

Модуль "Мониторинг» Лекция №2

Бочаров Филипп

Руководитель центра мониторинга и наблюдаемости в MTC Digital

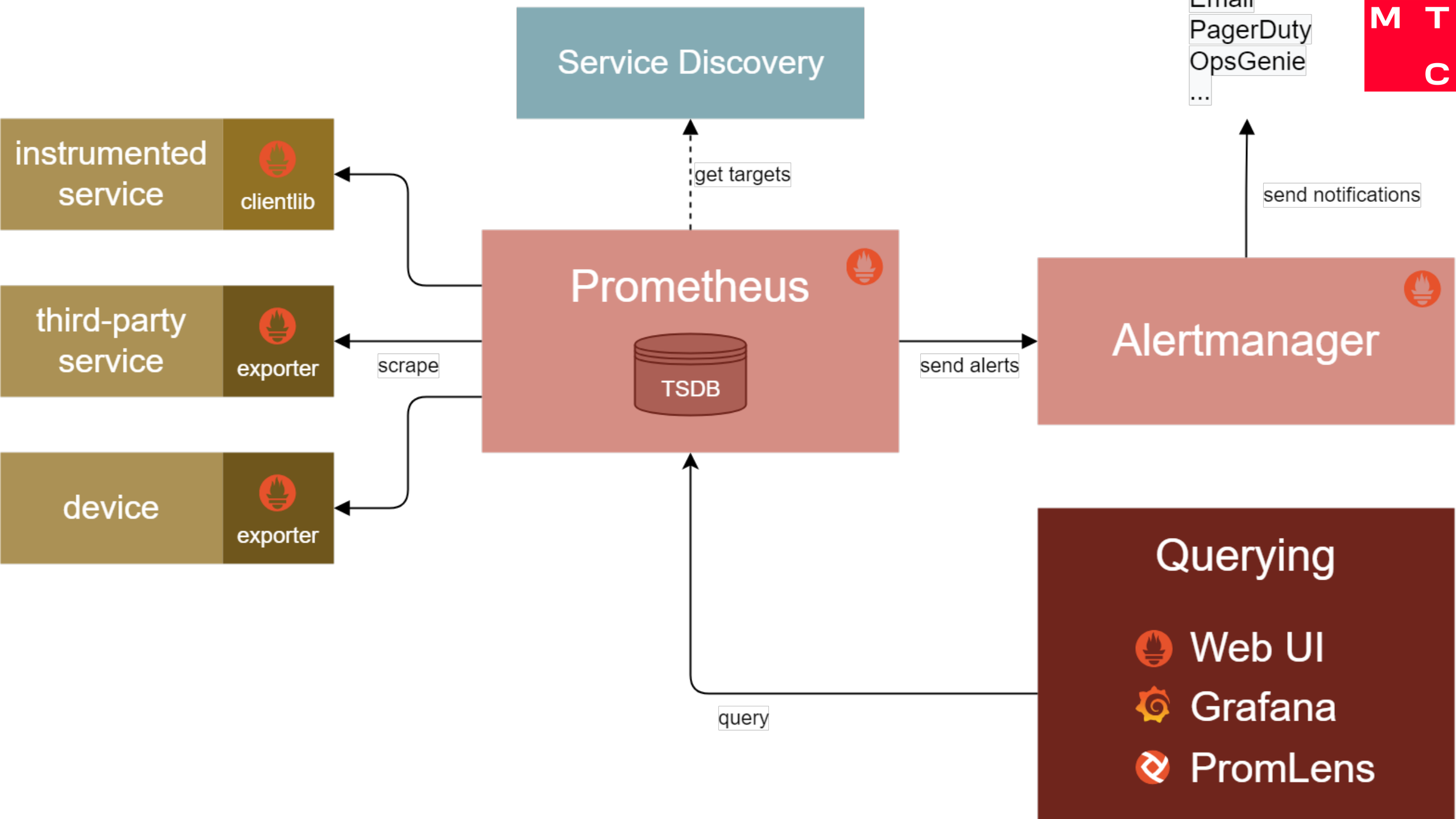
Занимаюсь разработкой платформы Наблюдаемости. Помогаю продуктовым командам сделать работу сложных распределенных систем понятной и прозрачной.

Спикер Highload++, Dotnext, TechleadConf ...



Тема 4

Подробно про Prometheus



Формат метрик Prometheus (OpenMetrics)

```
← → ↻ 🏠 ⓘ localhost:3000/metrics

# TYPE http_server_requests_total counter
# HELP http_server_requests_total The total number of HTTP requests handled by the Rack application.
http_server_requests_total{code="200",method="get",path="/"} 1.0
# TYPE http_server_request_duration_seconds histogram
# HELP http_server_request_duration_seconds The HTTP response duration of the Rack application.
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.005"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.01"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.025"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.05"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.1"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.25"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.5"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="1"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="2.5"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="5"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="10"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="+Inf"} 1.0
http_server_request_duration_seconds_sum{method="get",path="/"} 0.251396
http_server_request_duration_seconds_count{method="get",path="/"} 1.0
# TYPE http_server_exceptions_total counter
# HELP http_server_exceptions_total The total number of exceptions raised by the Rack application.
```

Модель данных Prometheus



Правильно выбираем разрезы метрик

Метрика **action_duration_ms** – время работы метода REST API

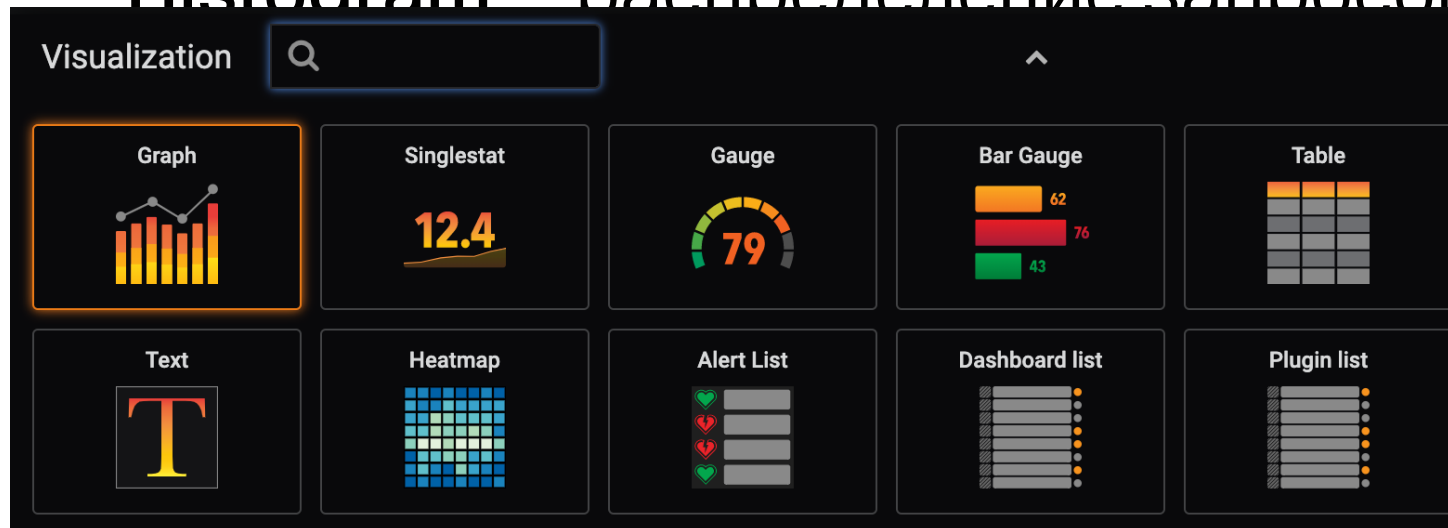
Метки:

- productID 90 продуктов x
- tenantID 2 контура x
- serviceName 10 сервисов x
- controller 2 контроллера x
- action 10 методов

Итого Cardinality = 36 000

Модель данных Prometheus

- **Counter** – кол-во обработанных запросов
- **Summary** – кол-во запросов, обработанных за последние 10 минут
- **Gauge** – кол-во запросов в очереди
- **Histogram** – распределение запросов по длительности



Инструментирование собственного кода

Разработчик может самостоятельно выставлять метрики из кода приложения:

- Prometheus SDK (множество языков)
- App Metrics (.net)
- OpenTelemetry SDK (множество языков)
- Micrometer (java)

Многие SDK интегрируются с фреймворками и выставляют метрики автоматически.

```
private static readonly Counter ProcessedJobCount = Metrics
    .CreateCounter("myapp_jobs_processed_total", "Number of processed jobs.");

ProcessJob();
ProcessedJobCount.Inc();
```

Скрейпинг

```
global:
  scrape_interval: 10s
scrape_configs:
  - job_name: python-app
    static_configs:
      - targets:
          - localhost:8000
    labels:
      my_new_target_label: foo
```

- Задаем интервал скрейпа
- Задаем URL адреса для скрейпа
 - Статически
 - Динамически
 - Из файла
 - Из конечной точки
 - Из оркестраторов и провайдеров

Язык PromQL

```
histogram_quantile(  
  0.9,  
  sum by(le, method, path) (  
    rate(  
      demo_api_request_duration_seconds_bucket{job="demo"}[5m]  
    )  
  )  
)  
)
```

Root of the query, final result, approximates a quantile.
1st argument to histogram_quantile(), the target quantile.
2nd argument to histogram_quantile(), an aggregated histogram.
Argument to sum(), the per-second increase of a histogram over 5m.
Argument to rate(), the raw histogram series over the last 5m.

```
histogram_quantile(...) 5 results - method:2, path:3  
├── 0.9  
└── sum by(le, method, path) (...) 130 results - le:26, method:2, path:3  
    ├── rate(...) 702 results - instance:3, job:1, le:26, method:2, path:3, status:3  
    └── demo_api_request_duration_seconds_bucket{job="demo"}[5m] 702 results - instance:3, job:1, le:26, method:2, path:3, status:3
```

Производные метрики

```
groups:  
- name: exampleGroup  
  rules:  
- record: rate5m:http_request_duration_bucket:sum:without_instance  
  expr: sum without(instance) (rate(http_request_duration_bucket[5m]))  
  
- record: node_memory_MemFree_percent  
  expr: (100 * node_memory_MemFree_bytes / node_memory_MemTotal_bytes)  
  labels:  
    [ resource: memory ]
```

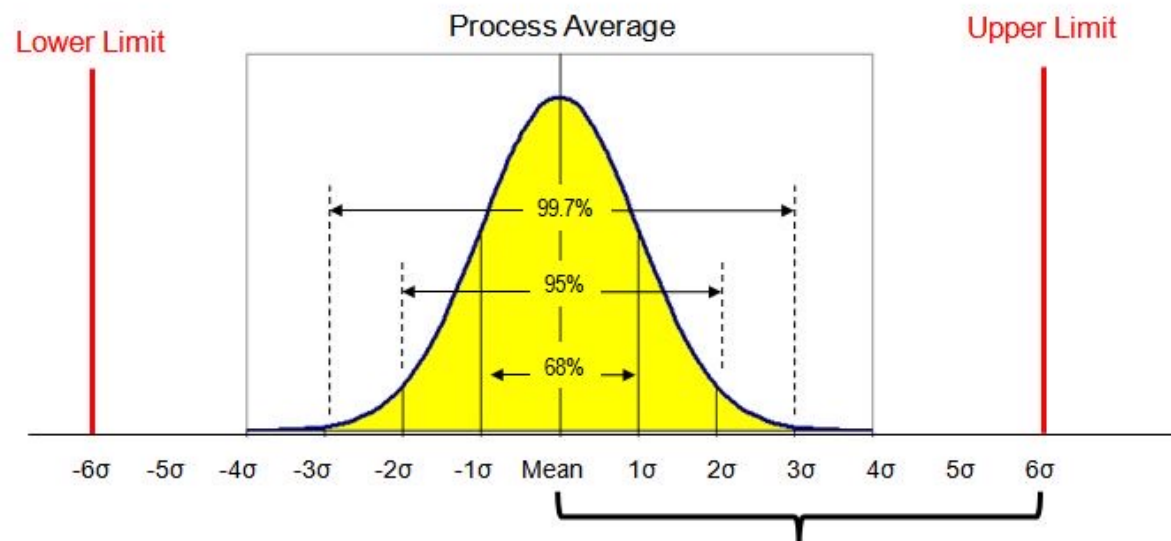
Alertmanager

```
groups:  
- name: example  
  rules:  
  - alert: HighRequestLatency  
    expr: avg(request_latency_seconds) > 0.5  
    for: 10m  
    labels:  
      severity: warning  
    annotations:  
      summary: High request latency
```

- Выражение из исходных метрик
- Пороговое значение
- Описание ответного действия *
- Задержка срабатывания *
- Метаданные *

Подбор порогового значения

- На основании норматива / SLA / требования регулятора
- На основании анализа долгосрочных трендов:
 - Взять MAX/MIN за период
 - Использовать baseline – автоматически вычисляемый допустимый порог



Тема 5

**Дашборды, на которые не
страшно смотреть**

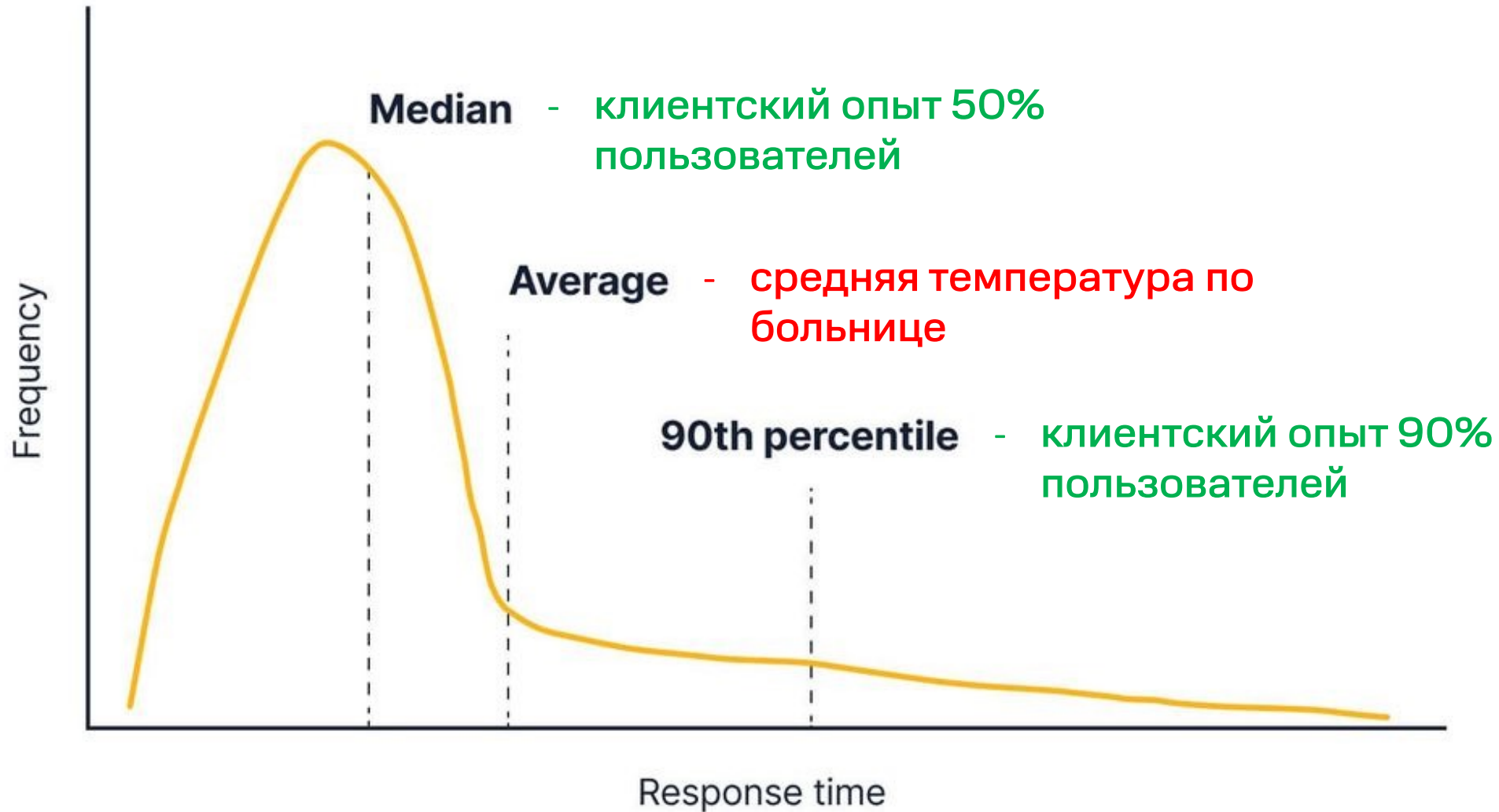
Организация дашбордов

- **Стратегия мониторинга первична** - структура дашбордов и каталогов следует стратегии и архитектуре продукта
- **Иерархическая структура** - от общего к частному, от overview к деталям
- **Простой поиск** - понятные названия, группировка по каталогам и теги

Хороший дашборд

- Панели сгруппированы в логические группы
- Указаны единицы измерения
- Правильно выбран масштаб
- Интуитивно понятные цвета: **красный** - проблема, **зеленый** - ОК
- Если метрик несколько - добавлена легенда
- Указаны пороги *
- Добавлены аннотации *
- Добавлены panel links и data links *

Перцентиль vs среднее



Тема 6

Индикаторы качества SLI/SLO

SLI/SLO/SLA

- **Service Level Indicator (SLI)** – количественная оценка качества работы сервиса

Где 100 % - хорошо, а 0% - ужасно

- **Service Level Objectives (SLO)** – целевое значение нашего SLI
- **Service Level Agreement (SLA)** - публичное значение SLO
- **Error Budget** – степень невыполнения наших SLO

Как может выглядеть подход к SLI

Продукт MegaTravel

Сценарий Поиск билетов

Индикатор Время поиска билетов < 500 мс в 99% случаев

Индикатор Успешный поиск билетов в 99.9% случаев

Индикатор Отставание изменений в базе билетов < 30 секунд

Сценарий "Оплата заказа"

Индикатор ...

















































Пример индикатора

- Индикатор по золотому сигналу **Ошибки** - делим количество ошибочных запросов на общее количество запросов
- Используем метрику количества запросов, проходящих через ingress k8s `nginx_ingress_controller_requests`

```
sum(rate(
  nginx_ingress_controller_requests{status!~"[4-5].*", k8s_cluster="my_product"}
[1m])) by (ingress) /
sum(rate(
  nginx_ingress_controller_requests{k8s_cluster="my_product"}
[1m])) by (ingress)
```



Status page

North America	South America	Europe	Africa	Asia Pacific	Middle East						
Service	RSS	⏪	Today	5 Oct	4 Oct	3 Oct	2 Oct	1 Oct	30 Sep		
Amazon EventBridge Scheduler (Canada-Central)											
Amazon EventBridge Scheduler (N. California)											
Amazon EventBridge Scheduler (N. Virginia)											
Amazon EventBridge Scheduler (Ohio)											
Amazon EventBridge Scheduler (Oregon)											
Amazon API Gateway (Canada-Central)											

Заключение

Checklist постановки на мониторинг

1. Выделить ключевые бизнес операции (сценарии) продукта. Наметить требуемый уровень качества SLA
2. Определить какие индикаторы качества нужны для контроля сценариев
3. Нарисовать архитектурную схему продукта. Выделить компоненты, связи, внешние зависимости и инфраструктуру
4. Поставить на мониторинг компоненты, влияющие на ключевые сценарии:
 - Покрыть публичные API, web-интерфейсы и внешние зависимости black box мониторингом
 - Установить экспортеры/агенты для сбора метрик с ОС и стороннего ПО
 - Инструментировать собственный код при необходимости
5. Создать типовые дашборды для каждого компонента из п4
6. Описать индикаторы из п2 на основе метрик п4. Реализовать дашборд качества
7. Настроить алертинг на нарушение SLA и ключевые метрики компонент

После каждой аварии, не выявленной мониторингом –
проходить по чек-листу снова и снова 😊

Полезные ссылки

- <https://prometheus.io/docs/prometheus/latest/querying/basics/>
- <https://prometheus.io/docs/instrumenting/exporters/>
- <https://github.com/prometheus-net/prometheus-net>
- <https://sre.google/sre-book/monitoring-distributed-systems/>
- <https://www.dynatrace.com/news/blog/why-averages-suck-and-percentiles-are-great/>
- <https://grafana.com/docs/grafana/latest/dashboards/build-dashboards/best-practices/>
- <https://docs.influxdata.com/telegraf/v1/plugins/>
- <https://sloth.dev/>
- <https://www.atlassian.com/ru/incident-management/kpis/sla-vs-slo-vs-sli>
- <https://web.dev/i18n/ru/vitals/>