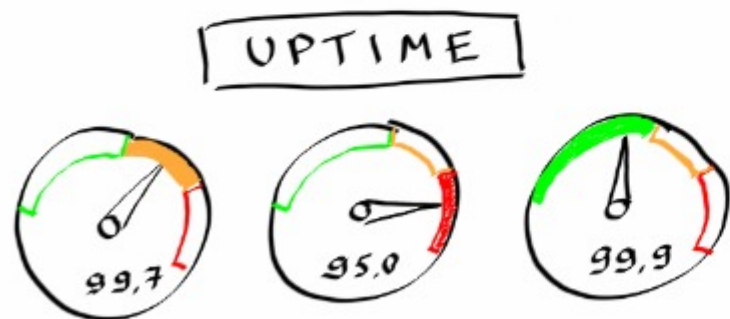


Все про НФТ

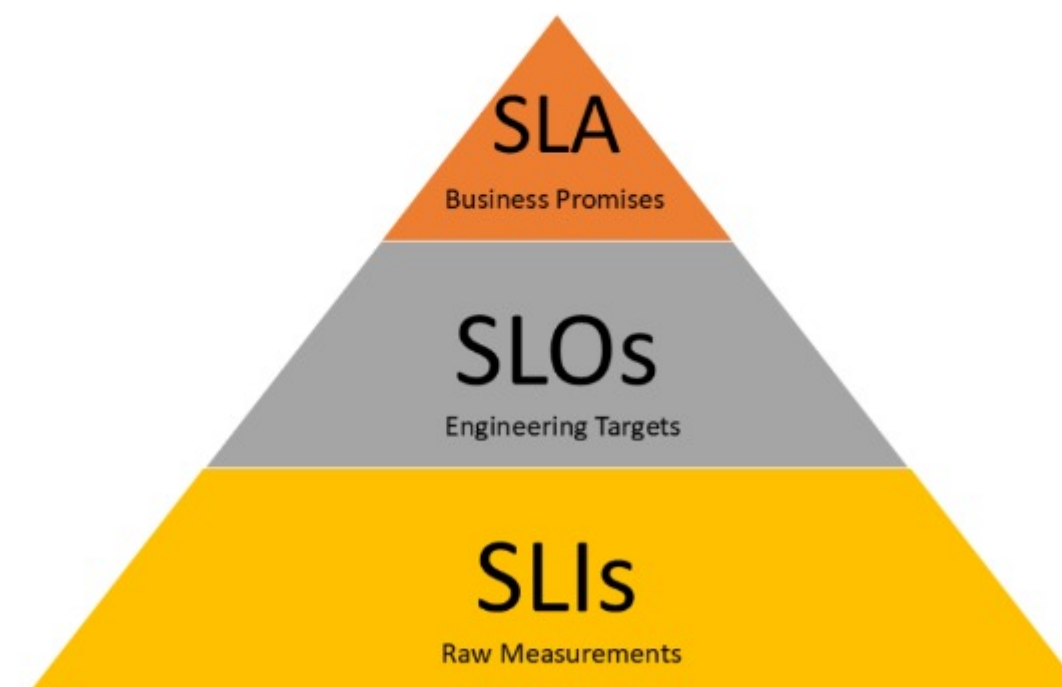
нефункциональные требования

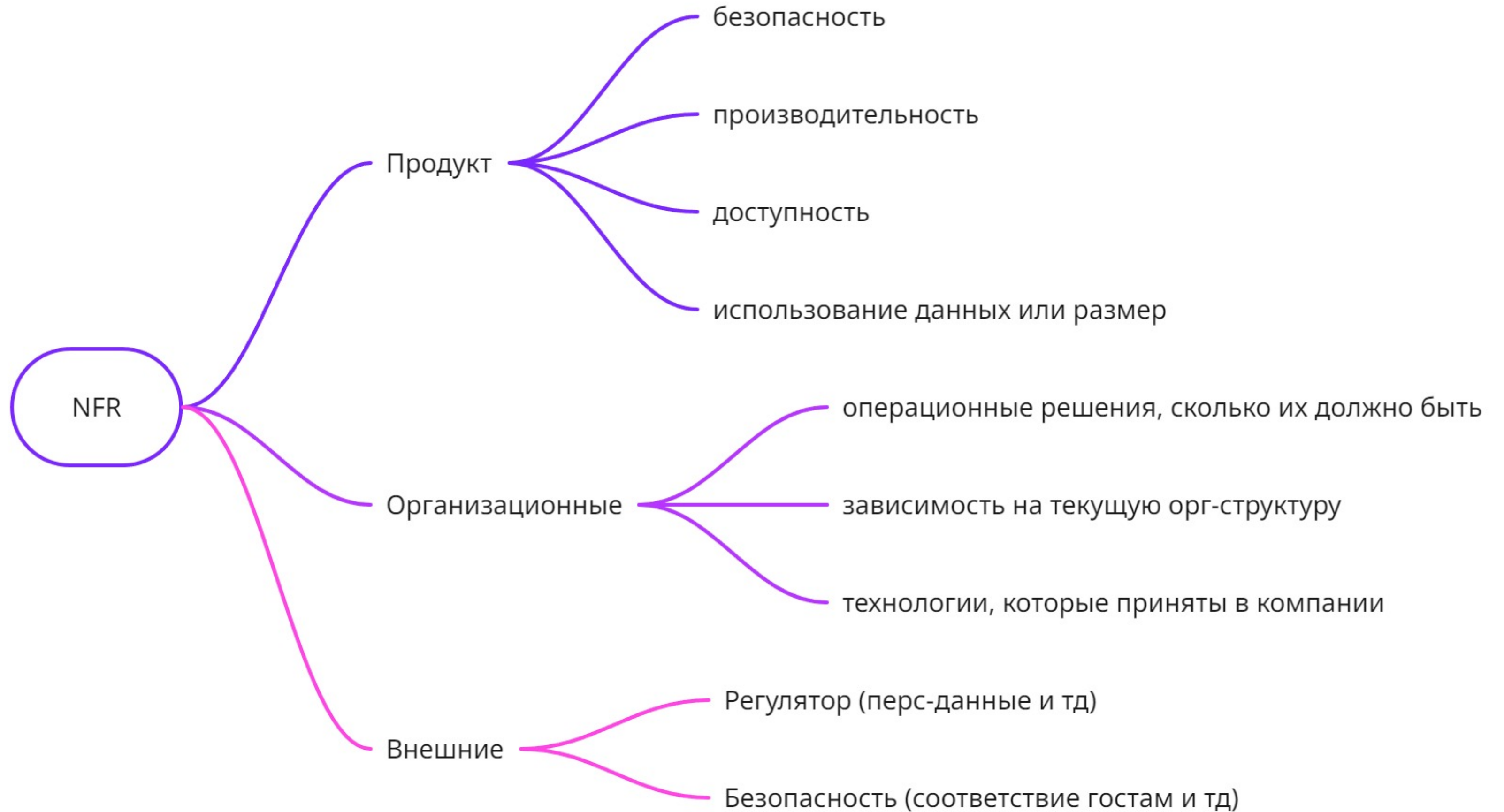
мифические 99.99

Почему вам это не надо и как правильно
сформулировать и замерять



99.99





Приходится думать о

- скорости работы
- количестве хранимых данных
- фреймворках
- технических решениях
- стоимости разработки



В индустрии принято считать доступность девятками

- 99 -> две
- 99.9 -> три

НЕ ВСЕМ СИСТЕМАМ НУЖНО 99.99 или даже 99

<https://uptime.is/99.9>

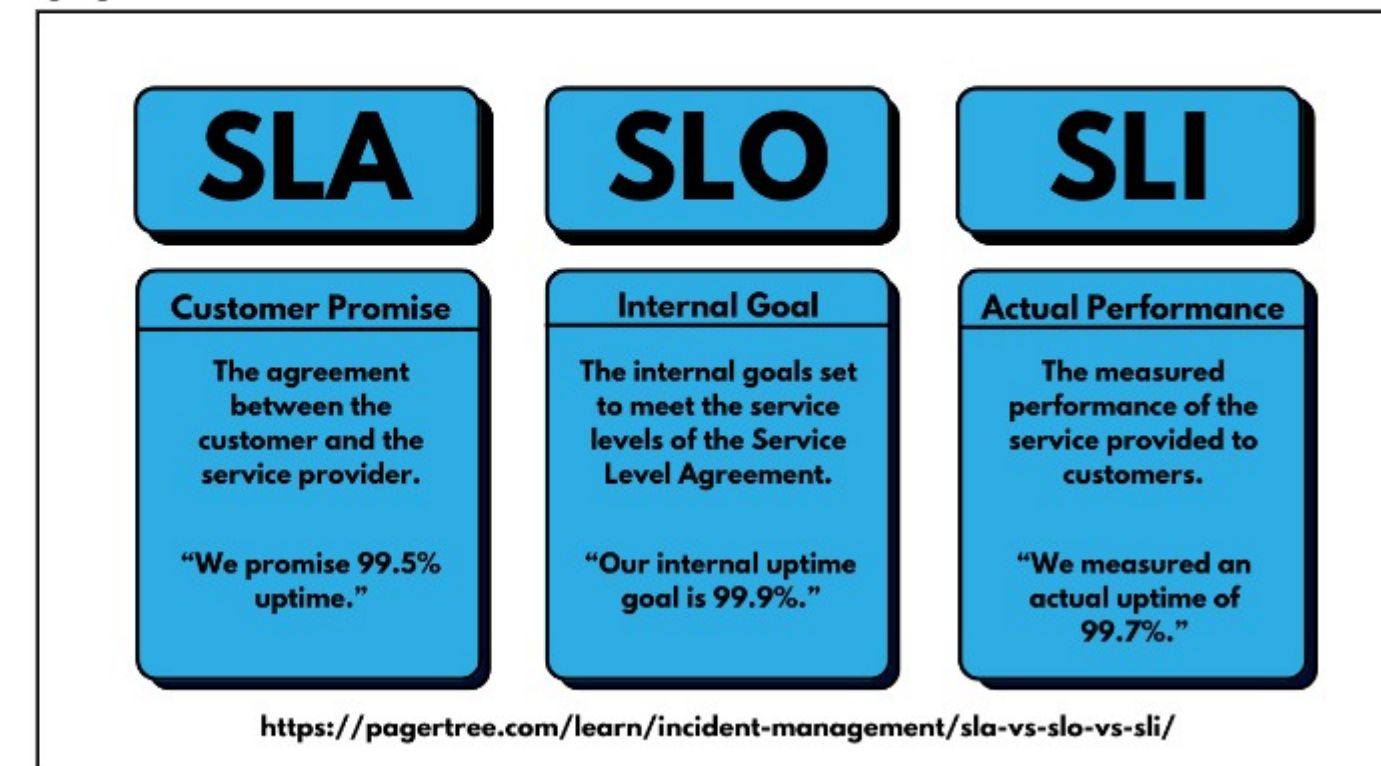
<https://andrewmatveychuk.com/why-99-99-uptime-or-sla-is-bad>

Это нужно для оценки качества решения

- И есть 3 ключевых слова про это SLI, SLO, SLA
- SLO - что мы декларируем, желаемые значения каких-то метрик (например 99 перцентиль времени ответа не более 300мс или количество 500 не более 1%)
- SLI - фактическое значение метрик, которые мы замерили. Время ответа, коды ответов
- SLA - договоренности для клиентов (что делаем когда SLI меньше SLO), на практике сталкивался со скидками на подписки в следующий месяц

<https://cloud.google.com/blog/products/devops-sre/sre-fundamentals-sli-vs-slo-vs-sla>

<https://www.atlassian.com/incident-management/kpis/sla-vs-slo-vs-sli>



Что и где считаем

- Замеры производим не в самом сервисе, а на прокси, балансере или сайдкаре сервис меша
- Выбираем надежность которая нас интересует, **я советую для старта 99%**
- Для RT (**по перцентилю**) до 99%, **дальше нужно больше данных**
 - По каждому эндпоинту фиксируем время ответа
 - Собираем метрики времени ответа (RT) нужного перцентиля для нас 99
 - Считаем соответствие фактического RT и заданного
- **По бюджету** (
 - Фиксируем время ответа
 - **SLI = good_requests / total_requests**
 - good_requests - RT < X ms
 - total_requests - все запросы
 - остальные запросы плохие, их должно быть не больше чем 100% - NFR%, их можно считать в штуках и вести бюджет
 - По простому, если у нас 10000 запросов и 99%, то у нас 100 плохих запросов

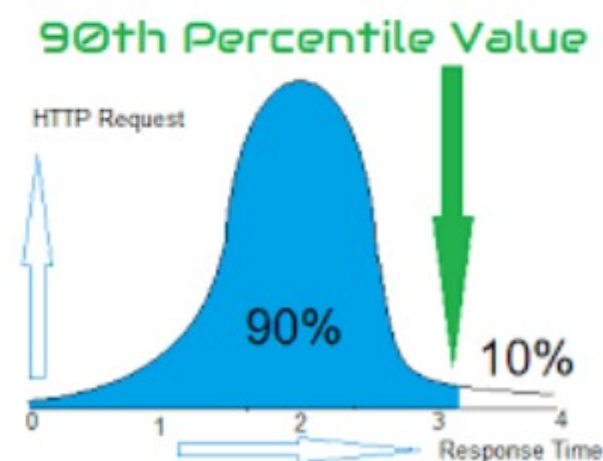
<https://habr.com/ru/companies/avito/articles/742960/>

GET /v1/users -> users[], 200, 400

Фиксируем RT в 1с

Считаем фактически по метрикам
99 перцентилю RT получаем
900мс

Все хорошо



Что и где считаем

- Для кодов ответов

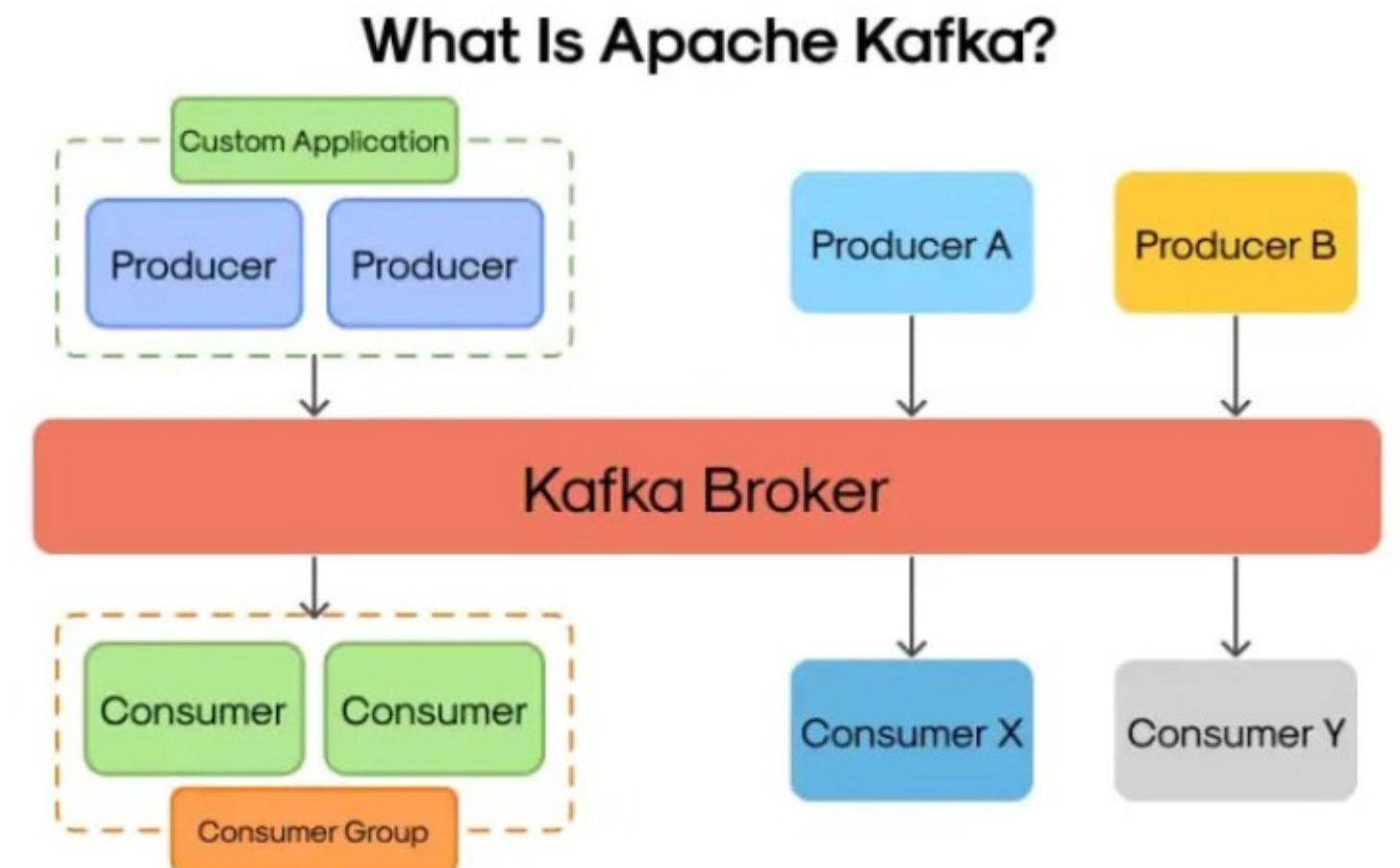
- SLI по кодам ответов - это **успешность обработки запросов**.
- Надежность из прошлого пункта 99%
- Выбираем какие коды ответов считаем ошибками для старта пойдет 500, 503, 504, 400 (посмотреть много ли плохих запросов)
- **SLI = good_responses / total_responses**
- Замеры проводим за интервал
 - обычно 30 дней или 1 день
- Для RPC считается аналогично
- Иногда для упрощения считают комбинированный SLI, `status=ok and latency < X ms` и **это итоговая надежность**



GET /v1/users -> users[], 200, 400 → 8999 ответов 200
299 ответов 400
103 ответа 500 → $SLI = 8999 / (8999 + 299 + 103) = 0.957$

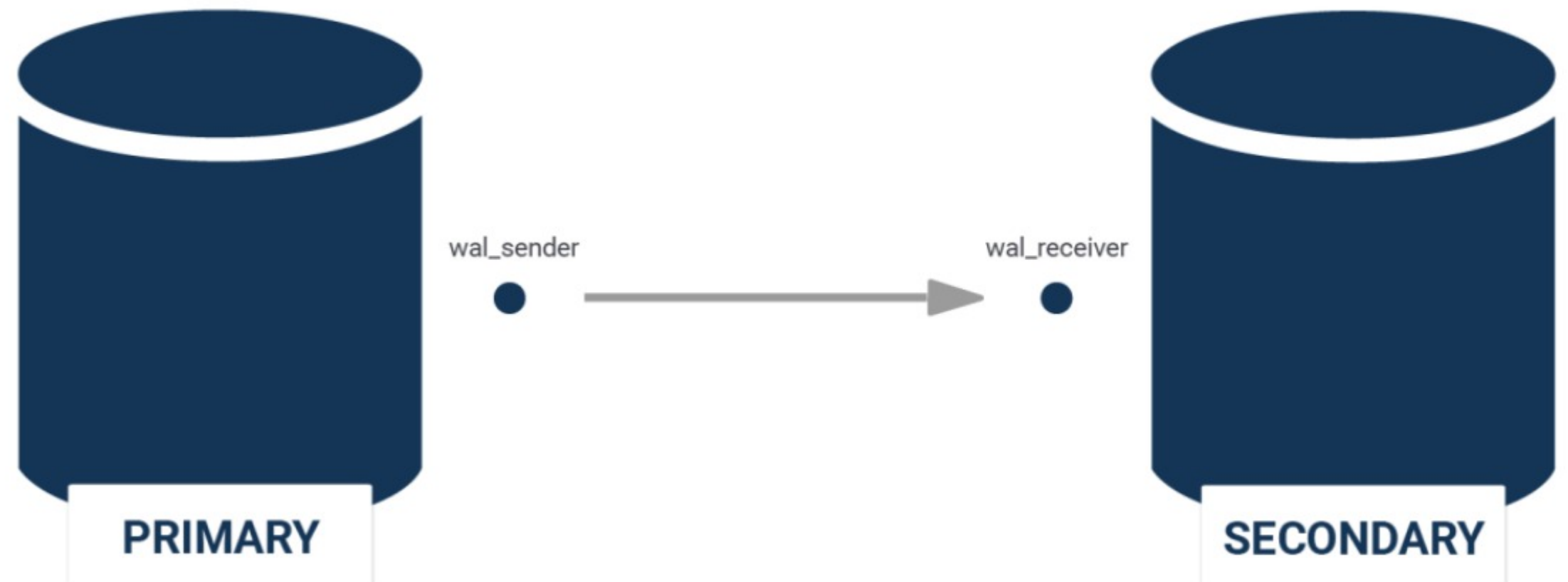
Что и где считаем

- Для очередей
 - **SLI = good_messages / total_messages**
 - total_messages - все сообщения, опубликованные в очередь
 - good_messages - сообщения, которые:
 - доставлены consumer'у
 - за допустимое время
 - обработаны успешно*
 - не ушли в DLQ*
- Как считать latency
 - producer ставит produced_at
 - consumer фиксирует processed_at (time.now)
 - latency = processed_at - produced_at



Что и где считаем

- Для баз данных
 - **$SLI_availability = \text{successful_queries} / \text{total_queries}$**
 - `successful_queries` - запросы, которые завершились без ошибок
 - **`total_queries` - все запросы**
 - **$latency = \text{query_completed_at} - \text{query_started_at}$**
 - **$replication_lag = \text{primary_tx_id} - \text{replica_tx_id}$**
- Придется собирать и логировать запросы, для расчёта надежности, в зависимости идем когда не понятно где что-то идет не так



NFR - инструмент управления разработкой

- Позволяет определить как система **работает для большинства** пользователей
- Помогает **принимать решения** в разработке
 - Нужна ли эта оптимизация или преждевременно
 - Нужен ли срочный хотфикс и доработки
 - Можно ли катить доработки
- Оперативная реакция на изменения NFR помогает предотвратить **массовые инциденты** или обнаружить проблему
- Есть разные методики расчёта - самый простой **перцентили** и продвинутый **бюджет ошибок**
- С бюджетом ошибок появляются хаки когда можно катить потенциально опасные фиксы или ломать инфраструктуру, в **малоактивные часы или безопасные**, чаще всего по ночам пользователей меньше и при полном отказе ошибок в штуках не станет уж очень много
- Считать метрики необходимо отдельно от приложения
- Каждая дополнительная 9 - это **большой прирост в цене** для бизнеса, метрики могут обосновать затраты

